RUDN Journal of Psychology and Pedagogics
Вестник РУДН. Серия: Психология и педагогика
ISSN 2313-1683 (Print); ISSN 2313-1705 (Online)
2024 Vol. 21 No. 3 858–886
http://journals.rudn.ru/psychology-pedagogics

DOI: 10.22363/2313-1683-2024-21-3-858-886
EDN: HBHWQQ
UDC 159.9.072.5

Исследовательская статья

Coding Lessons and the Development of Computational Thinking in Schoolchildren in the Post-Pandemic Educational Landscape: A Review on Research Challenges and Perspectives

Kristina A. Nikiforova

Sirius University of Science and Technology, *Sirius Federal Territory, Russian Federation*
✉ kkrisinger1990@gmail.com

Abstract. Despite the rapid growth of technology and the constant demand for IT specialists, the cognitive processes underlying computational thinking and the brain's ability to understand code remain poorly understood, especially in younger children. Following the Covid-19 pandemic, many countries have included coding lessons into their curricula. Coding is closely linked to complex cognitive skills in STEM (science, technology, engineering, and mathematics), such as computational and algorithmic thinking. However, confusion persists regarding the relationship between these forms of thinking and other cognitive skills. This review has two objectives: first, to investigate the methodologies used by cognitive scientists in studying the transfer effects of coding lessons on children's computational thinking skills; and, second, to examine contemporary research related to coding lessons and computational thinking. Our findings indicate that many teachers lack adequate training in coding and digital literacy, resulting in low competence and confidence in teaching these subjects. In addition, the absence of universal teaching platforms and methods complicates the implementation of coding lessons in primary schools. Finally, there is also a general shortage of longitudinal studies (over six months) focusing on the cognitive skills developed through coding lessons. Addressing these issues is essential for improving educational practices in coding and computational thinking.

Keywords: cognitive skills, K-12 curricula, computational thinking, coding lessons, neuroscience, schoolchildren, COVID-19, EEG

Introduction

Programming languages are designed specifically for conveying commands and solving issues with computers, and children as young as 3–4 years old are already able to comprehend basic coding concepts (Relkin et al., 2021). Schools

around the world have begun to include coding lessons into their K-12 curricula, but no standardized assessment protocols have been proposed as yet. Teaching methods for coding are still in their infancy. Moreover, most studies of computational thinking and code comprehension persist to focus on adult participants. Code comprehension and programming are widely associated with computational thinking. Although it is difficult to define, computational thinking is a problem-solving process that involves breaking down complex problems into smaller, easier-to-interpret parts and using algorithmic thinking and programming concepts (such as loops, conditionals, or functions) to analyze them and develop solutions (Scherer, et al., 2021). It is a fundamental skill in computer science and other fields that involve solving problems using computational tools and techniques (Relkin *et al.*, 2021).

Educational programs that develop computational thinking in middle and high schools are becoming increasingly popular. At the same time, there is still much room for improvement in this area, as more initiatives and programs are needed for younger students and their teachers. In particular, more research is required to understand the impact of coding on children's cognitive development and to determine the best ways to advance computational thinking skills (Relkin *et al.*, 2021). In adults, neuroimaging methods have shown that constant experience in one's field of expertise affects an individual's cognitive skills. In their eye-tracking study, D. K. Davis and F. Zhu (2022) analyzed the varying strategies that advanced programmers use when coding by examining eye-tracking data. Experienced programmers tend to have more efficient and focused gaze patterns than novice programmers. They spend less overall time gazing at irrelevant areas of the code, such as whitespace or non-functional areas, and they fixate on relevant areas of the code for a shorter period. The experienced programmers tend to make fewer fixations to understand a certain part of the code due to their ability to recognize patterns and familiarity with programming languages. Moreover, experienced programmers tend to fixate more frequently on function and variable names, since they need to read these identifiers to know what the program is doing. Unlike their novice counterparts, the experienced programmers also look at code blocks more often, as they read its entirety at once rather than line by line. However, novice programmers often need to read code more than once to understand each line and the code syntax, the structure of loops or conditional statements, and how to assign variables (Davis & Zhu, 2022). fMRI studies focusing on brain activity during coding, although rare, have been immensely informative. J. Castelhano and colleagues (2022) report activity in the insula during deep source-code comprehension. Specifically, when there were no errors in the participants' code spreadsheet, the dominant causal directions were mostly bottom-up, but when errors occurred, there were notable top-down effects from the frontal regions, particularly the anterior cingulate cortex (Castelhano et al., 2022). However, to date, very few

neuroimaging studies have been conducted with younger participants learning to code. Recent advances in eye tracking methods have led to the development of less intrusive devices, creating an opportunity to better understand how children interact with digital technologies, providing fresh insights into their cognitive functioning. (Sim & Bond, 2021). Unfortunately, such studies remain rare.

Literature search procedure

In this review, we conducted a comprehensive literature search using Google Scholar and PubMed as our primary databases. We used a combination of keywords and terms related to cognitive skills in coding lessons at school, including "coding education," "K-12", "computational thinking," (and\or) "algorithmic thinking,", "and "cognitive skills." We also searched for relevant articles by reviewing the identified reference lists. Our search strategy was designed to capture all the relevant studies published up to the search date. We were particularly interested in papers published after 2019, and inquiries that used extensive neuroimaging, psychophysiological or testing methods.

Inclusion criteria:

– Studies published after 2018;

– Inquiries that were conducted on schoolchildren, grades K-12;

– Studies that employed extensive cognitive skill testing or neuroscreening methodology;

– Works that focused on computational thinking training.

Exclusion criteria:

– Works published in a language other than English;

– Inquiries that involved participants over school age;

– Review articles, scoping reviews or meta-analyses;

– Studies that focused on neurodivergent or atypically developing students.

Although our study primarily focused on the educational landscape in the post-pandemic context, we included two research papers from 2018 and 2019. This inclusion is explained by our aim to examine the methodologies used by cognitive scientists in investigating the concept of computational thinking (CT). The selected papers represent distinctive methodological approaches that contribute valuable insights to our work. The papers included in our review can be seen in Table 1.

Table 1

Current works in the field of coding in schools where neuroscreening and/or testing methods are also used (summary of included studies)

Author(s)	Research disign	Cognitive skills	Age of parti-cipants	Aims of study	Sample size	Results	Methodology
Mousa, M., & Molnar, G. (2020)	Evaluation	Inductive reasoning	9–11 yrs.	To describe a training program that used computer technology to teach inductive reasoning skills. The program included interactive tasks integrated with mathematical content.	$N = 118$	The experimental group outperformed the control group after the intervention by a whole standard deviation.	Computer-based inductive reasoning test (pre- and post-test)
Relkin, E., de Ruiter, L. E., & Bers, M. U. (2021).	Longitudinal	Computational thinking	5–9 yrs.	Used the CAL-KIBO curriculum to improve students' computational thinking (CT) skills; also used *TechCheck* to assess the skills post-intervention.	The experimental group: $N = 667$; the control group: $N = 181$	The experimental condition showed improvement in CT skills ($M_{change} = 0.94$, $p < .001$) when compared to the control group ($M_{change} = 0.27$, $p = .07$)	*TechCheck* assess-ment instrument
Robledo-Castro, C., Castillo-Ossa, L. F., & Hederich-Martinez, C. (2023)	Pilot study	Computational thinking Executive function	10–11 yrs.	The experimental condition underwent an 8-week computational thinking intervention. Changes in their cognitive skills were assessed using the BANFE-2 neuropsychological battery of executive functions and frontal lobes. The aim was to identify neuropsychological changes in the brain.	$N = 30$	Positive effects were observed in the expe-rimental condition after the intervention on metacognitive functions, executive functions and working memory. No effects were observed on inhibition and risk-benefit management.	BANFE-2

