International Journal of Educational Research 114 (2022) 102008

Contents lists available at ScienceDirect

International Journal of Educational Research

journal homepage: www.elsevier.com/locate/ijedures

Computer science education and K-12 students' computational thinking: A systematic review

Sang Joon Lee *, Gregory M. Francom, Jeremiah Nuatomue

Instructional Systems and Workforce Development, Mississippi State University, Box 9730, Mississippi State, U.S.A.

ARTICLE INFO

Keywords:
Computer science education
Computational thinking
Programming
Systematic review
K-12

ABSTRACT

This systematic review examined how computer science (CS) education was implemented in schools and its efficacy for developing students' computational thinking (CT). Sixty-six papers were selected for this systematic review and analyzed for patterns in relation to the implementation of CS education in K-12 schools and its impact on the development of students' CT skills. Although educational interventions have not always been successful in CS education, this review provides strong evidence that overall, CS education promotes the development of students' CT in the K-12 setting while improving their creative and critical thinking skills. We recommend early access to CS education, various innovative instructional approaches to CS education and appropriate support and guidance for student learning.

1. Introduction

The hardware and software that make up computer systems are critical for the functioning of our society in the Information Age. These systems depend heavily on the computer science (CS) skills of professionals who are able to develop and maintain them. Because of the critical need for CS knowledge, CS education has become more prevalent in schools worldwide (Blikstein & Moghadam, 2018; Price & Price-Mohr, 2018). Evidence of the increasing prevalence of CS education includes initiatives from major technology companies, including Apple and Google, to fund and support the teaching and learning of CS concepts in K-12 education over the past years. In addition, the CS for All (CSforALL) initiative was launched in New York City and then expanded to the full United States. The CSforALL initiative has a mission to make CS education available for all K-12 students' learning experience and improve advancement to future CS study and careers (Santo et al., 2019).

1.1. Computer Science Education

Though computers have been used in education for over 50 years, beginning with Papert's use of LOGO for kids (Papert, 1980, 1993), modern CS education has taken decades to fully emerge (Blikstein & Moghadam, 2018). Current CS education experiences are more focused on developing students' coding skills to match those that CS professionals use from day to day. These approaches to CS education have covered a variety of areas and have used block-based and text-based coding along with a variety of physical and virtual tools (Durak, 2020; Hsiao et al., 2019; Weintrop & Wilensky, 2019). Many of these approaches have included the learning of CT concepts, such as sequences, events, parallelism, conditionals, and operators (Deng et al., 2020; Garneli & Chorianopoulos, 2018).

* Corresponding author. Sang Joon Lee, Instructional Systems and Workforce Development, Mississippi State University, USA.
E-mail address: slee@colled.msstate.edu (S.J. Lee).

https://doi.org/10.1016/j.ijer.2022.102008
Received 16 March 2022; Received in revised form 3 May 2022; Accepted 23 May 2022
Available online 29 May 2022

Though the coding tools have become more advanced over the years, the constructionist approach – proposed years ago (Kafai, 2006; Papert, 1993) – is still the dominant theoretical foundation for modern CS education (see Weintrop & Wilensky, 2019; Wong & Cheung, 2020).

1.2. Computational Thinking

Wing (2006) popularized the term, *Computational Thinking* (CT), and explained that CT should be a fundamental skill for all students to learn. In her seminal work, Wing further contrasted CT from simply learning about coding, but as broader ways of thinking about problems and the use of CS to help solve them. Thus, CT includes conceptualizing problems in ways that they can be effectively worked through with the help of computers. CT is about using creativity and new ideas based on an understanding of CS to complete meaningful actions (Wing, 2006).

Brennan and Resnick (2012) further developed the CT concept by situating it within CS education and breaking it down into three dimensions: CT concepts, CT practices, and CT perspectives. CT concepts include ideas that are used by CS professionals while developing software, including sequences, events, parallelism, conditionals, operators, and data (Brennan & Resnick, 2012). CT practices are the activities that designers of software systems work through. As observed by Brennan and Resnick in CS education, these practices may include; (a) iteratively developing software, (b) testing and debugging, (c) reusing and remixing, and (d) abstracting and modularizing. CT perspectives are the perspectives that CS professionals form about themselves and the world around them (Brennan & Resnick, 2012). Brennan and Resnick observed that CT perspectives in CS education may include expressing yourself through programming, connecting socially, and using computation to ask questions to make sense of the world.

Overall, CT is proposed as a way to avoid a narrow focus on simply learning coding skills in CS education. The concept of CT helps to broaden CS education efforts to include ways of thinking and acting that relate to real-world problems that can be solved using computers (Wing, 2006, 2008). Thus, a variety of CS education approaches have been implemented that go beyond simply learning coding skills to cover CT concepts, practices and perspectives. In addition, various CT-based methods have been used to measure the efficacy of CS learning (e.g. Baek et al., 2019; Chiazzese et al., 2019; Durak et al., 2019). These methods have ranged from specific assessment of individual student projects, to developed instruments that can assess improvement in CT.

In this systematic review, we examine research reports of CS education to determine how it has been implemented in K-12 schools and to evaluate how it has affected K-12 students' CT. We also discuss the challenges that teachers and students have run into when implementing CS education and provide recommendations for CS educators.

1.3. Findings and Limitations of Previous Reviews

There have been a limited number of systematic reviews to understand how the implementation of CS education impacted CT at various educational levels. In a systematic review, Tang et al. (2020) found that there were fewer CT assessments examined in high school, college, and professional development than in elementary school. Fidai et al. (2020) conducted a meta-analysis of 11 quantitative studies that implemented Arduino- and Scratch-enabled interventions to determine how the use of such tools helped to improve CT skills in K-12 and post-secondary settings. The study indicated that the open-source instructional interventions improved students' CT skills specifically in the areas of problem-solving, creative thinking, application of engineering concepts, computer programming, and cognitive abilities. In another review study, Xia and Zhong (2018) examined the robotics content knowledge in K-12 schools and concluded that most empirical studies took place at the elementary level, with LEGO being the dominant tool for teaching and learning.

Additionally, a few review studies have targeted different audiences. A systematic review by Kakavas and Ugolini (2019) focused on CT among elementary school students and found that most of the studies used programming for both plugged and unplugged activities to develop students' CT skills. Similarly, Barcelos et al. (2018) reported how mathematics learning was used to cultivate CT skills, but the target audience of the reviewed studies ranged from elementary to graduate programs. Likewise, very few systematic reviews studies have been conducted to examine how CS education has been implemented in K-12 schools and the extent to which CS education helps students develop their CT skills.

1.4. Research Questions

It has been more than five years since the CSforALL initiative was announced to expand CS education and provide high-quality CS education in the US. During this time frame, CS education and programming activities have increased in K-12 classrooms internationally as well as in the US. Thus, it is important to review how CS education has been implemented in K-12 schools and examine whether it helps K-12 students develop their CT skills.

A systematic review is a research method that provides an up-to-date and complete summary of primary research and answers research questions (Higgins et al., 2021). This systematic review aims to gain knowledge about the implementation of CS education in K-12 schools and examine whether CS education helps K-12 students increase their CT. This review was guided by the following research questions:

1 How was CS education implemented in K-12 schools in 2016–2020?
2 How has CS education impacted the development of K-12 students' CT?

2. Method

A systematic review mainly involves three essential activities: (a) identify relevant research studies to answer research questions, (b) critically appraise them in a systematic manner, and (c) synthesize, interpret, and present the findings (Gough et al., 2012). All three authors identified the objectives and the research questions, created a review protocol, and decided on inclusion and exclusion criteria for considering studies for this review as shown in Table 1.

2.1. Search Strategy

Eight different databases were used to identify eligible studies that met the inclusion criteria. These databases were ERIC (EBSCO), Academic Search Complete (EBSCO), PsycINFO (EBSCO), Psychology and Behavioral Sciences Collection (EBSCO), ScienceDirect, Scopus, JSTOR, and IEEE Xplore.

