Education Tech Research Dev (2024) 72:557–584
https://doi.org/10.1007/s11423-023-10312-2

RESEARCH ARTICLE

Different programming approaches on primary students' computational thinking: a multifactorial chain mediation effect

Lihui Sun[1] · Junjie Liu[2]

Accepted: 10 October 2023 / Published online: 18 October 2023

Abstract
This study investigated the effects of the single programming approach (plugged-in and unplugged) and the mixed programming approach (plugged-in-first and unplugged-first) on the computational thinking (CT) skills of first-grade students. However, focusing only on the programming learning approach itself is insufficient. Therefore, the influences of students' gender, programming experience, programming interest, and programming confidence factors on CT skills were also examined. 121 students from China were divided into four experimental and one control groups and engaged in the programming activities intervention for 10 weeks. The data consisted mainly of students' CT skill scores before and after the programming activities intervention. The results showed that both single and mixed programming approaches significantly improved students' CT skills, with the mixed programming approaches being more effective. Furthermore, the study found that the implementation of unplugged activities in the first stage attenuated the effects of programming experience. Furthermore, it was found that the unplugged-first programming approach was able to diminish the effect of students' programming experience on the development of CT skills and could be an essential condition to promote the development of equal CT skills. We also clarified the important role of programming interest and programming confidence in students' CT development. More importantly, a chain mediation effect of programming experience and programming interest between programming confidence and CT was also found. Finally, this study further discusses ideas and approaches for the future of CT education for primary school students and provides certain practical suggestions and insights for teachers and researchers.

Keywords Primary school students · Computational thinking · Plugged-in · Unplugged · Mixed programming approach

Lihui Sun and Junjie Liu have share the first authorship.

Extended author information available on the last page of the article

Introduction

Computational thinking (CT), deemed an indispensable foundational skill of digital intelligence, is fundamentally altering the way individuals approach problem-solving (Kong & Wang, 2020). Nowadays, educational reforms at the policy and curriculum scales are underway around the world intending to develop students' CT skills starting from the compulsory education level. As Bocconi et al. (2022) reported, 24 countries in the European Union have already incorporated CT into their curricula at the primary or middle school level. This is due to the digital benefits of CT, which not only helps students to fully understand the digital world around them but also to innovate based on it (European Commission, 2020). Consequently, this has motivated educators to search for the most effective way to develop CT at the compulsory education level, allowing every student to master it (Israel et al., 2015; Yadav et al., 2014). Actually, CT has become one of the important topics explored in the field of programming education. This is because CT involves abstraction, decomposition, algorithm design, and other competencies closely intertwined with programming. Therefore, programming has emerged as a pivotal means to develop students' CT skills (Liu et al., 2021). In the existing research, programming curricula are mostly implemented using plugged-in or unplugged programming (Sigayret et al., 2022). Unplugged programming activities are separated from electronic devices such as computers and represent concepts related to computer science through a series of logical tasks that deepen the learning of programming thinking. The advantage of the unplugged approach by reducing the abstraction of programming concepts, especially for students with no programming experience (Brackmann et al., 2017). Plugged-in programming usually involves interaction with programming software on the computer and allows for the development of students' programming skills by increasing their interest in programming through a gamified interactive interface (Ouahbi et al., 2015).

Plugged-in and unplugged programming are commonly used at the K-12 education level to develop students' CT skills. Plugged-in programming activities usually require students to have some basic knowledge of computer operations, and students need to embrace certain programming concepts in the process of coding. Therefore, it is usually recommended for implementation in upper elementary and middle school grades (Sigayret et al., 2022). By contrast, unplugged programming activities are usually carried out in primary schools, where on the one hand they can escape the issues that occur with exposure to computers, and on the other hand, they are not even influenced by the teacher's teaching level. Generally, most studies have used plugged-in or unplugged programming alone to develop CT skills (Polat & Yilmaz, 2022). It has been demonstrated that both single plugged-in programming activities and unplugged programming activities significantly enhance students' CT skills (del Olmo-Muñoz et al., 2020; Sun et al., 2021b). As research in programming education advances, the effects of combinations of plugged-in and unplugged programming approaches have attracted the attention of researchers. del Olmo-Muñoz et al. (2020) found that implementing unplugged activities before plugged-in activities improved CT skills more than fully plugged-in activities. Similarly, several researchers have demonstrated that this mixed programming approach significantly improves the CT skills of 6th and 7th graders (Hermans & Aivaloglou, 2017; Sun et al., 2021b). There may be current stereotypes about the implementation of the mixed programming approach, and it is often assumed that implementing the unplugged activities first is more effective (del Olmo-Muñoz et al., 2020). However, there is a lack of systematic empirical evidence on the impact of the sequences of implementation of the mixed programming approach on students' CT skills. In addition, relevant studies have mainly focused on the upper primary and junior secondary levels. According to Hsu et al. (2018), the cognitive abilities and knowledge structures of students at different ages vary greatly. In other words, it is difficult to replicate the same approach to CT development at different grade levels. Therefore, to better promote programming education, there is a need to systematically explore the effects of mixed programming approaches with different combination sequences on the CT skills of first-grade students.

Based on this, we designed a quasi-experimental study containing four experimental groups and one control group. The four experimental groups were designed with the single plugged-in and unplugged programming approach and the mixed programming approach with combined plugged-in and unplugged. Moreover, two different combination sequences were designed, i.e., the plugged-in-first and the unplugged-first programming approaches. An empirical study was conducted to compare the effects of the single and mixed programming approaches on students' CT skills, in addition to exploring the differences caused by the combination sequence of the mixed programming approaches. Ultimately, it was possible to determine the most effective way to develop CT skills in first-grade students, filling the gap in the current research field. Meanwhile, we also focused on the variability of students' gender when developing CT skills across different programming approaches (Webb et al., 2017). Students' programming experience was also an important factor influencing CT (Lye & Koh, 2014). However, the question of whether it can contribute to students' CT skills or play distinct roles in different programming approaches still needs to be answered. More importantly, it is worthwhile to investigate the factors that influence students' learning process, for example, whether students have interest in programming (Kong et al., 2018), or whether students have confidence in programming (Chiu & Klassen, 2010). Students with interest and confidence in programming generally have a strong intrinsic motivation and are constantly driven to explore themselves. This is accompanied by heightened self-efficacy and enhanced programming proficiency (Weber et al., 2005). However, these influences have not received much attention from researchers. Additionally, the potential relationships and influence pathways among these influences need further investigation. Therefore, we constructed a chain mediation effect model between programming experience, programming interest, programming confidence, and post-pre-test of CT. The effects of these influences on students' CT skills were explored further.

Literature review

Definition of computational thinking

CT has become the core skill that everyone should have in the 21st century, but there is no consensus in academia on the definition of CT (Lye & Koh, 2014; Wong & Cheung, 2020). Academics and institutions have also continued to redefine the concept of CT and applied it in practice as time evolves (Kalelioglu et al., 2016). The first introduction of the term computational thinking can be traced back to 1980 when Papert (1980) intended to develop powerful ideas in students through LOGO programming, but at this point, it was still "computer thinking". Wing (2006) clarified the significance of CT from the perspective of computer science, pointing out that CT is the use of computer-related concepts to design solutions to problems and to understand human behavior. At this point, CT has attracted widespread attention from the academic community, leading to a wave of research on CT. Several years later, Wing (2008) redefined CT as a way of thinking about the problem-solving process. Since then, Computer Science Teachers Association (CSTA) and International Society for Technology in Education (ISTE) (2011) have provided more explicit operational definitions, indicating that CT covers dimensions such as organizing data, data analysis, and using algorithms to automate problem-solving. Many scholars have also researched the components of CT. For example, Brennan and Resnick (2012) reconstructed the conceptual framework of CT based on Scratch programming activities. This includes computational concepts that learners use during programming, computational practices for developing or debugging projects, and computational perspectives for viewing things around them. Kalelioglu et al. (2016) proposed CT as the framework component of the problem-solving process and that it could be applied to different scenarios rather than only for solving computational tasks. Selby and Woollard (2013) focused on the thoughtful characteristics of CT, arguing that CT is a mental process involved in human problem-solving. Although there is no consensus on the conceptual definition of CT, CT has always been inseparable from the set of core skills involved in the problem-solving process, such as abstraction, deconstruction, algorithms, evaluation, and generalization (Barr & Stephenson, 2011; Kalelioglu et al., 2016; Selby & Woollard, 2013). Therefore, we used the CT framework that includes the above core skills and tested the students' CT skills through the Bebras Challenge project.