Continuation of table 1

Author(s)	Research disign	Cognitive skills	Age of participants	Aims of study	Sample size	Results	Methodology
Arslanyilmaz, A., & Sullins, J. (2021)	Experimental	Attention distribution Computational thinking	12–14 yrs.	To test the possibility of using the eye-tracking tool to assess attention distribution, while the participants perform computational thinking / coding tasks.	*N* = 29 were recruited, but only data from *N* = 16 were analyzed	Fixation durations were significant in testing the effectiveness of the participants' learning efficacy of computational thinking concepts. The participants returned to conditionals more often than to other concepts.	Eye tracker Quiz scores
Adamovic, M.D. & Ivetic, D. V. (2024)	Experimental	Analogy learning Computational thinking	7–9 yrs.	A simple video game combining traffic and basic programming concepts was designed and introduced to the schoolchildren. The aim of the study was to examine the possibility of teaching basic programming to schoolchildren using analogues.	*N* = 112	The participants exposed to the intervention displayed better post-test results than their control counterparts did.	Gamification Pre-test Re-test Post-test
Misirli, A. & Komis V. (2023)	Experimental	Debugging ability Computational thinking	4–6 yrs.	The aim was to examine the debugging strategies used by young children being taught to code a robot.	*N* = 526	Two categories of debugging knowledge were identified: knowledge related to coding syntax and knowledge related to coding semantics.	Klahr & Craven's framework (1988)

Continuation of table 1

Author(s)	Research disign	Cognitive skills	Age of participants	Aims of study	Sample size	Results	Methodology
Zhao, Li *et al*. (2022)	Quasi-experiment	Computational thinking Mind mapping	10–11 yrs.	A group of programming students were presented with two types of mind mapping methods (construct-by self vs. construct-on-scaffold) and their effects on computational thinking skills were compared.	$N = 73$ ($N = 33$ in the construct-by self (CBS) condition + $N = 40$ construct-on-scaffold (COS) condition)	The results suggested that coding lessons were not the only option to teach CT skills to younger students. Both conditions showed improvements in their CT tasks – CBS ($t = -3.022$, $p = 0.005$) and COS ($t = -8.594$, $p = 0.000$). The students in the COS group showed greater improvement in their CT skills when using the COS-MM (Mind Map) method compared to the students in the CBS group trained using the CBS-MM method.	Scratch tool Pre- and post-test CT surveys
Chan, S.-W. *et al*. (2021)	Quasi-experiment, non-equivalent groups	Computational thinking Mathematics	13 yrs.	To examine the potential impact of CT-based intervention on students' sense of number patterns.	$N = 106$ (the experimental group: $N = 70$ + the control group: $N = 36$)	The CT intervention had very little impact on the students' number sequencing abilities. However, several participants from the experimental condition showed drastic improvements in the post-test.	Math + C program Pre-test Post-test

Continuation of table 1

Author(s)	Research disign	Cognitive skills	Age of parti-cipants	Aims of study	Sample size	Results	Methodology
Papavlaso-poulou, S., *et al*. (2018)	Experimental	Attitude (learning, excitement, intention) towards coding Gaze patterns	8–17 yrs.	To investigate the participants' attitudes towards learning how to code and their gaze patterns during coding lessons.	*N* = 44	A definite correlation was found between the eagerness towards learning how to code and gaze patterns.	Eye tracking Surveys
Ozcan, M.S. *et al*. (2021)	Randomized experiment	Computational thinking Fluid intelligence Spatial orientation	M_{age} – 10 yrs.	To assess the impact of a 10-week coding program on the participants' cognitive skills (CT, fluid intelligence, spatial orientation)	*N* =174	Only the learn-to-code condition experienced significant impact on CT scores. Fluid intelligence was impacted for all conditions. Spatial orientation results remained unchanged.	The matrix reasoning task from the Wechsler Abbreviated Scale of Intelligence Measurement; computational thinking scale taken from Tran (2018); a subtest of the spatial reasoning task (Ramful, Lowrie, & Logan, 2017)
Heim, G. & Wang, O. J. (2023)	Experi-mental / Observational	Algorithmic thinking Mathematics programming	6th grade	To examine whether students would be able to envision programming possibilities following 2 lessons in math and food & health.	*N* = 44	Very few students were able to identify the link between maths, food& health, and the implications of these topics for programming.	Math lessons Food & health lessons Survey Unstructured observation

Continuation of table 1

Author(s)	Research disign	Cognitive skills	Age of participants	Aims of study	Sample size	Results	Methodology
Arfe, B. *et al*. (2019)	Longitudinal Experimental	Executive functions Planning Response inhibition	5–6 yrs.	The study was twofold: (1) it tested the potential transfer of skills from a 1-month coding intervention on the young participants' executive functions and planning abilities; the impact of one month of coding activities on the development of 17 second graders was examined within the experimental group longitudinally, comparing it to the 7-month standard activities experienced by the same children; and (2) the effects were compared between the experimental group and the control group engaged in standard STEM activities.	*N* = 80 (the experimental group: *N* = 44 + the control group: *N* = 36)	Very prominent effects of early coding activities on the students' executive functions and planning ability were observed. Further longitudinal data showed that the effects of only 1 month of coding lessons had a greater impact on the participants' planning and inhibition than 7 months of regular STEM-based activities.	Code.org Elithorn ToL Numeric Stroop task NEPPSY-II
Bai, H. *et al*. (2021)	Experimental	Computational thinking (concepts, practices and perspectives) Problem solving	8^{th} grade	To study the effects of the Problem-oriented learning (PoL) model vs. the lecture-and-practice (LaP) model on the participants' CT skills	*N* = 60 (the experimental group: *N* = 30 + the control group: *N* = 30)	The PoL method yielded better results than the LaP model. The most prominent effects were observed in the participants' CT concepts and perspectives, but to a lesser extent in CT practices.	Pre-test of CT perspectives scale Post-test of CT perspectives scale Bebras test Computer-based final test

Continuation of table 1

Author(s)	Research disign	Cognitive skills	Age of parti-cipants	Aims of study	Sample size	Results	Methodology
Theodoro-poulos, A. & Lepouras, G. (2022)	Experimental	Gender differences Personality traits (cognitive style, emotional intelligence) Learning perception	14–15 yrs.	To identify any correlation between the participants' gender, personality traits and preferred programming tools (Scratch, App inventor, Alice)	*N* = 163 (Group A: *N* = 57, Group B: *N* = 51, Group C: *N* = 55)	Definite changes were observed in gender, personality traits and preferred programming tools. Scratch was the most appropriate for the younger participants; App Inventor was recommended for introduction to more complex programming concepts, while Alice elicited mainly negative feelings from both genders.	Pre-questio-nnaire – Programming Experience Personality – cognitive style (short version of Myers and Briggs Type Indicator (MBTI) The Emotional Intelligence Questionnaire (Petrides et al., 2006) as a Personality Trait-short version (TEIQue-sf) Post-questi-onnaire –Learning Experience and Emotions
Perez-Marin, D.P. *et al*. (2020)	longitudinal pre- and post-test quasi-experiment	Computational thinking Programming	9–12 yrs.	To test the effectiveness of a metaphor-based Scratch methodology (MECOPROG) on the young participants' CT skills.	*N* = 132	Significant improvements in CT skills were observed during post-testing	CONT knowledge test the knowledge programming concept test CT tests (ROMT and PCNT)

Continuation of table 1

Author(s)	Research disign	Cognitive skills	Age of participants	Aims of study	Sample size	Results	Methodology
Cheng, Y-P. *et al.* (2023)	Experimental	Higher-order thinking Computational thinking Intrinsic & extrinsic motivation	Primary school	To investigate whether incorporating the student-generated question (SGQ) strategy into game-based learning (GBL) would improve primary students' CT skills and motivation.	$N = 53$	The experimental group, assigned to the SGQ and GBL condition, significantly outperformed their control counterparts in both CT, skills and motivation.	
Del Olmo-Munoz, J. *et al.* (2020)	Quasi-experiment	CT Motivation Gender	2nd grade	To examine possible correlations between students' motivation, gender and CT skills. Also, to test whether CT skills should be trained through plugged or unplugged activities in primary school students.	$N = 84$	The Mann–Whitney for the pre-test and post- test ($U = 369.00$, $p = .033$) suggested better CT performance for the unplugged condition on both easy ($U = 317.50$, $p < .001$) and difficult ($U = 538.50$, $p = .285$) problems. No significant differences were found for motivation in either the pre-test ($U = 718.00$, $p = .814$) or the post-test ($U = 715.50$, $p = .413$). No significant differences were observed in gender, but boys were slightly higher motivated in the plugged motivation domain ($U = 116.50$, $p = .030$).	Motivation: four-dimensional model ARCS (attention, relevance, confidence and satisfaction) CT skills: mixed difficulty Bebras Tasks