The search term was a combination of "computer science", "computational thinking", and "K12" in the title, abstract, or keyword. With the consultation of a librarian, the search string of *("computer science" OR programming OR coding) AND ("computational thinking") AND (k-12 OR k12 OR "primary school" OR "elementary school" OR "middle school" OR "junior high" OR "high school" OR "secondary school")* was used in the databases.

2.2. Selection of Studies

Fig. 1 shows a flow diagram of the study identification and selection process. The searches were conducted in September 2021 and retrieved 542 results. After 177 duplicate records were removed, 365 records were exported to Rayyan for eligibility screening, a web-based tool that helps research teams screen and select studies for systematic reviews. To reduce a single person's biases, all three authors were involved in all three steps to screen and select studies (Higgins et al., 2021). First, we reviewed the titles and abstracts independently and indicated what to include and exclude in Rayyan. There were 57 studies (15.6%) and 148 studies (40.6%) that all members agreed to include or exclude, respectively. There were conflicts in 160 studies (43.8%). Second, after discussing discrepancies and reviewing the eligibility criteria, all members determined to include 50 studies among the conflicting studies. After the first and second steps of the initial screening, 107 articles remained for full text screening. Finally, the full texts of all the remaining articles except one (106 studies) were retrieved and reviewed for consideration. After the full-text screening, 40 studies were removed based on the eligibility criteria. These studies were removed because they were conducted in informal settings, included interventions or outcomes that did not relate to CT, included the development of new interventions without measuring CT, or were instrument development or document analysis studies without interventions.

2.3. Data Collection and Analysis

In total, 66 papers were selected for this systematic review. Among them, 72.7% were published in 2019 and 2020: 3 (4.5%) in 2016, 5 (7.6%) in 2017, 10 (15.15%) in 2018, 22 (33.3%) in 2019, and 26 (39.4%) in 2020. Table 2 shows a list of journals with multiple articles.

The selected studies were divided and analyzed by each author. A data collection form was developed to help the authors extract necessary data from the assigned studies. Aim of study, study design, duration, study population, research setting and country, instructional method and tools, assessment, study results, challenges, and key conclusions of studies were collected. After collecting data from the selected articles, the data collection forms were shared among the authors. Each author began analyzing data by coding the data collection forms and identifying patterns in relation to the implementation of CS education in K-12 schools and its impact on the development of students' CT.

Table 1
Eligibility criteria.

	Inclusion Criteria	Exclusion Criteria
Population	• Students in K-12 schools (in the US & international countries)	• Students in higher education or adults • Teachers
Intervention	• CS education (programming or coding) • Formal, school-oriented programs • Studies with empirical data	• General STEM education • Informal, after-school activities • Meta-analysis • Systematic review
Outcomes	• Computational thinking	• Teachers' perceptions • General learning outcomes
Sources	• Scholarly journals in English	• Conference proceedings • Reports • Book chapters
Years	• 2016-2020	

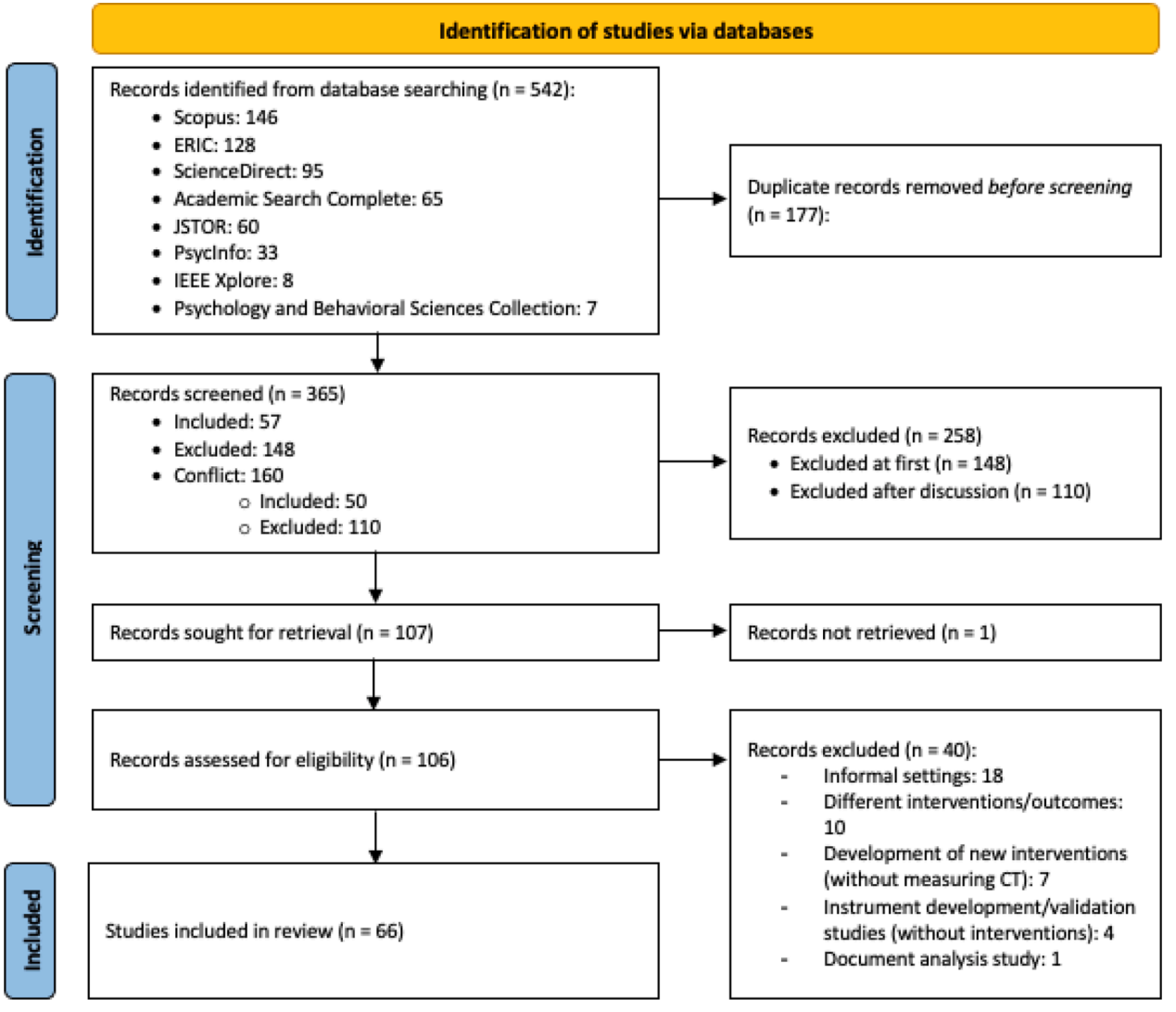

Fig. 1. Flow diagram of data collection adopted from Page et al. (2021)

Table 2
List of journals with multiple articles.

Journal	No. of articles selected in this study
ACM Transactions on Computing Education	5
Informatics in Education	5
Interactive Learning Environments	5
Computers & Education	4
International Journal of Child-Computer Interaction	4
International Journal of Computer Science Education in Schools	3
Journal of Educational Computing Research	3
Asia-Pacific Education Researcher	2
Educational Technology Research and Development	2
Journal of Computer Assisted Learning	2
Journal of Science Education and Technology	2
Participatory Educational Research	2

3. Results

3.1. Implementation of CS education in K-12 schools

The first research question for this study addresses how CS education has been implemented in K-12 schools in 2016–2020. An analysis of the articles reviewed reveals demographics of CS education learners, types of settings in which CS education has taken place, and most importantly, the instructional tools and methods used to support CS education.

