International Journal of STEM Education

RESEARCH Open Access

The effect of embedded structures on cognitive load for novice learners during block-based code comprehension

Xiaoxin Hao[1], Zhiyi Xu[1], Mingyue Guo[1], Yuzheng Hu[3] and Fengji Geng[1,2*]

Abstract

Background Coding has become an integral part of STEM education. However, novice learners face difficulties in processing codes within embedded structures (also termed nested structures). This study aimed to investigate the cognitive mechanism underlying the processing of embedded coding structures based on hierarchical complexity theory, which suggests that more complex hierarchies are involved in embedded versus sequential coding structures. Hierarchical processing is expected to place a great load on the working memory system to maintain, update, and manipulate information. We therefore examined the difference in cognitive load induced by embedded versus sequential structures, and the relations between the difference in cognitive load and working memory capacity.

Results The results of Experiment 1 did not fully support our hypotheses, possibly due to the unexpected use of cognitive strategies and the way stimuli were presented. With these factors well controlled, a new paradigm was designed in Experiment 2. Results indicate that the cognitive load, as measured by the accuracy and response times of a code comprehension task, was greater in embedded versus sequential conditions. Additionally, the extra cognitive load induced by embedded coding structures was significantly related to working memory capacity.

Conclusions The findings of these analyses suggest that processing embedded coding structures exerts great demands on the working memory system to maintain and manipulate hierarchical information. It is therefore important to provide scaffolding strategies to help novice learners process codes across different hierarchical levels within embedded coding structures.

Keywords Embedded structures, Hierarchical complexity, Cognitive load, Working memory, Novice learners

Introduction

Coding has become an integral part of STEM education as it not only supports the development of technical proficiency (the "T" component), but also enables the interdisciplinary connections with science, engineering, and mathematics (Liu & Schunn, 2020; Tucker-Raymond et al., 2019; Ye et al., 2023). Computational thinking (CT), primarily facilitated through coding education, is recognized as a trans-disciplinary competency that empowers individuals to address real-life problems and confront challenges within the STEM domains (Li et al., 2020a, 2020b; Ntemngwa & Oliver, 2018; So, 2023). Recently, block-based coding languages (e.g., Scratch) have become increasingly popular because they allow students to drag and drop code commands, reducing the burden of dealing with complex syntax involved in text-based programming (Hu et al., 2021; Weintrop, 2019; Xu et al., 2019). However,

*Correspondence:
Fengji Geng
gengf@zju.edu.cn
[1] Department of Curriculum and Learning Sciences, Zhejiang University, 866 Yuhangtang Road, Zijingang Campus, Hangzhou 310058, China
[2] Children's Hospital, Zhejiang University School of Medicine, National Clinical Research Center for Child Health, Hangzhou 310052, China
[3] Department of Psychology and Behavioral Sciences, Zhejiang University, 866 Yuhangtang Road, Zijingang Campus, 310058 Hangzhou, China

novice learners still face many challenges when learning block-based coding (Qian et al., 2020; Wang et al., 2021; Wiggins et al., 2021). One such challenge regards processing nested structures, where one control structure (e.g., repeat) is placed inside another (Fig. 1). Such structures are also referred to as embedded structures. Novice learners generally have greater difficulty in using nested structures compared to sequential structures that organize two or more control structures in a flat form (Fig. 1) (Bers et al., 2019; Kelleher & Hnin, 2019; Mladenović et al., 2018; Yamashita et al., 2016). To better understand and address this challenge, this study amis to explore the cognitive mechanism underlying the processing of nested coding structures.

