International Journal of Innovative Research in Education

Volume 9, Issue 2, (2022) 142-155

www.ijire.eu

Exploring the compelling rationale to include programming in the K-12 curriculum

Iman H. Aljameel[*], College of Education King Saud University, Riyadh, Saudi Arabia

Suggested Citation:

Aljameel, I. H. (2022). Exploring the compelling rationale to include programming in the K-12 curriculum. *International Journal of Innovative Research in Education*. *9*(2), 142-155 https://doi.org/10.18844/ijire.v9i2.8460

Received from May 13, 2022; revised from September 20, 2022; accepted from November 10, 2022.

Selection and peer review under the responsibility of Assoc. Prof. Dr. Zehra Ozcinar, Ataturk Teacher Training Academy, Cyprus.

Abstract

This study aims to conduct a literature review to explore the justification for including programming and coding in K–12 curricula. Additionally, it considers the value of teaching computational thinking and programming principles, pedagogical strategies, and the advantages and drawbacks of teaching programming to young students. Most studies acknowledge that programming courses effectively motivate and engage students and achieve learning objectives. However, research reveals that teachers lack the skills to teach programming, and even seasoned educators may find it challenging to develop and implement the necessary pedagogical strategies. Researchers recommend using frameworks based on sociocultural theories of learning, increased resources and training, student-centered instruction, and lowering teacher workload, which has led various nations to re-evaluate their curricula. Finally, the review concludes by suggesting the need for more research on how teaching and learning programming affects learning in other academic disciplines.

Keywords: Computer education, computational thinking, pedagogical strategies, programming, K–12 curricula;

[*] ADDRESS OF CORRESPONDENCE: Iman H. Aljameel, College of Education King Saud University, Riyadh, Saudi Arabia,

Email address: eman.h.jameel@gmail.com

1. Introduction

It is commonly accepted that learning programming is difficult (Luxton-Reilly, 2016; Tsukamoto et al., 2016). School children who are novice programmers struggle with typical programming tasks such as projecting program output, determining the proper order of commands, or designing a simple program to address a problem (Luxton-Reilly, 2016). Moreover, teaching programming to such learners is challenging as the majority of teachers lack sufficient training (Gal-Ezer & Stephenson, 2014; Neil et al., 2014; Sentance et al., 2019; Szabo et al., 2019).

1.1. Purpose of study

In this context, it is essential to understand why schools are implementing programming and coding in the K-12 curriculum. This necessity warranted a comprehensive review of current literature to explore the rationale for introducing programming into the K-12 curriculum. The research question guiding this literature review is:

What is the compelling rationale for including programming and coding in the K-12 curriculum?

The key objectives of this review paper are to Examine the significance of introducing programming in the K-12 curriculum; Examine the significance of teaching programming principles to K-12 students; Explore the advantages and disadvantages of teaching programming in K-12; Explore the lessons learned from different countries that have included programming in K-12 curriculum; Investigate the changes needed in the educational context to teach programming.

2. Materials and methods

To answer the research question and to meet the objectives of this paper, a comprehensive review of introductory-programming literature was carried out. The search terms used were 'programming', 'introductory', 'K-12', 'curriculum', 'schools', 'programming principles', 'pros and cons', and 'teaching programming'. The databases included EBSCOhost, Eric, Sage, Taylor & Francis, ACM Full-Text Collection, IEEE Xplore, Science Direct, Springer Link, Scopus database as well as Google Scholar. The screening and eligibility criteria for selecting, including, and excluding articles were:

1. Including articles from peer-reviewed journals, conferences, and working papers
2. Including articles relevant to the topic of introducing programming in K-12
3. Excluding articles that were not in English
4. Excluding articles that did not explicitly discuss and evaluate introductory programming in K-12
5. Including quantitative and qualitative studies

3. Results

3.1. Introducing programming in K-12 curricula

Computer programming in K-12 has become not just necessary, but mandatory, and countries around the world are including programming and coding as part of their curricula (Dong et al., 2019; Floyd, 2019; Moreno-León et al., 2016; Scherer et al., 2020; Sentance et al., 2019; Szabo et al., 2019). The rationale for the increased interest in computer programming is the growing need for specialists with coding abilities, as well as the need for individuals with a basic understanding of computer science in the digitalised society of the future (Balanskat & Engelhardt, 2015; Lindberg et al., 2019). Programming is being introduced into K-12 under the assumption that it will help students acquire 21st-century skills which include creative thinking, self-efficacy, problem-solving, critical thinking, communication, and cooperation skills (Ananiadou & Claro, 2009; Ceker & Ozdamli, 2016; Duncan & Bell, 2015; Grover & Pea, 2013; Psycharis & Kallia, 2017; Scherer et al., 2020).