3.1.1. Level and Age

Interestingly, of the studies included in this review, most occurred in elementary schools with studies in the upper elementary grades (4-6) comprising about two-thirds of the reported elementary studies, as shown in Table 3. Studies completed in middle schools made up the next largest group, with junior high or high school studies featured less prominently. Ages of CS education recipients – where reported – range from 5 to 18 years, with a median age of 12 years. This analysis suggests that much of CS education is happening on the upper elementary and middle school levels.

3.1.2. Gender

Gender has been a prominent topic of discussion in CS education, with encouragement for greater participation in coding by girls (Hur et al., 2017; Zagami et al., 2015). Information on the gender of CS education participants – where available in the studies – was analyzed. Findings showed that about 53.7% of participants in CS education were male, while about 46.2% were female. Though it is important to note that in many cases, male students self-selected to be in the CS education experiences to a greater degree than female students.

3.1.3. Setting

The settings for CS education studies reviewed here included K-12 schools internationally and in the United States. The United States accounted for CS education experiences in about 28% of these studies, however studies came from other countries such as Turkey, Greece, the United Kingdom, Spain, Taiwan, Korea and more. Where the type of class was indicated, a variety of names for classes in which CS education occurred were given, including information technology, software, robotics, programming, computer science, mathematics and informatics. In terms of duration, approximately 73% of the interventions reported in the selected studies lasted less than three months with about 21% having five or fewer meetings.

3.1.4. Instructional Methods and Tools

In the studies we reviewed, there were a wide variety of instructional methods and tools used in CS education. Summaries of the studies were analyzed for patterns that indicated the methods of instruction, the tools used, and other notable characteristics of the featured CS education experience.

Theoretical Foundation. A link to constructionist or project-based learning theories and practices was often made in the reviewed

Table 3
The number of selected studies by educational level.

Education Level		No. of studies	%
Elementary School	Lower Level	9	48.5%
	Upper Level	21	
	Lower & Upper Level	2	
Middle School		19	28.8%
Junior High and High School	Junior High	2	22.7%
	Junior High & High	2	
	High School	11	

studies (e.g., Weintrop & Wilensky, 2019; Wong & Cheung, 2020). Constructionist theory suggests that students benefit from the rich learning processes that occur as they are asked to create artifacts that show their knowledge of a topic (Kafai, 2006; Papert & Harel, 1991). This theory emerged from the intersection of mathematics and computer science, and provides an effective framework for hands-on CS learning experiences.

For example, Weintrop and Wilensky (2019) discussed a constructionist CS education experience in which students transitioned from the Pencil Code programming editor to a full Java learning environment. In this experience, students followed the *beauty and joy of computing* curriculum provided by UC Berkeley, a constructionist approach to programming (Weintrop & Wilensky, 2019). Another example of constructionist CS education comes from Wong and Cheung (2020), who describe an experience in which 4-6th graders created a game-based project to showcase their programming learning. A key component of this approach included encouraging students to remix and personalize their programs (Wong & Cheung, 2020). In addition to these studies, various other CS education experiences reported using approaches compatible with constructionism, in which students created artifacts – whether digital or physical – to show their learning of programming concepts.

Learning processes. One pattern that emerged from the analysis of studies involved the process of CS learning that took place. Many of the studies outlined a general process that involved a scaffolded approach to learning the tools of CS education. The general approach included: (a) introducing the new tool, including showing features and explaining how it works; (b) allowing for free play and exploration by students; (c) challenging students with tasks to complete using the tool; (d) having students test out and adjust or "debug" their solutions, and; (e) inviting students to reflect on the experience (see Baek et al., 2019; Chou, 2020; Durak, 2020; Merkouris & Chorianopoulos, 2019; Shen et al., 2020; Soleimani et al., 2019).

Many example studies follow this general process, with an initial phase in which the basics of featured tools are introduced to students and then free play and exploration are encouraged (Durak, 2020; Shen et al., 2020; Soleimani et al., 2019). For instance, Merkouris and Chorianopoulos (2019) outlined an approach to teaching app inventor, Scratch and LEGO Mindstorms that included introducing the user interface, allowing students to program, inviting them to add specific new features, and then involving students in a semi-open problem-solving task. Chou (2020) described a similarly scaffolded CS education experience in which third-grade students were introduced to programming concepts and practices, which included students learning about programming tools and concepts in homework assignments. At the start of class, students discussed the homework, and then the instructor introduced a more open-ended project along with requisite programming skills. Finally, students modified the programming they had completed with their own ideas (Chou, 2020). Generally, the learning processes reported in these studies followed the above-mentioned pattern in which students were given more control of the learning process as their skills increased.

CT concepts. CT concepts include ideas that are important to coding, such as sequences, events, parallelism, conditionals, operators, and data (Brennan & Resnick, 2012). Many of the studies reported teaching students these CT concepts as part of formal CS education, especially among middle and high school students (Deng et al., 2020; Garneli & Chorianopoulos, 2018; Lockwood & Mooney, 2017). For instance, Garneli and Chorianopoulos (2018) discussed a CS education experience in which middle school age students in Greece learned the CT concepts of synchronization (parallelism), coordination, variables, conditionals, variables, and events using the popular Scratch block-based programming tool. Similarly, Lockwood and Mooney (2017) reported teaching programming concepts to high school students, which included algorithms, searching, sorting and cryptography.

Subject area integration. Another important pattern observed in these reviewed studies was the integration of other subject areas into the CS education experience. Many of the studies described experiences in which students learned programming concepts and also integrated scientific or mathematical concepts in the process (see Aksit & Wiebe, 2020; Chiazzese et al., 2019; Rodríguez-Martínez et al., 2020; Sáez-López et al., 2019).

Notable examples of this include studies by Aksit and Wiebe (2020) and Sáez-López et al. (2019), in which students completed programming activities about the science concepts of force and motion. Another science-based CS education implementation reported by Chiazzese et al. (2019) required fourth-grade students to find a way to recreate the life cycle of a frog using LEGO WeDo robots after learning about programming concepts. This study showed the value of using physical robotics as a way to transfer programming skills to realistic situations. In the area of mathematics, Rodríguez-Martínez et al. (2020) reported on a CS education experience in which students learned programming concepts, then were required to work on a mathematical learning standard using Scratch. In this study students in the Scratch group significantly increased in their ability to solve word problems related to least common multiple and greatest common divisor.

In addition to mathematical and scientific subjects, a small subset of studies also integrated storytelling into CS education experiences. Many of these studies followed a process that first taught basic coding ability using a block-based coding tool, then challenged students to tell a story using the coding tool (Chang, 2019; Moschella & Basso, 2020; Saritepeci, 2020). Interestingly, Saritepeci (2020) presented a study in which a programming learning group was compared to a storytelling learning group and found that both groups improved the same amount in CT, problem-solving and creativity.

Physical and virtual tools. One interesting contrast that emerged from the analysis of studies was between physical and virtual tools. While a majority of the studies used exclusively virtual tools such as Scratch, Pencil, and Code.org, a smaller number of studies employed physical tools for learning such as LEGO Mindstorms, Arduino microcontrollers, Vex IQ robots and even textiles (Chevalier et al., 2020; Durak et al., 2019; Hsiao et al., 2019; Karaahmetoglu & Korkmaz, 2019; Kert et al., 2019; Richard & Giri, 2019). While the use of virtual CS education tools allows students to focus on programming operations and syntax, physical CS education tools help students concretely see the results of their code.

Often these studies outlined an approach in which students first learned about a coding tool, then learned about the physical object that can be controlled with the code, then worked on challenges for coding the physical object to perform key tasks (Durak et al., 2019; Hsiao et al., 2019). Overall, both block-based programming and physical robotics learning methods seem to work to improve CS

student outcomes (Kert et al., 2019). However, the use of physical objects that can be programmed may particularly improve student motivation, confidence and problem-solving skills (Durak et al., 2019; Karaahmetoglu & Korkmaz, 2019; Kert et al., 2019).