Unplugged and plugged-in programming teaching methods

In current research trends, students' CT skills are improved mainly through plugged-in and unplugged programming activities. Unplugged programming involves learning computer science through outdoor activities, card games, or puzzles, separated from the computer (Brackmann et al., 2017). For example, Kim et al. (2014) used paper and pencil programming, which is programming in the form of symbols and flowcharts to help students understand data structure algorithms and improve CT skills. Other researchers implemented a quasi-experimental study with upper primary and seventh-grade students in middle school and revealed that students who participated in unplugged programming activities had significantly improved CT skills, especially problem-solving and logical thinking skills, compared to non-participating students (Brackmann et al., 2017; Sun et al., 2021a). Li et al. (2022) analyzed 29 pieces of literature related to unplugged programming using meta-analyses, where the findings indicated that unplugged programming activities are more applicable to primary school students. It is worth noting that while unplugged programming activities provide students with an understanding of programming-related concepts with a simple approach, they may be removed from the practice of CT skills in programming (Bell & Vahrenhold, 2018). In addition, Shang et al. (2023)d ez-López et al. (2016) used robotics programming and Scratch programming to conduct upper primary students with plugged-in activities, and both found that students' CT skills, and mastery of programming concepts, significantly improved. Therefore, the effectiveness of plugged-in programming activities has also been demonstrated.

Despite the different learning approaches of plugged-in and unplugged programming, both aim to improve CT skills by promoting students' understanding of programming concepts or enhancing their abilities in algorithms, deconstruction, and problem-solving. In other words, plugged-in and unplugged programming are intended to be congruent and complementary, not opposed to each other. Saxena et al. (2019) conducted two stages of programming activities in kindergarten which revealed that the unplugged programming activities implemented in the first stage facilitated the concepts of pattern recognition, sequencing, and algorithm design to pave the way for the plugged-in programming activities in the second stage. Sun et al. (2021b) implemented unplugged-first and plugged-in-first programming activities in seventh grade which showed that students' CT skills improved significantly after both programming approaches intervention, and students who implemented unplugged-first programming activities intervention had the most improvement in CT skills. Therefore, the combination of plugged-in and unplugged programming approaches may be more effective in enhancing students' CT skills. However, the existing experimental design of mixed programming is simple, focusing mainly on the comparison of the effects of the single unplugged or plugged-in programming approaches with the mixed programming approaches, and the effects caused by the combined sequence of the mixed programming have not yet been explored in depth. In addition, most research on the mixed experiments of unplugged and plugged programming has focused on the upper primary and middle school levels, and fewer experimental studies have been conducted in the early primary grades. Due to differences in educational environments and individual students, the existing research findings cannot be directly transferred (Hsu et al., 2018). Therefore, the effects of these programming approaches on students' CT skills in the lower primary grades still need to be further explored.

Multiple factors influencing computational thinking

Gender

Gender, as the most prevalent demographic factor, has been extensively studied in CT and programming education. For example, Mouza et al. (2020) found that after the same intervention of plugged-in programming sessions, boys' CT skills were significantly higher than girls' in Grades 4–6. In fact, the stereotype that girls are inferior to boys in CT and programming is always held (Passey, 2017). However, some researchers have found that after the plugged-in programming intervention, girls were able to achieve the same level of CT skills as boys (Sun et al., 2022a), and even showed higher CT skills than boys (Atmatzidou & Demetriadis, 2016). Sun et al. (2021a) also found that unplugged programming activities were able to provide boys and girls with equitable opportunities for CT development. These disparities in findings may be attributed to the manner in which CT is cultivated. However, most of the current research has focused on the effect of gender on the single programming approach and little attention has been paid to the effect of gender on the mixed programming approach of plugged and unplugged.

Programming experience

In the learning process, knowledge acquisition was transformed by experience (Kolb et al., 2014). As Papert (1980) argued, children actively participate in the construction of knowledge through previous experiences. Similarly, programming experience is also usually a factor to be considered in the learning process of plugged-in and unplugged programming. Some researchers have found that students with programming experience have an easier time completing similar programming tasks and performing higher CT skills (Fessakis et al., 2013; Sun et al., 2021b). This is probably attributed to the fact that the neural activity processes of students with programming experience are more conducive to CT skill development (Helmlinger et al., 2020). It is evident that programming experience is emerging as an important factor in widening the differences in CT skills among students. In addition, Sun et al. (2022b) found that appropriate programming instructional approaches can reduce the impact of students' previous programming experiences on the development of CT skills. Therefore, we should consider the impact of students' programming experience in different programming learning approaches, so that students with different programming experiences can better engage in programming learning.

Programming interest

It was as early as the 19th century that researchers noticed the importance of interest in the learning process (Hidi, 2006). As the study progressed, the researcher found that interest both contributed to sustained learning and was a key factor in student achievement (Dewey, 1913; Schiefele, 2008). Similarly, students with programming interests are more motivated to explore the programming learning process, which helps to produce positive learning outcomes (Deci et al., 2017). Meanwhile, they will spend more time finding solutions to complex programming problems, thus improving their CT skills. Students without programming interest will resist participating in programming activities, thus creating a vicious circle in terms of CT and programming skills (Beyer, 2014). Kong et al. (2018) constructed a structural equation about the programming interest of primary school students and found that students with high programming interest would regard programming as a meaningful activity, which is conducive to the improvement of CT skills. In other words, students with different programming interests may have different CT skills. However, different programming approaches may have different effects on primary students. For example, computer-based plugged-in programming takes time for students to get used to, which can reduce the positive CT impact of programming interest (Sigayret et al., 2022). Therefore, future research needs to explore the effects of programming interest in different programming approaches.

Programming confidence

Students' intrinsic psychological factors affect the development of their CT skills, and confidence is an important psychological factor. Within the field of programming education, confidence enables students to believe that they are capable of performing certain programming tasks or acquiring certain programming knowledge well (Bandura & Wessels, 1994). Jong et al. (2020) also found that developing primary school students' confidence in facing difficult programming problems is as important as improving CT skills. This is because students' ability to solve complex problems is closely linked to their level of programming confidence. Tsai et al. (2020) also found that in computer programming, students' increased programming confidence contributes to the acquisition of algorithmic skills, logical thinking skills, and thus better development of CT skills. Whilst the important role of programming confidence is clear, teachers who use an inappropriate approach to teaching programming may not have positive impacts on students' CT skills (Gunbatar & Karalar, 2018). Therefore, there is a need to further explore the specific role that programming confidence plays in different programming approaches.

The interactions between programming experience, programming interest, programming confidence and computational thinking

As mentioned above, programming experience, interest, and confidence are all closely related to students' CT skills. Generally, people who maintain confidence in programming have an active interest in it. According to Kong et al. (2018), students with high levels of programming confidence find programming activities more interest, which promotes positive learning outcomes. Mason and Rich (2020) also found significant correlation between programming interest and confidence. In addition, when students utilize 'prior knowledge' to construct CT skills, the drive generated by confidence can facilitate positive adjustments (Papert, 1980). Also, students with programming experience tend to invest more interest and energy in similar programming activities. This means that there may be correlations between programming confidence, programming experience, and programming interest. Therefore, to deeply explore the influence mechanism between them, we developed a chain mediation effect model.

Research objectives

In conclusion, the main purpose of this research is to investigate programming approaches that are appropriate for the development of CT skills of first-grade students. The four experimental groups in this study implemented the single programming activities (plugged-in and unplugged) and the mixed programming activities (plugged-in-first and unplugged-first) and measured students' CT skills before and after the programming activities intervention. Meanwhile, we also considered multiple factors that might influence students' CT skills during instruction, including students' gender, previous programming experience, and interest and confidence in programming. More importantly, a chain mediation effect model of programming experience and programming interest was constructed to explore the mechanism of influence between these factors and students' CT post-pre-test. The research questions of this study are as follows:

RQ1 Can single programming approaches (plugged-in, unplugged) and mixed programming approaches (plugged-in-first, unplugged-first) improve the CT skills of primary school students?

RQ2 Can different programming approaches be influenced by students' gender, programming experience, programming interest, and programming confidence in improving their CT skills?

RQ3 Is there a chain mediation effect of students' programming experience and programming interest in the effect of programming confidence on the CT post-pre-test?

Method

Research design

Based on the purpose of this study and considering that the study was conducted in an authentic educational environment, it was not possible to completely disrupt the existing classes and randomize them. Therefore, all groups in this study followed the original class configuration. Ultimately, five classes were randomly selected to conduct a quasi-experimental study lasting 10 weeks. It included four experimental groups and one control group. The programming activities throughout the experiment are divided into two stages, each with 4 lessons, for a total of 8 lessons. Each stage had 4 plugged-in programming sessions or unplugged programming sessions, each of which lasted 40 min. The plugged-in group implemented 8 sessions of plugged-in programming activities and the unplugged group implemented 8 sessions of unplugged programming activities. The other two groups, the plugged-in-first and unplugged-first groups, which received interventions in a mixed programming approach, received four plugged-in programming sessions and four unplugged programming sessions. In the first stage, the plugged-in-first group conducted 4 sessions of plugged-in programming, while the unplugged-first group conducted 4 sessions of unplugged programming. In the second stage, the plugged-in-first group then conducts 4 unplugged programming sessions; while the unplugged-first group conducts 4 plugged-in programming sessions.

It should be noted that both the experimental and control groups were taught in the IT curriculum. The students in the control group were still taught the regular IT curriculum and the students in the experimental group received the corresponding programming activities instead. China's IT Curriculum Standards for Compulsory Education clearly state that CT skills are one of the four core literacies that must be developed (Ministry of Education, 2022). Therefore, the regular IT curriculum also includes CT concepts and skills. Moreover, CT skills were measured for all students, and demographic information was collected before the first stage of the programming sessions were conducted. At the end of the second stage of programming sessions, students' CT skills were measured again. Quantitative results were analyzed to investigate the impact of the programming activities intervention on students' CT skills. The research route of this study is shown in Fig. 1.