End of table 1

Author(s)	Research disign	Cognitive skills	Age of parti-cipants	Aims of study	Sample size	Results	Methodology
Gerosa, A. *et al*. (2021)	Cross-sectional correlational design	Higher-order thinking Computational thinking Intrinsic & extrinsic motivation Fluid intelligence, Working memory, Planning, Sequencing, Mental rotation, Vocabulary, Early math precursors (numerical transcoding and symbolic magnitude comparison) Parents' attitude towards technology use Computational thinking	4–6 yrs.	To study the impact of CT skill acquisition on the development of other cognitive skills in early childhood	*N* = 102	Temporal sequencing ability and symbolic magnitude comparison were found to be the most effective predictors of CT competence.	Yune Tran's CT questionnaire for 7 year old children; Raven's colored progressive matrices (tablet-based version); Corsi Block Tapping Test (Tablet-based version); Tower of London task (tablet-based version); Langdon and Coltheart's subset of mechanical stimuli (paper-based); Peabody Picture Vocabulary Test (tablet-based); Classical task by Moyer and Landauer (tablet-based); Transcoding task; Mental rotation task; the Parental Perceptions of Technology Scale; Parent's attitudes towards computer use scale

Through this structured approach, we aim to provide a thorough examination of the literature relevant to our topic. By synthesizing the results of various studies, this review highlights effective practices in coding education and identifies gaps in the current research landscape. Ultimately, our goal is to inform educators and policymakers about the cognitive benefits of coding lessons and to advocate for improved training and resources for teachers in this critical area of education. The COVID-19 pandemic has significantly accelerated the shift to online learning and the integration of digital tools into educational frameworks. This shift has made computational thinking a critical competency for navigating modern educational environments (Koh & Daniel, 2022). Both students and teachers have been forced to adapt to remote learning platforms, which requires the development of problem-solving, algorithmic thinking, and data analysis skills. These skills are essential for effectively engaging with educational content and facilitating collaborative interactions in a virtual context. To date, it is still unknown how this shift has affected cognitive skills.

Results

We report a total of 18 studies that examine various aspects of coding education, focusing on both plugged and unplugged programming approaches. The studies used a range of methodologies to assess the effectiveness of these teaching methods for developing cognitive skills in K-12 students. Notably, only a few of the included studies employed neuroscreening methods, highlighting a potential gap in the literature regarding the neurocognitive impacts of programming instruction. This diversity in research methods highlights the complexity of evaluating coding education and its effects on learning outcomes. We will synthesize the principal concepts derived from current research in the following sub-chapters.

Coding and cognition. Technology and coding have been incorporated into modern curricula to develop a variety of cognitive skills, from reading ability to mathematics (McCray, & Chen, 2012). There is an inherent relationship between computational thinking and mathematics, particularly in terms of logical structure and the ability to explore and create models for mathematical relationships. Integrating computational thinking into the teaching of mathematics has the potential to improve and broaden understanding of both subjects (Chan et al., 2021). According to C. Robledo-Castro and colleagues (2023), computational thinking has become more widely recognized in recent years due to its role in facilitating the growth and development of STEM (Science, Technology, Engineering, and Mathematics) competencies. The meta-study suggests that, indeed, computer use and programming lessons from an early age have had long-lasting positive effects on students' logic, reasoning and problem-solving skills. These effects may be due to the fact that the parts of the prefrontal cortex responsible for executive control are highly dependent on the stimuli the brain receives from the environment, and

computer lessons in early childhood have been shown to facilitate the maturation of the prefrontal cortex. These findings have important implications for computer intervention programs targeting children. Although studies of such interventions are rare, they have shown that learning to code is correlated with schoolchildren's performance in tasks involving working memory and creative thinking (Robledo-Castro et al., 2023). Few studies have focused on the interaction between computational thinking and other cognitive skills in younger children. A. Gerosa and colleagues (2021) recruited 102 (N = 102) kindergarteners aged 4–6. They combined various cognitive test batteries with a robotics-based intervention designed for young children. Their study aimed to determine which cognitive skills served as potent predictors of CT competence. Two cognitive skills were found to be highly correlated with CT competence, *i.e.*, temporal sequencing ability (assessed by the Langdon and Coltheart task) and symbolic magnitude comparison (assessed by the Moyer and Landauer task). Temporal sequencing ability refers to an individual's ability to understand and reproduce the correct order of events or stimuli in relation to time. The authors note that, although their findings are definitive and serve to improve our understanding of the interactions between CT and other early cognitive skills, more research is needed (Gerosa et al., 2021).

Computational skills have been linked to mathematical ability. Moreover, computers have become indispensable in modern mathematics, influencing the way mathematicians conduct research, teach, and apply mathematical concepts across disciplines. S.-W. Chan and colleagues (2021) investigated whether integrating computational thinking concepts into math lessons would improve students' number pattern skills. They recruited 106 (N = 106) Singaporean secondary school students (13 years old). The participants attended classes, where they were taught number sequences. They passed both pre- and post-testing. During the intervention, the students were exposed to both unplugged and plugged activities. During the post-test, the Rasch analysis showed that the mean score for the experimental condition was 1.49, while for the control group it was 1.48. The authors argue that the similarity of the results was unexpected, as previous studies have shown some gains in math ability after CT intervention. One possible explanation for this lack of improvement may be the short duration of the intervention, i.e., the for the plugged activities it was only an hour and a half, while for the unplugged activities it was less than 2 hours. Additionally, the authors report several extreme improvements only in the experimental conditions, with no such outliers observed in the control group, which never received the intervention (Chan et al., 2021).

M.Ş. Özcan and colleagues (2021) conducted their inquiry on 4th grade students (M_{age} = 10). They recruited students from Turkey (N = 174), because this country introduced coding lessons into its mandatory curricula in 2018. The authors tested how a 10-week coding intervention affected the participants' cognitive skills. On the one hand, the authors hypothesized that coding could promote the development

of students' near-transfer skills (in this case, computational thinking). On the other hand, they suggest some positive effects could be observed on far-transfer skills (in this study, fluid intelligence and spatial ability). The participants were divided into three conditions: "learn-to-code", "reading" and "maths". They completed both pre- and post-tests. The testing consisted of the Matrix Reasoning task from the Wechsler Abbreviated Scale of Intelligence (WASI), the Computational Thinking scale taken from Tran (2018); and the Spatial Reasoning task subtest (Ramful et al., 2017). The results of a one-way ANOVA indicate that the "learn-to-code" condition showed higher scores at the post-test (M = 3.67, SD = 2.14) than at the pre-test (M = 3.08, SD = 1.71, p = .04, d = .29). The students in the "math" condition also showed higher results at the post-test (M = 3.56, SD = 1.84) than at the pre-test (M = 3.11, SD = 1.58), although the difference was not significant at p = .10, d = .26. A slight improvement was noted in the "reading" condition, which was also treated as a control group in the present study: at the pre-test (M = 3.00, SD = 1.77), p = .26, d = .15 compared to the post-test (M = 3.26, SD = 1.64), p = .26, d = .15. The study demonstrated some effects on computational thinking after the coding intervention, but no significant effects on far-transfer skills, such as fluid intelligence or, surprisingly, spatial ability (Özcan et al., 2021). In summary, coding lessons, math ability and computational thinking are interconnected, with coding facilitating mathematical thinking and problem-solving skills, and computational thinking skills being a part of mathematical thinking. Integrating coding into various subjects can promote computational thinking and enhance students' understanding of the subject matter.