The explosion of innovative technologies, such as extended technologies which comprise virtual reality, augmented reality, and mixed reality, and the extensive use of mobile devices over the past decade has increased the need for the rapid integration of programming (Sterling, 2016; Wohlgenannt et al., 2020). This is because a skilled workforce specialised in programming languages (for example, HTML, JavaScript, C#, ReactJS, Scratch, and Python, to name a few) is needed to effectively participate in a world full of digital objects, and, in particular, creating the apps that run on mobile devices (Moreno-León et al., 2016).

The new solutions provided by technology also contribute to the development of such skills, notably, teaching programming or coding training, which has been considered to have the capacity to compel or enact many changes in education at various stages (Bers et al., 2014; Giordano & Maiorana, 2015). The ability to program is the foundation of coding, and the foundation of programming is the algorithm; these skills are built on problem-solving abilities, one of the essential tools that enable students to succeed in real life (Tondeur et al., 2019; Trilling & Fadel, 2009). Although coding and programming are synonymously or interchangeably used in the literature to refer to the written instructions utilised to control a computer (Lloyd & Chandra, 2020), 'coding in the curriculum' appears to be the desired alternative to programming at present (Sterling, 2016, p. 79).

In recent years, coding training has gained importance for use in visual programming languages (for example, Alice, Kodu, Code.org, and Scratch), to improve students 'computational thinking skills (Babori et al., 2016; Curzon, 2013; Resnick et al., 2010). It is strongly suggested that students receive algorithm training prior to receiving coding training since what is vital is to be able to visualise and sequentially follow the procedures that establish the foundation of the program one by one (Babori et al., 2016).

One of the crucial skills is computational thinking, as it is considered valuable in STEM-related academic disciplines, such as mathematics and technology (Campbell & Heller, 2019; Scherer et al., 2020). Consequently, there has been an extraordinary effort to rekindle interest in STEM education, and much of the debate around technology, or the 'T 'in STEM education, has concerned whether programming and coding should become a required component of the school curriculum (Sterling, 2016). It is for these reasons that a plethora of empirical research has focused on this topic, to attempt to explain the factual relationship between teaching programming or coding and computational thinking skills (Bers et al., 2014; Campbell & Heller, 2019; Dong et al., 2019; Pears et al., 2017; Resnick et al., 2010). Lindberg et al. (2019) also point out an increasing global trend in learning computational thinking in education, which reveals the importance of embedding programming within the school curriculum.

3.2. The significance of teaching programming principles

3.2.1. The principles

The principles of a programming language are a set of rules and norms governed by communicating instructions (high-level instruction or assembly-level instruction) to a machine or, particularly, a computer (Balanskat & Engelhardt, 2015). A programming language is a vocabulary and a set of grammatical rules programmers use to give instructions so that computers can execute or perform specific tasks (Chen, 2020). Programming languages are classified as machine-level (used by computers and consists of only two symbols 1 & 0), assembly-level (for example MIPS, INTEL64/AMD64, ARM64, THUMPs, 6502), or high-level languages (for example, languages, such as BASIC, C, C++, COBOL, Java, FORTRAN, Ada, Pascal). Although many languages share similarities, each has its own syntax (Chen, 2020). Syntax is the grammar or the rules that define the structure of a language, while semantics assigns computational meaning to valid strings in a programming language (Harper, 2016). Without syntax, the meaning or semantics of a language is nearly impossible to understand (Harper, 2016).

3.2.2. The tools

A major distinction between instructional programming tools is whether they offer text-based (for example, Python, Java and JavaScript, C++) or block-based (for example, Scratch, Blockly, Snap) visual programming (Lindberg et al., 2019). Block-based programming, an introductory system, is usually considered to be easier for beginners to simulate coded language by using colorful forms in a canvas work area which can be dragged and dropped with a mouse instead of typing to form a sequenced program (Lindberg et al., 2019; Moors et al., 2018). Text-based programming tools—such as Cobol, FORTRAN, Python, Java, Action Script, Arduino Integrated Development Environments, and Processing—are used in professional contexts (Lindberg et al., 2019; Sentance et al., 2019). As students develop, they can use blocks to create a program and understand the complexity of text-based language. However, students must learn the principles of programming, such as events, sequencing, loops/iteration, functions, and variables, throughout K-12 teaching (Moors et al., 2018).

3.2.3. The significance of computational thinking

Today, computational thinking is emphasised in K-12 education because it is deemed vital to computer programming and problem solving, abilities that enable students to be creators of digital material rather than just users. Computational thinking is a problem-solving process (Pears et al., 2017) that includes: decomposition, pattern recognition, algorithm design, and abstraction. Decomposition is a method of breaking down a task or problem into smaller steps or components. Pattern recognition entails breaking down a problem into its constituent parts, for example, to identify the pattern behind raw data: 2, 5, 8, 11, 14, 17, 20, 23, 26, and so on. Although number sets are easy, identifying patterns involving shapes, sounds or images might be difficult (Pears et al., 2017). Abstraction is a method for managing computer-system complexity. It works by establishing a level of complexity at which a user interacts with the system, hiding more complex features below the present level (Pears et al., 2017). Algorithms involve independent instructions or writing the steps in a pseudo-code that a computer can use to solve the problem (Pears et al., 2017).