Plugged-in and unplugged activities. Another pattern that emerged in the reviewed studies included plugged-in vs. unplugged activities. Plugged-in activities involve activities that can only be done with a powered device such as a computer. In contrast, unplugged activities include games, paper-based activities, and other challenges that teach CS concepts and processes, but can be done without a computer or external power (see Ardito et al., 2020; Chen & Chi, 2020; del Olmo-Muñoz et al., 2020; Kuo & Hsu, 2020; Rijke et al., 2018). Some examples of unplugged CS education activities included coding-themed board games (Chen & Chi, 2020; Kuo & Hsu, 2020), paper-based activities (del Olmo-Muñoz et al., 2020; Rijke et al., 2018), and activities that use other common office supplies (Ardito et al., 2020). Unplugged CS education activities were more often used among younger learners whose computer knowledge is still developing (Chen & Chi, 2020; Rijke et al., 2018).

A few studies strategically combined the use of plugged-in and unplugged activities. For example, Ardito et al. (2020) outlined a CS learning process that used both plugged-in and unplugged activities. First, the unplugged activities were introduced to students and then students completed plugged-in activities that built upon the same concepts learned in the unplugged activities (Ardito et al., 2020). Del Olmo-Muñoz et al. (2020) compared the use of unplugged and plugged-in activities and found that younger students, and girls in particular, seemed to benefit more from unplugged activities, yet measures of CT improvement showed no significant difference between unplugged and plugged-in groups.

Block-based vs. text-based programming. Various reviewed studies also discussed the challenges and benefits of block-based versus text-based programming (Gillott et al., 2020; Weintrop & Wilensky, 2018, 2019). Block-based programming tools such as Scratch allow a low threshold for beginning CS learners to jump into programming activities (Weintrop & Wilensky, 2018). However, these tools may not include more advanced features, and they are less applicable to professional programming activities.

Weintrop and Wilensky (2019) implemented a study comparing a block-based programming group to a text-based programming group and found that confidence and enjoyment were the same in both groups. However, the block-based programming group showed quicker programming learning at the outset, which diminished over time until ultimately both groups learned the same amount (Weintrop & Wilensky, 2019). Weintrop and Wilensky (2018, 2019) have suggested that a hybrid approach that allows learning of both block- and text-based programming may be the best approach to support the development of programming knowledge.

3.2. Impact of CS Education on the Development of K-12 Students' CT

The second research question for this study addresses what impact CS education has had on the development of students' CT in K-12 schools from 2016 to 2020. Most studies used a combination of quantitative (e.g., tests, surveys, and questionnaires) and qualitative (e.g., interviews, observations, and artifact analysis) methods to measure students' CT skills, such as abstraction, decomposition, algorithmic thinking, systems thinking, pattern recognition, problem solving, and logical reasoning. The Computational Thinking Scale (see Atun & Usta, 2019; Durak et al., 2019; Tonbuloğlu & Tonbuloğlu, 2019; Uşengül & Bahçeci, 2020) and the Computational Thinking Test (see Aksit & Wiebe, 2020; Rodríguez-Martínez et al., 2020; Taylor & Baek, 2019; Zhao & Shute, 2019) are the two CT measurement instruments used in multiple studies. Overall, research has shown that CS education promotes the development of students' CT knowledge and skills. Furthermore, CS education in K-12 schools can influence students' performance in CS courses in colleges (Burgiel et al., 2020). Additionally, CS education improves students' academic achievement, creative thinking, critical thinking, reasoning, and problem-solving skills (Atun & Usta, 2019; Miller, 2019; Wong & Cheung, 2020).

3.2.1. CT by Intervention

CT skills are not naturally developed; educational interventions are necessary to help children develop basic computational concepts (Rodríguez-Martínez et al., 2020). Thus, various interventions using different approaches and tools have been used in CS education. Among the most implemented interventions in CS education are educational robotics, block-based programming, and educational games.

Educational robotics. Educational robotics (ER) have been an effective tool for teaching STEM subjects and also used in CS education to increase students' CT skills in K-12 schools (Chiazzese et al., 2019). Learning programming is a highly conceptual process and ER enables students to materialize their computational concepts through computational practices (Min & Kim, 2020). Recent studies provide evidence that ER in programming education can be an effective tool for the development of CT skills (Ardito et al., 2020; Hsiao et al., 2019; Noh & Lee, 2020; Shen et al., 2020). For example, Hsiao et al. (2019) reported on a robot-based learning experience in which elementary school students were asked to make a robot and use Scratch programming to control it to complete assigned tasks. Hsiao et al. found that students in the robotics program achieved significantly better learning performance and CT scores than others who learned through lectures. Measuring students' CT skills before and after a virtual robotics programming curriculum in secondary schools, Witherspoon et al. (2017) found that the robotic curriculum brought about significant gains, particularly for those who advanced to more complex tasks. Furthermore, those who were introduced to the structural logic of programming showed significant learning gains and developed programming knowledge that could be transferred to dissimilar contexts or non-robotics computing tasks (Witherspoon et al., 2017, 2018).

Additionally, ER has had a positive effect on creativity and subject matter learning (Miller, 2019; Noh & Lee, 2020; Sáez-López et al., 2016, 2019). Sáez-López et al. (2019) implemented activities that integrated programming and robotics in sciences and mathematics in primary education schools. The results showed statistically significant improvements not only in the acquisition and understanding of computational concepts but also in science and math. Miller (2019) also found that coding instruction using robotics led to a significant increase in mathematical thinking.

Some of the positive effects of ER in CS education can be attributed to an increase in student motivation (Sáez-López et al., 2019). Atmatzidou and Demetriadis (2016) investigated the development of students' CT skills through ER and showed that working with robots motivated students to program and helped them develop a deeper understanding of programming concepts. With 81 students in secondary education, Çınar and Tüzün (2020) compared the effects of object-oriented and robot programming activities on programming achievement, abstraction, problem solving, and motivation. While there was no significant difference between the groups on programming achievement scores, abstraction, and problem solving, the robotic group showed statistically significantly higher motivation scores than those in the object-oriented programming group.

The motivation factor of ER, however, may not last long. Witherspoon et al. (2018) tested the effects of different programming content within a virtual robotic context on motivational changes in middle schools. The results showed declines in programming interest, identity as a programmer, and beliefs in their programming abilities, especially for those who were involved in simple tasks. In a robotics programming training with 6-7th graders, Durak et al. (2019) found that students who used robots in programming education for the first time were likely more motivated and made a higher effort to learn programming than those who had participated in ER before.

Also, the effect of ER may not be the same on all students. Students' different individual traits such as intrinsic and extrinsic motivation, self-efficacy, and executive functions as well as environmental factors are related to the development of CT skills with ER (Baek et al., 2019; Chiazzese et al., 2019; Noh & Lee, 2020; Roberson et al., 2020). For example, Noh and Lee (2020) showed that teaching programming using a robot improved CT significantly, but the difference between pre- and post-test CT scores was not significant for high-achieving students. Examining the effect of an ER laboratory on the development of CT skills in primary school students, Chiazzese et al. (2019) found that students participating in the robotics laboratory spent more time on their assignments and obtained significantly higher total scores on the assigned tasks than students attending regular lessons.

Compared with other approaches, ER has shown similar or more positive outcomes in CS education. Kert et al. (2020) compared the effects of ER to block-based programming on sixth-graders' academic achievement, CT skills, and conceptual knowledge levels in programming education. Their findings showed that ER was more effective for developing CT skills than block-based programming. In a study by Çınar and Tüzün (2020) comparing the effects of object-oriented and robot programming activities, both robotic and object-oriented programming groups showed significant increases in programming achievement scores, abstraction, and problem solving.

