Chapter 4

COMPUTER SCIENCE EDUCATION: CURRENT SITUATION, PEDAGOGICAL APPROACHES AND TOOLS[1]

Yüksel Deniz ARIKAN[2]
Figen EĞİN[3]

INTRODUCTION

Computer science is the science of computers and computational systems. Unlike engineering, computer scientists primarily work on software, software development, and the theory, design development, and applications of these fields. Computer science, which dates to the 1960s, has many different definitions. Forsythe (1967) defines computer science as the art of processing information with digital computers. Booth (2001) states that computer science is a mathematics-based technical science. Jain (2018) said that computer science is the science and art of software development.

There are significant differences between countries in terms of course content and concepts in computer science education. For example, programming education (Robins et al., 2003), algorithmic thinking (Knuth, 1985), and computational thinking (Wing, 2006; Çetin & Berigel, 2017). It is seen that concepts such as programming education, computer science education, informatics education, technology education, informatics applications, and computer applications are mentioned in the educational field (Hubwieser at All, 2015). In Türkiye, computer science concepts and information and communication technologies are used. However, computer science has generally become widespread in recent years (Gülbahar, 2017).

---

[1] This article is derived from Figen Eğin's master's thesis entitled "An investigation of information technologies teachers' opinions on coding teaching", conducted under the supervision of Asst. Prof. Dr. Yüksel Deniz Arıkan.

[2] Assist. Prof. Dr., Ege University, Faculty of Education, Department of Computer and Instructional Technologies Education, y.deniz.arikan@gmail.com, ORCID iD: 0000-0002-7151-5381

[3] Teacher, Turgutlu Science and Art Centers (BİLSEM), figenkaya@gmail.com, ORCID iD: 0000-0003-4865-5789,

Reusink (2018) emphasized the positive contribution of computer science to all areas of society by mentioning its benefits, such as problem-solving and solution development, protecting individuals and institutions, improving education, and improving communication. In our digital age, there is software coded for different purposes. It is a fact that we now live in a world surrounded by software, from our phones to our televisions. In such an environment, it is essential to teach coding to provide individuals with skills such as problem-solving, logical thinking, and creativity for real life, in short, to support computational thinking. According to García-Peñalvo, Reimann, Tuul, Rees, and Jormanainen (2016), coding instruction is a comprehensive tool for acquiring computational thinking.

Computational thinking is a skill everyone should have, such as writing, reading, or arithmetic, and is involved in all areas of life (Wing, 2006). In the report published by the International Society for Teaching and Learning (ISTE, 2023), computational thinking, which is one of the standards that students should have, is expressed as follows:

- Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models, and algorithmic thinking in exploring and finding solutions. Students collect data and use digital tools to analyze data.
- Students collect data or identify relevant data sets, use digital tools to analyze them and represent data in various ways to facilitate problem-solving and decision-making.
- Students break problems into parts, extract critical information, and develop descriptive models to understand complex systems or facilitate problem-solving.
- Students understand how automation works and use algorithmic thinking to develop steps to create and test automated solutions.

When the studies in the literature are examined, it is seen that it is possible to gain problem-solving skills, which is considered one of the 21st-century skills under the umbrella of computational thinking, through computer science education (Chao, 2016; Kalelioğlu & Gülbahar, 2014; Kukul & Gökçearslan, 2014; Fessakis et al., 2013, Clements & Gullo; 1984). In their study, Au and Leung (1992) found that programming increased problem-solving skills with children aged 8-12. Reed and Palumbo (1992) found similar results in studies conducted with older age groups.

Considering the contributions of computer science to the individual and society, it can be said that it is essential that all members of society. Forsythe (1967)

stated that computer science education should be included in the education given to everyone. When the literature is analyzed, it is seen that computer science education provides many thinking skills, especially computational thinking skills. For this reason, it is seen that countries add computer science courses to their curriculum.

COMPUTER SCIENCE EDUCATION IN THE WORLD

The importance of programming education in early childhood has been increasing significantly since 2010 (Balanskat & Engelhardt, 2015; Saygıner & Tüzün, 2017a). According to the report published by the European School Network in 2015, Austria, Bulgaria, Czech Republic, Denmark, Estonia, France, Hungary, Ireland, Israel, Lithuania, Malta, Spain, Poland, Portugal, Slovakia and the UK have integrated coding into their curriculum at national, regional or local level. Estonia, Israel, and Slovakia have integrated coding at all levels, while ten countries, including France and Spain, have integrated coding at the primary level (Balanskat & Engelhardt, 2015). The inclusion of computer science education in school curricula cannot be interpreted only as steps to train computer software developers or individuals who are experts in this field. Duncan and Bell (2015) state that computer science is integrated into the school curriculum in many countries to provide students with computational thinking skills.

When the UK's informatics course curriculum is examined, it is seen that it is generally structured based on computational thinking skills (Barut & Kuzu, 2017). Within the scope of the aims of the informatics course, which is compulsory for students aged 11-14, it is aimed to develop students' algorithmic thinking structures to enable them to produce solutions with programming software in practice. Thus, individuals are taught programming logic early (Barut & Kuzu, 2017). The Sun, a British newspaper, launched the "Let us learn coding" campaign with the support of the UK government, and the BBC published documents for coding learning following the new curriculum on the "https://www.bbc.com/bitesize" website in the UK (Demirer & Sak, 2016). In addition, 2014 was declared the "Year of Coding" in the UK, and the objectives related to software and algorithm logic were given a vast place in the informatics course curriculum, thus paving the way for raising a productive generation in this field by creating programming logic from an early age (Barut & Kuzu, 2017).

With the guidance and programming law no. 2015-595 enacted on 8 July 2013 to restructure schools in France, regulations for many innovations have been made as of September 2016. In this context, all students completing compulsory

education in France are expected to "know the basic principles of algorithms and coding" (French Government, 2015).

In Finland, the learning of programming and algorithmic thinking takes place in the context of mathematics. In Grades 1 and 2, students learn to give commands step by step, while in Grade 3, they use visual programming tools. In the last years of primary education (grades 7-9), they create algorithms and compare the usefulness of different algorithms, progressing gradually from simple tasks to more complex tasks (Finnish National Board of Education, 2016).

Singapore and Australia have also set clear policies and frameworks for computer programming in K12 schools (Bers, 2017). In Austria, the curriculum developed for secondary schools in Informatics takes problem-solving as its primary objective. Students are expected to understand the theoretical foundations and learn the working principles of machines and the basic principles of algorithms and programming (Sabitzer, Antonitsch, & Pasterk, 2014).

In Cyprus, computer programming and coding are part of the computer science curriculum. As of 2001-2003, algorithmic thinking and programming have started to be taught compulsorily to students aged 13-16, and in addition, learning in other subjects is supported by computing. However, it is not taught separately in the primary school curriculum (Webb et al., 2017).

The Introduction to Computer Science course in Israel's high school education emphasizes the fundamentals of algorithmic thinking (Bargury et al., 2012). While the curriculum consists of a series of compulsory and elective modules, the aim is not to train all students as programmers but to ensure that all students learn the basics of computer science. In recent years, programming teaching computer science has been launched for secondary schools (grades 7-9) and primary schools (grades 4-6) starting from the 2016-2017 school term (Armoni and Gal-Ezer, 2014).

In South Korea, it was announced by the Ministry of Science and Future Planning that coding will be taught progressively at both primary and high school levels (Özçakmak, 2014). Although only seven states have computer science curriculum standards in K12 schools in the USA, the code.org project has reached 182 million students at all levels from 198 countries (Bers, 2017). In addition, for the first time, this project was supported by a US president by writing code. For students aged eight and above, the online visual programming platform scratch.mit.edu hosts millions of projects (Scratch, 2017).

Looking at the coding studies carried out in the world, it is seen that many countries include computer science education in their curriculum or support

computer science education through various organizations. While some countries have recently renewed their curriculum and emphasized computer science education, it can be said that some countries have completed these studies before. In Türkiye, it is seen that the importance given to computer sciences has increased recently.

COMPUTER SCIENCE EDUCATION IN TÜRKİYE

In Türkiye, the Ministry of National Education renewed the Information Technologies and Software curriculum in 2017. Within the scope of this program, two weekly compulsory lessons are taught in 5th and 6th grades, and elective lessons are taught in 7th and 8th grades. The general objectives of the curriculum (MEB, 2017) include "Developing an understanding of algorithm design and expressing it verbally and visually, selecting and applying the appropriate programming approach to solve problems, building technical knowledge in programming, using at least one of the programming languages at a good level."

In secondary education, the curriculum, organized as two courses within the scope of the Computer Science Course, started to be taught in Fine Arts and Sports High Schools in 2016-2017 and other secondary education institutions in 2017-2018. Among the skills that the Computer Science Course Curriculum aims to provide students with (MEB, 2018) are computational thinking, critical thinking, algorithmic thinking, mathematical thinking, creative thinking, problem-solving, algorithm design, software development, and analytical thinking skills. Within the scope of the course, which is implemented for two hours a week, text-based programming is used at the level of Course 1, and for Course 2, teaching continues with any approach suitable for teaching the programming subject. Accordingly, Course 1 includes the units "Ethics, Security, Society, Problem Solving and Algorithms, Programming"; Course 2 includes "Robot Programming, Web Based Programming, Mobile Programming."