Computational thinking vs. algorithmic thinking. Computational thinking and algorithmic thinking are related concepts, but they focus on different aspects of problem-solving and problem analysis. On the one hand, computational thinking involves using a mixture of creativity, logic and problem-solving skills to solve complex problems in a way that a computer or a machine can understand and execute efficiently. It is a broader concept that encompasses various abilities, such as breaking problems down into smaller components, identifying patterns and abstractions, designing algorithms and models, and making use of logical and analytical reasoning (Angeli, 2022). Algorithmic thinking, on the other hand, specifically focuses on the design and analysis of algorithms, i.e., a step-by-step procedure for solving a problem or completing a task. Algorithmic thinking emphasizes the formulation of steps or instructions that can be executed in a specific sequence to efficiently solve a problem (Bacelo & Gómez-Chacón, 2023). It involves determining the appropriate data structures, identifying efficient steps and considering issues such as time complexity and space complexity. In other words, computational thinking is a more general approach to problem solving in a computational context, while algorithmic thinking is a narrower focus on the design and analysis of algorithms to efficiently solve problems. However, the scientific

community continues to debate the precise definition of both computational and algorithmic thinking, as what we know today is vague and highly context-dependent. This lack of a clear definition leads to lackluster guidelines on how to measure and evaluate computational thinking, which is a cause for concern and should be acknowledged. Without appropriate assessment methods, computational thinking is unlikely to be effectively integrated into any educational program. Furthermore, to determine the success of a curriculum that includes computational thinking, it is essential to establish reliable measures that will allow educators to assess students' learning outcomes (Román-González et al., 2017). Another contentious issue is, simply put, what to teach and when. Previous research has shown that introducing the concept of algorithmic thinking as a first step to computational thinking enhances the learning experience, thereby emphasizing the importance of teaching programming from an early age at all educational levels (Angeli, 2022).

Plugged vs. unplugged programming in school curricula. Most authors agree that, when it comes to teaching CT in schools, it is no longer a question of "if", but "when" and "how": the demand for IT professionals is constantly growing, and even primary school students are able to acquire some elements of programming code (Zeng e*t al.*, 2023). Another focus of interest in modern scientific literature is the issue of plugged and unplugged programming. In unplugged programming, the activities typically do not require a computer at all. Instead, they include offline activities and games to explain programming concepts, logic, algorithms, computational thinking, and more. They are often used with beginners or younger learners to introduce, explain, and analogize complex concepts in a tangible, hands-on way without the layer of abstraction or potential distractions that a computer may introduce (Chen et al., 2023). Conversely plugged programming involves the use of actual computer hardware and software. It can include writing code in a specific programming language (such as Python or JavaScript), working with a graphical interface in block-based programming environments (like Scratch or Blockly), and using specific educational robotics kits or programmable devices. This is a more traditional and direct method of learning programming, where students write code, execute it and see the results immediately (Kirçali & Özdener, 2023). It remains unclear at what age plugged programming should be introduced. J. Del Olmo-Muñoz and colleagues (2020) examined whether computational skill lessons would yield better results if second graders were exposed to plugged or unplugged programming. Their study was twofold: (1) to test the effects of plugged vs. unplugged programming lessons on the students' CT; and (2) to examine any possible correlations between the participants' gender, CT skills and motivation. They recruited 84 participants ($N = 84$) from the second grade. During the initial session, the students completed a computational thinking pre-assessment to determine their initial competence in the relevant skills. Following this assessment, a three-session instructional period commenced. During this phase, the control

group, referred to as the unplugged group, engaged in activities without the use of computers. In contrast, the experimental group, referred to as the plugged-in group, participated in activities that involved the use of computers. Following this instructional phase, the students completed a mid-assessment focusing on their computational thinking abilities and a survey designed to assess their level of interest and enthusiasm for the tasks they had just had. The second stage spanned two sessions and standardized the activity type for all the participants to be computer-based (plugged-in). After this second instructional phase, the students completed final assessments to re-assess their computational thinking (CT) abilities and motivation levels. The unplugged group showed better CT scores for both easy ($U = 317.50$, $p < .001$) and difficult ($U = 538.50$, $p = .285$) problems. There were no statistically significant differences in motivation at the pre-test ($U = 718.00$, $p = .814$) or at the post-test ($U = 715.50$, $p = .413$). Additionally, no significant differences were found in terms of gender, but it was concluded that the boys demonstrated slightly higher motivation in the plugged motivation domain ($U = 116.50$, $p = .030$). The authors suggested combining the plugged and unplugged activities for younger students as this approach improved both students' CT skills and motivation levels (Del Olmo-Muñoz et al., 2020). Unplugged activities, which involve learning computing concepts without digital tools, are particularly useful for younger children. These activities help them understand fundamental concepts such as algorithms, logical prediction, debugging, problem decomposition, structure recognition, and algorithm design. Unplugged activities are recommended as a starting point before moving on to the plugged activities to build a solid foundation in computational thinking and programming skills.

Teacher competence. Another contentious issue regarding the introduction of coding into school curricula is teacher readiness. Most primary and/or secondary school teachers do not have experience with computers or computer science, as they are not required to take any related courses during their studies (Erümit, & Sahin, 2020). Moreover, they do not have formal training or exposure to instructional strategies to effectively teach computational thinking, which may reduce their confidence in this area (El-Hamamsy et al., 2023). Additionally, students who are less experienced with computers or have learning disabilities may find it difficult to keep pace with the lesson plan (Chan et al., 2021). S.-C. Kong and colleagues examined how well a teacher development program could promote critical thinking in primary education, and whether it is scalable and sustainable. The 2023 report covered two separate studies. The first one evaluated whether two different programming environments ("Scratch" and "App Inventor") could effectively develop teachers' computational thinking skills. A total of 245 teachers ($N = 245$) from several primary schools participated in two 12-hour sessions that used the Technological Pedagogical Content Knowledge (TPACK) framework. This framework is a theoretical model that highlights both the complex interactions and

integration between technology, pedagogy and content knowledge in education. It was developed to provide a basis for understanding how technology could be effectively used to enhance teaching and learning. TPACK suggests that effective technology integration requires teachers to have knowledge and understanding of the interactions between technology, pedagogy and content, as well as the ability to apply this knowledge in practice. In summary, the TPACK framework identifies a set of knowledge domains that teachers need to master in order to effectively integrate technology into their teaching. These domains include technology, pedagogy and content, all of which interact in complex ways in the classroom. The research found that the program significantly improved the teachers' knowledge and understanding of content-related dimensions, and helped them grasp advanced computational thinking concepts such as "data structures and procedures" (Kong et al., 2023). The second study conducted a thematic analysis on computational thinking strategies used in 47 primary schools during 94 school visits. The most commonly mentioned strategies included the "forming teaching teams", "lesson co-planning", and "integrating computational thinking with subject teaching". The most frequently encountered challenges were "teacher readiness, lesson time, and diversity in learners' abilities", interests, and approaches (Kong et al., 2023). The results suggested that a training program using different programming environments and teaching experience could effectively improve teacher's skills. However, ongoing support was needed to help the teachers implement the strategies they learned after completing the program. Addressing the diversity in the learners' abilities and interests and integrating computational thinking with subject teaching requires continued support. Specifically, for computer science education, it is important for teachers to be technologically literate, as they may be required to teach computer science even if they have no experience in the subject. Moreover, it is crucial that school and district administrators emphasize teacher's digital literacy to avoid policies that simply mandate technology use step by step. Instead, digitally literate teachers should be encouraged to see technology for all its creative potential and collaborate with peers to improve their students' learning outcomes.

Coding in schools. As mentioned earlier, computational thinking can be introduced into the curricula in a variety of ways. A meta-review conducted by Z. Zhan and colleagues (2022) sought to find the optimal trajectory through which programming could potentially be taught in schools. Their answer was gamification, "a learning process in which learners solve problems and overcome challenges in game-based settings to achieve desired learning outcomes" (Zhan et al., 2022). The authors reviewed 21 studies published over the last decade. The studies included in this paper proposed a variety of game-based teaching methods that addressed computer technology/programming lessons in schools and considered different age groups of learners. For instance, although it can be argued that programming is a very tedious subject for schoolchildren, many unusual techniques and methods

have been introduced ranging from already existing apps and games for children to more complex activities designed to teach students to create their own on-line games. Z. Zhan and colleagues (2022) examined the effects of various interactive coding- and computer-based games on students' learning motivation, academic performance and thinking skills. The results of the study showed that gamification had a greater overall impact on teaching code programming compared to graphical programming (Zhan et al., 2022). The authors concluded that introducing games into computer classes improved students' motivation (*SMD* = 0.77), academic performance (*SMD* = 0.75) and thinking skills (*SMD* = 0.48).