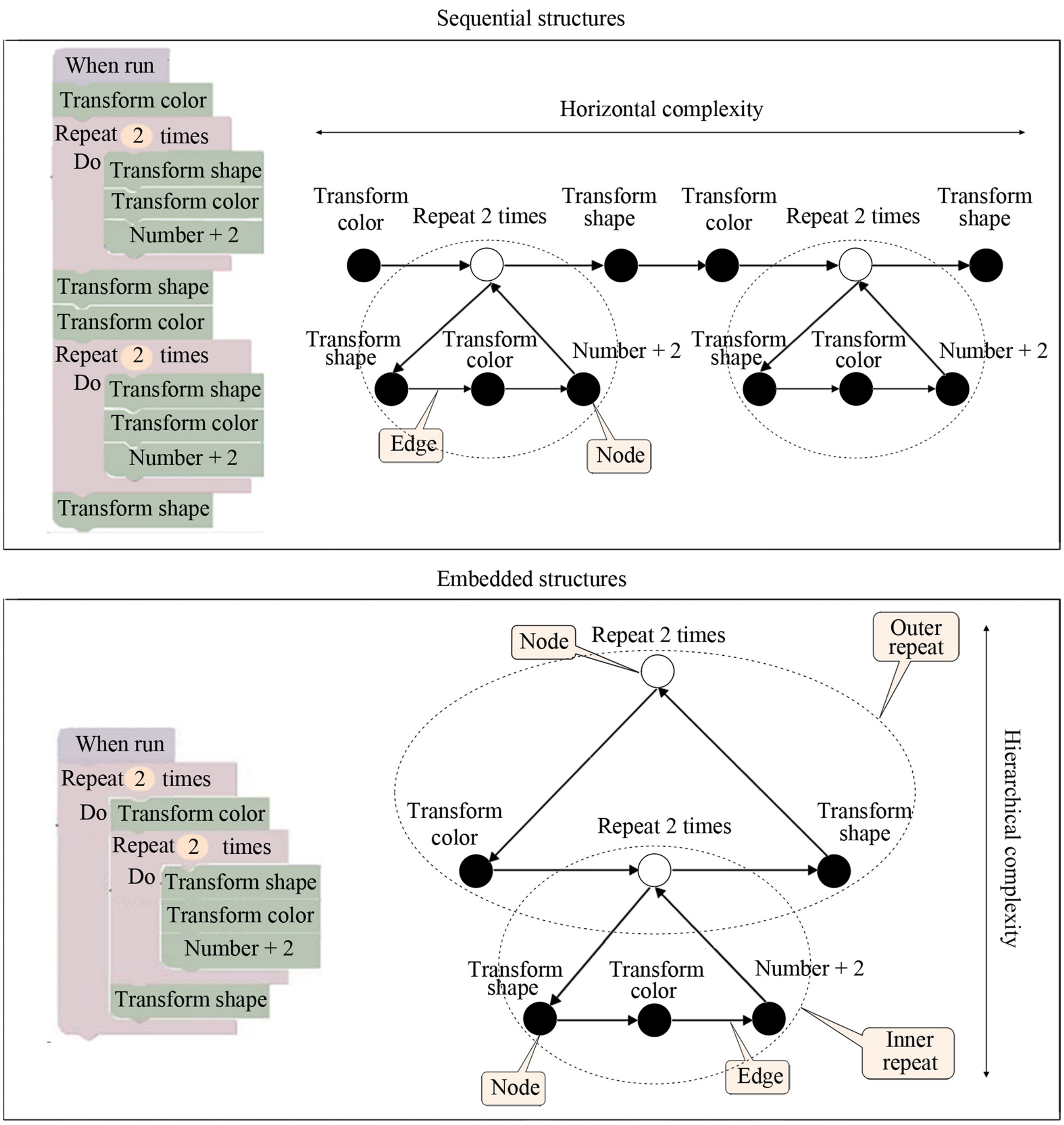


Fig. 1 Topological maps of embedded and sequential coding structures

Theoretical framework

According to hierarchical complexity theory (Commons, 2007; Commons et al., 1998), nested and sequential control structures differ in their hierarchical complexity and horizontal complexity, which can be clearly depicted using nodes and edges rooted in graph theory (Uddén et al., 2020; West, 2001). As shown in Fig. 1, there are two types of nodes. The first type refers to commands (e.g., "Transform color") that explicitly specify operations (represented by ●), while the second type represents commands (e.g., "Repeat 2 times") that indicate the number of times to execute each individual command within a loop body (represented by ○). Hierarchy is formed when the latter type of commands are placed at the higher level to coordinate the former type of commands at the lower level.