Block-based programming environments. Computer programming has been considered a significant factor for developing students' CT skills (Chiazzeses et al., 2018). Block-based programming is an effective method for programming education in that students learn programming concepts more quickly (Kert et al., 2020; Weintrop & Wilsensky, 2019). Block-based programming allows students to drag-and-drop visual objects and create games and animations. Because learners do not need to memorize and write a code, block-based programming allows them to use more cognitive resources to understand programming and CT concepts (Deng et al., 2020; Weintrop & Wilensky, 2018, 2019). Therefore, block-based programming environments such as Scratch, Alice, and Code.org have become some of the most popular tools used in CS education for children.

Block-based programming promotes the development of CT, reflective thinking, and problem-solving abilities (Deng et al., 2020). For example, in a two-year case study that implemented Scratch in primary schools, Sáez-López et al. (2016) found that block-based programming resulted in a significant improvement in programming concepts, logic, and computational practices. In an eight-week CT educational training, Chou (2020) integrated ScratchJr into a programming curriculum of an elementary school. The findings included significant improvement in students' CT competence and high retention one month after the completion of the training.

Different programming tools affected the development of students' CT differently (Deng et al., 2020). Durak (2020) compared the effects of two popular block-based programming tools, Scratch and Alice, and programming teaching practices on student engagement, reflective thinking and problem-solving, and CT. The results showed that while both programming environments contributed a significant improvement in CT skills, teaching with Scratch affected student engagement, reflective thinking and problem-solving skills more positively than teaching with Alice. Studies also have compared block-based programming with text-based programming. Deng et al. (2020) examined the effects of the Pencil Code (block-based) and the Visual Basic (text-based) tools on students' CT and computer learning attitude in a high school. The results showed that students in the block-based environment had higher scores and better performance on CT than did those learning in the text-based environment. Furthermore, students learning in the block-based environment showed more positive attitudes toward programming.

Educational games. Creating and playing educational games engages students' participation in developing CT skills and motivates students to learn more about programming, particularly for young students (Allsop, 2019; Altanis et al., 2018; Chiazzese et al., 2018; Garneli & Chorianopoulos, 2018; Gomes et al., 2018; Zhao & Shute, 2019). While developing and playing high-quality games, students can apply complex programming commands and concepts to strengthen their CT skills (Altanis et al., 2018). In a cross-curricular programming and science project, Garneli and Chorianopoulos (2018) found that students in a video game group became more motivated to learn coding and science concepts over time and showed higher CT skills than those creating a simulation.

Research has shown that students improve their thinking skills, problem solving skills and creativity while designing and creating their own games (Wong & Cheung, 2020). Allsop (2019) conducted a game-making project using Scratch and Alice with primary school students and evaluated their CT skills in terms of computational concepts, metacognitive practices, and learning behaviors. Students' completed games showed that they were able to use programming concepts including sequences, loops, parallelism, conditionals, operators and events. Students also applied planning, monitoring, and evaluation of metacognitive practices in the game-making project. Furthermore, while making games, students demonstrated collaboration, communication, perseverance, problem solving, and creativity skills (Allsop, 2019).

3.2.2. CT by Instructional Design and Approach to CS Education

CS education does not always result in positive outcomes (Gillott et al., 2020; Lockwood & Mooney, 2017; Weintrop & Wilensky, 2019). For instance, Lockwood and Mooney (2017) found that the introduction of a CS course in a secondary school, consisting of unplugged and Scratch programming lessons, did not improve students problem-solving skills and even lowered it for those without prior programming experience. Gillott et al. (2020) also found that there was little difference in the CT practices between students taking CS and those who did not take CS in secondary schools. Therefore, it is important to look at how educational interventions are designed and implemented in CS education.

Teachers' teaching styles and methods, instructional materials and activities affect the development of CT skills (Fronza et al., 2017; Kert et al., 2019; Strawhacker et al., 2018; Witherspoon & Schuss, 2019). For example, unplugged CS activities including board games, drama, and guided discovery can be used to promote CT skills (Chen & Chi, 2020; Kert et al., 2019; Kuo & Hsu, 2020; Tonbuloğlu & Tonbuloğlu, 2019). Design-based learning (DBL) activities can also be an alternative in developing CT skills in K-12 schools, especially for those who have negative attitudes toward programming (Saritepeci, 2020). Jun et al. (2017) applied DBL to an experimental group and compared changes in CT with a control group of students who received a traditional, direct method of teaching. The results showed significant positive CT changes in the DBL group over those of the control group. Similarly, Saritepeci (2020) compared the effects of DBL activities and programming training on CT skills. The results showed that the DBL group had a similar positive improvement in CT skills to the programming group. Further analysis showed that the programming group developed more in algorithmic thinking and critical thinking whereas participants in the DBL group showed improvement in problem solving and creativity.

As the use of group work has been encouraged in problem-based learning, Taylor and Baek (2019) conducted a robotics project with elementary school students and examined the impact of grouping by gender and group roles on robotics performance, CT skills, and learning motivation towards computer programming. They found that while grouping by gender did not make a difference, group roles made a significant impact on robotics performance, CT skills, and motivation. When students' roles were fixed at the start of the project and remained through the entire project, student performance, CT skills, and motivation were increased. In a study by Allsop (2019), students working in pairs used variables, loops, conditionals and event constructs more than those who worked alone.

3.2.3. CT by Gender

Gender has been discussed as an important factor in STEM-related subjects, including CS education. Some studies have found that boys learn and perform better in CT related tasks than girls. For instance, Atmatzidou and Demetriadis (2016) investigated the development of students' CT skills through robotics in two different age groups across gender. They found that although students eventually developed the same level of CT skills, girls needed more time to reach the same skill level as boys. Allsop (2019) found that while variables were challenging for both boys and girls, boys were able to use programming constructs including loops, parallelism, conditionals, and events more than girls.

However, other research studies show no significant difference between boys and girls on CT outcomes (Chiazzeses et al., 2018). Noh and Lee (2020), for example, designed a course in programming a robot and investigated its effectiveness on the CT of elementary school students. There was no statistically significant difference in CT between boys and girls. Similarly, examining the impact of grouping by gender on robotics performance of 4-5th grade students, Taylor and Baek (2019) could not find a significant difference between boys and girls in their CT skills. Also, in the middle school level, Witherspoon et al. (2017) measured students' CT skills before and after a virtual robotics programming curriculum. Although there were significant gains between pre- and post-test scores, no significant differences were found between boys and girls.

Furthermore, there have been studies showing superior performance in CS learning among girls. In a programming training of robotic activities with 6-7th graders, Durak et al. (2019) found that girls had higher skill levels of CT, programming self-efficacy and reflective thinking aimed at problem solving than boys. Witherspoon et al. (2018) also found that girls showed higher gains and performance in their programming knowledge than boys in the context of a virtual robotics in a middle school. In a project developing video games with primary school students, Chiazzese et al. (2018) found that while girls did not spend more time on games than boys, girls achieved higher and more positive learning outcomes from the project.

Boys and girls may be influenced by different factors or focus on different aspects of CT skills (Türker & Pala, 2020). For example, Witherspoon and Schunn (2019) found that girls showed significantly higher learning gains in CT than boys when having teachers who valued CT skills more in a robotics programming curriculum in middle schools. Ardito et al. (2020) investigated the collaborative development of CT skills in a robotics program with sixth-grade students. They found that while boys focused more on the operational aspects of building and coding their robots, girls focused more on group dynamics.

3.2.4. CT by Age

Although block-based programming, educational robotics and educational games are the most commonly implemented interventions in CS education, different approaches have been used based on educational level. Table 4 shows various interventions used in elementary and secondary schools. While 15.6% of the selected studies focusing on elementary students use unplugged activities, only 8.8% of the studies in middle, junior high, and high schools utilize unplugged activities. In general, secondary schools implement CS-related courses (e.g., Computer Science, Programming, and ITS courses) and text-based programming more than elementary schools (see, Günbatar, 2020; Psycharis & Kallia, 2017).