It is important to note that although there are few separate interventions that focus solely on coding, the introduction of computers into the classroom is no longer contentious point for educators or cognitive scientists. A study by M. Mousa and colleagues (2020) presented an educational program that used computer-based training to help develop the inductive reasoning skills in 9- to 11-year-old students. The study evaluated the program and its outcomes. It was designed based on Klauer's model and the Cognitive Training for Children approach to inductive reasoning. It included 120 engaging problem-solving exercises that were presented in an online environment. All the problems were integrated into mathematical content, making the program easily applicable during regular mathematics lessons (Mousa & Molnár, 2020). The results showed that the implementation of this program resulted in measurable improvements in the students' academic performance, regardless of gender and/or maternal education level, which were additional variables in the study, compared to the control group (M_{exp} = 58.6, SD_{exp} = 14.5, t = 13.1, $p < .001$). It should be noted that although there are few interventions that focus exclusively on programming and/or coding, many use techniques and exercises derived from programming.

Indeed, since coding involves a variety of cognitive abilities, coding lessons allow students to practice not only computational skills but also writing and mathematics. According to J. Thompson & G. Childers (2021), today's rapidly evolving technologies have impacted every aspect of the modern written language, which has changed our views on literacy. In their work, the authors examined a group of fifth-graders that were enrolled in school-based summer sessions focused on storytelling. The school district's summer program conducted instructional sessions that focused on creating stories using coding. The learners were assessed before and after their writing sessions regarding their (1) writing ability, (2) improvement in idea generation, writing organization, syntax and usage, as well as mechanical skills, and (3) writing endurance. The results showed that there were definite improvements in their endurance and overall descriptive abilities, while interviews revealed an increase in their motivation and desire to continue their coding lessons (Thompson & Childers, 2021). Similar results were obtained by E. Relkin and colleagues (2022). Their study involved a large sample size (N = 667

in the experimental condition vs. $N = 181$ in the control condition) and was aimed at examining the benefits of teaching age-appropriate coding to first- and second-grade schoolchildren. The participants' computational skills were assessed *post-hoc*. The "Coding as Another Language" or "CAL" curriculum was designed to last for seven weeks and employed the KIBO robot in a way that combined programming and literacy skills, essentially treating coding as a language. The KIBO robot is an educational tool designed for young children aged 4–7 years to introduce them to coding, robotics, and STEM concepts. This interactive robot can be programmed using colorful blocks, allowing children to learn programming concepts through physical play. KIBO includes motors, sensors and sound to perform actions based on programs children create using tangible blocks. Designed for use in the classroom or at home, KIBO encourages hands-on exploration and experimentation, and teaches children important skills in critical thinking, problem-solving and creativity. The results showed that CAL-KIBO increased the children's competences in algorithms, modularity and representation in the computational thinking domains ($M_{change} = 0.94$, $p < .001$) compared to the control group ($M_{change} = 0.27$, $p = .07$). These results suggest that a context-appropriate curriculum for children to learn coding can improve their computational thinking abilities (Relkin *et al.*, 2021).

We know very little about the effects of computational thinking interventions on students' brain development. This is partly because there have been few studies on school-aged participants exposed to CT interventions, especially studies based on neurophysiological methodology. We know that successful coding requires potent executive functions, which largely rely on the coder's frontal lobes (Arfé et al., 2019). C. Robledo-Castro and colleagues (2023) used the Neuropsychological Battery of Executive Functions and Frontal Lobes (BANFE - 2) to test how an 8-week CT-based intervention affected the anterior prefrontal cortex (aPFC) and the dorsolateral cortex (dlPFC) in schoolchildren. These are two parts of the brain located within the prefrontal cortex, the area responsible for many aspects of executive functions. The dlPFC has many interbrain connections, allowing it to integrate information from different resources. This region is heavily involved in executive functions, particularly working memory and cognitive flexibility. It helps manage tasks, when multiple steps, adjustments or simultaneous goals are required. In other words, the dlPFC plays a key role in coordinating thoughts and actions in accordance with internal goals. The aPFC, also sometimes referred to as the frontopolar prefrontal cortex, is another region that is critical to many aspects of executive function. In particular, we know that it contributes to high-level functions such as multi-tasking, integrating information over time, thinking about future outcomes, and analyzing complex situations. The aPFC is often considered the most evolutionarily advanced part of our brain. The exact extent and nature of functional specialization within these regions are ongoing areas of research within cognitive neuroscience. It is also important to note that the brain functions as a

highly interconnected network; therefore, while one can speak meaningfully of a certain region being "involved" in certain functions, this does not mean that the functions are strictly "localized" to that region only (Panikratova et al., 2020). Following the intervention, the authors reported pre- to post-test changes in the executive functions of the experimental condition controlled by the anterior prefrontal and the dorsolateral cortex (F (1, 28) = 22.00 $p < .001$ $\omega^2 = 0.13$). However, C. Robledo-Castro and colleagues reported no statistically significant changes in the executive functions of the experimental condition controlled by the orbitofrontal cortex (Robledo-Castro et al., 2023).

B. Arfé and colleagues (2019) conducted two studies to investigate the effects of a 1-month coding intervention on the planning and response inhibition skills in first and second-grade students. In the first study, they compared the performance of 76 first graders ($N = 76$) who participated in coding activities to that of a control group who participated in standard STEM activities. In the second study, they compared the performance of 17 second graders ($N = 17$) who participated in coding activities to that of the same children who participated in standard activities over an extended period, as well as to that of a control group of 19 second graders ($N = 19$) who participated in standard STEM activities. A significant correlation was found between the reduction in planning time for coding tasks from the first pre-test to the post-test and coding accuracy r (76) = −0.61, $p < 0.001$. Furthermore, there were significant correlations between the reduction in planning time and improvements in accuracy for the Elithorn and ToL (Tower of London) tasks from the pre-test to the post-test with r (76) = −0.29, $p = 0.01$ and r (76) = −0.31, $p < 0.01$, respectively. The changes in coding accuracy between the pre-test and the post-test were positively linked with the changes in accuracy on the Elithorn task r (76) = 0.26, $p < 0.05$. Moreover, the reduction in planning time between the post-test and the delayed post-test was significantly associated with the improvements in encoding accuracy in the same time interval r (76) = −0.70, $p < 0.001$. The improvements in accuracy on the Elithorn and ToL tests r (76) = −0.38, $p = 0.001$ and r (76) = −0.47, $p < 0.001$ were associated with the reductions in inhibition errors on the NEPSY-II r (76) = 0.23, $p < 0.05$ and Stroop tasks r (76) = 0.45, $p < 0.001$. Furthermore, improvements in coding accuracy between the post-test and the delayed post-test were positively associated with improvements in accuracy on the Elithorn r (76) = 0.33, $p < 0.005$ and ToL tasks r (76) = 0.42, $p < 0.001$ and were negatively associated with the reductions in inhibition errors on the Stroop test r (76) = −0.35, $p < 0.005$. The authors concluded that just one month of coding lessons had a greater effect on the participants' planning and inhibition than 7 months of regular STEM-based activities (Arfé et al., 2019).

One way to introduce coding to younger students is through analogies. Analogies serve as tools to convey understanding through meaningful depictions across various subjects (Harsch & Kendeou, 2023). The biggest challenge lies in identifying

valuable correlations between distinct symbolic portrayals of subjects that allow knowledge to be shared quickly and effectively. Learning through analogies is generally split into two subdivisions:

1) Near transfer, a situation where the origin and the desired area of knowledge are already similar, allowing solutions to be conveyed almost word-for-word.

2) Far transfer, a context where the domains differ significantly at the superficial level, requiring knowledge to be transferred through deeper abstractions.