Participants

The 121 students in this study were all first-grade students in a primary school in a Chinese city. They all participated in this experiment voluntarily and with parental consent. The study was divided into 5 groups with 121 students (63 boys, 58 girls) participating in the experiment with an average age of 7.50 years (SD = .824). 4 experimental groups

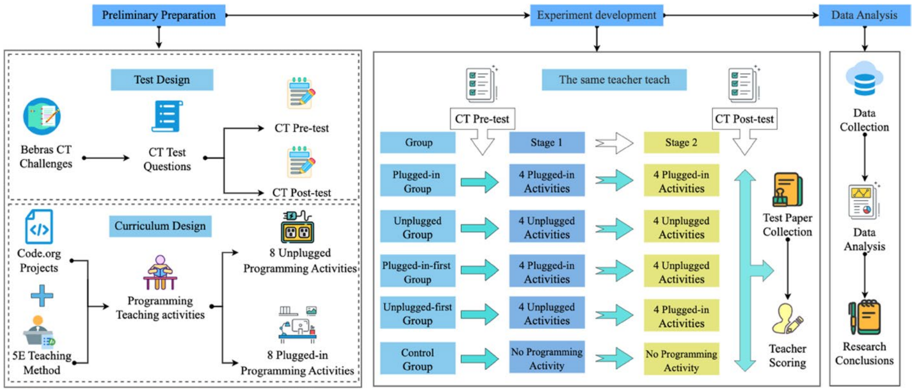


Fig. 1 Research route map

implemented the plugged-in programming activities (13 boys, 12 girls), the unplugged programming activities (14 boys, 10 girls), the plugged-in-first programming activities (11 boys, 11 girls), and the unplugged-first programming activity (10 boys, 12 girls). The control group (15 boys, 13 girls) was taught the regular IT curriculum and had no additional plugged-in and unplugged programming activities. Additionally, we collected students' demographic information through a questionnaire, which mainly included programming experience, programming interest, and programming confidence (1 represents have, 2 represents no). Table 1 shows the demographic information of the students in each group. 43.0% of the students had participated in programming activities, indicating that most of the students had not yet been exposed to programming learning. 60.3% of the students reported interest in programming. However, only 46.3% of the students had confidence in programming, indicating that most of the students may feel that programming is difficult.

Research instruments

Computational thinking test

The introduction of CT into K-12 education has also contributed to the development and research of CT assessment tools. Since the concept of CT has not reached consensus in the academic community, many scholars have developed different assessment approaches based on different conceptual frameworks, such as the Computational Thinking Scale (CTS) (Korkmaz et al., 2017), the automated assessment tool Dr. Scratch (Moreno-León et al., 2015), and the Bebras CT Challenge (Izu et al., 2017). Among them, the Bebras CT Challenge questions can assess students' problem-solving skills in different contexts, and numerous studies have validated its effectiveness in measuring students' CT skills (del Olmo-Muñoz et al., 2020). The two sets of CT skills tests in this study were taken from the Bebras CT Challenge, which was divided into different age groups: Kits (age 6–8), Castors (age 8–10), Benjamins (age 10–12), and Cadets (age 10–12). − 12), Cadets (age 12–14), Juniors (age 14–16), and Seniors (age 16–18). Simultaneously, the questions were classified according to the difficulty level as A level (easy), B level (medium), and C level (difficult). As the experimental participants in this study were first-grade students, aged between 6 and 8 years old. Therefore, the test questions in this study were selected from previous Kits level questions to test the students' CT skills before and after the programming sessions intervention. Eventually, each set consisted of 12 questions from the Kits level, including the 6 "A" level questions, 4 "B" level questions, and 2 "C" level questions. Moreover, different scores were assigned according to the difficulty of the questions, with 1 score for "A" level questions, 2 scores for "B" level questions, and 3 scores for "C" level questions, for a total of 20 scores. To determine the validity of the two developed tests, the reliability of the questions was checked using the IRT package in R language. The results indicated that Cronbach's alpha coefficient was .713 for the CT pre-test questions and .732 for the CT post-test questions (both > .700). This demonstrated that both sets of questions exhibited good reliability and validity (Cronbach & Meehl, 1955). Example questions for the CT pre-test and post-test are shown in Figs. 2 and 3.

Code.org

With the popularization of programming education, there is an increasing variety of platforms used to teach programming, such as Scratch, ScratchJr, Alice, and even various

Table 1 Demographic information of students

Group	N	Gender		Programming experience		Programming Interest		Programming Confidence	
		Boys	Girls	Have	No	Have	No	Have	No
Plugged-in	25	13	12	8	17	14	11	9	16
Unplugged	24	14	10	10	14	16	8	10	14
Plugged-in-first	22	11	11	11	11	14	8	10	12
Unplugged-first	22	10	12	8	14	14	8	15	7
Control Group	28	15	13	15	13	15	13	12	16
Total	121	63 (52.1%)	58 (47.9%)	52 (43.0%)	69 (57.0%)	73 (60.3%)	48 (39.7%)	56 (46.3%)	65 (53.7%)

问题：商店里来了几只动物，其中一只偷偷地在商店里面走来走去，留下了一串脚印。**请你猜猜这是哪只小动物的脚印呢？（　　）**

Question: Several animals came to the store, and one of them secretly walked around inside the store, leaving a series of footprints. **Please guess which animal's footprints are these ? (　　)**

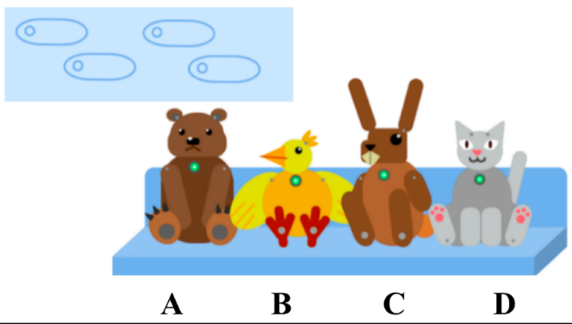
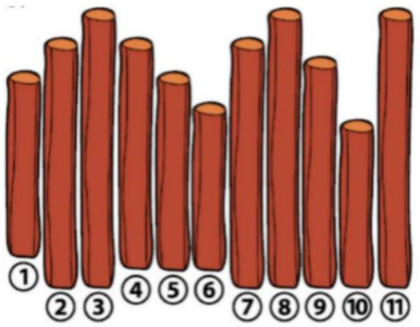


Fig. 2 Pre-test questions of computational thinking skills ("A" level)

问题：小海狸要用一堆木头盖房子，他将每根木头摆放整齐，并比较了木头的长短。如果该木头比它右边的木头长，那么这跟木头就会被用来制作屋顶。**小朋友猜猜哪些木头被用来制作屋顶了呢？（　　）**

Question: Little beaver wants to build a house with a pile of wood, he placed each log neatly and compared the length of the logs. If the log is longer than the log to its right, then that log will be used to make the roof. **Guess which pieces of wood were used to make the roof ? (　　)**



A.④⑤①②③　　B.④⑨⑥②⑤

C.③④⑤⑧⑨　　D.①③④⑤②

Fig. 3 Post-test questions of computational thinking skills ("B" level)

unplugged programming methods. In particular, the Code.org platform uses drag-and-drop programming blocks to teach students programming. It attracted the attention of many programming educators as soon as it was launched in 2013 (Code.org, 2013). Code.org has designed different course content for students aged 4–18, including an online version of plugged-in programming activities and an offline version of unplugged programming activities (Kalelioğlu, 2015). These programming courses covered programming concepts such as sequences, loops, events, conditional statements, functions, etc. As mentioned by Kale and Yuan (2020), Code.org enables students to understand the correspondence between programming blocks and visual objects by manipulating them. Students' CT skills and problem-solving abilities improved after the Code.org curriculum intervention. Kalelioğlu (2015) also observed that Code.org has a good motivational function that helps students to master computer science, especially for those who have no previous programming experience. Therefore, this study developed plugged-in and unplugged programming sessions for first-grade students based on the Code.org project.

Instructional design

As described earlier, we designed 8 plugged-in programming sessions and 8 unplugged programming sessions, with 40 min each, for the single plugged-in and unplugged programming instruction. Moreover, four lessons were selected from each of them to form the mixed plugged-in-first and unplugged-first programming sessions, and only the order of the combination of sessions was changed. The effect of different programming approaches on the CT skills of first-grade students was investigated. The plugged-in programming sessions were selected from the "Crash Course for Preschoolers" (https://studio.code.org/s/pre-express-2019), and the unplugged programming sessions were selected from the "Unplugged Version of Computer Science Basics" (https://code.org/curriculum/unplugged). The programming activities we have chosen cover sequences, algorithms, loops, and events, and they have been proven suitable for first-grade students. It is important to note that the Code.org platform offers combinations and substitutions between different forms of programming activities. Hence we ensured that the programming knowledge taught in each session was the same, only with various teaching approaches (del Olmo-Muñoz et al., 2020). The programming sessions of this study were organized as shown in Fig. 4.