Age is related to performance in programming processes (Durak et al., 2019; Rijke et al., 2018). Different age group students show different levels of abstraction skills and programming comprehension (Moore et al., 2020; Rijke et al., 2018; Strawhacker et al., 2018). For example, older students had higher levels of CT, programming self-efficacy, and reflective thinking. Students gradually become

Table 4
CS interventions by education level.

	Elementary ($N = 32$)	Secondary ($N = 34$)
Storytelling	6.3%	5.9%
Games	18.8%	17.6%
Block-based programming	53.1%	58.8%
Robots	40.6%	35.3%
CS-related courses	0%	17.6%
Unplugged	15.6%	8.8%
Text-based programming	3.1%	17.6%

Note. The total % exceeds 100% as multiple interventions were used in most studies.

more skilled in abstraction tasks as they become older (Durak et al., 2019; Rijke et al., 2018).

Several studies have shown the importance of CS education in elementary school (Arfé et al., 2019, 2020; Sáez-López et al., 2016; Tran, 2019). Teaching coding early and increasing access to CS education is important to promote children's CT, increase motivation to learn programming, and foster positive attitudes toward CS education (Arfé et al., 2019, 2020; Tran, 2019). For students who do not have coding help available at home, providing CS education is particularly important (Burgiel et al., 2020). In a coding program, Tran (2019) examined over 200 elementary school students' CT and found that early access to CS lessons resulted in increases in CT skills. Participants showed positive attitude about CS lessons, developed CT and problem-solving skills, and became more interested in future CS studies. Furthermore, the benefits of early access to CS can be expanded to other STEM subjects (Tran, 2019).

4. Discussion

This systematic review selected 66 academic research papers on CS education in the K-12 setting. Although educational interventions have not always been successful in CS education, this review provides strong evidence that CS education promotes the development of students' CT in the K-12 setting while improving students' creative and critical thinking skills. When integrated into other subjects, CS education increases students' motivation, commitment, participation, and interest in the subject matter area (Sáez-López et al., 2019). CT knowledge and skills developed during CS education are likely retained longer (Burgiel et al., 2020; Chou, 2020; Kwon et al., 2021) and transferred to other contexts (Witherspoon et al., 2017, 2018).

In CS education, instructional design, teaching methods, materials, and activities affect the development of CT skills (Jun et al., 2017; Saritepeci, 2020). Many interventions use different approaches and tools to develop students' CT, including robotics, block-based or text-based programming, games, physical and virtual tools, and unplugged activities. While educational levels and grades are related to CT performance, the difference between genders is not conclusive. In our review, there were studies reporting that boys learned and performed better (see Allsop, 2019; Atmatzidou & Demetriadis, 2016) whereas other studies showed no difference (see Noh & Lee, 2020; Taylor & Baek, 2019; Witherspoon et al., 2017). Still other studies showed that girls had higher learning outcomes (see Chiazzese et al., 2018; Durak et al., 2019; Witherspoon et al., 2018).

4.1. Implications for Practice

4.1.1. CS Education Challenges

There have been challenges while implementing CS education in K-12 settings. First, many students struggle with programming concepts or tools (Allsop, 2019; Chang, 2019; Durak et al., 2019; Garneli & Chorianopoulos, 2018). For example, in a study by Chang (2019), elementary school students did not develop more advanced CT concepts such as variables within the Scratch environment. Second, the limited time for educational interventions was mentioned as one of the major barriers to CS education (Arfé et al., 2019; Fagerlund et al., 2020; Witherspoon et al., 2017; Xia & Zhong, 2018). In a previous systematic review, Xia and Zhong (2018) also found that most empirical studies lasted for less than two months. Third, educators found it difficult to effectively manage students, class materials, and various devices while conducting CS instruction (Lockwood & Mooney, 2017; Strawhacker et al., 2018; Tonbuloğlu & Tonbuloğlu, 2019). Lastly, teachers may not have in-depth knowledge about CS concepts and need professional development opportunities for CS education (Jun et al., 2017; Miller, 2019; Witherspoon & Schunn, 2019).

4.1.2. Recommendations for CS Educators

Based on the review of CS education methods, we provide some recommendations for CS educators. It is important to design and develop meaningful CS educational environments, instructional tools, strategies, and activities for CS education (Kert et al., 2019; Kynigos & Grizioti, 2018). These interventions should be appropriate for different age groups and genders (del Olmo-Muñoz et al., 2020; Tran, 2019). Inquiry-oriented curricula with authentic problems may help students promote their CS skills and their real-world application (Chiazzese et al., 2019; Kwon et al., 2021; Richard & Giri, 2019; Shen et al., 2020). Group work also seems to enhance student learning in CS lessons while encouraging communication, collaboration, and problem-solving skills (Allsop, 2019; Atmatzidou & Demetriadis, 2016; Kuo & Hsu, 2020; Lai & Wong, 2022; Pellas & Peroutseas, 2016).

Perhaps the more innovative CS education approaches are those that bridge the gap between the physical and virtual world through the use of virtual coding to manipulate physical objects (Durak et al., 2019; Karaahmetoglu & Korkmaz, 2019; Kert et al., 2019). Approaches that support the learning of coding along with manipulation of robots and physical objects are also likely to be more

motivating and support better problem-solving skills. Similarly, coding tools that allow for both block-based and text-based programming may better support the development of CT (Weintrop & Wilensky, 2018). For younger learners, it may be best to begin with unplugged coding-related CS education activities that are later applied in plugged-in coding activities (Ardito et al., 2020; del Olmo-Muñoz et al., 2020).

Approaches that scaffold students into learning CS are likely to be effective. These practices include clearly introducing programming concepts, introducing the programming tool, allowing free play and exploration by students, challenging students with tasks to complete using the tool, having students test out their solutions, and inviting students to reflect on the experience (Baek et al., 2019; Chou, 2020; Durak, 2020; Merkouris & Chorianopoulos, 2019; Shen et al., 2020; Soleimani et al., 2019). Also, because metacognition such as self-regulation and self-monitoring contributes to students' engagement in CT (Allsop, 2019; Robertson et al., 2020), Chen et al. (2021) suggest that CS education should incorporate metacognitive scaffolding.

4.2. Limitations

There were a few limitations in this study. First, because we decided to review formal, school-oriented CS education in K-12 settings, we excluded many studies involving informal or after-school CS programs. Second, because we targeted scholarly journal articles published in 2016–2020, this review left out studies reported after the year of 2020. Considering that over 72% of the selected papers were published in 2019–2020, we expect more studies have been published since the time frame of this systematic review. Finally, the review focused on the development of K-12 students' CT and did not include studies reporting different perspectives of students, teachers, and administrators.

5. Conclusion

CS education is recognized as one of the critical skills needed for the 21st century, and it is important to provide a high-quality CS education in K-12 schools. While the number of K-12 schools offering CS education has been increasing, there are still disparities in access to CS education experiences (Code.org, CSTA, & ECEP Alliance, 2021). We recommend early access to CS education, various innovative approaches to CS education and appropriate guidance for student learning. Also, there is a need for increased and improved professional development opportunities for CS educators.

CS education and programming activities have increased in K-12 classrooms, yet there have been limited studies on their practices, impacts, and outcomes (Chen et al., 2021; Çınar & Tüzün, 2020; Grover & Pea, 2013; Lye & Koh, 2014). For future research, it will be important to include multiple dimensions of CT skills (Allsop, 2019; Brennan & Resnick, 2012; Tang et al., 2020) and expand our understanding of many aspects of CS education and its impact on student learning in the long run. Furthermore, it will be important to listen to more diverse voices and perspectives to improve student experiences and increase the quality of CS education.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References[1]

*Aksit, O., & Wiebe, E. N. (2020). Exploring force and motion concepts in middle grades using computational modeling: A classroom intervention study. *Journal of Science Education and Technology, 29*(1), 65–82.

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

*Altanis, I., Retalis, S., & Petropoulou, O. (2018). Systematic design and rapid development of motion-based touchless games for enhancing students' thinking skills. *Education Sciences, 8*(1), 18–32.