Processing coding structures with multiple hierarchical levels is expected to generate substantial cognitive load, which refers to the cognitive demands imposed on learners' working memory system (Anmarkrud et al., 2019; Paas & Van Merriënboer, 1994). Based on previous studies (Badre & Nee, 2018; O'Reilly & Frank, 2006), there are three critical cognitive demands associated with processing hierarchical structures. The first demand involves rapid encoding of information at the lower levels of a hierarchy (e.g., the commands represented by • in Fig. 1). The second demand involves robust maintenance of information at the higher levels of a hierarchy (e.g., commands as represented by ○ in Fig. 1). The third demand involves selective updating of specific information in working memory while simultaneously maintaining others across different hierarchical levels (Murty et al., 2011). For example, when executing codes within nested structures, participants need to selectively update the number of repetition times for the inner loop while simultaneously maintaining the repetitions of the outer loop.

Nested structures possess greater hierarchical complexity than sequential ones when other factors are well-matched because they organize commands across more hierarchical levels (Fig. 1). Thus, executing commands in nested structures necessitates more iterative switches between different hierarchical levels compared to sequential structures. The execution of these iterative switches engages the working memory system to maintain and selectively update hierarchical information (e.g., the repetition times for the inner and outer loops). In contrast, sequential structures present greater horizontal complexity than nested structures (Fig. 1). While processing horizontal complexity also consumes working memory resources, such demand primarily arises from the rapid encoding of stimuli rather than the maintenance and updating required for processing hierarchical complexity. Therefore, we aimed to test whether processing nested coding structures compared to sequential structures would result in greater cognitive load due to the different demands on the working memory system (Prediction 1). Meanwhile, as there are great individual variations in working memory capacity (Baddeley, 1992; Barrett et al., 2004), we also predicted that the amount of extra cognitive load generated by processing nested versus sequential coding structures was significantly related to individual working memory capacity (Prediction 2).

In addition, we further examined the extra cognitive load induced by greater horizontal complexity in both nested and sequential conditions. A different number of code commands were inserted in these two conditions to manipulate horizontal complexity. Unlike hierarchical complexity, processing horizontal complexity mainly poses cognitive demands on rapid encoding. Therefore, we predicted that increasing the number of inserted commands would lead to extra cognitive load (Prediction 3). However, such cognitive load would not be associated with working memory capacity, as measured by tasks emphasizing the maintenance and selective updating of stimuli information (Prediction 4). Moreover, the rapid encoding of more inserted commands may interfere with information maintenance and updating, especially for nested conditions. Accordingly, we predicted that the difference in cognitive load between nested and sequential conditions would be modulated by the number of inserted commands (Prediction 5).

In summary, testing the aforementioned predictions would enhance our understanding of how individuals process nested structures and provide valuable insights to guide the teaching and learning of coding. Novice learners often face challenges in learning nested coding structures due to the complex hierarchical information involved. Processing such hierarchical complexity is thought to impose significant demands on the working memory system for information maintenance and manipulation, thus leading to increased cognitive load. Hence, effective teaching or learning strategies should be incorporated into educational practices to support novice learners in processing the hierarchical information involved in nested coding structures.

Literature review

Nested structures exist not only in coding, but also in many other domains, including natural language, artificial grammar, and music (Hochmann et al., 2008; Koelsch et al., 2013; Lakretz et al., 2020). Studies in these domains have widely investigated the cognitive and neural mechanisms involved in the processing of nested structures. The current study mainly drew on the findings in these fields to investigate the cognitive mechanism underlying novices' comprehension of block-based codes within nested structures. To be consistent with the research in other domains, the term "nested structures" is replaced by the term "embedded structures" below.