In Türkiye, Information Technologies and Software courses taught in secondary schools and Computer Science courses taught in high schools are taught by information technology teachers. In parallel with this change in the course contents in secondary and high schools, with the changes made in 2018 in the Computer Education and Instructional Technology Teaching Undergraduate Program in higher education, it is seen that "Algorithm Design and Development," "Electronic Circuit Elements," "Programming Teaching Approaches," "Physical Programming" and "Mobile Programming" courses are included as field education

courses in the program. The contents of these courses (YÖK, 2018) include block and text-based programming environments, electronic circuit elements, electronic circuit installation, sample applications for programming teaching, robot types, educational robots, and concepts related to mobile programming.

In Türkiye, coding is taught within the scope of the computer course in Science and Art Centers (BİLSEM), opened to enable gifted and talented students to develop their talents and increase their capacities outside their formal education. In these centers, students are trained in three stages: support, individual talent recognition, and exceptional talent development. In the support group, the first stage in the BİLSEM Computer Course Framework Program, basic algorithm and coding training are provided using code.org, Scratch, and Small Basic coding environments and tools, respectively. Object-oriented programming and Python programming language are taught in the program to recognize individual talents. The unique skills development program covers object-oriented programming, Python, and PHP programming languages. In addition, basic electronics and digital electronics topics are also included in the framework program (MEB, 2020).

In Türkiye, it is also seen that coding teaching projects are carried out in many provinces and districts such as KodlaManisa (Kodlamanisa, 2020, KodlaRize (Kodlarize, 2017), Keşf@ Kodlama Keşfediyorum Projesi (Keşfet, 2014), Düzce Kodluyor Projesi (MEB, 2023). Considering the studies conducted in Türkiye, it can be said that the importance given to computer science education has increased. In addition to gifted students, in line with the objectives added to the curriculum in secondary and high schools, problem-solving skills are supported, and it is aimed that students can use at least one programming language. In line with the renewed curriculum, changes were made in teacher training programs in higher education. In addition, coding workshops and opportunities provided within the scope of various projects provide an opportunity for students who want to further their learning in this field. Pedagogy in computer science education can be considered an issue that must be discussed. The following section presents approaches to computer science education in the world and in Türkiye.

PEDAGOGICAL APPROACHES TO COMPUTER SCIENCE EDUCATION

In computer science education, the need to employ learning sciences to provide students with computational thinking skills has been recognized (Grover & Pea, 2013). Much research has been conducted in line with this need, and many

approaches have been tried. However, it cannot be said that there is a single approach to effectively teaching computer science at the K-12 level (Goode, 2008). This section presents research on computer science education in the literature. The literature was searched through Web of Science, EBCHOST, and Google Scholar using the keywords "computer science," "pedagogy," "coding education," "computer science education," "coding teaching," "computer science," and "programming teaching." In this context, the studies in the literature are summarized in this section.

In a study by Sabitzer (2011), neuro-didactic methods were proposed in computer science education. Neuro-didactics, a relatively young discipline, represents the interaction between neuroscience and didactics. Based on brain research findings, it offers principles and recommendations for effective teaching and learning. Sabitzer summarizes the situations in which all students learn effectively as follows:

- When they experience,
- When their social relationship needs are fulfilled, and they are honored,
- When positive emotions accompany their learning,
- When they can utilize their innate capacities,
- When information is embedded in real-life situations,
- When their attention is concentrated,
- When they are given time to think about it,
- When they are presented with multiple experiences to remember,
- When developmental differences are considered,
- When the environment is supportive and challenging,
- When their talents and capacities come into play.

He also states that the brain will establish its own rules and pathways and that students will learn better in learning environments that carry parts of their lives and environments. He emphasizes that every new learning should be associated with old concepts and the importance of practice. According to Sabitzer (2011), when the principles of neuro-didactics are used in computer science education, more effective education can be provided.

Fokides (2017) investigated whether game programming supports learning computer science concepts. In this context, 138 students aged 10-11 were made to program games with Microsoft Kodu Game Lab for a school year. The students were divided into three groups and worked in pairs. The first group needed teacher support, while the second needed more support. In the third group, teacher-

directed activities were carried out. The data related to the research were obtained through questionnaires and analyses of the games. As a result of the research, it was found that the most used programming concepts in all three groups were conditions, variables, and loops, while logical expressions and function concepts were used less frequently. The most problematic concepts were logical expressions and loops. Conditions and variables were the most problematic concepts. It was observed that the errors decreased as the programming concepts used in the game increased. It was concluded that the students learned basic programming concepts, enjoyed the activity, and developed a positive attitude towards learning programming by developing games. The teaching method did not have any effect on the learning outcomes.

In a study conducted by John and Rani (2015) with undergraduate students, an innovative approach was demonstrated by using a smartphone environment. Firstly, a questionnaire was applied to reveal the devices used by the students. As a result of the survey, it was found that 70% of the students used smartphones, 20% used tablets, and 6% used regular phones. 4% of the students did not have any mobile device. It was observed that the students were familiar with smartphone and tablet technology. The rest of the study aimed to teach Java programming language by designing an application on the Android platform. When students were introduced to the concept of learning Java on a mobile platform, they paid more attention, and this attention increased, especially during the process of transferring their work to smartphones. During the implementation, the researchers used STPT (Share Time Pair Teaching) in addition to the paired mentor method. The study observed the positive contribution of application development to students' motivation. The researchers stated that this approach can support students' creativity and entrepreneurship.

According to Jenson and Droumeva (2016), game-based learning is essential for middle and high school students to learn computational thinking in computer science education. In their experimental design research, 60 6th-grade students designed games using Game Maker. Data were collected both qualitatively and quantitatively. When the results were compared, it was stated that game programming is helpful in computer science education.

In a study conducted at the K-12 level in Oman, programming was taught with Alice within the scope of a computer science course. Within the scope of the study conducted in experimental design, students were asked to fulfill the tasks given to Alice. In addition, their opinions about including Alice in their lessons were taken. It was observed that the students were willing to fulfill the given

tasks. Students found Alice's programming tool easy to use and understand. The only difficulty experienced by the students was that the program needed Arabic support. They were reluctant to add dialogue to some characters because of their difficulty understanding English. Nevertheless, most students expressed positive opinions about the Alice programming tool. The researchers stated that after this application, students' motivation toward computer science education increased more than expected (Hayat et al., 2017).

In a study by Abad (2008), a non-traditional method was applied that can be used in upper-level computer science courses such as distributed systems, operating systems, computer architecture, artificial intelligence, etc. Students were asked to design an interactive learning object that would help their classmates (and future students of the course) understand one of the topics covered in the course. Within the scope of the course, the undergraduate class of 34 students was divided into two, and the students were asked to choose from the specified topics. The learning objects, which their instructors reviewed, had to include an objective section, a pre-test, and a theoretical section, followed by a test or more interactive activities (such as code tutorials, simulations, or line-of-code exercises). Students took notes on the learning objects' content, presentation, and usability. Peer review is critical for the quality of the product. Therefore, students were asked to evaluate at least five learning objects, and the groups made necessary adjustments based on these evaluations. At the end of the course, students stated that more laboratory lessons, animations, and interactive activities could facilitate their understanding of the concepts. They also needed more in-class activities to understand the theoretical concepts. The students commented that they found the learning objects developed by their peers helpful, instructive, and fun. The groups developed the learning objects that received negative comments from their peers regarding competence and interaction. The results showed that the students well received the project and that designing learning objects and using the learning objects designed by their peers helped them better understand the content of the course.

Raab, Rasala, and Proulx (2000) designed a toolkit for a Java programming language course. They stated that standard Java components needed to be improved to create quality user interfaces in introductory courses and that the toolkit they designed would make it easier for beginner students to complete their projects.

Bashir and Hoque (2016) state that programming skills are one of the basic skills of computer science students. Developing an effective learning/teaching

pedagogy for programming languages is vital. Problem-based learning is an effective pedagogy that allows students to learn through self-directed practice. However, researchers say traditional problem-based learning is unsuitable for computer science education. In this study, traditional problem-based learning and e-learning were integrated. A problem bank was established to support teaching more than one programming language. In the system, which can be logged in as a student, teacher, or system user, teachers can design problems, see student performances, and access the evaluation module. On the other hand, the learning module allows students to select a problem, receive material support to solve the problem, participate in online courses, make self-assessments, or see the teacher's evaluation. The problem bank, which is the basis of the whole system, consists of application, interface, and database layers. Problems can be structured, semi-structured, or unstructured. The researchers stated that problem-based learning in an e-learning environment is student-centered. The system controls the learning environment, and the possibility of inefficient activities is very low.

Hansen, Iveland, Carlin, Harlow, and Franklin (2016), in their design-oriented research with 123 students aged between 9 and 12 studying with the same computer science curriculum and the same teacher, had students create digital stories with user-centered design by using a visual block-based programming tool. LaPlaya, a modified version of Scratch, was used as the programming tool. The application consists of two modules: Digital storytelling and game design. Each module contains three activities: activities of gradually increasing complexity involving a small task with a computer, activities without a computer where they can relate the concepts learned to everyday life, and engineering design lessons that emphasize the iterative nature of programming and teach the engineering design process. At the end of both modules of 12 hours each, students designed and programmed their designs. In the user-orientated design lessons, students performed three small tasks. In the first task, students were asked to find hidden code in completed programming. In the second task, students were introduced to the user interface concept. In the last task, students were asked to design a more accessible interface for a new user. As a result of the analyses made on the students' projects, it was seen that the students could make user-oriented designs, but they were more successful from the 5th grade onwards. Although this result is not surprising, this study provides the most appropriate age range for teaching user-centered design. The researchers suggested waiting until the 5th grade (10-11 years old) for this concept related to computer science.