*Ardito, G., Czerkawski, B., & Scollins, L. (2020). Learning computational thinking together: Effects of gender differences in collaborative middle school robotics program. *TechTrends, 64*(3), 373–387.

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

*Arfé, B., Vardanega, T., & Ronconi, L. (2020). The effects of coding on children's planning and inhibition skills. *Computers & Education, 148*. https://doi.org/10.1016/j.compedu.2020.103807

*Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems, 75*, 661–670.

*Atun, H., & Usta, E. (2019). The effects of programming education planned with TPACK framework on learning outcomes. *Participatory Educational Research, 6*(2), 26–36.

*Baek, Y., Yang, D., & Fan, Y. (2019). Understanding second grader's computational thinking skills in robotics through their individual traits. *Information Discovery and Delivery, 47*(4), 218–228.

Barcelos, T. S., Muñoz-Soto, R., Villarroel, R., Merino, E., & Silveira, I. F. (2018). Mathematics learning through computational thinking activities: A systematic literature review. *Journal of Universal Computer Science, 24*(7), 815–845.

Blikstein, P., & Moghadam, S. H. (2018). *Pre-College Computer Science Education: A Survey of the Field*. Google, LLC. https://goo.gl/gmS1Vm.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In , *1. Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada* (p. 25).

*Burgiel, H., Sadler, P. M., & Sonnert, G. (2020). The association of high school computer science content and pedagogy with students' success in college computer science. *ACM Transactions on Computing Education, 20*(2), 1–21.

[1] *Studies included in this systematic review

*Chang, C. H. (2019). Does the learning of computational thinking concepts interact with the practice of digital curation in children? A preliminary case study. *Journal of Educational Media and Library Sciences, 56*(1), 45–68.
Chen, C.-H., Liu, T.-K., & Huang, K. (2021). Scaffolding vocational high school students' computational thinking with cognitive and metacognitive prompts in learning about programmable logic controllers. *Journal of Research on Technology in Education*, 1–18. https://doi.org/10.1080/15391523.2021.1983894
*Chen, K.-Z., & Chi, H.-H. (2020). Novice young board-game players' experience about computational thinking. *Interactive Learning Environments*. https://doi.org/10.1080/10494820.2020.1722712
*Chevalier, M., Giang, C., Piatti, A., & Mondada, F. (2020). Fostering computational thinking through educational robotics: a model for creative computational problem solving. *International Journal of STEM Education, 7*(39). https://doi.org/10.1186/s40594-020-00238-z
*Chiazzese, G., Arrigo, M., Chifari, A., Lonati, V., & Tosto, C. (2019). Educational robotics in primary school: Measuring the development of computational thinking skills with the Bebras tasks. *Informatics, 6*(4), 43. https://doi.org/10.3390/informatics6040043
*Chiazzese, G., Fulantelli, G., Pipitone, V., & Taibi, D. (2018). Engaging primary school children in computational thinking: Designing and developing videogames. *Education in the Knowledge Society, 19*(2), 63–81.
*Chou, P. (2020). Using ScratchJr to foster young children's computational thinking competence: A case study in a third-grade computer class. *Journal of Educational Computing Research, 58*(3), 570–595.
*Çınar, M., & Tüzün, H. (2020). Comparison of object oriented and robot programming activities: The effects of programming modality on student achievement, abstraction, problem solving, and motivation. *Journal of Computer Assisted Learning, 37*(2), 370–386.
Code.org, CSTA, & ECEP Alliance. (2021). *2021 State of computer science education: Accelerating action through advocacy*. Retrieved from https://advocacy.code.org/2021_state_of_cs.pdf.
*del Olmo-Muñoz, J., Cózar-Gutiérrez, R., & González-Calero, J. A. (2020). Computational thinking through unplugged activities in early years of primary education. *Computers and Education, 150*. https://doi.org/10.1016/j.compedu.2020.103832
*Deng, W., Pi, Z., Lei, W., Zhou, Q., & Zhang, W. (2020). Pencil Code improves learners' computational thinking and computer learning attitude. *Computer Application in Engineering Education, 28*(1), 90–104.
**Durak, H. Y. (2020). The effects of using different tools in programming teaching of secondary school students on engagement, computational thinking and reflective thinking skills for problem solving. *Technology, Knowledge and Learning, 25*(1), 179–195.
*Durak, H. Y., Yilmaz, F. G. K., & Bartin, R. Y. (2019). Computational thinking, programming self-efficacy, problem solving and experiences in the programming process conducted with robotic activities. *Contemporary Educational Technology, 10*(2), 173–197.
*Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2020). Assessing 4th grade students' computational thinking through Scratch programming projects. *Informatics in Education, 19*(4), 611–640.
Fidai, A., Capraro, M. M., & Capraro, R. M. (2020). Scratch"-ing computational thinking with Arduino: A meta-analysis. *Thinking Skills and Creativity, 38*. https://doi.org/10.1016/j.tsc.2020.100726
*Fronza, I., Ioini, N. E., & Corral, L. (2017). Teaching computational thinking using agile software engineering methods: A framework for middle schools. *ACM Transactions on Computing Education, 17*(4), 1–28.
*Garneli, V., & Chorianopoulos, K. (2018). Programming video games and simulations in science education: Exploring computational thinking through code analysis. *Interactive Learning Environments, 26*(3), 386–401.
*Gillott, L., Joyce-Gibbons, A., & Hidson, E. (2020). Exploring and comparing computational thinking skills in students who take GCSE Computer Science and those who do not. *International Journal of Computer Science Education in Schools, 3*(4), 3–22.
*Gomes, T. C. S., Pontual Falcão, T., & Tedesco, P. C. A. R (2018). Exploring an approach based on digital games for teaching programming concepts to young children. *International Journal of Child-Computer Interaction, 16*, 77–84.
Gough, D., Oliver, S., & Thomas, J. (2012). *An introduction to systematic reviews*. London: Sage.
Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43.
*Günbatar, M. S. (2020). Computational thinking skills, programming self-efficacies and programming attitudes of the students. *International Journal of Computer Science Education in Schools, 4*(2).
*Hsiao, H. S., Lin, Y. W., Lin, K. Y., Lin, C. Y., Chen, J. H., & Chen, J. C. (2019). Using robot-based practices to develop an activity that incorporated the 6E model to improve elementary school students' learning performances. *Interactive Learning Environments*. https://doi.org/10.1080/10494820.2019.1636090
Higgins, J. P. T., Thomas, J., Chandler, J., Cumpston, M., Li, T., Page, M. J., & Welch, V. A. (Eds.). (2021). *Cochrane Handbook for Systematic Reviews of Interventions* (version 6.2). Cochrane. Available from www.training.cochrane.org/handbook.
Hur, J. W., Andrzejewski, C. E., & Marghitu, D. (2017). Girls and computer science: Experiences, perceptions, and career aspirations. *Computer Science Education, 27*(2), 100–120.
*Jun, S., Han, S., & Kim, S. (2017). Effect of design-based learning on improving computational thinking. *Behaviour & Information Technology, 36*(1), 43–53.
Kafai, Y. B. (2006). Playing and making games for learning: Instructionist and constructionist perspectives for game studies. *Games and Culture, 1*(1), 36–40.
Kakavas, P., & Ugolini, F. C. (2019). Computational thinking in primary education: A systematic literature review. *Research on Education and Media, 11*(2), 64–94.
*Karaahmetoglu, K., & Korkmaz, Ö. (2019). The effect of project-based arduino educational robot applications on students' computational thinking skills and their perception of basic STEM skills levels. *Participatory Educational Research, 6*(2), 1–14.
*Kert, S. B., Erkoç, M. F., & Yeni, S. (2020). The effect of robotics on six graders' academic achievement, computational thinking skills and conceptual knowledge levels. *Thinking Skills and Creativity, 38*. https://doi.org/10.1016/j.tsc.2020.100714
*Kert, S. B., Kalelioğlu, F., & Gülbahar, Y. (2019). A holistic approach for computer science education in secondary schools. *Informatics in Education, 18*(1), 131–150.
*Kuo, W.-C., & Hsu, T.-C. (2020). Learning computational thinking without a computer: How computational participation happens in a computational thinking board game. *Asia-Pacific Education Researcher, 29*(1), 67–83.
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 & Development, 69*(5), 2761–2787.
*Kynigos, C., & Grizioti, M. (2018). Programming approaches to computational thinking: Integrating turtle geometry, dynamic manipulation and 3D space. *Informatics in Education, 17*(2), 321–340.
Lai, X., & Wong, G. K.-W. (2022). Collaborative versus individual problem solving in computational thinking through programming: A meta-analysis. *British Journal of Educational Technology, 53*(1), 150–170.
*Lockwood, J., & Mooney, A. (2017). A pilot study investigating the introduction of a computer-science course at second level focusing on computational thinking. *Irish Journal of Education, 43*, 108–127.
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.
*Merkouris, A., & Chorianopoulos, K. (2019). Programming embodied interactions with a remotely controlled educational robot. *ACM Transactions on Computing Education, 19*(4), 1–19.
*Miller, J. (2019). STEM education in the primary years to support mathematical thinking: using coding to identify mathematical structures and patterns. *ZDM, 51*(6), 915–927.
**Min, S. H., & Kim, M. K. (2020). Developing children's computational thinking through physical computing lessons. *International Electronic Journal of Elementary Education, 13*(2), 183–198.
*Moore, T. J., Brophy, S. P., Tank, K. M., Lopez, R. D., Johnston, A. C., Hynes, M. M., & Gajdzik, E. (2020). Multiple representations in computational thinking tasks: a clinical study of second-grade students. *Journal of Science Education and Technology, 29*(1), 19–34.
*Moschella, M., & Basso, D. (2020). Computational thinking, spatial and logical skills. An investigation at primary school. *Journal of Theories and Research in Education, 15*(2), 69–89.