In modern education, several examples of coding analogies are presented, namely maps, electrical grids, correspondence and traffic (Adamović & Ivetić, 2024). In terms of teaching software design and programming (*e.g.*, using Scratch), both metaphors and analogies are often used. They can develop students' understanding of abstract computing concepts by relating them to tangible real-world examples. Students can be taught such complex programming concepts as variables, conditional statements, loops, and debugging strategies using examples from their everyday lives. M. Đ. Adamović & D. V. Ivetić (2024) presented a video game that combined programming concepts and traffic for a group of children aged 7–9 years (N = 112). Similarly, D. Pérez-Marín and colleagues validated a pedagogical methodology that combined metaphors with Scratch, a block-based visual programming language primarily aimed at children (Pérez-Marín, D. at al., 2020). Created by the Lifelong Kindergarten group at the MIT Media Lab, Scratch allows users to create projects using a block-like interface (Dúo-Terrón, 2023). In their study, D. Pérez-Marín and colleagues (2020) explained programming using food- and recipe-based analogies (called "metaphors" by the authors). They recruited 132 (N = 132) participants aged 9–12 years. The authors used three tests both before and after the intervention: a test that assessed children's computational thinking skills, validated for this age group ("ROMT"); a test created specifically for the pre-assessment ("CONT"); and a new test, based on scientific literature, created to test the participants' computational thinking ("PCNT"). After a 6-week (1 hour per week) intervention, a significant improvement (Rosenthal r) was noted for the 4th grade condition in the CONT test (r = 0.62). The 5th grade condition showed a small increase for the PCNT variable (r = 0.27) as well as for the ROMT variable (r = 0.23), and a notable improvement in the CONT variable (r = 0.57). The 6th grade condition showed a large effect (r = 0.55) (Pérez-Marín et al., 2020). This was not the first instance that educators used recipe or food comparison to illustrate programming in a school setting. In their 2023 observational study, G. Heim & O.J. Wang (2023) analyzed the feedback from a group of 6th grade students (N = 44). The students were part of two classes that participated in lessons on two topics: mathematics (where the students were introduced to block programming), and food/health (where students followed a recipe, an example of unplugged programming). Since Norway introduced programming into their school curricula, the authors wanted to know whether students would be able to envision

uses for coding within the food and health topic. Although only 36 students provided feedback ($N = 36$), seven participants answered in a way that suggested they could see the connection between the topics, correctly indicating that they followed steps in a recipe that they thought were similar to the blocks in their coding classes. However, the small number of students who were able to see some connection was not necessarily a fault in the analogy used (Heim & Wang, 2023). Since algorithmic thinking is a large category within computational thinking, the recipe analogy is applicable when it comes to programming, as it serves to implement many aspects of algorithms in a way that is easy for younger students to understand. Like coding, following a recipe involves following instructions, doing things in the correct order and analyzing the results of each step.

Debugging is an essential aspect of programming and software development, as it helps identify and fix errors or bugs in the software source code. It is crucial to determine why an operating system, application, or program is misbehaving and plays a significant role in improving both software quality and end-user experience. Studies have shown that different programmers have their own strategies when it comes to the debugging process, namely, experience in the area affects eye movement patterns while searching for code errors (Davis & Zhu, 2022). A. Misirli & V. Komis (2023) suggested that young children can develop their own debugging strategies, even those with no prior experience. Of the 526 recruited participants aged 4–6 years ($N = 526$), 84 ($f = 284$, $rf = 53.99$) demonstrated fully consistent programming behavior without errors in their programs. Furthermore, 184 of the 526 children ($f = 184$, $rf = 34.98$) demonstrated semantic or logical errors, and 36 ($f = 36$, $rf = 6.84$) showed a combination of syntactic and semantic/logical errors, while the remaining 22 children ($f = 22$, $rf = 4.18$) had only syntax errors. The authors concluded that the participants in their study, regardless of their age, were guided to identify and correct errors in a way that was consistent with their intuition and logical reasoning, allowing them to adjust their programs and solve the perceived problem (Misirli & Komis, 2023).

This work is intended for educators, researchers and policymakers interested in improving coding education in K-12 settings. Our results distinguish computational thinking from algorithmic thinking, highlighting that, while both are essential for effective coding instruction, they serve different cognitive purposes. Additionally, we have explored the benefits of both plugged and unplugged programming approaches, noting that each method offers unique advantages for engaging students in coding concepts. However, a significant challenge identified in the literature is the lack of universal methods for assessing coding skills and cognitive development, which complicates the ability to consistently measure the effectiveness of various instructional strategies. This gap highlights the need for standardized assessment tools to better understand and improve coding education practices.

Discussion

The current inquiry is an attempt to review scientific articles that not only describe computational thinking in the K-12 curricula but also use cognitive or neuroscientific methodology. The author was particularly interested in papers published in the post-Covid era, as it has been a paramount turning point in modern education. Global social distancing efforts have led to a shift in education with increased screen time and reliance on technology for learning (Koh & Daniel, 2022). We acknowledge that this review is multi-faceted in nature, but this is because this is the state of current research in the field of K-12 programming lessons and computational thinking. The articles addressing these topics explore different aspects of the problem, and the definition itself of computational thinking remains vague. It is also unclear whether computational and algorithmic thinking can be improved and, if so, what methods should be implemented (Sun et al., 2021). The author reports several drawbacks that continue to persist when it comes to computational thinking and programming education. First, one of the major drawbacks is that coding lessons are not universally available in all countries. The United Kingdom introduced computing into its national curriculum as a compulsory subject in 2014, and France followed suit in 2016 (Grout & Houlden, 2014). Given the prevalence of technology in our lives, it is expected that more curricula will include coding in the coming years. However, these schools will face significant challenges, as little is known about coding as a cognitive ability.

Another contentious point is assessment. While it is possible to assess older students' understanding of program by asking them to complete a project, younger students are unable to perform such complex activities. One possible assessment method would be to test younger students' understanding by breaking down a coding-based task into its computational and algorithmic parts. There are challenges related to the mismatch between the types of skills assessed, the complexity of tasks, and the age groups, which makes it difficult to draw consistent conclusions. Assessing computational thinking is an evolving field, and ongoing research is underway to develop new assessment methods and tools (Tang et al., 2020). Unfortunately, there is currently no established way to accurately measure how well a student has learned computational thinking concepts. This lack of standardization may make it difficult for educators and researchers to accurately assess the effectiveness of their teaching methods or the efficacy of different learning materials. Furthermore, while there have been previous attempts to assess computational thinking concepts, such assessments have often failed to consider the role of visual engagement in the learning process. Eye-gaze measures, for example, can provide valuable insights into how students interact with various concepts and learning materials. However, these measures are often overlooked in traditional assessments of computational thinking concepts, leading to potential gaps in our understanding of how students learn and retain these fundamental skills (Jarodzka et al., 2021). In light of these challenges, there is a growing need for new and innovative approaches to measuring and assessing computational thinking concepts. By incorporating eye-

gaze measures and other advanced evaluation techniques, we can gain a more comprehensive understanding of how students interact with computational thinking concepts and identify areas for improvement in teaching methods and learning materials (Arslanyilmaz & Sullins, 2023). As technology continues to advance, coding will become increasingly important. By teaching children to code, we can help prepare them for a future where technology will play an even greater role in our lives (Sim & Bond, 2021). Finally, what seems to matter is children's attitude towards coding. According to a 2018 study that recruited 44 participants aged 8–17 years, their attitude towards coding impacted their gaze patterns during a coding exercise to a great extent (Papavlasopoulou et al., 2018).

Future studies should focus on neuroimaging and psychophysiological methods to expand our understanding of the effects of coding on brain development. Additional research efforts should be directed at defining the concepts of computational and algorithmic thinking, and identifying the cognitive processes most involved in both. Assessments of schoolchildren's computational and algorithmic thinking have been proposed, but at the time of publication of this article, none have been formally implemented. The PISA 2024 Science framework, for example, includes a set of competencies related to informatics that could be considered for inclusion within the PISA 2024 Science framework (OECD, 2024). These competencies include understanding the nature of problems that are worthy of an algorithmic solution, being able to assess the efficiency and correctness of simple algorithms, as well as defining, implementing, and validating programs and systems that model or simulate simple physical systems or familiar processes that occur in the real world or are studied in other disciplines.