In the literature, it has been observed that research on computer science

pedagogy in Türkiye is limited. Arabacıoğlu, Bülbül, and Filiz (2007) designed a "tutorial system" in Turkish to overcome the adverse effects of programming languages being in English on learning (Figure 1). The system's most important feature is that it provides the concepts in Turkish. The size of the tool, which can run in a Windows operating system environment, is relatively small. Visuality was given importance in the design. The study aimed to make the work done on paper enjoyable by moving it to the computer environment. It is argued that this application will increase success as it will reduce the mental load of the student.

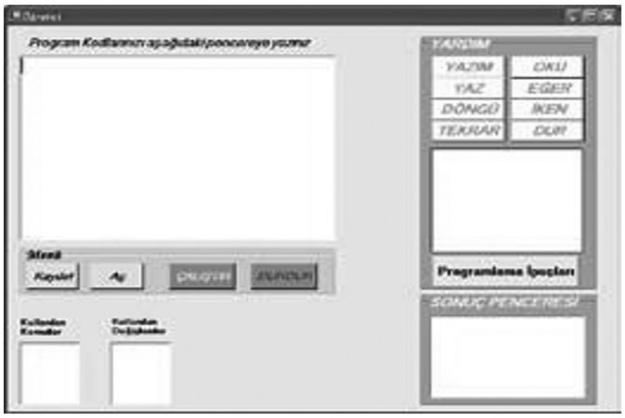

Figure 1. Turkish tutorial system (Arabacıoğlu et al., 2007)

Ersoy, Madran, and Gülbahar (2011) stated that theoretical information is given to the students through direct expression in computer science education. Then, it is tried to be taught in a way based on application and demonstration techniques, which may negatively affect student motivation. For this reason, they proposed a model using robot programming techniques to increase motivation and success. Robot programming keeps student interest high and enables the use of methods based on new learning theories, such as competition and teamwork. Arduino board was preferred for the research. As a result, it is stated that the difficulties caused by the student's continuous work with abstract concepts in programming teaching can be overcome with the robot programming model.

In the literature, it is seen that many approaches to computer science education are applied. In these studies, the game programming approach is more frequently used in computer science education (Fakides, 2012; Jenson & Droumeva, 2016). In addition, teaching with toolkits has also been observed (Raab et al., 2000; Arabacıoğlu et al., 2007). There are also studies in which students designed learning

objects (Abad, 2008), wrote digital stories (Hansen et al., 2016), and developed mobile applications (John & Rani, 2015). In addition, robot programming (Ersoy et al., 2011), problem-based learning in an e-learning environment (Bashir & Hoque, 2016), the use of the Alice programming tool (Hayat et al., 2017), and neuro-didactic methods (Sabitzer, 2011) are recommended approaches in computer science education. It is seen that methods such as design-oriented research, reflection, paired mentor, peer assessment, and STPT are used (Abad, 2008; John & Rani, 2015; Hansen et al., 2016), while methods such as lecture and demonstration are not recommended (Ersoy et al., 2011). Like the diversity in the applied approaches, it is seen that many tools can be used in computer science education. The following section presents tools that can be used in computer science education.

TOOLS FOR COMPUTER SCIENCE EDUCATION

Emphasizing the importance of computer science education has led to many projects. Projects such as Alice, Scratch, and code.org are frequently used in schools and are the subject of academic research. Armoni, Meerbaum-Salant, and Ben-Ari (2015) grouped the tools recommended in computer science education under kinesthetic activities, visual programming environments, and robotics.

Kinesthetic activities are activities performed without the use of a computer. It is also called computer science without a computer in the literature. The studies on computer science activities without computers emphasized that the activities should be more related to computer sciences. Gallenbacher (2012) showed Abenteur Informatic as a positive example of computer science activities without computers. Visual programming environments provide a simple environment and increase interest thanks to their symbolic interfaces. Scratch and Alice are examples of visual programming environments. Robotics activities are widely used to introduce science and technology to young students. Robots are suitable for introducing computer science, physics, and math to students. Algorithms and programming created with these tools are associated as virtual characters on the screen, as in Scratch and Alice, and with concrete objects.

Another classification of the tools used in computer science education was made by Weinberg (2013). Weinberg stated that computational thinking skills can be taught through computer science activities without computers, block-based programming tools, robot programming, and interdisciplinary applications. Kalelioğlu and Keskinkılıç (2017) added text-based environments to this

classification. In the next section, the tools used in computer science education will be discussed under computer science titles without computers, block-based tools, text-based tools, and robotic tools.

A. Computer Science without Computers

According to the constructivist theory, learning is an active process in which the student constructs knowledge by combining it with previous learning. When constructivist theory is applied to computer science education, the active, subjective, and constructivist learning characteristics are emphasized, and the student is placed at the center of the learning process (Ben-Ari, 1998). Computer science without computers also appears as mobile activities in which students actively participate. With this feature, it is based on constructivist theory. Computer science without computers includes activities in which computer science concepts and computational thinking skills can be taught to students independently of computers and programming languages. The activities are based on game-based learning and learning by discovery (Kalelioğlu & Keskinkılıç, 2017). Computer-less computer science activities differ from others in enabling students to focus on concepts rather than the tool used. Computer-less computer science projects are implemented in many countries, and the reasons for the worldwide interest in these activities are the increasing interest of students, the desire to apply new teaching methods, and a method that can be applied in case of limited access to computers (Bell, Alexander, Freeman, & Grimley, 2009). A study conducted with primary and secondary school students observed that these activities increased interest and motivation (Nishida, Idosaka, Hofuku, Kanemune, & Kuno, 2009). Computer science activities without computers first appeared in New Zealand with the CS Unplugged project (Bell et al., 2009; Nishida et al., 2009). Some of the other noteworthy projects that aim to teach computer science without using computers are code.org unplugged, Keşf@ Kodlamaya Keşfediyorum, and Bilge Kunduz (Kalelioğlu & Keskinkılıç, 2017).

1. CS Unplugged

This project, which includes many activities developed by Bell et al. (2009) under CS Unplugged, originated at Canterbury University. The resources provided free of charge include many materials and puzzles. In one of these activities, the orientation and locking activity, each student wears a different colored t-shirt. There are nine fruits in five different colors, and one of the students has one empty hand. The students change the fruits in their hands in a circular motion until the

color of the T-shirt matches the color of the fruit in the student's hand. The rule here is that the neighbor can give some fruit to the student with an empty hand. To complete the activity, students sometimes must give the fruit even if the color of the fruit in their hands is correct. Otherwise they experience a deadlock. At the end of the activity, a discussion on buffer memory and deadlock can be initiated.

2. Code.org Unplugged

The code.org project, implemented in 2013 to provide computer science education to everyone with block-based activities, also includes computer science activities without computers. One of these activities, "Binary Images", involves students creating pictures with a binary coding system. Students are divided into groups and try to find the hidden pictures on the cards created according to binary coding. In this activity, 0s are colored black, and 1s are left in white (Figure 2).

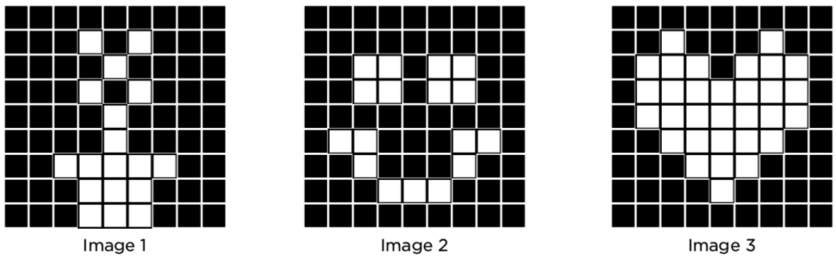

Figure 2. Binary pictures event - Code.org (2018)
(https://code.org/curriculum/course4/17/Activity17-BinaryImages.pdf)

3. Keşf@ I Discover Coding

I Discover Coding project consists of many computerized and non-computerized coding activities that aim to provide 5th grade students with computational thinking skills. Some of the project objectives are to enable students to recognize the basic concepts and approaches in problem-solving, analyze problems, determine and solve appropriate steps, use operators, expressions, and equations in problem-solving, recognize the concept of algorithms and flowcharts. All materials and plans that may be needed for a complete teaching are available on the website. In "Game Analysis," one of the activities provided, games such as football, table tennis, hide and seek are analyzed and the constants and variables of the games are revealed. For example, in the game of hide and seek, "the number of midgets" and "the time given for hiding" are constants, while "the number of hiding places"

and "the time taken for the end of the game" are variables. With these analyses, it is aimed to teach students the concepts of "constant" and "variable" in computer science (Keşf@ Project, 2014).

4. Bilge Kunduz

The activity, which aims to teach computational thinking to students of all ages, was first organized in Lithuania in 2004. In 2012, it reached more than 500,000 students. More than 25 countries have participated in the event and many countries are planning to participate. The activity consists of tasks that students can solve even without any prior knowledge. High-level thinking skills such as calculation, analytical thinking, and problem solving should be used to solve these tasks (Bilge Kunduz, 2018). It is seen that the questions are asked from areas such as algorithms, programming, data processing, problem solving, digital literacy (Kalelioğlu and Keskinkılıç, 2017).

Figure 3. Displacement activity - Bilge Kunduz (2018) (http://www..org)

In this section, activities for computer science without computers are presented. Computer-less computer science activities seem to be useful in computer science education, especially in cases where computer labs are inadequate. However, Taub, Ben-Ari, and Armoni (2009) offer some suggestions for these activities to be effective:

- While creating the activities, students' previous learning and their views on computer science should be taken into consideration and care should be taken

to build on these learnings.