3.2.4. Strategies for teaching principles of programming

Denning (2003) argues that teaching approaches in computer science should include programming, modeling, validation, innovating, and applying. Teaching also entails the application of principles, for example, clarity, efficiency, reliability, and security, as well as the function of mechanics such as communication, computation, automation, and reuse. Thus, programming or coding in education is not a new concept. Rather, it has recently gained increased popularity as a result of newly designed programming tools that are easier for young students to understand. Sentance et al. (2019) developed a strategy by drawing on Vygotsky's sociocultural theory of human learning, to guide the teaching of the principles of programming. The authors focused on mediation or the use of language to connect students, peers, and teachers; scaffolding learning, and the use of models or exemplars to facilitate learning.

3.3. The benefits and disadvantages of teaching programming

The successful integration of programming in K-12 classrooms may open up new doors for significant opportunities in topics other than computer science. Programming, for example, could serve as a tool for knowledge-building and assisting students 'transition from passive consumers to active creators (Psycharis & Kallia, 2017; Tundjungsari, 2016).

One study which examined the benefits of introducing programming in K-12 education is that of Wilson and Moffat (2010). Overall, the study examined the impact of utilising Scratch to teach rudimentary programming to young children. When the researchers assessed the children's cognitive progress, they discovered that the students could quickly learn to write basic programs and enjoyed the exercise. The researchers also interviewed the teacher, who confirmed that several of the students exceeded his expectations. Despite the minimal cognitive gain, the fundamental advantage of Scratch in this study appears to be that its enjoyability makes learning how to program a pleasant experience, as opposed to the frustration and anxiety that so frequently seems to characterise the typical learning process.

The benefits of programming have long been acknowledged in several studies (Ananiadou & Claro, 2009; Ceker & Ozdamli, 2016; Duncan & Bell, 2015; Grover & Pea, 2013). These include developing students 'digital literacy; improving resourcefulness and creativity; enabling both outcomes- and process-oriented thinking; motivating learners; internalising knowledge with long-term memory use; acquiring problem-solving, spatial thinking, and critical thinking skills; as well as learning to work as part of a team (Akpınar & Altun, 2014; Framework for 21st Century Learning, 2017).

One of the reasons for the preference for programming is that teachers consider it to be a fun activity that can engage students (Humble et al., 2020). Recent studies have confirmed these findings. For instance, teachers have integrated programming with gamification to maximise student motivation and engagement by making them feel excited and enthusiastic (Lindberg et al., 2018; Pontes et al., 2019).

Recent research evidence is consistent with previous findings (Wilson & Moffat, 2010) that suggest that the cognitive benefits of learning to program are promising (Scherer et al., 2020, 2021). Through their research, Scherer et al. (2020, 2021) also argue that students' coding skills, as well as other skills such as creative thinking and mathematical problem-solving, can be transferred to other disciplines or domains.

The incorporation of programming into the K-12 curriculum has also introduced new challenges, such as limited access to technology and the internet, a lack of enthusiasm, and a lack of digital competence (Tsukamoto et al., 2016). However, multiple studies have found that textual programming is more complex and difficult to master for new programmers than block programming (Lindberg et al., 2019; Moors et al., 2018; Sentance et al., 2019). Textual programming tools can be difficult to learn and teach, even with well-chosen teaching and learning resources, because many individuals struggle with the syntax of textual programming (Sentance et al., 2019). When teaching programming to elementary-school pupils, textual programming did not appeal to students when compared to block programming in terms of motivation and confidence (Tsukamoto et al., 2016). This implies that text-based programming may be less suitable for teaching coding to younger students. Therefore, researchers have suggested the use of unplugged coding or learning programming skills without the use of technology.

The aforementioned challenges are also why teachers tend to compare textual programming tools to block programming tools concerning opportunities and challenges (Lindberg et al., 2019). The second reason is that unplugged programming is perceived as fun and easy but might be limited to younger students (Humble et al., 2020). In contrast to these claims, and that of Tsukamoto et al. (2016) in particular, Moors et al. (2018) found that block-based programming may reduce children's confidence as a result of the sudden shift toward syntax-heavy languages.

Results related to the pros and cons of block- and text-based programming suggest that teaching students about the program may be a complex process (Szabo et al., 2019). Nevertheless, recent studies have discovered low dropout rates among computing students (Watson & Li, 2014), and it has been suggested that the difficulties encountered by beginners may be the result of unrealistic expectations rather than inherent subject complexity (Luxton-Reilly, 2016), implying the need to begin teaching programming as early as possible.