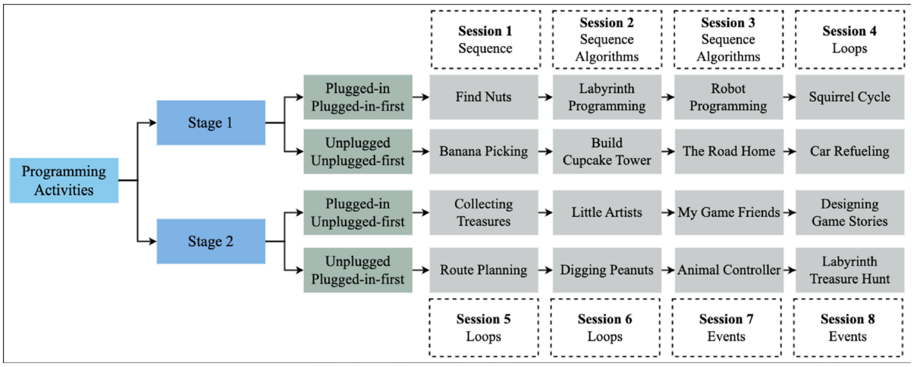


Fig. 4 Programming sessions arrangement map

The content of the control group is designed according to the standards of the IT curriculum, which mainly includes experiencing the process of problem-solving using digital devices and knowing the various ways of expressing information. For the given task, it is possible to identify the main steps in the implementation of the task and to express them in the form of graphical symbols, etc.

After determining the teaching contents, a suitable teaching model is needed to organize the teaching and learning process. The traditional model of teaching programming is usually lecture-based and students are passive recipients of programming knowledge (Tsai, 2019). This may lead to a lack of learning motivation, which is detrimental to the development of students' CT skills. Therefore, to fully develop students' CT skills, we designed the teaching activities according to the "5E" teaching model, which includes five stages: engagement, exploration, explanation, elaboration, and evaluation. The "5E" teaching model returns the main body of the classroom to the students, providing them with the opportunity to actively explore programming practices and motivating them to solve programming problems (Bybee et al., 2006). In addition, the "5E" teaching model can help students understand programming concepts in a step-by-step manner through inquiry, thus better helping them master CT skills. Gao and Hew (2021) also found that courses designed according to the "5E" instructional model significantly improved primary students' understanding of CT concepts and programming problem-solving skills. Eventually, we designed an instructional guide for each lesson, including the theme, key instructional content, instructional objectives, and instructional focus. An example of the instructional guide is shown in Table 2.

Intervention fidelity

Intervention fidelity refers to the degree to which the intervention is implemented as intended, and the key to effective experimental interventions depends on increasing the fidelity of the intervention. If the intervention fidelity is not valued, it may lead to a diminished effectiveness of the intervention, making it difficult to conclude reliably (Clements et al., 2015). In this study, we considered intervention fidelity across all aspects. First, to ensure the fidelity of the course implementation, the same teacher experienced in teaching programming in our research team taught the experimental and control groups. Second, the curriculum guidelines for both the experimental and control groups were designed following the "5E" teaching model, and teachers were trained to implement the curriculum in strict accordance with the guidelines. In addition, the content was carefully designed to ensure that the experimental groups were taught the same content in each lesson, differing only in the way it was implemented (del Olmo-Muñoz et al., 2020).

Data analysis

Data analysis for this study was completed in SPSS 25.0 and Amos Graphics 26.0. In the current study, the data were primarily the students' CT test scores, separately collected before and after the 8-week programming activity intervention. In addition, demographic information about the students was collected before the programming activities intervention. First, descriptive analyses were used to present the pre-test and post-test scores of students in the plugged-in, unplugged, plugged-in-first, unplugged-first, and control groups, to observe the changes in students' CT scores. Second, paired-samples T-tests were used to investigate the differences in students' CT skills before and after the programming activities

Table 2 Example of instructional guide for programming activities

Content	Description	Example
Teaching themes	The theme of this lesson	Building Cup Tower
Teaching content	Programming knowledge or programming activities covered in each plugged-in or unplugged programming session	Learning and practice of "sequences" and "algorithms" in programming concepts
Teaching objectives	Programming knowledge or concepts that students need to master in each session	Understand the meaning of programming concepts "sequence" and "algorithm", and be able to design the sequence of building paper cups and then write the correct programming instruction symbols
Teaching highlights	Programming concepts that students need to focus on or have difficulty understanding	Understand the meaning of "algorithm" in programming concepts; design and write correct programming instruction symbols
Activity preparation	Materials that students may need in the course of their studies	Paper cups, "Building Cup Tower" activity paper, pens
Engagement (8 min)	Teachers can provide fun activities to engage students' interest	The teacher shows the "Building Cup Tower" activity paper used in this lesson and tells students that they need to write symbolic instructions to move the paper cups
Exploration (20 min)	In this session, students are required to explore and complete the tasks of the programming activity on their own	Based on their understanding of the symbols on the programming instruction cards, students write a series of programming instruction symbols as required by the "Building Cup Tower" programming task
Explanation (7 min)	At the end of the exploration, students are required to explain the programming concepts from the programming activity based on their understanding. Subsequently, the teacher gives an accurate and well-defined definition of the programming concept, as well as a correct understanding	The teacher identifies the concepts of "sequence" and "algorithms" involved in the programming activity after the students have completed the task. Students experience the programming concepts in the context of the teacher's explanations and activities
Elaboration (3 min)	Teachers guide students to think about how these programming concepts relate to real-life situations	The teacher also asks questions to guide students to relate the "algorithms" to real-life situations. For example, ask students to describe what the process of cleaning at home is like
Evaluation (2 min)	Teachers need to observe students' understanding and application of programming concepts	Students answer questions posed by the teacher, and the teacher records and evaluates the students' responses

intervention to validate the effectiveness of the different programming approaches. Further, independent sample T-tests were used to explore the variability between the different programming approaches and to visualize the variability between the groups' pre-test, post-test, and post-pre-test of CT. To explore the most suitable programming approach for the development of CT skills in first-grade students. Finally, Multifactor Analysis of Variance (MANOVA) was used to explore the effects of students' gender, programming experience, programming interest, and programming confidence on CT skills. The above data analyses were done in SPSS 25.0. More importantly, a chain mediation effect model of programming experience and programming interest was established in Amos Graphics 26.0 while exploring the effect of students' programming confidence on the CT post-pre-test. It provides evidence to support our study for the mechanisms of influence of these factors.

Results

Comparison of CT skill scores before and after instructional intervention

First, we began by exploring the effectiveness of the various programming approaches in RQ 1. Table 3 shows the pre-test and post-test scores of CT skills for students in the single programming approach (plugged-in and unplugged) and the mixed programming approach (plugged-in-first and unplugged-first), control group. The mean scores of students in the pre-test were, from highest to lowest, for the plugged-in group (M = 11.08, SD = 3.16), the plugged-in-first group (M = 10.55, SD = 2.84), the unplugged group (M = 9.83, SD = 3.67), the control group (M = 9.79, SD = 2.54), and the unplugged-first group (M = 9.55, SD = 2.37). Figure 5 shows the results of independent sample T-tests between the pre-test and post-test, and the post-pre-test of CT skills for each group. It can be seen that there was no significant difference between the pre-test of CT for each group ($p > .05$). In summary, it was possible to directly compare the differences between the students' CT skills of post-test and post-pre-test.

As shown in Table 3 for each group on the CT post-test, the highest CT scores after the instructional intervention were found in the unplugged-first group (M = 17.91, SD = 1.80), followed by the plugged-in-first group (M = 17.23, SD = 2.79). Next was the unplugged group (M = 13.25, SD = 4.31) and the plugged group (M = 12.36, SD = 3.12). The control group had the lowest score (M = 10.21, SD = 2.69). To explore the effect of different programming approaches, paired-samples T-tests were conducted on the post-test and pre-test

Table 3 Pre-test and post-test scores for each group of computational thinking

Group	Pre-test			Min_{CT}	Max_{CT}	Post-test		Min_{CT}	Max_{CT}
	N	M	SD			M	SD		
Plugged-in	25	11.08	3.16	6	16	12.36	3.12	7	19
Unplugged	24	9.83	3.67	3	17	13.25	4.31	5	20
Plugged-in-first	22	10.55	2.84	5	18	17.23	2.79	11	20
Unplugged-first	22	9.55	2.37	4	13	17.91	1.80	14	20
Control	28	9.79	2.54	4	15	10.21	2.69	6	17

Min_{CT} represents the lowest CT score for the group; Max_{CT} represents the highest CT score for the group