- Activities should be linked to the main concepts of computer science.
- Although not directly related to the activities, students should be told about career opportunities in computer science.
- In another study, the principles that should be taken into consideration when developing computer science activities without computers were stated as follows:
- It should focus on computer science concepts,
- Group work should be emphasized,
- It should be designed to be fun,
- Costs must be kept low,
- It should be as interesting for female students as it is for male students,
- They should be enabled to play games through concepts,
- Debugging should be included in teaching as a part of the activity (Bell at al, 2009).

B. Block-based Tools

Computer science is one of the courses in which students have the most difficulty. Gomes and Mendes (2007) pointed out that the reason for this is the shortcomings and mistakes in teachers' teaching methods and the fact that teachers try to teach the syntax of a programming language instead of focusing on developing students' algorithmic thinking skills. In addition, the abstract nature of programming and complex syntax can reduce students' motivation. The use of block-based visual environments is recommended to overcome these difficulties (Saygıner & Tüzün, 2017b).

Block-based coding tools are designed to make it easier for the student to understand the basic concepts of programming by preventing the student from getting lost in complex lines of code. Thanks to block-based coding tools, the student does not make syntax errors because he/she uses the drag and drop method. By focusing only on the design, they can develop their computational thinking skills in a fun way. Block-based coding tools appear as desktop, mobile or web-based applications. In this section, some of the block-based coding tools will be introduced.

1.Scratch

The Scratch programming tool was developed at the Massachusetts Institute of Technology (MIT) Media Laboratory in 2003. Designed especially for the 8-16

age group, Scratch has a very widespread use. The code blocks created by drag and drop method enable the use of concepts such as loop, variable, condition statements related to algorithms and programming in computer science. It is possible to create both animations and interactive software with Scratch. A different version of Scratch Jr, which is available for preschool students, can be used in mobile environment. It is possible to programming the Arduino board with the Scratch 4 Arduino version of Scratch. The installation of the Scratch coding tool is quite simple. It can also be used via "scratch.mit.edu" address. This site, where students can both programming and share, provides a social learning environment for students. Users can see and comment on the applications and codes of other users through the site. They can create their own projects without any installation on their computers. They can indicate that they like other projects by pressing the "I like this project" or "This project is one of my favorites" buttons. If they wish, they can make changes on other users' projects and publish them.

When we look at the literature, it is possible to come across many studies using Scratch coding tool. In one study, the transition process from Scratch to a professional text-based programming language (C# or Java) was examined. It was observed that students with Scratch learning experiences needed less time to learn new topics and reached higher cognitive levels in understanding many concepts. As a result of this application, enrolment in computer science classes increased. This study revealed that providing computer science education in secondary schools with visual programming tools increased students' self-confidence and motivation (Armoni et al., 2015). Scratch is found interesting by students and useful by teachers (Vieira and Magana, 2013). In another study conducted with pre-service teachers, Scratch application was found to be positive in terms of motivation, usefulness, and ease of use by pre-service teachers (Yükseltürk & Altıok, 2016). A different approach is seen in the activities of creating and playing music using Scratch. In a study conducted with 109 6th grade students, positive results were obtained in teaching computer concepts to students (Lopez & Gutierrez, 2017).

Scratch offers an environment where students can both reveal their creativity and develop their computational thinking skills without dealing with the complex structure of programming languages, thanks to its block structure carried by drag and drop method. Studies in the literature reveal that Scratch application is an effective, interesting, and motivational tool that can be used while teaching

computer science concepts.

2. mBlock

It is a software developed especially for STEM activities. Inspired by the Scratch 3.0 application, it offers both visual programming and text-based programming. It allows students to design games and animations as well as programming Makeblock and similar robots. It supports Python language. It allows the codes written to be shared online (mBlock, 2018). Although there is no study in the literature on mBlock coding tool, its similar features with Scratch suggest that it will have positive effects in computer science education.

3. Alice

Alice is a 3D, interactive, visual programming tool. Users can learn the concepts of object-oriented programming while designing 3D animations and games. Alice offers students the opportunity to make designs without falling into syntax errors by drag and drop method. The programming and tutorial documents are available as open source on the Alice website. It offers a creative and engaging way to teach computer science concepts (Ward, Marghitu, Bell, & Lambert, 2010). It encourages students to learn through discovery. It is a suitable tool for introduction to object-oriented programming (Alice, 2017). It is possible to make 3D designs, animations, and games using Alice. It can be used to teach students logical and computational thinking skills by teaching the basic principles of programming. As a matter of fact, in a study in which 325 secondary school students programmed games with Alice for one semester, it was revealed that students grasped high-level computer science concepts (Werner, Campe, & Denner; 2012). Alice can be seen as a suitable tool to teach computer science concepts to students.

4. AppInventor

App Inventor, a mobile application development platform for Android-based devices, was developed at Google Labs by a team led by Hal Abelson. It enables application development using visual blocks and allows beginners or non-programmers to design applications that can be used in real life and install them on mobile devices or publish them on Google Play (Figure 4). App Inventor can be used at all levels, from K-12 to undergraduate level. Sample curriculum, sample applications, and news can also be found on the MIT App Inventor website (Wolber, Abelson, & Friedman, 2015).

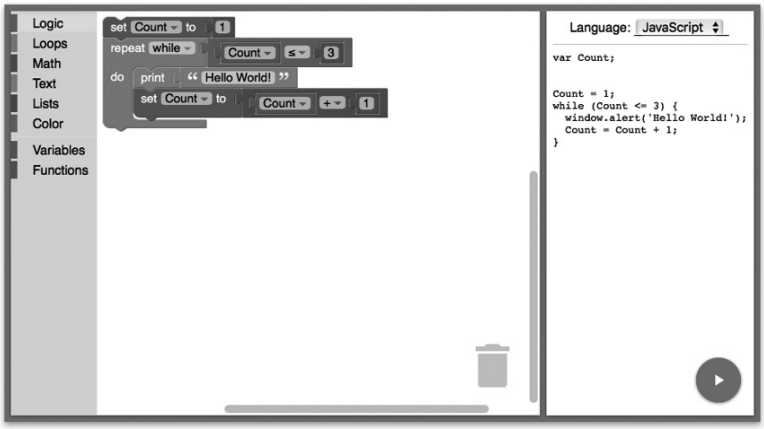

Figure 4. App Inventor Coding Environment - Massachusetts Institute of Technology (2017) (http://appinventor.mit.edu/explore/front.html)

App Inventor is a powerful and highly accessible environment that can be used in introductory computer science courses at both K-12 and high school levels (Morelli, De Lanerolle, Lake, Limardo, Tamotsu, & Uche, 2011). Considering that mobile device usage rates are increasing every year (Güler, Şahinkayası, & Şahinkayası, 2017), it can be said that students' interest and motivation towards computer science will increase if they can develop applications for mobile devices that are so much a part of our lives. In addition, the fact that the application can be easily transferred to the mobile device, or the result can be displayed instantly with the help of its own simulator facilitates the debugging and design process. With these features, it can be seen as a tool that can be used in computer science education. In a study conducted by Abuah, Schilder, Sherman, and Martin (2018) with 44 students, it was revealed that App Inventor improves programming skills.

5. Code Org

Since its establishment in 2013, it has been offering students the opportunity to learn coding online. As a block-based programming platform, code.org includes coding activities that progress from simple to complex with 34 different language support. Feedback is given to the student when each activity is completed. The student can easily follow his/her progress through his/her account (Figure 5). The teacher, on the other hand, can both benefit from the lesson plans offered and closely follow the progress of the students enrolled in his/her class. It guides the student throughout all levels and provides motivational and informative videos featuring famous people such as Mark Zuckerberg and Bill Gates to

attract students' interest in computer science. It contains educational materials prepared for teaching coding to illiterate students. To increase the motivation of the students, the characters of popular games were used in the activities. When a complete course is completed, it rewards the student with a certificate.

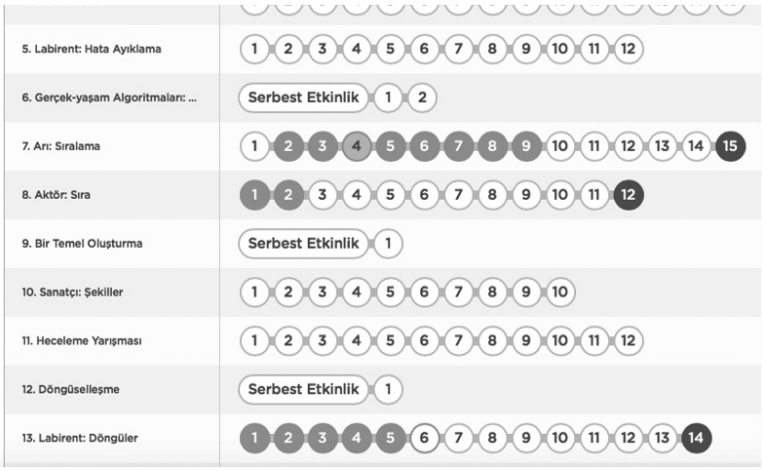

Figure 5. Code.org completed activities table - Code.org (2017) (https://code.org/)

The website, which has been created for students aged 6-14, provides more than 20 million students with not only algorithm knowledge but also other information such as how computers and the internet work, how to solve an error in a programming (Code.org, 2017).

It can be said that block-based coding tools are useful and common tools in computer science education, especially at the K12 level. Although they are effective in teaching concepts, some problems that students may experience when they switch to text-based coding tools (Levis, 2010; Weintrop & Wilensky, 2015; Meerbaum-Salant, Armoni, & Ben-Ari, 2011) draw attention to text-based coding tools that can be used for educational purposes that offer students real code writing experience instead of drag and drop method. In the following section, some of the text-based coding tools used for educational purposes will be introduced.