Considering the results of previous studies, we can assume that it is difficult to design and conduct introductory programming classes, especially when K-12 students with weak or no literacy abilities are involved. This difficulty is exacerbated by a lack of recognition of computing as a field in K-12 curricula, as well as the fact that computing modules might be interdisciplinary and taught by various teachers (Gal-Ezer & Stephenson, 2014; Neil et al., 2014). Furthermore, teachers may lack the requisite expertise as well as access to adequate training and professional development (Neil et al., 2014). Another issue is that schools may have minimal infrastructure available for teachers (Gal-Ezer & Stephenson, 2014). Sentance et al. (2019) support these findings by suggesting that even experienced teachers with little or no experience in teaching programming may struggle to establish and implement the pedagogical strategies required to teach this subject.

Furthermore, when learning to code, instructors must deal with time constraints (Humble et al., 2020; Tundjungsari, 2016). Time constraints are a recurring finding and have also been reported in various countries (Hijón-Neira et al., 2017). Researchers assert that ignoring contextual factors, such as the above-mentioned, leads to the failure of the reform intended (Alshumaimeri, 2022; Fullan, 2007; Wedell & Alshumaimeri, 2014; Wedell & Malderez, 2013).

Several potential objections to introducing programming in the K-12 curriculum have been refuted by Sterling (2016). The first objection is that teaching programming or coding lacks a suitable pedagogical foundation. Sterling claims that the pedagogy is clearly defined and that most researchers agree that Scratch works effectively (Babori et al., 2016; Curzon, 2013; Resnick et al., 2010). However, the issue is using a block-based programming language, avoiding small syntax concerns, and wanting to see the results of executing programs rapidly. Sterling's second argument is that there is a misconception that there is a lack of evidence base establishing programming/coding as beneficial. However, research has provided evidence of the benefits of programming/coding (Scherer et al., 2021; Wilson & Moffat, 2010). Sterling (2016) contends that the demand for programming is motivated mostly by vested interests. However, advocating for programming in the curriculum is not the same as advocating for computers in schools. The drivers for programming are public interest groups as well as vendors, and there is no dearth of quality, free and open-source resources. Nonetheless, there is tremendous potential for research on discerning conflicting product claims and educational settings for programming.

3.4. Lessons learned from different countries that taught programming in K-12

In most countries (for example, Australia, the United States, and the UK), programming is an end in itself, as the emphasis is on the likely benefits of coding and the skills required for entering the labor market (Lloyd & Chandra, 2020). Other governments are considering programming from a different perspective, as they pay more attention to the educational impact of learning to program at an early age, using coding as a medium to learn various disciplines (Moreno-León et al., 2016). However, teachers lack training in teaching coding and computational thinking (Lloyd & Chandra, 2020). The study of Mouza et al. (2017, p. 61) from the United States revealed that teachers only have a 'surface understanding' of programming concepts and were unable to effectively transfer knowledge into practice. Research suggests that, shortly, there will be a dearth of well-trained teachers with relevant technological and pedagogical skills (Cabrera, 2019; Yadav et al., 2016). The importance of teaching programming implies that this is an urgent issue for teacher education (Yadav et al., 2016).

According to evidence from Sweden, teachers are unsure of what programming entails in terms of practice and concepts, as well as how programming knowledge could be evaluated (Vinnervik, 2022). Teachers face various hurdles during the transition process from block-based to text-based programming, which include their programming skills, the quality of course materials, and the realisation that their students lack prior programming knowledge. Furthermore, findings suggest that technology teachers tended to function in isolation and that there was a lack of collaboration. As a result, the expected cooperation between various disciplines was absent when preparing lessons for programming courses (Vinnervik, 2022).

In Sweden, a new curriculum for K-9 education approved by the government in 2018 emphasising interdisciplinary digital competence and programming was introduced (Heintz et al., 2017), although this lacked concrete guidelines on how programming should be involved in the mathematics and technology curricula (Humble et al., 2020). As identified in some earlier studies (Mouza et al., 2017; Yadav et al., 2016), most teachers found this transition demanding, as they lacked technological and pedagogical knowledge (Humble & Mozelius, 2019).

Training is an issue that has been emphasised in studies from all over the world. Just as in the case of Australia, the United States, and the UK, research on schools in Sweden (Heintz et al., 2017), Finland, and Lithuania (Pears et al., 2017) have also found that teachers 'in-service professional development and training were important to the success of integrating introductory programming into the curriculum. Likewise, teachers in Japan lacked confidence in introducing programming in school programs due to a lack of technological and pedagogical knowledge regarding the use of technology, lack of optimism, preference for traditional teaching methods, and teacher workload (Ohashi, 2017).

Concluding that teachers were not technically prepared or were not enthusiastic about the planned introduction of programming courses, several researchers have proposed solutions and frameworks. Ohashi advocates for increased resources and training, student-centered instruction, and lower teacher workloads. Webb et al. (2016) examined curriculum developments in several countries and discovered that teacher professional development is crucial for facilitating curriculum reform. This finding has been reiterated by many researchers (Aljameel, 2022; Graves, 2021; Soto, 2018; Wedell & Alshumaimeri, 2014; Wedell & Grassick, 2018).