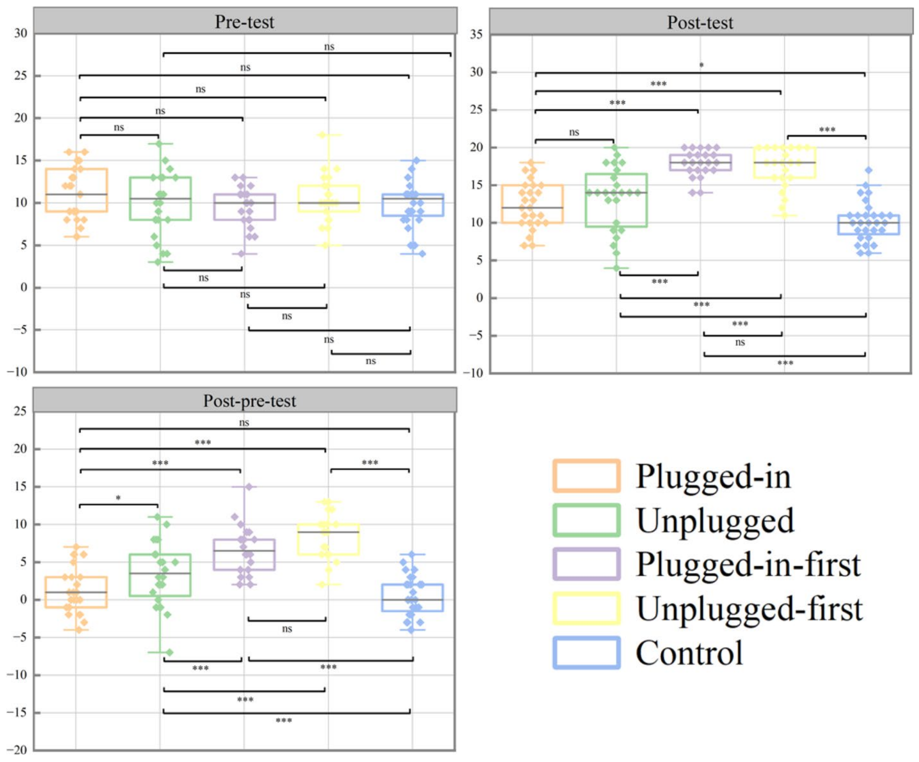


Fig. 5 The independent sample T-test for each group pre-test, post-test, and post-pre-test

Table 4 Paired samples T-test for each group of computational thinking

Group	Post-pre-test						
	N	M	SD	t	df	p	Cohen's d
Plugged-in	25	1.28	3.02	2.118*	24	.045	.424
Unplugged	24	3.42	4.21	3.975**	23	.001	.811
Plugged-in-first	22	6.68	3.17	9.893***	21	.000	2.109
Unplugged-first	22	8.36	2.92	13.433***	21	.000	2.864
Control	28	.43	2.70	.840	27	.408	.159

* = p < .05, ** = p < .01, *** = p < .001

scores of each group for CT, and the results are shown in Table 4. It was found that the plugged-in (t = 2.118, p < .05), unplugged (t = 3.975, p < .01), plugged-in-first (t = 9.893, p < .001), and unplugged-first (t = 13.433, p < .001) programming approaches were able to significantly improve students' CT skills, while the control group (t = .0840, p > .05) students' CT skills were not significantly improved. The unplugged-first group improved CT scores the most (M = 8.36, SD = 2.92), followed by the plugged-in-first programming group (M = 6.68, SD = 3.17). Following again were the unplugged programming group (M = 3.42, SD = 4.21) and the plugged-in programming group (M = 1.28, SD = 3.02). Meanwhile, the unplugged group, plugged-in-first group, and unplugged-first group all had large effect sizes, and the plugged-in group had medium effect sizes. In summary, both the mixed programming approach (Plugged-in-first and unplugged-first) and the single programming approach (Plugged-in and unplugged) proved to be effective in enhancing students' CT skills.

To further investigate which programming approach was more effective in enhancing students' CT skills, independent sample T-tests were conducted on the post-pre-test of CT for each group, and the results are shown in Fig. 5. When comparing the single programming approaches, we found significant differences between the plugged-in and unplugged groups ($p < .05$) and students in the unplugged group had higher CT skills ($M = 3.42$, $SD = 4.21$). While comparing the mixed programming approaches, there was no significant difference between the plugged-in-first and unplugged-first groups ($p > .05$). Furthermore, comparing the mixed programming approaches with the single programming approaches, we found that the mixed programming approaches (plugged-in-first, unplugged-first) were significantly better than the single programming approaches (plugged-in, unplugged) ($p < .001$). In summary, in terms of improving students' CT skills, the plugged-in-first and unplugged-first programming approaches were equally effective, but both were superior to the plugged-in and unplugged programming approaches. That said, plugged-in-first and unplugged-first programming approaches improve CT better than the always plugged-in or unplugged approaches.

Exploring multiple influencing factors in programming education

We used MANOVA to explore the effects of students' gender, programming experience, programming interest, and programming confidence on students' CT pre-test and CT post-pre-test scores in different groups. The independent variables were students' gender, programming experience, programming interest, and programming confidence, and the dependent variables were students' CT pre-test scores and CT post-pre-test scores. In addition, we tested the CT scores for each classification for all variables, and they all conformed to a normal distribution, so the MANOVA could be conducted. The results of MANOVA are shown in Table 5.

As seen in Table 5, there was no four- factor and three- factor interaction ($p > .05$) in any of the groups. We only observed significant interaction effects between gender and programming experience ($F = 7.471$, $p < .05$), and gender and programming confidence ($F = 8.090$, $p < .05$) on the CT pre-test in the plugged-in-first group. Therefore, we analyzed simple effects on gender, programming experience, and programming confidence in the plugged-in-first group. Therefore, the results of the simple effect analyses for gender, programming experience, and programming confidence in the plugged-in-first group are recorded in Table 5. Subsequently, we analyzed the impact of these factors in different programming approaches.

Gender

There was no significant difference for the gender factor in the CT pre-test in the control ($F = .041$, $p > .05$), plugged-in ($F = .996$, $p > .05$), unplugged ($F = 1.078$, $p > .05$), plugged-in-first ($F = 3.141$, $p > .05$), and unplugged-first ($F = 0.160$, $p > .05$) groups. Likewise, there was no significant difference in the gender factor on the post-pre-test of CT between the

Table 5 Results of MANOVA

Factors	Plugged-in		Unplugged		Plugged-in-first		Unplugged-first		Control	
	Pre-test F	Post-pre-test F	Pre-test F	Post-pre-test F	Pre-test F	Post-pre-test F	Pre-test F	Post-pre-test F	Pre-test F	Post-pre-test F
(A) Gender	.996	.024	1.078	0.030	3.141	1.506	0.160	3.363	0.041	0.128
(B) Programming Experience	.507	6.414*	4.274*	1.276	0.044	6.872*	5.253*	1.654	2.269	1.409
(C) Programming Interest	.523	1.513	0.240	10.027**	0.001	0.094	2.415	8.917*	0.916	0.024
(D) Programming Confidence	.470	.111	3.499	16.463**	1.097	7.173*	.105	6.059*	.796	.724
A × B	.914	.707	.726	.448	7.471*	1.506	1.492	.868	.168	.011
A × C	.672	1.409	.081	.773	.656	.069	1.895	2.637	.084	.280
A × D	.438	1.411	.546	1.908	8.090*	.309	1.324	.796	.009	.562
B × C	.039	.480	.018	.128	3.428	.621	.407	1.769	.046	.003
B × D	.102	.807	.062	1.209	3.712	.461	.010	.024	2.248	1.377
C × D	.327	.763	.546	1.095	.016	.405	.241	.000	.431	.005

* = p < .05, ** = p < .01. A represents the gender factor, B represents the programming experience factor, C represents the programming interest factor, and D represents the programming confidence factor

control group and the four experimental groups (p > .05). This suggested that the effectiveness of different programming education approaches is not affected by gender.

Programming experience

In both the CT pre-test and CT post-pre-test in the control group, no significant difference in programming experience were observed. In the CT pre-test, we found significant differences in programming experience in the unplugged (F = 4.274, p < .05) and unplugged-first groups (F = 5.253, p < .05), and no significant differences in the plugged-in group (F = .507, p > .05) and plugged-in-first group (F = .044, p > .05). However, in the CT post-pre-test, we found that programming experience showed significant differences in the plugged-in group (F = 6.414, p < .05) and the plugged-in-first group (F = 6.872, p < .05), and no significant difference in the unplugged group (F = 1.276, p > .05) and the unplugged-first group (F = 1.654, p > .05). In conclusion, programming experience had different effects during the intervention of different programming approaches.

Programming interest

In the CT pre-test, there was no significant difference in programming interest in any of the five groups (p > .05). In the CT post-pre-test, we only observed significant differences in the unplugged group (F = 10.027, p < .01) and unplugged-first group (F = 8.917, p < .05), which were not found in the control group (F = .024, p > .05), plugged-in group (F = 1.153, p > .05), and plugged-in-first group (F = .04, p > .05). This showed that programming interest can significantly affect students in the unplugged group and the unplugged-first group, and we found that students who were interested improved their CT skills more.

Programming confidence

In the CT pre-test, programming confidence showed no significant difference in any of the 5 groups (p > .05). In the CT post-pre-test, we observed significant differences in programming confidence in the unplugged group (F = 16.463, p < .01), plugged-in-first group (F = 7.173, p < .05), and unplugged-first group (F = 6.059, p < .05). However, there was also no significant difference in the control group (F = .724, p > .05) and the plugged-in group (F = 0.11, p > .05). Furthermore, students with programming confidence showed more improvement in CT skills.