C. Text Based Coding Tools

Although block-based coding tools are widely used for teaching programming at an early age, they have a significantly different structure from programming languages. Therefore, students experience difficulties in the transition from

block-based environments to programming languages (Levis, 2010; Weintrop & Wilensky, 2015; Meerbaum-Salant, Armoni, & Ben-Ari, 2011). It has been reported that students do not pay attention to syntax rules and have difficulty in error detection when they switch to text-based programming languages (Kandemir, 2017). For this reason, it can be said that it would be useful to use tools that both offer students the experience of writing real code and motivate them to learn. In this section, some of the coding environments that serve this purpose will be introduced.

1. HackerCan

HackerCan, which aims to teach coding at an early age and is a 100% Turkish and local platform, is built on a game-based learning model. With the education it provides for students aged 6-9 and over 9, it is aimed for students to easily adapt to the language they want. It also offers special panels, teaching plans, presentations, and all necessary academic material for instructors. The HackerCan platform offers students the experience of writing real code, which is missing in software that teaches algorithm development to primary and secondary school students and provides the basics of programming knowledge with a block-based learning model (Figure 6). The student completes the sections by writing real lines of code either in Turkish or in English. The fact that all lines of code are 100% Turkish is important for the student to gain the ability and self-confidence of programming in the native language.

Figure 6. HackerCan coding environment - Hacker Can (2023) (http://www.hackercan.com/tr/)

By using gamification elements, students are asked to use the features of various characters in different scenes and complete the tasks by writing code. It is aimed to keep the student's attention at the highest level with both the graphic quality and the animation and sound effects used, and awards and reinforcements are given in case of successful completion of the section (HackerCan, 2023).

2. Turkish Programming Language

It is a platform that can be coded over the web with commands close to the spoken language, free from unnecessary details and whose outputs can be observed. It provides a coding guide that gives basic programming facts. While the execution of the codes written on the coding page can be observed step by step, the values of the variables can be read from the "Variable Values" window. "Console" is the section where the output of the program is given (Figure 7). The syntax of the language was originally produced by being inspired by the Java/C++ language family and Python language (Akkuş, 2023).

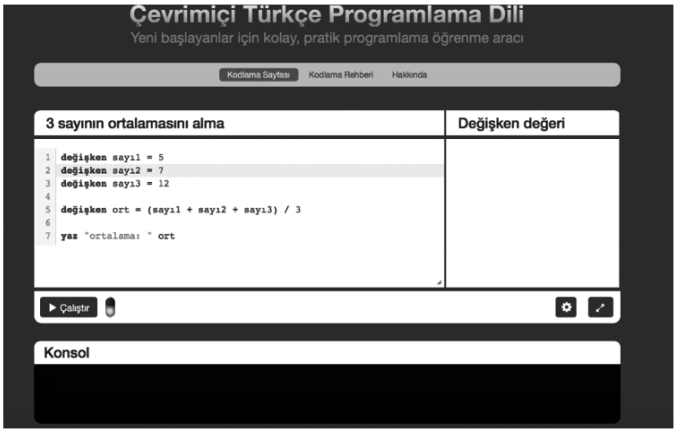

Figure 7. Online Turkish Programming Language (Akkuş, 2023)
(https://mehmet-akif-akkus.gitbook.io/turkce-programlama-diline-giris/

3. Code Combat

Codecombat.com, which offers the opportunity to learn coding with one of the Python or JavaScript languages, makes it possible to create a class from the teacher account and follow the student (Figure 8). While offering a real game experience, it provides students with the necessary gains for programming skills. Students can track their progress and the amount of code they write in real time. It was observed that the quality of the visual elements was kept quite high, and the teaching

material was supported with animations and sounds. Students can choose and customize their own game character. Each character has its own characteristics. Clear goals are given to the student with the storytelling technique. It offers rich feedback and help support (Figure 9). As the level is passed, the student is rewarded with items such as experience points, diamonds, and sword belts. The stages passed and future goals can be observed by the student through a complete map provided by the programming environment. It also provides teachers with educational resources and lesson guides (Code Combat, 2023).

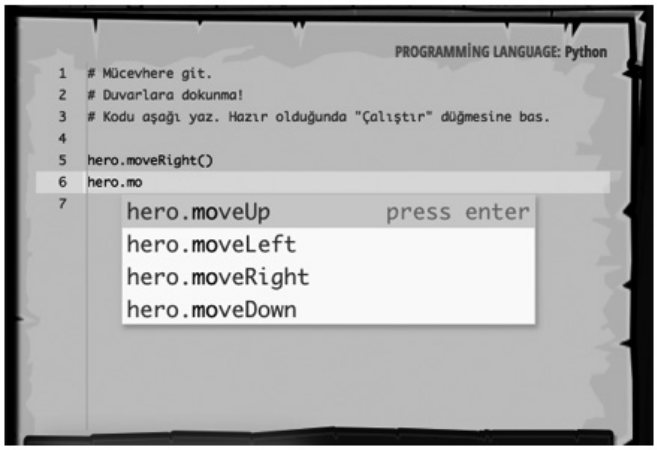

Figure 8. CodeCombat coding environment - Code Combat (2023) (https://codecombat.com/)

Figure 9. CodeCombat hero and programming language selection - Code Combat (2023) (https://codecombat.com/)

4. Code HS

Students can log in to the system either by entering the class code created by the teacher or by using their Google account. After creating an account, the teacher can open a class from one of the sections such as "Introduction to Computer Science, Introduction to Programming with Dog Karel, Introduction to Computer Science with Python, Web Design". Information about the contents and objectives of the sections are given under the section headings. After the class is created, students can register to this class with the code given by the system. While student registrations and student accounts can be controlled from the teacher account, it can also be determined which courses the student can access. When the student course screen was examined, it was determined that there were opportunities to run the written code, receive feedback about the written code, suggestions about similar applications to reinforce learning, suggestions about codes that can be used additionally, and the ability to send a message to the teacher from the help menu (Figure 10).

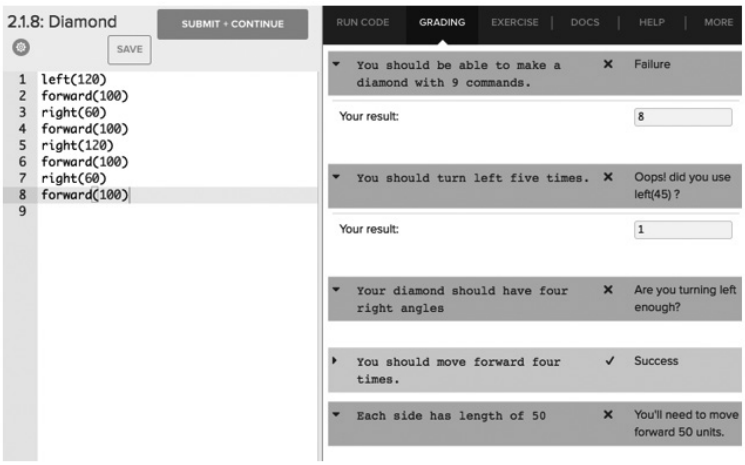

Figure 10. CodeHS course screen - CodeHS (2023) (https://codehs.com/)

The programming platform supported by video presentations does not have Turkish language support (CodeHS, 2023). For this reason, it is thought that the use of higher education level students will be more appropriate.

5. CodeMonkey

Animation and graphic quality are designed to maximize student motivation. In the training material, which has a very simple narrative, it can be said that

the objectives are very clearly stated, and the student is very well guided. Instant feedback is given, and the correct answer of the student is rewarded with stars. The feedback also includes animation and sound effects (Figure 11). Although it does not have Turkish support, it can be said that it can be used at primary level under teacher guidance. In addition, it is also possible for the student to write code by clicking buttons without using the keyboard (Code Monkey, 2023).

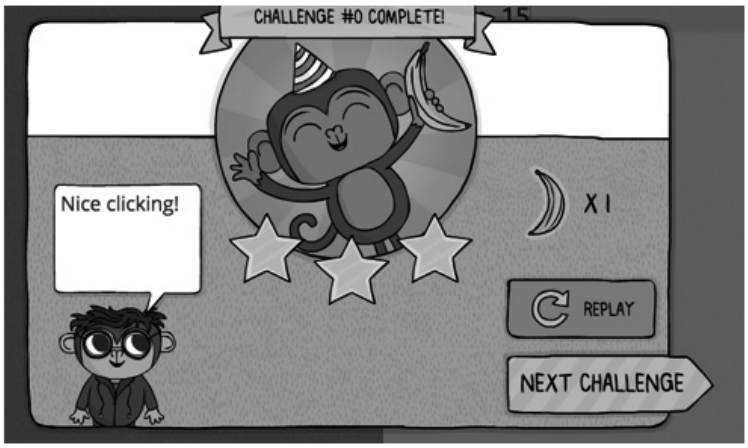

Figure 11. CodeMonkey feedback screen -Code Monkey (2023) (https://www.codemonkey.com/)

6. Kodris

In Kodris, two different types of accounts can be opened as teacher and student accounts. While it is possible to follow students, access lesson plans and create classes from the teacher account; while creating a student account, the student becomes a member by choosing one of the primary, secondary, and high school levels. It lists the schools affiliated to the Ministry of National Education during member registration and allows the student to complete his membership by selecting the name of the school he attends. Especially for students aged 8-16, it aims to provide students with the skills of creating algorithms and coding the created algorithm in Python language, computational thinking, and problem-solving skills. It is an e-teaching environment that offers both real and block coding opportunities suitable for the Turkish Education System, regardless of the platform used with topics ranging from easy to difficult, from simple to complex. In addition, teachers and students can design and share their own scenes. Teachers can also make online exams and access detailed information about students' task

solutions from the instructor panel. A feature that distinguishes Kodris application from other coding learning environments is that it offers live support with a chat plugin (Kodris, 2023).