In response to calls from researchers, as many as 16 countries in the European Union have re-evaluated their curriculum and revised it or are in the process of revising it so that their curricula include programming training, while the USA, Turkey and South Korea have started implementing initiatives (Uzunboylu et al., 2017). The aim is to teach programming or writing codes as early as possible so that the children will be in a position to design their projects and applications in the future (Demirer & Sak, 2016). However, Sterling (2016) was critical of the federal government in Florida (USA) for prioritising computer coding at the expense of foreign languages, and for failing to support science and mathematics as a better place for coding instruction in the school curriculum. Sterling considers this an attack on people's language and culture.

In the United Kingdom, Sentance et al. (2019) developed the PRIMM framework, which is based on Vygotsky's concepts of zone of proximal development (ZPD), language, more knowledgeable others (MKO), and scaffolding. PRIMM is a method for organising programming classes and activities that consists of five stages: predict, run, investigate, modify, and make. These phases can be utilised in lesson and activity preparation and are intended to help learners at all stages of learning programming in K-12 education.

3.4.1. Mediation through language

In the context of classroom learning, Vygotsky asserts that a child's development within a ZPD comprises social contact, discourse, and mediated engagement between learners and their teachers (Vygotsky, 1978). This can include instructors or peers, but it can also include artifacts such as class diagrams, data models, printed documents or scripts, books, wall displays, and online learning environments (Sentance et al., 2019). When understanding how programs work, students should be encouraged to share their findings through the social construction of knowledge. This can be accomplished through pair programming or collaborative tasks such as discussing parts of program code to determine their function. Teaching should promote concentrated discussion of programming constructs and concepts.

3.4.2. Moving learning from the social plane to the cognitive plane

Programming activities should be carefully scaffolded so that they are within the student's ZPD. Using starter programs and teaching code reading implies that the program code first exists on the social plane before it is understood internally. As the understanding deepens, more complicated programming assignments requiring independent problem-solving and creativity will become attainable.

3.4.3. The position of the MKO in the ZPD

There are several tools in programming education that provide scaffolding, such as block-based environments that prevent students from making syntax errors. Teachers, as MKO, must demonstrate (using a model or exemplars) how to solve an issue for students. When students collaborate, one peer can be designated as the MKO. In Vygotsky's terminology, the content (materials) and the framework (structure) are mediating activities, but they should be tailored to be within the learners 'ZPD. This necessitates a thorough understanding of the development and which topics are easier or more difficult for children to grasp.

It is evident from the literature that there is a worldwide push to teach introductory programming and several nations are taking the initiative to incorporate it into the K-12 curriculum. Much can be learned from these practices.

3.5. Changes needed for reshaping K-12 education by incorporating introductory programming

Indeed, in many of the world's education systems, teacher training remains a barrier to developing the necessary skills and confidence levels for the effective incorporation of introductory programming in K-12 education. Over the past two decades, numerous teaching methods have been employed in programming education, including developing algorithms to solve problems and code walkthroughs; code debugging which enables students to examine a given code in greater detail by spotting and resolving errors; and lecture-note reconstruction, metacognition, collaborative pair-programming learning, and active learning (Attard & Busuttil, 2020; Sentance & Csizmadia, 2017).

Attard and Busuttil (2020) provide valuable suggestions for introducing programming and assert that all educators have the responsibility to provide pupils with the finest learning opportunities available since education must change to keep up with the times. Schools and policymakers must exert greater effort in the area of programming education. To better understand the various and efficient teaching strategies for programming, more research is required. Furthermore, students should be given opportunities to solve a variety of problems, work as a team, and, finally, program. There is a need to broaden the scope of programming instead of focusing on the study of particular languages. Students need their programming education to remain current, and new and emerging technologies must be used in programming lessons.

It is evident from the suggestions of Attard and Busuttil (2020) that teachers need support. According to Humble et al. (2020), teachers must be provided with the time and resources they need to study and integrate programming into their lesson plans. To take advantage of the opportunities in both languages, teachers should be allowed to learn both block- and text-based programming. This might also help students prepare more thoroughly for the opportunities and difficulties that come with integrating programming into the classroom (Humble et al., 2020).

To address the challenges facing teachers, school systems should introduce continuous professional development, certification programs, and computer-science credentials at the pre-service teacher level (Campbell & Heller, 2019; Code.org, 2017; CSTA, 2013). Certification programs measure teachers' knowledge of their subject matter while also highlighting their distinctive qualifications (Code.org, 2017; CSTA, 2013). This then encourages educators to pursue training, creating a loop that produces stronger educators of computer science. In any innovation planning, policymakers need to develop an understanding of the complex, interdependent contextual factors related to the education system as a whole, and other factors related to recourses, schools, teachers, and learners (Wedell, 2022). More specifically, policymakers need to understand the pedagogic implications of a new curriculum, its classroom practices, and the roles of different implementers (e.g., teachers, inspectors, and principals). Understanding the context, the roles of the implementers and their needs, new curriculum principles, and pedagogical implications will help policymakers design implementation plans that provide the required support and are designed to overcome emerging innovation challenges (Alshumaimeri, 2022; Wedell, 2022).