Chain mediation effect model of programming experience, programming interest, programming confidence, and computational thinking

First, the correlations of programming experience, programming interest, programming confidence, and post-pre-test of CT were analyzed for the four experimental groups of students. We used the point biserial correlation coefficient to calculate correlations between the continuous variable CT post-pre-test scores and the categorical variables programming experience, programming interest, and programming confidence. In addition, we used the coefficient of contingency to calculate the correlation between categorical variables. The results are shown in Table 6. It was found that there were significant correlations (p < .05) among all of these factors, which provided the basis for later structural equations to explore the relationships among the influencing factors.

Table 6 Correlations Analysis of Programming Experience, Programming Interest, Programming Confidence and Computational Thinking

Variables	Computational thinking	Programming experience	Programming interest	Programming confidence
Computational Thinking	–			
Programming Experience	.363**	–		
Programming Interest	.377***	.303**	–	
Programming Confidence	.578***	.235*	.292**	–

* = p < .05, ** = p < .01, *** = p < .001

Then, we developed a multiple chain mediation effect model. This included the direct effect paths between programming confidence, programming interest, programming experience, and performance on the post-pre-test of CT. The direct path between programming experience and programming interest. As well as, the indirect path of influence between programming experience and programming interest. In this process, we excluded students' gender from the model because no significant difference due to gender was found in the one-way ANOVA in Table 5. The results of the chain mediation effect are shown in Table 7; Fig. 6. Since there was significant direct effect between programming confidence and the post-pre-test of CT (p < .01), and programming interest was able to play significant mediation effect (p < .05). Meanwhile, programming experience and programming interest also had a chain mediation effect (p < .05) so that programming experience can

Table 7 Results of Multiple Chain Mediation Effect (N = 93)

Effect paths	Direct effect	Indirect effect	P	Lower bounds	Upper bounds	Mediation effect
P1: PC-PE-CT		– .260	.068	– .909	.025	Partial mediation
P2: PC-PI-CT		– .408*	.023	– .039	– 1.203	
P3: PC-PE-PI-CT		– .105*	.019	– .099	– .455	
P4: PC-CT	– 4.250**		.001	– 5.728	– 2.687	

* = p < .05, ** = p < .01, *** = p < .001, *PC* Programming Confidence, *PE* Programming Experience, *PI* Programming Interest, *CT* Computational Thinking of Post-pre-test. The lower and upper limits of the 95% confidence interval (bootstrap = 5000). If the 95% confidence interval does not contain 0, the path is established

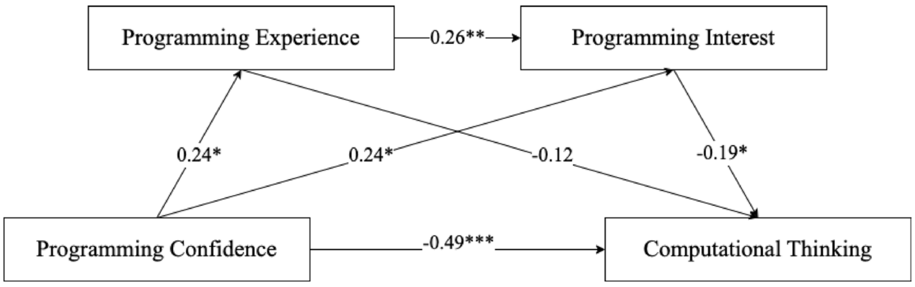


Fig. 6 Multiple chain mediation effect model (N = 93)

play significant mediation effects as well. Therefore, programming confidence was able to influence CT through programming experience and programming interest, in addition to directly influencing CT. Thus, there is a partial mediation effect between students' programming experience, programming interest, programming confidence, and CT. In other words, programming experience and programming interest were able to moderate the relationship between programming confidence and CT to a certain extent.

Discussion

The purpose of this study was to determine the effectiveness of different programming approaches on students' CT skill development. To investigate whether the effects of different programming approaches on students' CT are related to gender, programming experience, programming interest, and programming confidence factors, in addition to clarifying what the mechanisms of effect between those factors are. Therefore, we conducted a 10-week quasi-experimental study among 121 first-year students. We explored the effects of plugged-in, unplugged, plugged-in-first, unplugged-first, and regular IT courses on students' CT skills. We also collected demographic information about the students and their CT scores before and after the intervention.

In RQ1, quantitative results showed that CT skills improved significantly in all four experimental groups after the intervention. This proved the effectiveness of plugged-in, unplugged, plugged-in-first, and unplugged-first programming in improving the CT skills of first-grade students. This is consistent with the findings of several current studies, where many researchers have similarly found that plugged-in and unplugged programming are well suited to developing CT skills in students at the primary education level, and can even enhance students' ability to solve programming problems (Kale & Yuan, 2020; Kalelioğlu, 2015). We have also demonstrated the effectiveness of Code.org's instructional materials in enhancing students' CT skills. However, no significant difference in CT skills was shown in the control group which surprised us. In fact, the Ministry of Education has already put in place policies to develop students' CT skills (Ministry of Education, 2018). This situation may be attributed to the lack of systematic programming curriculum design, which makes it difficult to achieve the objective of developing students' CT skills. Sun et al. (2022a) also proposed that policy level and programming curriculum implementation level can only be improved together to provide opportunities for students to fully develop their CT skills. More interestingly, we found that the mixed plugged-in and unplugged programming methods work equally well, but better than the always plugged-in or unplugged programming methods. This is consistent with the findings of del Olmo-Muñoz et al. (2020) who also found that the mixed programming approach gave better results than the always plugged-in programming approach. This may be related to the style of programming activities, where repeating the same style of plugged-in or unplugged programming activities may lead to a loss of interest among students (Jiang & Wong, 2021). The mixed programming approach consistently engages students' attention and promotes their active engagement in programming activities. In summary, the mixed programming approach of plugged-in and unplugged is also an effective way to improve CT skills for primary students.

We also found that the gender of students does not affect their CT skills. In other words, there was no significant difference between boys' and girls' CT skills after the experimental group's activity intervention. However, some researchers have found that boys' CT skills are significantly higher than girls' after the same programming intervention (Gao et al., 2022), and that it even takes girls more time to achieve the same level of CT as boys (Atmatzidou & Demetriadis, 2016). Similarly, according to Gur et al. (2012), boys and girls have different rates of neurological brain function development, which can lead to differences in how boys and girls deal with reasoning problems and spatial problems. This gender difference usually becomes apparent with age. However, the younger age of the first graders in this study and the fact that their cognitive skills were not yet fully developed may be the reason why significant differences were not observed. As a matter of fact, breaking gender stereotypes in programming education is what we have been waiting for. In summary, these programming approaches can be considered as one of the ways to achieve equitable CT instruction.

In addition, it was found that the plugged-in and plugged-in-first approaches amplified the impact of students' programming experience on CT skills, where students with programming experience improved their CT skills more. Indeed, as Sun et al. (2021a) argued, there is a positive relationship between CT performance and programming experience, where students with programming experience are more likely to understand the abstract concepts involved in programming activities. Also, students with programming experience perform better on various cognitive tests (Liao & Bright, 1991). In contrast, the unplugged and unplugged-first approaches drove students with different programming experience to achieve the same CT improvement, weakening the effect of programming experience on CT skills. This could be caused by the implementation of different types of programming activities in the first stage of instruction. The implementation of unplugged activities first can provide the programming conceptual foundation required for the development of subsequent plugged activities, which facilitates the mastery of complex CT concepts (Saxena et al., 2019). Meanwhile, unplugged programming activities can reduce the cognitive load of students using digital products and help to smooth the transition of children to complex digital programming environments (Sigayret et al., 2022; Tsarava et al., 2017). In other words, implementing unplugged programming activities first allows students to adapt to programming most familiarly, and can diminish the impact of prior experience as well as the psychological aspects.

Moreover, we found that programming interest and programming confidence significantly affect CT skills. Specifically, students who were interested and confident in programming had higher CT skills. This confirms the statement shown by Sun et al. (2022b) that besides cognitive factors such as programming experience, psychological dispositional factors such as interest and confidence also significantly affect students' CT skills. This was probably attributed to the fact that, in the face of negative attitudes or negativity carried by programming learning difficulties, students with interest and confidence in programming would actively adjust and work towards achieving the set learning goals (Gunbatar & Karalar, 2018). Kong (2016) even designed an interest-driven programming curriculum framework to better develop students' CT skills. This showed that interest and confidence are important factors that should be considered in CT education and programming education. However, this phenomenon was not observed in the plugged-in group. Plugged-in programming has a certain threshold of computer operation and requires sitting in front of the computer for periods, which is not conducive to a child's instincts for a first-grade student (Bray, 2018). This may diminish the positive impact of student programming interest and programming confidence on CT.