Conclusion

Many countries around the world have integrated coding lessons into their educational curricula, but many more have yet to do so. Administrators of these future schools will face numerous challenges that many educators and researchers are already struggling with. Technology is of paramount importance in the modern era, and coding is called a new form of literacy. This poses many questions to the scientific community that researchers continue to ask. First, most teachers do not receive adequate training in coding and digital literacy. This, in turn, often leads to their lack of competence and confidence in teaching related subjects. Moreover, the lack of universal teaching platforms and methods creates additional challenges when it comes to implementing coding lessons in primary schools. The results show that many teachers do not have sufficient training in coding and digital literacy, resulting in low competence and confidence in teaching these subjects. Additionally, the absence of universal teaching platforms and methods complicates the implementation of coding lessons in primary schools.

In terms of research, longitudinal studies (over 6 months) on the cognitive skills of school-based coding lessons are limited due to various factors. One reason is the relatively recent integration of coding into school curricula, which means there has

not been enough time to conduct long-term studies that track students' cognitive development over several years. Additionally, the dynamic nature of technology and coding languages makes it difficult to design studies that can accurately measure the long-term impact of coding lessons on cognitive skills. Longitudinal studies, especially those involving psychophysiological methods, are needed to better understand the effect of code comprehension on brain development. Coding can be a great outlet for students who enjoy creative activities. By learning to code, children can develop skills like critical thinking, perseverance, and attention to detail.

References

Adamović, M. Đ., & Ivetić, D. V. (2024). Streamlined approach to 2nd/3rd graders learning basic programming concepts. *Entertainment Computing, 48,* 100604. https://doi.org/10.1016/j.entcom.2023.100604

Angeli, C. (2022). The effects of scaffolded programming scripts on pre-service teachers' computational thinking: Developing algorithmic thinking through programming robots. *International Journal of Child-Computer Interaction, 31,* 100329. https://doi.org/10.1016/j.ijcci.2021.100329

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

Arslanyilmaz, A., & Sullins, J. (2023). Eye-gaze data to measure students' attention to and comprehension of computational thinking concepts. International *Journal of Child-Computer Interaction, 38,* 100414. https://doi.org/10.1016/j.ijcci.2021.100414

Bacelo, A., & Gómez-Chacón, I. M. (2023). Characterising algorithmic thinking: A university study of unplugged activities. *Thinking Skills and Creativity, 48,* 101284. https://doi.org/10.1016/j.tsc.2023.101284

Bai, H., Wang, X., & Zhao, L. (2021). Effects of the problem-oriented learning model on middle school students' computational thinking skills in a python course. *Frontiers in Psychology, 12,* 771221. https://doi.org/10.3389/fpsyg.2021.771221

Carvalho, A. R., & Santos, C. (2022). Developing peer mentors' collaborative and metacognitive skills with a technology-enhanced peer learning program. *Computers and Education Open, 3,* 100070. https://doi.org/10.1016/j.caeo.2021.100070

Castelhano, J., Duarte, I. C., Couceiro, R., Medeiros, J., Duraes, J., Afonso, S., Madeira, H., & Castelo-Branco, M. (2022). Software bug detection causes a shift from bottom-up to top-down effective connectivity involving the insula within the error-monitoring network. *Frontiers in Human Neuroscience, 16,* 788272. https://doi.org/10.3389/fnhum.2022.788272

Chan, S.-W., Looi, C.-K., Ho, W. K., Huang, W., Seow, P., & Wu, L. (2021). Learning number patterns through computational thinking activities: A Rasch model analysis. *Heliyon, 7*(9), e07922. https://doi.org/10.1016/j.heliyon.2021.e07922

Chen, P., Yang, D., Metwally, A. H. S., Lavonen, J., & Wang, X. (2023). Fostering computational thinking through unplugged activities: A systematic literature review and meta-analysis. *International Journal of STEM Education, 10*(1), 47. https://doi.org/10.1186/s40594-023-00434-7

Cheng, Y.-P., Lai, C.-F., Chen, Y.-T., Wang, W.-S., Huang, Y.-M., & Wu, T.-T. (2023). Enhancing student's computational thinking skills with student-generated questions strategy in a game-based learning platform. *Computers & Education, 200,* 104794. https://doi.org/10.1016/j.compedu.2023.104794

Davis, D. K., & Zhu, F. (2022). Analysis of software developers' coding behavior: A survey of visualization analysis techniques using eye trackers. *Computers in Human Behavior Reports, 7,* 100213. https://doi.org/10.1016/j.chbr.2022.100213

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

Dúo-Terrón, P. (2023). Analysis of scratch software in scientific production for 20 years: Programming in education to develop computational thinking and STEAM disciplines. *Education Sciences, 13*(4), 404. https://doi.org/10.3390/educsci13040404

El-Hamamsy, L., Bruno, B., Audrin, C., Chevalier, M., Avry, S., Zufferey, J. D., & Mondada, F. (2023). How are primary school computer science curricular reforms contributing to equity? Impact on student learning, perception of the discipline, and gender gaps. *International Journal of STEM Education, 10*(1), 60. https://doi.org/10.1186/s40594-023-00438-3

Erümit, A. K., & Sahin, G. (2020). Plugged or unplugged teaching: A case study of students' preferences in the teaching of programming. *International Journal of Computer Science Education in Schools, 4*(1), 3–32. https://doi.org/10.21585/ijcses.v4i1.82

Gerosa, A., Koleszar, V., Tejera, G., Gómez-Sena, L., & Carboni, A. (2021). Cognitive abilities and computational thinking at age 5: Evidence for associations to sequencing and symbolic number comparison. *Computers and Education Open, 2,* 100043. https://doi.org/10.1016/j.caeo.2021.100043

Grout, V., & Houlden, N. (2014). Taking Computer Science and Programming into Schools: The Glyndŵr/BCS Turing Project. *Procedia - Social and Behavioral Sciences, 141,* 680–685. https://doi.org/10.1016/j.sbspro.2014.05.119

Harper, F. K., Caudle, L. A., Flowers, C. E., Rainwater, T., & Quinn, M. F. (2023). Centering teacher and parent voice to realize culturally relevant computational thinking in early childhood. *Early Childhood Research Quarterly, 64,* 381–393. https://doi.org/10.1016/j.ecresq.2023.05.001

Harsch, R. M., & Kendeou, P. (2023). Learning from refutation texts about scientific topics with analogical and causal explanations. *Contemporary Educational Psychology, 73,* 102172. https://doi.org/10.1016/j.cedpsych.2023.102172

Heim, G., & Wang, O. J. (2023). Block and unplugged programming can be mutually beneficial: A study of learning activities in a 6th grade class in Norway. *Frontiers in Education, 8,* 1138285. https://doi.org/10.3389/feduc.2023.1138285

Jarodzka, H., Skuballa, I., & Gruber, H. (2021). Eye-tracking in educational practice: Investigating visual perception underlying teaching and learning in the classroom. *Educational Psychology Review, 33*(1), 1–10. https://doi.org/10.1007/s10648-020-09565-7

Kirçali, A. Ç., & Özdener, N. (2023). A comparison of plugged and unplugged tools in teaching algorithms at the K-12 level for computational thinking skills. *Technology, Knowledge and Learning, 28*(4), 1485–1513. https://doi.org/10.1007/s10758-021-09585-4

Koh, J. H. L., & Daniel, B. K. (2022). Shifting online during COVID-19: A systematic review of teaching and learning strategies and their outcomes. *International Journal of Educational Technology in Higher Education, 19*(1), 56. https://doi.org/10.1186/s41239-022-00361-7

Kong, S.-C., Lai, M., & Li, Y. (2023). Scaling up a teacher development programme for sustainable computational thinking education: TPACK surveys, concept tests and primary school visits. *Computers & Education, 194,* 104707. https://doi.org/10.1016/j.compedu.2022.104707

McCray, J. S., & Chen, J.-Q. (2012). Pedagogical content knowledge for preschool mathematics: Construct validity of a new teacher interview. *Journal of Research in Childhood Education, 26*(3), 291–307. https://doi.org/10.1080/02568543.2012.685123