4. Conclusion

This literature review elucidated the initiatives taken by schools and educators worldwide to incorporate programming into K–12 curricula. Overall, the evidence supporting the cognitive benefits of learning to code is promising. As with introductory programming courses in K-12 education, much of the work has focused on the difficulties in incorporating programming within the curriculum. Most studies have acknowledged the effectiveness of programming courses in engaging and motivating students and achieving learning outcomes.

The evidence related to teachers 'challenges centers on lack of training. This review discovered a considerable number of studies that identify a need to support teachers in teaching the programming curriculum and in overcoming other challenges. Studies have recommended instructional strategies for introducing programming in schools. However, further research is required to establish the relative effectiveness of these approaches.

Additionally, this review found several gaps in the literature. Compared to computers or computational thinking in K–12, there is a dearth of literature on programming instruction and learning. Moreover, there is a lack of research examining how studying programming affects learning other courses. These gaps in the literature point to the need for more thorough studies to determine the efficacy of teaching techniques for K–12 programming.

References

Akpınar, Y., & Altun, A. (2014). The need for programming education in knowledge-based society schools. *Primary Education Online Magazine, 13*(1), 1–4.

Aljameel, I. H. (2022). Computer-assisted language learning in Saudi Arabia: Past, present, and future. *International Education Studies, 15*(4), 95–107. https://doi.org/10.5539/ies.v15n4p95

Alshumaimeri, Y. (2022). Educational context: The factor for a successful change. *Journal of Education & Social Policy, 9*(1), 51–57. https://eric.ed.gov/?id=ED621782

Ananiadou, K., & Claro, M. (2009). 21st century skills and competences for new millennium learners in OECD countries. *OECD Education Working Papers* (No. 41). OECD Publishing. https://doi.org/10.1787/218525261154

Attard, L., & Busuttil, K. (2020). Teacher perspectives on introducing programming constructs through coding mobile-based games to secondary school students. *Informatics in Education, 19*(4), 543–568. https://doi.org/10.15388/infedu.2020.24

Babori, A., Fihri Fassi, H., Hariri, A., & Bideq, M. (2016). An e-learning environment for algorithmic: Toward an active construction of skills. *World Journal on Educational Technology: Current Issues, 8*(2), 82–90. https://un-pub.eu/ojs/index.php/wjet/article/view/819

Balanskat, A., & Engelhardt, K. (2015). *Computing our future: Computer programming and coding. Priorities, school curricula, and initiatives across Europe: European Schoolnet.* EUN Partnership AIBSL.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers Education, 72*, 145–157. https://www.sciencedirect.com/science/article/pii/S0360131513003059

Cabrera, L. (2019). Teacher preconceptions of computational thinking: A systematic literature review. *Journal of Technology and Teacher Education, 27*(3), 305–333. https://www.learntechlib.org/p/210234/

Campbell, L. O., & Heller, S. (2019). Building computational thinking design and making in teacher education. *Recruiting, preparing, and retaining STEM teachers for a global generation* (pp. 163–189). Brill Sense. https://doi.org/10.1163/9789004399990_007

Ceker, E., & Ozdamli, F. (2016). Features and characteristics of problem-based learning. *Cypriot Journal Educational Sciences, 11*(4), 195–202. https://doi.org/10.18844/cjes.v11i4.129

Chen, Y. (2020). Basic principles of programming languages. *Introduction to programming languages* (pp. 1–39). Kendal Hunt Publishing.

Code.org. (2017). *Recommendations for states developing computer science teacher pathways*. Code.org.

CSTA. (2013). https://www.csteachers.org/documents/en-us/3b4a70cd-2a9b-478b-95cd-376530c3e976/1

Curzon, P. (2013). *Computational thinking: Searching to speak*. Queen Mary University of London.

Demirer, V., & Sak, N. (2016). Programming education and new approaches around the world and in Turkey. *Journal of Theory and Practice in Education, 12*(3), 521–546. https://dergipark.org.tr/en/pub/eku/issue/26697/280853

Denning, P. J. (2003). The great principles of computing. *Communications of the ACM*, *46*(11), 369–372. https://dl.acm.org/doi/fullHtml/10.1145/948383.948400

Dong, Y., Cateté, V., Lytle, N., Isvik, A., Barnes, T., Jocius, R., Albert, J., Joshi, D., Robinson, R., & Andrews, A. (2019). Infusing computing: Analyzing teacher programming products in K-12 computational thinking professional development. *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 278–284). ACM. https://doi.org/10.1145/3304221.3319772

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

Floyd, S. P. (2019). Historical high school computer science curriculum and current K-12 initiatives. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 1287–1287). ACM. https://dl.acm.org/doi/abs/10.1145/3287324.3293772

Framework for 21st Century Learning. (2017). http://www.p21.org/our-work/p21-framework

Fullan, M. (2007). *The new meaning of educational change* (4th ed.). Teachers College Press.