In this section, tools that both save students from the boring and difficult environment of programming languages and offer real code writing experience are introduced. Robotic coding tools are tools that students can experience computer science education in a physical environment. Some of these tools will be introduced in the following section.

D. Robotic Coding Tools

Today, educational robots are used in the teaching of many course subjects and robot sets that learners can program practically are called educational robots (Üçgül, 2017). The idea of using educational robots in computer science teaching is not a new idea. When the literature is analyzed, "Logo Turtle", which was first introduced by Papert (1980) based on the studies of Piaget and Montessori, comes to the fore. After Logo Turtle, many different educational robots were produced and used. Educational robots such as Lego Mindstorms, Arduino and Makeblock sets, which are widely used in Türkiye, can be given as examples.

There are many studies on educational robots in the literature. Catlin (2010), in his study presenting many ways of how educational robots can be used in education, argued that activities designed with robots help students learn more, gain a deeper understanding, learn faster, remember longer and enjoy the experience. Benitti (2012) analyzed 197 studies in the literature and concluded that educational robots have a tremendous potential as a learning tool. In the light of these studies, it can be said that educational robots enrich learning environments, support the learning process from many different angles, and therefore, their use in computer science education will be very useful.

When it is decided to use many types of educational robots in computer science education, educators often ask the question "Which one should we choose?". To give an idea to educators in the selection of educational robots, it is necessary to evaluate these tools with their positive and negative aspects. In this section, Lego Mindstorms EV3, Arduino, Vex IQ, Makey Makey and Makeblock, which are widely used in our country for educational purposes, will be introduced.

1. Lego Education

According to Papert (1980), robots are the best tools for applying constructivist theory. The LOGO programming language developed by Papert has been used by

tens of millions of children around the world, and in the mid-1980s, because of the work carried out with the LEGO toy company, it was put forward as a tool where children could design and programming their own robots (Üçgül, 2017). Lego Mindstorms EV3 (Figure 12) is presented as a set with gear wheels and other parts that can be mounted to each other, as well as sensors, motors and programmable bricks that can measure light level, distance, rotation angle, temperature, etc. It is programmed with Lego Mindstorms Education or Lego Mindstorms Home Edition programming tools. The programming structure consists of interlocking and connecting blocks. The values read from the sensors can be displayed in the programming. The rotation angles of the motors or the number of turns can also be determined. Impellers and other parts allow the student to make many different designs. Thus, successful designs can be created in tasks such as transport and placement, which serve many different purposes. In addition, the Lego We Do set for primary school students can be considered as a set that can be used in computer science education at an early age with its simpler structure and parts compatible with Lego play sets (Lego Education, 2018).

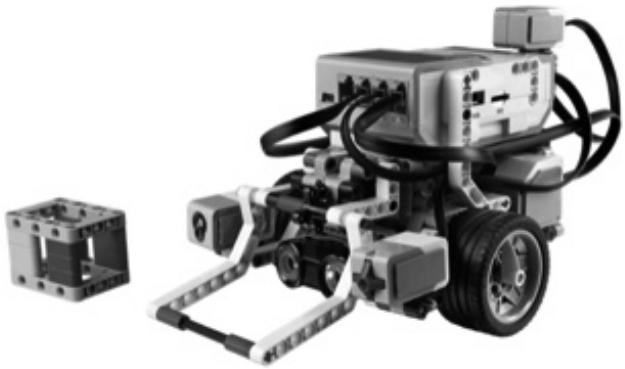

Figure 12. Lego Mindstorms Education EV3 - Lego Education (2018) (https://education.lego.com/en-us/middle-school/intro)

2. Arduino

Arduino is an easy-to-use and open-source electronic platform that includes both hardware and software. Arduino, the first open-source hardware project of this scale, is an application that can be easily used by anyone with basic electronics knowledge. By using the input pins of Arduino boards, data can be read with the help of sensors and any output unit such as motor, led can be controlled with the

output pins. Arduino, which allows designing many projects such as autonomous vehicles, smart home systems, toys, or smart agricultural applications, also increases the number of projects that can be put forward with the variety of sensors that can be used. Arduino is connected to the computer via USB cable and can be programmed either with its own programming tool or with Scratch for Arduino or mBlock program. There are many types of Arduino boards produced for different purposes, such as Arduino UNO (Figure 13), which is very useful for educational purposes and simple projects, Arduino Liliput, which can be used for wearable systems, or Arduino Mega, which has more input-output pins for larger projects. It can also be said that Arduino is a suitable tool for STEM studies (Arduino, 2018).

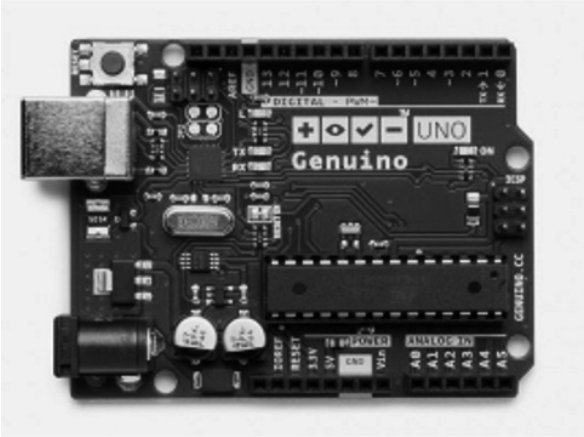

Figure 13. Arduino Genuino UNO - Arduino (2018) (https://www.arduino.cc)

In addition to its many advantages, the fact that it requires basic electronic knowledge and the need to use soldering or breadboard when making connections can be seen as difficulties that may be encountered when used in computer science education.

3. Vex Robotics Kits

The robot kits offered by VEX Robotics, which designs robot kits for international robot competitions (Figure 14), are sold as Vex IQ for middle school level and Vex EDR for high school level. Vex PRO is recommended for teams who want to participate in competitions. Vex EDR allows data to be received from many sensors with 12 input ports. The firmware is open source. Its own programming

tool can be used or ROBOTC or easyC languages can be used for programming. Free seminars and training materials are offered on its website (Takacs, Eigner, Kovács, Rudas, & Haidegger, 2016). With Vex Robotics kits, STEM education is emphasized, and these kits allow for many different designs with their attachable and detachable parts. Its own programming tool has a block-based structure. Therefore, it can be said that there are educational robots that students using Scratch-like programming tools can easily switch and programming.

Figure 14. VEX IQ - Vex Robotics (2018)
(https://www.vexrobotics.com/vexiq)

4. Makeblock

It is an open-source platform. Most of its products are presented with aluminium mechanical parts and electronic modules (Takacs et al., 2016). mBot, which can be supported with many sensors and parts, is an educational robot that can attract students' attention with its mobile structure. mBot Ultimate comes with more parts, sensors, and motors. Aluminum beams can be assembled in different ways using machine screws and evenly spaced holes and grids (Figure 15). In addition, various sensors, DC, and servo motors can be mounted on the design (Vandevelde, Saldien, Ciocci, Vanderborght, 2013). It can be easily programmed with the block-based mBlock programming. For teaching materials and support, they offer services through their website.

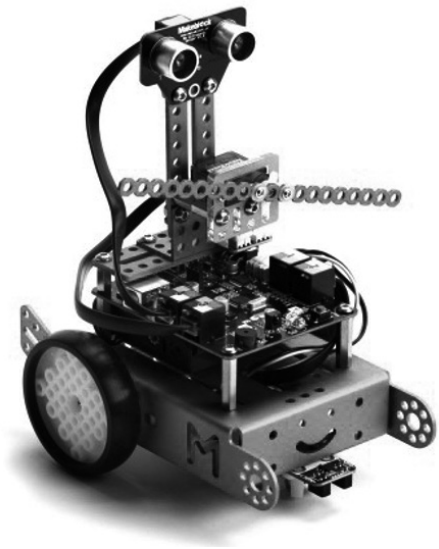

Figure 15. mBot - Makeblock (2018)
https://www.makeblock.com

5. Makey Makey

It is a project initiated by two students at the MIT Media Lab as both an academic and artistic project (Makey Makey, 2018). Makey Makey can transmit commands that can be given to the computer with a keyboard and mouse without the need for any programming. Therefore, it allows anything to be used as a keyboard or mouse. In other words, the object connected to the Makey Makey set (Figure 16) with a cable act as a mouse or keyboard when the circuit is completed. Thus, a physical user interface is designed without requiring any technical skills.