In addition, we found a chain mediation effect of programming experience and programming interest between programming confidence and CT skills. Indeed, programming confidence is generally recognized as being closely related to academic achievement or specific skills (Tsai et al., 2018). In this study, programming confidence demonstrated a strong relationship with CT skills, which can serve as an important factor in influencing CT directly, or indirectly through programming experience and programming interest. Piaget and Cook (1952) also argued that children's exploration of things or mastery of learning outcomes requires the stimulation of confidence-generating internal motivation. This is also consistent with some of the current results that programming confidence acts as a powerful internal driver of students' programming interest (Kong et al., 2018). In addition, when students have programming-related experience it increases their willingness and interest in learning programming, which is supported by the study of Usher and Pajares (2008). Mason and Rich's (2020) Attitude Scale for Elementary School Students, developed based on Expectancy Theory, similarly showed that programming experience affects programming interest. This showed that increased attention should be paid to the psychological aspects of students' confidence and interest in the programming learning process.

Implications for policy and practice

Generally, CT skill development relies on single plugged-in or unplugged programming. This study demonstrated the effectiveness of the mixed approach of plugged-in and unplugged programming to help better develop CT skills in primary students. Meanwhile, we clarified that programming experience, programming interest, and programming confidence are all important factors affecting students' CT skills. We explored the influence of these factors on CT and the chain mediation mechanism. These findings can provide references for programming education researchers and front-line IT teachers in primary schools. Firstly, we found that the mixed programming approaches (plugged-in-first, unplugged-first) are more effective than the single programming approaches (plugged-in, unplugged) in developing CT skills in primary school students, enriching the research findings in the fields of programming education and CT education. Secondly, it is more effective to use the mixed programming approach to develop students' CT skills in the primary school IT curriculum than in the regular IT curriculum. Therefore, education authorities can appropriately integrate plugged-in and unplugged activities into primary school IT curricula. Thirdly, given that most primary school students have no programming experience and are unfamiliar with computer operations, teachers need to pay attention to the sequence in which programming activities are implemented. We advocate implementing unplugged activities first to help students understand programming concepts more intimately at the beginning. This reduces the cognitive load and unfamiliarity with the complex digital programming environment when students are subsequently exposed to plugged-in programming activities. Finally, the important role of students' programming interest and programming confidence was clarified when it comes to CT skill development. Therefore, teachers should use sensible ways to maintain students' interest and confidence in programming and promote the strong development of CT skills.

Limitations and future research

This study also has some limitations. First, this study was conducted in the Chinese educational context with first-grade primary school students. Due to the different educational environments in different countries and the variation among individual students, careful thought is needed if the study findings are generalized. Second, this study only collected data on students' CT skills before and after the experimental intervention, and only studied the short-term effects of the programming activities. In the next step of the study, it is appropriate to consider whether there is a sustained impact and to track the changes in students' CT skills sometime after the programming activity. Further, only binary variable type data were collected when exploring the effects of students' interest and confidence on CT skills. Future research should use more accurate assessment scales to measure these characteristics of students while considering the possible effects of programming learning activities on these factors. Finally, this study focused only on students' development of CT skills during programming activities, but not on specific programming concepts. Future research could use instruments to measure students' mastery of specific programming concepts and CT subskills.

Conclusion

Considering that CT skills have become an essential core skill for every student in K-12 education, teachers and educators are constantly exploring ways to introduce CT into classroom education. Therefore, this study conducted a quasi-experimental study to investigate the effects of the single programming approach (plugged-in and unplugged) and the mixed programming approach (plugged-in-first and unplugged-first) on the CT skills of first-grade students. Also, students' factors (including gender, programming experience, programming interest, and programming confidence) were considered in addition to the effects of the programming learning approach itself. Our study revealed the effectiveness of mixed plugged-in and unplugged programming approaches on CT skill enhancement for first-year students. We also found that implementing unplugged programming in the first phase would provide students with the same opportunities for CT development, attenuating the impact of the programming experience. Moreover, programming interest and programming confidence will play an important role in the CT learning process. In addition, when exploring the relationship and mediation mechanisms between CT and the main factors influencing CT, it was found that programming experience and programming interest had a significant chain mediation between programming confidence and CT. This study enriches the research findings in the field of programming education and CT in primary schools. It also provides evidence for teachers and educators to teach programming.

Declarations

Conflict of interest The authors declare no conflicts of interest.

References

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, *75*, 661–670. https://doi.org/10.1016/j.robot.2015.10.008.

Bandura, A., & Wessels, S. (1994). *Self-efficacy* (Vol. 4). na.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *Acm Inroads*, *2*(1), 48–54.

Bell, T., & Vahrenhold, J. (2018). CS Unplugged—How Is It Used, and Does It Work? In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications* (pp. 497–521). https://doi.org/10.1007/978-3-319-98355-4_29.

Beyer, S. (2014). Why are women underrepresented in Computer Science? Gender differences in stereotypes, self-efficacy, values, and interests and predictors of future CS course-taking and grades. *Computer Science Education*, *24*(2–3), 153–192. https://doi.org/10.1080/08993408.2014.963363.

Bocconi, S., Chioccariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, K., Earp, J., Horvath, M. A., Jasutė, E., & Malagoli, C. (2022). *Reviewing computational thinking in Compulsory Education*. Joint Research Centre. https://digital-skills-jobs.europa.eu/en/inspiration/research/reviewing-computational-thinking-compulsory-education-jrc-2022-1 (Seville site).

Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017). Development of computational thinking skills through unplugged activities in primary school. Proceedings of the 12th workshop on primary and secondary computing education (pp.65–72).

Bray, M. (2018). *Plugged in: The dangers of modern technology*. https://go.gale.com/.

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

Bybee, R. W., Taylor, J. A., Gardner, A., Van Scotter, P., Powell, J. C., Westbrook, A., & Landes, N. (2006). The BSCS 5E instructional model: Origins and effectiveness. *Colorado Springs Co: BSCS*, *5*, 88–98.

Chiu, M. M., & Klassen, R. M. (2010). Relations of mathematics self-concept and its calibration with mathematics achievement: Cultural differences among fifteen-year-olds in 34 countries. *Learning and Instruction*, *20*(1), 2–17. https://doi.org/10.1016/j.learninstruc.2008.11.002.

Clements, D. H., Sarama, J., Wolfe, C. B., & Spitler, M. E. (2015). Sustainability of a Scale-Up intervention in early mathematics: A longitudinal evaluation of implementation fidelity [Article]. *Early Education and Development*, *26*(3), 427–449. https://doi.org/10.1080/10409289.2015.968242.

Code.org (2013). *Anybody can learn*. https://hourofcode.com/us/zh.

Computer Science Teachers Association (CSTA), & International Society for Technology in Education (ISTE) (2011). *Operational Defnition of Computational Thinking for K-12 Education*. http://www.iste.org/docs/pdfs/Operational-Defnition-of-Computational-Thinking.pdf.

Cronbach, L. J., & Meehl, P. E. (1955). Construct validity in psychological tests. *Psychological Bulletin*, *52*(4), 281.

Deci, E. L., Olafsen, A. H., & Ryan, R. M. (2017). Self-determination theory in work organizations: The state of a science. *Annual Review of Organizational Psychology and Organizational Behavior*, *4*, 19–43. https://doi.org/10.1146/annurev-orgpsych-032516-113108.

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*. https://doi.org/10.1016/j.compedu.2020.103832

Dewey, J. (1913). *Interest and effort in education*. London: Forgotten Books.

European Commission (2020). *Digital Education Action Plan 2021–2027*. https://education.ec.europa.eu/focus-topics/digital-education/about-digital-education.

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

Gao, X., & Hew, K. F. (2021). Toward a 5E-Based flipped Classroom Model for Teaching Computational thinking in Elementary School: Effects on Student Computational thinking and problem-solving performance. *Journal of Educational Computing Research*, *60*(2), 512–543. https://doi.org/10.1177/07356331211037757.

Gao, H., Hasenbein, L., Bozkir, E., Göllner, R., & Kasneci, E. (2022). Exploring gender differences in computational thinking learning in a VR Classroom: Developing machine learning models using Eye-Tracking Data and explaining the models. *International Journal of Artificial Intelligence in Education*. https://doi.org/10.1007/s40593-022-00316-z.

Gunbatar, M. S., & Karalar, H. (2018). Gender differences in middle school students' attitudes and self-efficacy perceptions towards mBlock programming. *European Journal of Educational Research*, *7*(4), 925–933. https://doi.org/10.12973/EU-JER.7.4.925.

Gur, R. C., Richard, J., Calkins, M. E., Chiavacci, R., Hansen, J. A., Bilker, W. B., Loughead, J., Connolly, J. J., Qiu, H., Mentch, F. D., Abou-Sleiman, P. M., Hakonarson, H., & Gur, R. E. (2012). Age group and sex differences in performance on a computerized neurocognitive Battery in children age 8–21. *Neuropsychology*, *26*(2), 251–265. https://doi.org/10.1037/a0026712.

Helmlinger, B., Sommer, M., Feldhammer-Kahr, M., Wood, G., Arendasy, M. E., & Kober, S. E. (2020). Programming experience associated with neural efficiency during figural reasoning. *Scientific Reports*, *10*(1), 13351. https://doi.org/10.1038/s41598-020-70360-z.

Hermans, F., & Aivaloglou, E. (2017). To Scratch or not to Scratch? Proceedings of the 12th Workshop on Primary and Secondary Computing Education (pp. 49–56).

Hidi, S. (2006). Interest: A unique motivational variable. *Educational Research Review*, *1*(2), 69–82.