Mousa, M., & Molnár, G. (2020). Computer-based training in math improves inductive reasoning of 9- to 11-year-old children. *Thinking Skills and Creativity, 37,* 100687. https://doi.org/10.1016/j.tsc.2020.100687

OECD. (2024). PISA 2024 Science Strategic Vision Proposal [PDF]. Retrieved from https://www.oecd.org/pisa/publications/PISA-2024-Science-Strategic-Vision-Proposal.pdf

Panikratova, Y. R., Vlasova, R. M., Akhutina, T. V., Korneev, A. A., Sinitsyn, V. E., & Pechenkova, E. V. (2020). Functional connectivity of the dorsolateral prefrontal cortex contributes to different components of executive functions. *International Journal of Psychophysiology, 151,* 70–79. https://doi.org/10.1016/j.ijpsycho.2020.02.013

Papavlasopoulou, S., Sharma, K., & Giannakos, M. N. (2018). How do you feel about learning to code? Investigating the effect of children's attitudes towards coding using eye-tracking. *International Journal of Child-Computer Interaction, 17,* 50–60. https://doi.org/10.1016/j.ijcci.2018.01.004

Pérez-Marín, D., Hijón-Neira, R., Bacelo, A., & Pizarro, C. (2020). Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children? *Computers in Human Behavior, 105,* 105849. https://doi.org/10.1016/j.chb.2018.12.027

Ramful, A., Lowrie, T., & Logan, T. (2017). Measurement of spatial ability: Construction and validation of the spatial reasoning instrument for middle school students. *Educational Psychology, 37*(2), 193–211. https://doi.org/10.1080/01443410.2016.1148776

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

Robledo-Castro, C., Castillo-Ossa, L. F., & Hederich-Martínez, C. (2023). Effects of a compu-tational thinking intervention program on executive functions in children aged 10 to 11. *International Journal of Child-Computer Interaction, 35,* 100563. https://doi.org/10.1016/j.ijcci.2022.100563

Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior, 72,* 678–691. https://doi.org/10.1016/j.chb.2016.08.047

Scherer, R., Siddiq, F., & Sánchez-Scherer, B. (2021). Some evidence on the cognitive benefits of learning to code. *Frontiers in Psychology, 12,* 559424. https://doi.org/10.3389/fpsyg.2021.559424

Sim, G., & Bond, R. (2021). Eye tracking in Child Computer Interaction: Challenges and opportunities. *International Journal of Child-Computer Interaction, 30,* 100345. https://doi.org/10.1016/j.ijcci.2021.100345

Sun, L., Hu, L., & Zhou, D. (2021). Improving 7th-graders' computational thinking skills through unplugged programming activities: A study on the influence of multiple factors. *Thinking Skills and Creativity, 42,* 100926. https://doi.org/10.1016/j.tsc.2021.100926

Tang, X., Yin, Y., Lin, Q., Hadad, R., & Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education, 148,* 103798. https://doi.org/10.1016/j.compedu.2019.103798

Theodoropoulos, A., & Lepouras, G. (2022). Game design, gender and personalities in programming education. *Frontiers in Computer Science, 4,* 824995. https://doi.org/10.3389/fcomp.2022.824995

Thompson, J., & Childers, G. (2021). The impact of learning to code on elementary students' writing skills. *Computers & Education, 175,* 104336. https://doi.org/10.1016/j.compedu.2021.104336

Tran, Y. (2018). Computational thinking equity in elementary classrooms: What third-grade students know and can do. *Journal of Educational Computing Research, 57*(1), 3–31. https://doi.org/10.1177/0735633117743918

Zeng, Y., Yang, W., & Bautista, A. (2023). Teaching programming and computational thinking in early childhood education: a case study of content knowledge and pedagogical knowledge. *Frontiers in Psychology, 14,* 1252718. https://doi.org/10.3389/fpsyg.2023.1252718

Zhan, Z., He, L., Tong, Y., Liang, X., Guo, S., & Lan, X. (2022). The effectiveness of gamification in programming education: Evidence from a meta-analysis. *Computers and Education: Artificial Intelligence, 3,* 100096. https://doi.org/10.1016/j.caeai.2022.100096

Zhao, L., Liu, X., Wang, C., & Su, Y.-S. (2022). Effect of different mind mapping approaches on primary school students' computational thinking skills during visual programming learning. *Computers & Education, 181,* 104445. https://doi.org/10.1016/j.compedu.2022.104445

Article history:
Received 10 April 2024
Revised 15 June 2024
Accepted 16 June 2024

For citation:
Nikiforova, K. A. (2024). Coding lessons and the development of computational thinking in schoolchildren in the post-pandemic educational landscape: A review on research challenges and perspectives. *RUDN Journal of Psychology and Pedagogics, 21*(3), 858–886. http://doi.org/10.22363/2313-1683-2024-21-3-858-886

Conflicts of interest:
The author declares that there is no conflict of interest.

Bio note:
Kristina A. Nikiforova, Postgraduate Student, Junior Researcher, Scientific Center for Cognitive Research, Sirius University of Science and Technology (1 Olimpiyskiy Ave., Sirius, 354340, Krasnodar Region, Russian federation). ORCID: 0009-0000-4302-4406. E-mail: kkrisinger1990@gmail.com

DOI: 10.22363/2313-1683-2024-21-3-858-886
EDN: HBHWQQ
УДК 159.9.072.5

Обзорная статья

Уроки программирования и развития вычислительного мышления школьников в пост-пандемическом образовательном ландшафте: аналитический обзор вызовов и перспектив исследований

К.А. Никифорова

Научно-технологический университет «Сириус», *федеральная территория «Сириус», Российская Федерация*
✉ kkrisinger1990@gmail.com

Аннотация. Несмотря на быстрый рост технологий и постоянный спрос на IT-специалистов, когнитивные процессы, лежащие в основе вычислительного мышления и способности мозга понимать коды, остаются плохо изученными, особенно у детей младшего возраста. После пандемии Covid-19 школы многих стран включили уроки программиро-

вания в свои учебные планы. Программирование тесно связано со сложными когнитивными навыками в области STEM (наука, технологии, инженерия и математика), такими как вычислительное и алгоритмическое мышление. Однако в литературе существует путаница в отношении взаимосвязи между этими формами мышления и другими когнитивными навыками. Цели обзора: проанализировать методологии, используемые когнитивными учеными для изучения эффектов переноса навыков, полученных на уроках программирования, на развитие навыков вычислительного мышления у детей; рассмотреть современные исследования, направленные на изучение проблемы связи занятий программированием и развитием вычислительного мышления. Наши результаты показали, что многим учителям не хватает адекватной подготовки в области программирования и цифровой грамотности, что приводит к низкой компетентности и неуверенности в преподавании этих предметов. Кроме того, отсутствие универсальных платформ и методов обучения усложняет внедрение уроков программирования в начальных школах. Существует также нехватка лонгитюдных исследований (более шести месяцев), которые изучают когнитивные навыки, развиваемые в ходе уроков программирования. Решение этих проблем важно для улучшения образовательных практик.

Ключевые слова: когнитивные навыки, учебные планы K-12, вычислительное мышление, уроки программирования, школьники, нейронаука, COVID-19, ЭЭГ

История статьи:
Поступила в редакцию 10 апреля 2024 г.
Доработана после рецензирования 15 июня 2024 г.
Принята к печати 16 июня 2024 г.

Для цитирования:
Nikiforova K.A. Coding lessons and the development of computational thinking in schoolchildren in the post-pandemic educational landscape: A review on research challenges and perspectives // Вестник Российского университета дружбы народов. Серия: Психология и педагогика. 2024. Т. 21. № 3. С. 858–886. http://doi.org/10.22363/2313-1683-2024-21-3-858-886

Заявление о конфликте интересов:
Автор заявляет об отсутствии конфликта интересов.

Сведения об авторе:
Никифорова Кристина Андреевна, аспирант, младший научный сотрудник, Научный центр когнитивных исследований, Научно-технологический университет «Сириус», Российская Федерация, 354340, Краснодарский край, федеральная территория «Сириус», Олимпийский проспект, д. 1. ORCID: 0009-0000-4302-4406. E-mail: kkrisinger1990@gmail.com