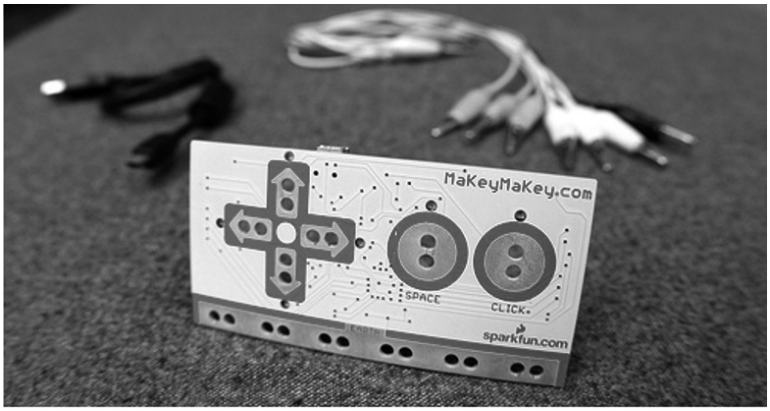

Figure 16. Makey Makey set - Makey Makey LLC (2018) (https://makeymakey.com)

Although there are not many studies on its effects on learning in the literature (García-Peñalvo et al., 2016), it is recommended to be used especially in young age groups with its easy use. Examples of activities that can be done using this set include making a joystick or playing the piano with different objects. Another activity that can be designed with this set is interactive maps. By using a coding tool like Scratch, the student can develop a software that can provide information on the computer when he/she touches the shapes he/she has prepared for different courses. In this way, geographical maps, organs, or any area where information is desired can be presented interactively. It is a kit that can be used to create a fun learning environment blended with artistic activities (Rogers et al., 2014).

REFERENCES

Abad, C. L. (2008). Learning through creating learning objects: Experiences with a class project in a distributed systems course. *ACM SIGCSE Bulletin*, *40*(3), 255-259.

Abuah, C., Schilder, D., Sherman, M. & Martin, F. (2018). The tablet game: an embedded assessment for measuring students' programming skill in app inventor, *Journal of Computing Sciences in Colleges*, 33(6): 9-21.

Akkuş, M. A. (2023). Türkçe programlama diline giriş. https://mehmet-akif-akkus.gitbook.io/turkce-programlama-diline-giris/ (Date: 2023, July)

Alice (2017). https://www.alice.org (Date: 2018, May).

Arabacıoğlu, C., Bülbül, H. & Filiz, A. (2007, Şubat). Bilgisayar programlama öğretiminde yeni bir yaklaşım. *Akademik Bilişim 2007 Konferansı*, Dumlupınar Üniversitesi, Kütahya.

Retrieved from http://ab.org.tr/ab07/kitap/arabacioglu_bulbul_AB07.pdf.

Ardunio (2018). https://www.arduino.cc (Date: 2018 May)

Armoni, M. & Gal-Ezer, J. (2014). High school computer science education paves the way for higher education: The Israeli case. *Computer Science Education*, *24*(2-3), 101-122. doi:10.1080/08993408.2014.936655

Armoni, M., Meerbaum-Salant, O. & Ben-Ari, M. (2015). From scratch to "real" programming. *ACM Transactions on Computing Education*, *TOCE*, *14*(4), 25. doi:10.1145/2677087

Au, W. K. & Leung, J. P. (1992). Problem solving, instructional methods and Logo programming. *Journal of Educational Computing Research*, *7*(4), 455–467.

Balanskat, A. & Engelhardt, K. (2015). *Computing our future: Computer programming and coding-priorities, school curricula and initiatives across Europe*. Belgium: European Schoolnet.

Bargury, I. Z., Muller, O., Haberman, B., Zohar, D., Cohen, A., Levy, D. & Hotoveli, R. (2012, October). Implementing a new computer science curriculum for middle school in Israel. In Frontiers in Education Conference, FIE, 2012, IEEE, 1-6.

Barut, E. & Kuzu, A. (2017). Türkiye ve İngiltere bilişim teknolojileri öğretim programlarının amaç, kazanım, etkinlik, ölçme ve değerlendirme süreçleri açısından karşılaştırılması. *Trakya Üniversitesi Eğitim Fakültesi Dergisi*, 7 (2) , 721-745. doi: 10.24315/trkefd.303156

Bashir, G. M. M. & Hoque, A. S. M. L. (2016). An effective learning and teaching model for programming languages. *Journal of Computers in Education*, *3*(4), 413-437. doi:10.1007/s40692-016-0073-2

Bell, T., Alexander, J., Freeman, I. & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *New Zealand Journal of Applied Computing and Information Technology*, *13*(1), 20–29.

Ben-Ari, M. (1998, 26 February- 1 March). Constructivism in computer science education. *Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education*. Association for Computing Machinery, New York, NY, USA.

Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education*, 58(3): 978-988.

Bers, M. U. (2017). *Coding as a playground: Programming and computational thinking in the early childhood classroom*. Oxon: Routledge. doi:10.4324/9781315398945

Bilge Kunduz (2018). http://www.bilgekunduz.org (Date: 2018, June).

Booth, S. (2001). Learning computer science and engineering in context. *Computer Science Education*, 11(3): 169-188.

Catlin, D. & Blamires, M. (2010). The principles of educational robotic applications (ERA): a framework for understanding and developing educational robots and their activities. in: Clayson, J. and Kalaš , I. (ed.) Constructionism 2010: Constructionist Approaches to Creative Learning, Thinking and Education: Lessons for the 21st Century: Proceedings for Constructionism 2010: The 12th EuroLogo Conference, 16-20 August, 2010 Paris, France Paris The 12th EuroLogo conference.

Chao, P. Y. (2016). Exploring students' computational practice, design & performance of problem-solving through a visual programming environment. *Computers & Education*, (95), 202-215. doi:10.1016/j.compedu.2016.01.010

Clements, D. H. & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology*, *76*(6), 1051-1058.

CodeCombat (2023, June). https://codecombat.com

Codehs (2023, June). https://codehs.com

Codemonkey (2023, June). https://www.codemonkey.com

Codeorg (2017, May). https://code.org

Çetin., İ. &Berigel, M. (2017). Bilgisayar bilimi eğitiminde kavram ve kuramlar, Y. Gülbahar (Ed.), *Bilgi işlemsel düşünmeden programlamaya in* (ss. 102-131). Ankara: Pegem Akademi.

Demirer, V. & Sak, N. (2016). Programming education and new approaches around the world and in Turkey. *Eğitimde Kuram ve Uygulama*, 12(3), 521-546.
Retrieved from https://dergipark.org.tr/tr/pub/eku/issue/26697/280853

Duncan, C. & Bell, T. (2015). A pilot computer science and programming course for primary school students. *In Proceedings of the Workshop in Primary and Secondary Computing Education*, ACM, 39-48.

Ersoy, H., Madran, R. O. & Gülbahar, Y. (2011, Şubat). *Programlama dilleri öğretimine bir model önerisi: Robot programlama.* XIII. Akademik Bilişim Konferansı, Malatya, Türkiye.
Retrieved from https://ab.org.tr/ab11/kitap/ersoy_madran_AB11.pdf.

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

Finnish National Board of Education (2016). *Curriculum reform 2016*, http://www.oph.fi/english/education_development/current_reforms/curriculum_reform_ 2016 (Date: 05.09.2017).

Fokides, E. (2017). Students learning to program by developing games. Results of a year-long project in primary school settings. *Journal of Information Technology Education: Research,* (16), 475-505. doi:10.28945/3893

Forsythe, G. E. (1967). A university's educational program in computer science. *Communications of the ACM*, *10*(1), 3-11.

French Government (2015). *Socle commun de connaissances, de compétences et de culture*, http://www.education.gouv.fr/pid25535/bulletin_officiel.html?cid_bo=87834, (Date: 05.09.2017).

García-Peñalvo, F. J., Reimann, D., Tuul, M., Rees, A., & Jormanainen, I. (2016). *An overview of the most relevant literature on coding & computational thinking with emphasis on the relevant issues for teachers*. Belgium: TACCLE3 Consortium. doi:10.5281/zenodo.165123

Gallenbacher, J. (2012). Abenteuer Informatik. Hands-on exhibits for learning about computational thinking. Paper Presented at WiPCSE 2012, Germany, 149- 150. https://doi.org/10.1145/2481449.2481487

Gomes, A. & Mendes, A. J. (2007, September). Learning to program-difficulties and solutions. In International Conference on Engineering Education–ICEE, Vol. 2007.

Goode, J. (2008). Increasing diversity in K-12 computer science: Strategies from the field. *In ACM SIGCSE Bulletin*, 40(1), ACM, 362-366.

Grover, S. & Pea, R. (2013, March). Using a discourse-intensive pedagogy and android's app inventor for introducing computational concepts to middle school students. *In Proceeding of the 44th ACM technical symposium on Computer science education*, ACM, 723-728.

Gülbahar, Y. (2017). Bilgi işlemsel düşünme ve programlama konusunda değişim ve dönüşümler, Gülbahar (Ed.), *Bilgi işlemsel düşünmeden programlamaya in* (ss. 396-410). Ankara: Pegem Akademi.

Güler, H., Şahinkayası, Y. & Şahinkayası, H. (2017). Internet ve mobil teknolojilerin yaygınlaşması: Fırsatlar ve sınırlılıklar. *Sosyal Bilimler Dergisi*, 7(14).

HackerCan (2023, May). http://www.hackercan.com/tr

Hansen, A. K., Iveland, A., Carlin, C., Harlow, D. B. & Franklin, D. (2016). User-centered design in block-based programming: Developmental & pedagogical considerations for children. In *Proceedings of the 15th International Conference on Interaction Design and Children* (pp. 147-156). doi:10.1145/2930674.2930699

Hayat, K., Al-Shukaili, N. A. & Sultan, K., (2017). Alice in Omman. *Education and Information Technologies*, 22(4): 1553-1569.

Hubwieser, P., Giannakos, M. N., Berges, M., Brinda, T., Diethelm, I., Magenheim, J. & Jasute, E. (2015, July 4-8). A global snapshot of computer science education in K-12 schools. *In Proceedings of the 2015 ITICSE on Working Group Reports. ITICSE-WGR'15*. ACM, New York, NY, USA, 65-83.