Gal-Ezer, J., & Stephenson, C. (2014). A tale of two countries: Successes and challenges in K-12 computer science education in Israel and the United States. *ACM Transactions on Computing Education, 14*(2), 8. https://doi.org/10.1145/2602483

Giordano, D., & Maiorana, F. (2015). Teaching algorithms: Visual language vs flowchart vs textual language. *Global Engineering Education Conference (EDUCON), 2015 IEEE* (pp. 499–504). IEEE.

Graves, K. (2021). Mind the gap: A tale of two curriculum fallacies. *Language Teaching,* 1–13. https://www.cambridge.org/core/journals/language-teaching/article/mind-the-gap-a-tale-of-two-curriculum-fallacies/81BEF3AE54D320D3906CE065EF022B63

Grover, S., & Pea, R. (2013). Computational thinking in K–12. A review of the state of the field. *Educational Researcher, 42*(1), 38–43. https://journals.sagepub.com/doi/abs/10.3102/0013189x12463051

Harper, R. (2016). *Practical foundations for programming languages* (2nd ed.) Cambridge University Press.

Heintz, F., Mannila, L., Nordén, L. Å., Parnes, P., & Regnell, B. (2017). Introducing programming and digital competence in Swedish K-9 education. *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 117–128). Springer. https://link.springer.com/chapter/10.1007/978-3-319-71483-7_10

Hijón-Neira, R., Santacruz-Valencia, L., Pérez-Marín, D., & Gómez-Gómez, M. (2017). An analysis of the current situation of teaching programming in primary education. *2017 International Symposium on Computers in Education (SIIE)* (pp. 1–6). IEEE. https://ieeexplore.ieee.org/abstract/document/8259650/

Humble, N., & Mozelius, P. (2019). Teacher perception of obstacles and opportunities in the integration of programming in K-12 settings. *Proceedings of International Conference on Education and New Learning Technologies* (pp. 350–356). IATED. http://lib.uib.kz/edulearn19/files/papers/125.pdf

Humble, N., Mozelius, P., & Sällvin, L. (2020). The introduction of programming in K-12 technology and mathematics teacher choice of programming tools and their perceptions of challenges and opportunities. *Education Applications & Developments: Advances in Education and Educational Trends Series,* 117–126. https://www.diva-portal.org/smash/record.jsf?pid=diva2:1456930

Lindberg, R., Laine, T., & Haaranen, L. (2019). Gamifying programming education in K-12: A review of programming curricula in seven countries and programming games: Gamifying programming education in K-12. *British Journal of Educational Technology, 50*. https://doi.org/10.1111/bjet.12685

Lloyd, M., & Chandra, V. (2020). Teaching coding and computational thinking in primary classrooms: Perceptions of Australian preservice teachers. *Curriculum Perspectives, 40*, 189–201. https://doi.org/10.1007/s41297-020-00117-1

Luxton-Reilly, A. (2016). Learning to program is easy. *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '16)* (pp. 284–289). ACM. https://doi.org/10. 1145/2899415.2899432

Moors, L., Luxton-Reilly, A., & Denny, P. (2018). Transitioning from block-based to text-based programming languages. *International Conference on Learning and Teaching in Computing and Engineering* (pp. 57–64). https://10.1109/LaTICE.2018.000-5.

Moreno-León, J., Robles, G., & Román-González, M. (2016). Code to learn: Where does it belong in the K-12 curriculum? *Journal of Information Technology Education Research, 15*, 283–303. http://www.informingscience.org/Publications/3521

Mouza, C., Yang, H., Pan, Y. C., Ozden, S. Y., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology, 33*(3). https://doi.org/10.14742/ajet.3521

Neil, C. C., Brown, S., Sentance, T., Crick, & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education, 14*(2), 9. https://doi.org/10.1145/2602484

Ohashi, Y. (2017). Preparedness of Japan's elementary school teachers for the introduction of computer programming education. *Informatics in Schools: Focus on Learning Programming* (pp. 129–140). Springer International Publishing. https://link.springer.com/chapter/10.1007/978-3-319-71483-7_11

Pears, A., Dagiene, V., & Jasute, E. (2017). Baltic and nordic K-12 teacher perspectives on computational thinking and computing. In *Informatics in Schools: Focus on Learning Programming* (pp. 141–152). Springer International Publishing. https://link.springer.com/chapter/10.1007/978-3-319-71483-7_12

Pontes, R. G., Guerrero, D. S., & Figueiredo, J. (2019). Analyzing gamification impact on a mastery learning introductory programming course. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 400–406). https://doi.org/10.1145/3287324.3287367

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. https://link.springer.com/article/10.1007/s11251-017-9421-5

Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch programming language and environment. *ACM Transactions on Computing Education (TOCE), 10*(4), 16. https://doi.org/10.1145/1868358.1868363

Scherer, R., Siddiq, F., & Sánchez-Scherer, B. (2021) Some evidence on the cognitive benefits of learning to code. *Frontiers in Psychology, 12*. https://doi.org/10.3389/fpsyg.2021.559424

Scherer, R., Siddiq, F., & Viveros, B. S. (2020). A meta-analysis of teaching and learning computer programming: Effective instructional approaches and conditions. *Computers in Human Behavior, 109*, 106349. https://doi.org/10.1016/j.chb.2020.106349.

Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education & Information Technologies, 22*, 469–495. https://doi.org/10.1007/s10639-016-9482-0.

Sentance, S., Waite, J., & Kallia, M. (2019). Teaching computer programming with PRIMM: A sociocultural perspective. *Computer Science Education, 29*(2–3), 136–176. https://doi.org/10.1080/08993408.2019.1608781

Soto, M. A. (2018). Imaginary realities: Curriculum change that ignores classroom contexts. In M. Wedell, & L. Grassick (Eds.), *International perspectives on teachers living with curriculum change* (pp. 205–228). Palgrave Macmillan.

Sterling, L. (2016). Coding in the curriculum: Fad or foundational? *Research Conference 2016 – Improving STEM Learning: What will it take?* (pp. 79–83). https://research.acer.edu.au/research_conference/RC2016/9august/4

Szabo, C., Sheard, J., Luxton-Reilly, A., Simon, B., Becker, B., & Ott, L. M. (2019). Fifteen years of introductory programming in schools: A global overview of K-12 initiatives. *19th Koli Calling International Conference on Computing Education Research* (p. 9). ACM. https://doi.org/10.1145/3364510.3364513

Tondeur, J., Scherer, R., Baran, E., Siddiq, F., Valtonen, T., & Sointu, E. (2019). Teacher educators as gatekeepers: preparing the next generation of teachers for technology integration in education. *British Journal of Education & Technology, 50*, 1189–1209. https://doi.org/10.1111/bjet.12748

Trilling, B., & Fadel, C. (2009). *21st century skills: Learning for life in our times*. John Wiley & Sons.

Tsukamoto, H., Takemura, Y., Oomori, Y., Ikeda, I., Nagumo, H., Monden, A., & Matsumoto, K.I. (2016). Textual vs. visual programming languages in programming education for primary school children. *IEEE Frontiers in Education Conference (FIE)* (pp. 1–7). IEEE. https://ieeexplore.ieee.org/abstract/document/7757571/

Tundjungsari, V. (2016). E-learning model for teaching programming language for secondary school students in Indonesia. *13th International Conference on Remote Engineering and Virtual Instrumentation (REV)* (pp. 262–266). IEEE.

Uzunboylu, H., Kınık, E., & Kanbul, S. (2017). An analysis of countries which have integrated coding into their curricula and the content analysis of academic studies on coding training in Turkey. *TEM Journal, 6*(4), 783–791. https://doi.org/10.18421/TEM64-18

Vinnervik, P. (2022). Programming in school technology education: The shaping of a new subject content. *International Journal of Technological Design & Education*. https://doi.org/10.1007/s10798-022-09773-y

Vygotsky, L. S. (1978). *Mind in society*. Harvard University Press.

Watson, C., & Li, F. W. B. (2014). Failure rates in introductory programming revisited. *Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education (ITiCSE '14)* (pp. 39–44). https://dl.acm.org/doi/abs/10.1145/2591708.2591749

Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Sysło, M. M. 2016. Computer science in K-12 school curricula of the 2lst century: Why, what and when? *Education and Information Technologies, 22*(2), 445–468. https://link.springer.com/article/10.1007/s10639-016-9493-x

Wedell, M. (2022). Innovation in ELT revisited. *ELT Journal, 76*(2), 272–275. https://doi.org/10.1093/elt/ccac003

Wedell, M., & Alshumaimeri, Y. (2014). Putting out the fires: Supervisors ’experiences of introducing primary English in Saudi Arabia. *System, 46*, 120–130. https://doi.org/10.1016/j.system.2014.07.014

Wedell, M., & Grassick, L. (2018). *International perspectives on teachers living with curriculum change.* Palgrave Macmillan.

Wedell, M., & Malderez, A. (2013). *Understanding language classroom contexts: The starting point for change.* Bloomsbury Publishing.

Wilson, A., & Moffat, D. (2010) Evaluating scratch to introduce younger schoolchildren to programming. *Proceedings of the 22nd Annual Workshop of the Psychology of Programming Interest Group*. http://scratched.gse.harvard.edu/ sites/default/files/wilson-moffatppig2010-final.pdf

Wohlgenannt, I., Simons, A., & Stieglitz, S. (2020). Virtual reality. *Business and Information Systems Engineering, 5,* 455–461. https://link.springer.com/article/10.1007/s12599-020-00658-9

Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: Understanding teacher experiences and challenges. *Computer Science Education, 26*(4), 235–254. https://doi.org/10.1080/08993408.2016.1257418.