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, *126*, 296–310. https://doi.org/10.1016/j.compedu.2018.07.004.

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education*, *82*, 263–279. https://doi.org/10.1016/j.compedu.2014.11.022.

Izu, C., Mirolo, C., Settle, A., Mannila, L., & Stupuriene, G. (2017). Exploring Bebras tasks Content and performance: A multinational study. *Informatics in Education*, *16*(1), 39–59. https://doi.org/10.15388/infedu.2017.03.

Jiang, S., & Wong, G. K. W. (2021). Exploring age and gender differences of computational thinkers in primary school: A developmental perspective. *Journal of Computer Assisted Learning*, *38*(1), 60–75. https://doi.org/10.1111/jcal.12591.

Jong, M. S. Y., Geng, J., Chai, C. S., & Lin, P. Y. (2020). Development and predictive validity of the computational thinking disposition questionnaire. *Sustainability*. https://doi.org/10.3390/su12114459

Kale, U., & Yuan, J. (2020). Still a new kid on the Block? Computational thinking as Problem solving in Code.org. *Journal of Educational Computing Research*, *59*(4), 620–644. https://doi.org/10.1177/0735633120972050.

Kalelioglu, F., Gulbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, *52*, 200–210. https://doi.org/10.1016/j.chb.2015.05.047.

Kim, B., Kim, T., & Kim, J. (2014). Paper-and-Pencil Programming Strategy toward Computational thinking for non-majors: Design your solution. *Journal of Educational Computing Research*, *49*(4), 437–459. https://doi.org/10.2190/EC.49.4.b.

Kolb, D. A., Boyatzis, R. E., & Mainemelis, C. (2014). Experiential learning theory: Previous research and new directions. *Perspectives on thinking, learning, and cognitive styles* (pp. 227–248). England: Routledge.

Kong, S. C. (2016). A framework of curriculum design for computational thinking development in K-12 education. *Journal of Computers in Education*, *3*(4), 377–394. https://doi.org/10.1007/s40692-016-0076-z.

Kong, S. C., & Wang, Y. Q. (2020). Formation of computational identity through computational thinking perspectives development in programming learning: A mediation analysis among primary school students. *Computers in Human Behavior*. https://doi.org/10.1016/j.chb.2019.106230

Kong, S. C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers & Education*, *127*, 178–189. https://doi.org/10.1016/j.compedu.2018.08.026.

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior*, *72*, 558–569. https://doi.org/10.1016/j.chb.2017.01.005.

Li, F., Wang, X., He, X., Cheng, L., & Wang, Y. (2022). The effectiveness of unplugged activities and programming exercises in computational thinking education: A Meta-analysis. *Education and Information Technologies*, *27*(6), 7993–8013. https://doi.org/10.1007/s10639-022-10915-x.

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

Liu, Y. C., Huang, T. H., & Sung, C. L. (2021). The determinants of impact of personal traits on computational thinking with programming instruction. *Interactive Learning Environments*. https://doi.org/10.1080/10494820.2021.1983610

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

Mason, S. L., & Rich, P. J. (2020). Development and analysis of the elementary student coding attitudes survey. *Computers & Education*. https://doi.org/10.1016/j.compedu.2020.103898

Ministry of Education (2022). Compulsory Information Technology Curriculum Standards http://www.gov.cn/zhengce/zhengceku/2022-04/21/content_5686535.htm.

Ministry of Education (2018). *Education Informatization 2.0 Action Plan*. http://www.moe.gov.cn/srcsite/A16/s3342/201804/t20180425_334188.html.

Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED Revista de Educación a Distancia, 46*, 1–23.

Mouza, C., Pan, Y. C., Yang, H., & Pollock, L. (2020). A multiyear investigation of Student Computational thinking concepts, practices, and perspectives in an after-School Computing Program. *Journal of Educational Computing Research*, *58*(5), 1029–1056. https://doi.org/10.1177/0735633120905605.

Ouahbi, I., Kaddari, F., Darhmaoui, H., Elachqar, A., & Lahmine, S. (2015). Learning basic programming concepts by creating games with scratch programming environment. *Procedia-Social and Behavioral Sciences*, *191*, 1479–1482. https://doi.org/10.1016/j.sbspro.2015.04.224.

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

Passey, D. (2017). Computer science (CS) in the compulsory education curriculum: Implications for future research. *Education and Information Technologies*, *22*, 421–443. https://doi.org/10.1007/s10639-016-9475-z.

Piaget, J., & Cook, M. (1952). *The origins of intelligence in children* (Vol. 8). United States: International Universities Press.

Polat, E., & Yilmaz, R. M. (2022). Unplugged versus plugged-in: Examining basic programming achievement and computational thinking of 6th-grade students. *Education and Information Technologies*, *27*(7), 9145–9179. https://doi.org/10.1007/s10639-022-10992-y.

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using scratch in five schools. *Computers & Education*, *97*, 129–141. https://doi.org/10.1016/j.compedu.2016.03.003.

Saxena, A., Lo, C. K., Hew, K. F., & Wong, G. K. W. (2019). Designing Unplugged and plugged activities to cultivate computational thinking: An exploratory study in early Childhood Education. *The Asia-Pacific Education Researcher*, *29*(1), 55–66. https://doi.org/10.1007/s40299-019-00478-w.

Schiefele, U. (2008). Lernmotivation und Interesse. *Handbuch Der pädagogischen Psychologie*, *10*, 38–49.

Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition.

Shang, X., Jiang, Z., Chiang, F. K., Zhang, Y., & Zhu, D. (2023). Effects of robotics STEM camps on rural elementary students' self-efficacy and computational thinking. *Educational Technology Research and Development*. https://doi.org/10.1007/s11423-023-10191-7.

Sigayret, K., Tricot, A., & Blanc, N. (2022). Unplugged or plugged-in programming learning: A comparative experimental study. *Computers & Education, 184*, 104505.

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

Sun, L., Hu, L., & Zhou, D. (2021b). Single or combined? A study on programming to promote Junior High School Students' computational thinking skills. *Journal of Educational Computing Research*, *60*(2), 283–321. https://doi.org/10.1177/07356331211035182.

Sun, L., Hu, L., & Zhou, D. (2022aa). The bidirectional predictions between primary school students' STEM and language academic achievements and computational thinking: The moderating role of gender. *Thinking Skills and Creativity*. https://doi.org/10.1016/j.tsc.2022.101043

Sun, L., Hu, L., & Zhou, D. (2022b). Programming attitudes predict computational thinking: Analysis of differences in gender and programming experience. *Computers & Education*. https://doi.org/10.1016/j.compedu.2022.104457

Tsai, C. Y. (2019). Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy [Article]. *Computers in Human Behavior*, *95*, 224–232. https://doi.org/10.1016/j.chb.2018.11.038.

Tsai, M. J., Wang, C. Y., & Hsu, P. F. (2018). Developing the Computer Programming Self-Efficacy Scale for Computer Literacy Education. *Journal of Educational Computing Research*, *56*(8), 1345–1360. https://doi.org/10.1177/0735633117746747.

Tsai, M. J., Liang, J. C., & Hsu, C. Y. (2020). The computational thinking scale for Computer Literacy Education. *Journal of Educational Computing Research*, *59*(4), 579–602. https://doi.org/10.1177/0735633120972356.

Tsarava, K., Moeller, K., Pinkwart, N., Butz, M., Trautwein, U., & Ninaus, M. (2017). Training computational thinking: Game-based unplugged and plugged-in activities in primary school. European conference on games based learning (pp. 687–695).

Usher, E. L., & Pajares, F. (2008). Sources of self-efficacy in school: Critical review of the literature and future directions. *Review of Educational Research*, *78*(4), 751–796. https://doi.org/10.3102/0034654308321456.

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

Weber, K., Martin, M. M., & Cayanus, J. L. (2005). Student interest: A two-study re-examination of the concept. *Communication Quarterly*, *53*(1), 71–86. https://doi.org/10.1080/01463370500055996.

Wing, J. M. (2006). Computational thinking. *Communications of the Acm*, *49*(3), 33–35.
Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical Physical and Engineering Sciences*, *366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118.
Wong, G. K. W., & Cheung, H. Y. (2020). Exploring children's perceptions of developing twenty-first century skills through computational thinking and programming. *Interactive Learning Environments*, *28*(4), 438–450. https://doi.org/10.1080/10494820.2018.1534245.
Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education (TOCE)*, *14*(1), 1–16. https://doi.org/10.1145/2576872.

Lihui Sun is an associate professor in the Department of Educational Technology, School of Education, Minzu University of China. His research interests include computational thinking and programming education for children.

Junjie Liu is a master degree student in the Key Laboratory of Child Development and Learning Science, Ministry of Education, Southeast University.

Authors and Affiliations

Lihui Sun[1] · **Junjie Liu**[2]

Lihui Sun
slhphd@yeah.net

Junjie Liu
liujunjieny@foxmail.com

[1] School of Education, Minzu University of China, No. 27 Zhongguancun South Avenue, Beijing 100081, China

[2] Key Laboratory of Child Development and Learning Sciences, Ministry of Education, Southeast University, Nanjing 210096, China