ISTE (2023). *ISTE Standarts: Students*. https://www.iste.org/standards/iste-standards-for-students

Jain. A. (2018). *What is computer science?* Retrieved from http://cs.boisestate.edu/~amit/teaching/121/handouts/WhatIsComputerScience.pdf

Jenson, J. & Droumeva, M. (2016). Exploring media literacy and computational thinking: A game maker curriculum study. *The Electronic Journal of e-Learning, 14*(2), 111-121.

John, M. S. & Rani, M. S. (2015). Teaching Java programming on smartphone-pedagogy and innovation; Proposal of its ontology-oriented implementation. *Procedia-Social and Behavioral Sciences*, (176), 787-794. doi:10.1016/j.sbspro.2015.01.541

Kalelioğlu, F. & Gülbahar, Y. (2014). The effects of teaching programming via Scratch on problem solving skills: A discussion from learners' perspective. *Informatics in Education, 13*(1), 33-50.

Kalelioğlu, F. & Keskinkılıç, F. (2017). Bilgisayar bilimi eğitimi için öğretim yöntemleri. Y. Gülbahar (Ed.), *Bilgi işlemsel düşünmeden programlamaya içinde* (ss. 155-182). Ankara: Pegem Akademi. doi:10.14527/9786052411117.07

Kandemir, C. M. (2017). Metin tabanlı programlama. Y. Gülbahar (Ed.), *Bilgi işlemsel düşünmeden programlamaya* içinde(ss. 267-294). Ankara: Pegem Akademi. doi:10.14527/9786052411117.11

Keşfet. (2018, May 21). http://www.kesfetprojesi.org/hakkimizda

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

KodlaRize. (2017, September 7). http://kodlarize.gov.tr

KodlaManisa. (2020, April). *Kodlamanisa Projesi*. Retrieved from http://www.kodlamanisa.gov.tr

Kodris. (2023, May). https://www.kodris.com

Kukul, V. ve Gökçearslan, Ş. (2014, September). *Scratch ile programlama eğitimi alan öğrencilerin problem çözme becerilerinin incelenmesi*. Retrieved from https://goo.gl/6ucqn1

Lego Edycation (2018). https://education.lego.com (Date, 2018 April)

Lewis, C. M. (2010, March). How programming environment shapes perception, learning and goals: logo vs. scratch. *In Proceedings of the 41st ACM technical symposium on Computer science education,* New York: Association for Computing Machinery, 346-350.

Lopez, J. M. & Cozar Gutierrez, R. (2017). Computational thinking and visual programming through blocks in the elementary school classroom, *EDUCAR,* 53(1): 129-146.

Makeblock. (2018, April). https://www.makeblock.com

MakeyMakey. (2018, April). https://makeymakey.com

Mblock. (2018, May). http://www.mblock.cc/

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2011, June). *Habits of programming in scratch*. ITICSE'11 Proceedings of the 16th annual joint conference on Innovation and technology in computer science education, (168-172). New York: Association for Computing Machinery.

Millî Eğitim Bakanlığı [MEB]. (2023, August). Düzce Kodluyor. Retrieved from https://duzce.meb.gov.tr/www/duzce-kodluyor/icerik/5325,

Millî Eğitim Bakanlığı [MEB]. (2017, September 2). *Öğretim programlarını izleme ve değerlendirme sistemi*, Retrieved from http://mufredat.meb.gov.tr/ProgramDetay.aspx?PID=152

Millî Eğitim Bakanlığı [MEB]. (2018, June 23). *Öğretim programlarını izleme ve değerlendirme sistemi,* Retrieved from http://mufredat.meb.gov.tr/Dosyalar/2018120203611364-BILGISAYAR%20BILIMI%20 DERSI%20OGRETIM%20PROGRAMI.pdf

Millî Eğitim Bakanlığı [MEB]. (2020, April). *Bilişim teknolojileri çerçeve programı.* Retrieved from http://bilsem.meb.gov.tr

Morelli, R., De Lanerolle, T., Lake, P., Limardo, N., Tamotsu, E. & Uche, C. (2011). Can android app inventor bring computational thinking to k-12. *In Proc. 42nd ACM technical symposium on Computer science education, SIGCSE'11*, 1-6.

Nishida, T., Kanemune, S., Idosaka, Y., Namiki, M., Bell, T. & Kuno, Y. (2009). *A CS unplugged design pattern*. Proceedings of the 40th ACM Technical Symposium on Computer Science Education, (pp. 231). SIGCSE 09. doi:10.1145/1508865.1508951.

Özçakmak, Ş. (2014). *Bilgisayar kullanımı çocukta bağımlılık yapar mı?* Retrieved from http://www.haberturk.com/polemik/haber/973204-bilgisayar-kullanimi-cocukta-bagimlilik-yapar-mi?

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Raab, J., Rasala, R. & Proulx, V. K. (2000). Pedagogical power tools for teaching Java. *ACM SIGCSE Bulletin*, *32*(3), 156-159.

Reed, W. M. & Palumbo, D. B. (1992). The effect of basic instruction on problem-solving skills over an extended period of time, *Journal Of Educational Computing Research*, 8(3): 311-325.

Reusink, M. (2018, May 17). Why is computer science important? Retrieved from "http://www.rasmussen.edu/degrees/ technology/blog/ways-computer-science-benefits-society/

Robins, A., Rountree, J. & Rountree, N. (2003) Learning and teaching programming: A review and discussion. *Computer Science Education, 13(*2): 137-173.

Rogers, Y., Paay, J., Brereton, M., Vaisutis, K. L., Marsden, G. & Vetere, F. (2014) Never too old: Engaging retired people inventing the future with MaKey MaKey. *In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* ACM, 3913-3922.

Sabitzer, B. (2011, March). *Neurodidactics–a new stimulus in ICT and computer science education*. Conference paper. *5th International Technology, Education and Development Conference*, Valencia, Spain.

Sabitzer, B., Antonitsch, P. K. & Pasterk, S. (2014). November, Informatics concepts for primary education: Preparing children for computational thinking. *In Proceedings of the 9th Workshop in Primary and Secondary Computing Education,* ACM, 108-111.

Saygıner, Ş. & Tüzün, H. (2017, February 8-10). Programlama eğitiminde yaşanan zorluklar ve çözüm önerileri. *Akademik Bilişim '17. Konferansı*, Aksaray.

Saygıner, Ş. & Tüzün, H. (2017, February 8-10). İlköğretim düzeyinde programlama eğitimi: yurt dışı ve yurt içi perspektifinden bir bakış. *Akademik Bilişim '17. Konferansı*, Aksaray.

Scratch. (2017, December 9). *Scratch hakkında*, Retrieved from https://scratch.mit.edu/about:

Takacs, A., Eigner, G., Kovács, L., Rudas, I. J. & Haidegger, T. (2016). Teacher's kit: Development, usability, and communities of modular robotic kits for classroom education. *IEEE Robotics & Automation Magazine*, 23(2): 30-39.

Taub, R., Ben-Ari, M. & Armoni, M. (2009). The effect of CS unplugged on middle-school students' views of CS. *ACM SIGCSE Bulletin*, 41(3): 99-103.

Üçgül, M. (2017). Eğitsel robotlar ve bilgi işlemsel düşünme. Y. Gülbahar (Ed.), *Bilgi işlemsel düşünmeden programlamaya* içinde (ss. 295-317). Ankara: Pegem Akademi.

Vandevelde, C., Saldien, J., Ciocci, M. C. & Vanderborght, B. (2013). Overview of technologies for building robots in the classroom. In *International conference on robotics in education* (pp. 122-130).

Vieira, Camilo & Magana, Alejandra. (2013). *Colombian Elementary Students' Performance and Perceptions of Computing Learning Activities with Scratch*. 10.18260/1-2--19315.

Ward, B., Marghitu, D., Bell, T. & Lambert, L., 2010, Teaching computer science concepts in Scratch and Alice. *Journal of computing Sciences in Colleges, 26(2):* 173-180.

Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Sysło, M. M. (2017). Computer science in K-12 school curricula of the 2lst century: Why, what and when? *Education and Information Technologies*, *22*(2), 445-468. doi:10.1007/s10639-016-9493-x

Weinberg, A. E. (2013). *Computational thinking: An investigation of the existing scholarship & research.* Retrieved from https://search.proquest.com/docview/1413309206?accountid=10699

Weintrop, D. & Wilensky, U. (2015, June). *To block or not to block, that is the question: Students' perceptions of blocks-based programming.* Full paper, 14 International Conference on Interaction Design and Children, Medford, MA, USA. doi:10.1145/2771839.2771860

Werner, L., Campe, S. & Denner, J. (2012). Children learning computer science concepts via Alice game-programming. *In Proceedings of the 43rd ACM technical symposium on Computer Science Education,* ACM, 427-432.

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

Wolber, D., Abelson, H. & Friedman, M. (2015). Democratizing computing with app inventor. *GetMobile: Mobile Computing and Communications*, 18(4): 53-58.

Yükseköğretim Kurulu [YÖK]. (2018). *Bilgisayar ve Öğretim Teknolojileri Öğretmenliği Lisans Programı,* Retrieved from http://yok.gov.tr/documents/10279/41805112/Bilgisayar_ve_Ogretim_Teknolojileri_Ogretmenligi_Lisans_Programi.pdf

Yükseltürk, E. & Altıok, S. (2016). Bilişim teknolojileri öğretmen adaylarının programlama öğretiminde scratch aracının kullanımına ilişkin algıları. *Mersin Üniversitesi Eğitim Fakültesi Dergisi*, 12(1): 39-52.