*Noh, J., & Lee, J. (2020). Effects of robotics programming on the computational thinking and creativity of elementary school students. *Educational Technology Research and Development, 68*(1), 463–484.
Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., …, & Moher, D. (2021). The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *Systematic Reviews, 10*(89). https://doi.org/10.1186/s13643-021-01626-4
Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books.
Papert, S. (1993). *The Children's Machine: Rethinking School in the Age of the Computer*. Basic Books.
Papert, S., & Harel, I. (1991). Situating constructionism. *Constructionism*. Ablex Publishing Corporation.
*Pellas, N., & Peroutseas, E. (2016). Gaming in Second Life via Scratch4SL: Engaging high school students in programming courses. *Journal of Educational Computing Research, 54*(1), 108–143.
*Price, C. B., & Price-Mohr, R. M. (2018). An evaluation of primary school children coding using a text-based language (Java). *Computers in the Schools, 35*(4), 284–301.
*Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional science, 45*(5), 583–602.
*Richard, G. T., & Giri, S. (2019). Digital and physical fabrication as multimodal learning: Understanding youth computational thinking when making integrated systems through bidirectionally responsive design. *ACM Transactions on Computing Education, 19*(3), 1–35.
*Rijke, W. J., Bollen, L., Eysink, T. H., & Tolboom, J. L. (2018). Computational thinking in primary school: An examination of abstraction and decomposition in different age groups. *Informatics in Education, 17*(1), 77–92.
*Robertson, J., Gray, S., Martin, T., & Booth, J. (2020). The relationship between executive functions and computational thinking. *International Journal of Computer Science Education in Schools, 3*(4).
*Rodríguez-Martínez, J. A., González-Calero, J. A., & Sáez-López, J. M. (2020). Computational thinking and mathematics using Scratch: An experiment with sixth-grade students. *Interactive Learning Environments, 28*(3), 316–327.
*Sáez, L., J, M., Román, G. M., & Cano, E (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education, 97*, 129–141.
*Sáez-López, J. M., Sevillano-García, M. L., & Vazquez-Cano, E. (2019). The effect of programming on primary school students' mathematical and scientific understanding: educational use of mBot. *Educational Technology Research and Development, 67*(6), 1405–1425.
Santo, R., Vogel, S., & Ching, D. (2019). *CS for what? Diverse visions of computer science education in practice*. New York, NY: CSforALL.
*Saritepeci, M. (2020). Developing computational thinking skills of high school Students: Design-based learning activities and programming tasks. *The Asia-Pacific Education Researcher, 29*(1), 35–54.
*Shen, J., Chen, G., Barth-Cohen, L., Jiang, S., & Eltoukhy, M. (2020). Connecting computational thinking in everyday reasoning and programming for elementary school students. *Journal of Research on Technology in Education*, 1–21. https://doi.org/10.1080/15391523.2020.1834474
*Soleimani, A., Herro, D., & Green, K. E. (2019). CyberPLAYce–A tangible, interactive learning tool fostering children's computational thinking through storytelling. *International Journal of Child-Computer Interaction, 20*, 9–23.
**Strawhacker, A., Lee, M., & Bers, M. (2018). Teaching tools, teachers' rules: exploring the impact of teaching styles on young children's programming knowledge in ScratchJr. *International Journal of Technology and Design Education, 28*(2), 347–376.
Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education, 148*. https://doi.org/10.1016/j.compedu.2019.103798
*Taylor, K., & Baek, Y. (2019). Grouping matters in computational robotic activities. *Computers in Human Behavior, 93*, 99–105.
*Tonbuloglu, B., & Tonbuloglu, I. (2019). The effect of unplugged coding activities on computational thinking skills of middle school students. *Informatics in Education, 18*(2), 403–426.
*Tran, Y. (2019). Computational thinking equity in elementary classrooms: What third-grade students know and can do. *Journal of Educational Computing Research, 57*(1), 3–31.
*Türker, P. M., & Pala, F. K. (2020). A study on students' computational thinking skills and self-efficacy of block-based programming. *i-managers Journal on School Educational Technology, 15*(3), 18–31.
*Uşengül, L., & Bahçeci, F. (2020). The Effect of LEGO WeDo 2.0 Education on Academic Achievement and Attitudes and Computational Thinking Skills of Learners toward Science. *World Journal of Education, 10*(4), 83–93.
*Weintrop, D., & Wilensky, U. (2018). How block-based, text-based, and hybrid block/text modalities shape novice programming practices. *International Journal of Child-Computer Interaction, 17*, 83–92.
*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*. https://doi.org/10.1016/j.compedu.2019.103646
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.
*Witherspoon, E. B., Higashi, R. M., Schunn, C. D., Baehr, E. C., & Shoop, R. (2017). Developing computational thinking through a virtual robotics programming curriculum. *ACM Transactions on Computing Education, 18*(1). https://doi.org/10.1145/3104982
*Witherspoon, E. B., & Schunn, C. D. (2019). Teachers' goals predict computational thinking gains in robotics. *Information and Learning Sciences, 120*(5/6), 308–326.
*Witherspoon, E. B., Schunn, C. D., Higashi, R. M., & Shoop, R. (2018). Attending to structural programming features predicts differences in learning and motivation. *Journal of Computer Assisted Learning, 34*(2), 115–128.
*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.
Xia, L., & Zhong, B. (2018). A systematic review on teaching and learning robotics content knowledge in K-12. *Computers & Education, 127*, 267–282.
Zagami, J., Boden, M., Keane, T., Moreton, B., & Schulz, K. (2015). Girls and computing: Female participation in computing in schools. *Australian Educational Computing, 30*(2).
*Zhao, W., & Shute, V. J. (2019). Can playing a video game foster computational thinking skills? *Computers & Education, 141*. https://doi.org/10.1016/j.compedu.2019.103633