Embedded structures in the domain of natural language

In the domain of natural language, many studies have focused on the difficulty in processing embedded sentences. Behavioral studies have consistently found that processing sentences in embedded structures (e.g., [The boy [the girl chased] kicked the ball]) is more difficult than processing nonembedded sentences (e.g., The boy kicked the ball on the grass), as evidenced by slower responses (Holmes et al., 1987) and lower accuracy (Opitz & Friederici, 2007). Additionally, neurological studies indicated that the processing of embedded sentences, compared to nonembedded sentences, activated the left inferior frontal gyrus to a greater extent, suggesting increased cognitive demands (Meyer & Friederici, 2016; Shetreet et al., 2009). Furthermore, cognitive efforts induced by other factors involved in embedded sentences have been examined. For example, a neuroimaging study compared the processing of embedded sentences with varying dependency distance, which signifies the distance between the subject noun and its verb in the main sentence. The results showed that processing embedded sentences with a long dependency distance (e.g., "Maria who loved Hans who was good looking kissed Johann") enhanced the functional coupling between the left inferior frontal gyrus and other brain regions compared to embedded sentences with a short dependency distance (e.g., "Maria who cried kissed Johann and that was yesterday night"). This finding suggests that processing long embedded sentences is so demanding that greater interaction is required between different brain regions (Makuuchi et al., 2009).

Embedded structures in the domain of grammar learning

In the domain of artificial grammar, the neural mechanism underlying the processing of embedded structures has been widely explored using two types of symbolic sequences: nonembedded sequences following the adjacent dependency rule $(AB)^n$ (e.g., A1B1A2B2) and embedded sequences following the hierarchical dependency rule A^nB^n (e.g., A1A2B2B1) (Fitch & Hauser, 2004; Levelt, 2020; Perruchet & Rey, 2005; Poletiek et al., 2021). For example, an electrophysiology study used auditory sequences organized according to the A^nB^n rule to measure infants' ability to process embedded structures with different levels of complexity: 5 versus 7 tones. Each level included standard sequences conforming to the embedded rule (e.g., A1A2CB2B1) and deviant sequences violating the embedded rule (e.g., A1A2CB1B2). The results showed that mismatch responses to deviant tones within the 7-tone embedded sequences occurred approximately 90 ms later than those within the 5-tone embedded sequences, indicating that processing embedded structures with greater complexity recruited more cognitive resources compared to those with less complexity (Winkler et al., 2018).

Embedded structures in the music domain

In the music domain, previous studies have also explored the cognitive complexity of embedded transposition chords (Koelsch et al., 2013; Ma et al., 2018). For example, an electrophysiology study separately compared the difference in neural responses of music experts and nonexperts when processing musical chords with and without embedded transposition (Ma et al., 2018). The results indicated that nonexperts exhibited larger amplitudes of the early right anterior negativity (ERAN) and the late N5 components when processing embedded chords compared to nonembedded chords, suggesting that the difficulty in interpreting embedded structures appeared at both early (i.e., a larger ERAN) and late (i.e., a larger N5) processing stages. In contrast, experts showed significant differences between embedded and nonembedded conditions in beta activity, which have been regarded as an indicator of top-down cognitive effort (Bressler & Richter, 2015; Wang et al., 2012). These findings suggest differences in the processing of embedded musical chords between experts and nonexperts.

Embedded structures in the programming domain

In the programming domain, the difficulty in processing embedded coding structures has been repeatedly observed (Asenov et al., 2016; Cetin, 2015; Ginat, 2004; Kelleher & Hnin, 2019). For example, when processing embedded structures, one common mistake made by students was that they ignored the embedded relations between inner and outer repeats but just executed them separately (Izu et al., 2016; Mladenović et al., 2018). Another study analyzed the code scripts generated by novice learners to solve computational problems (Chao, 2016). The results indicated that novices preferred sequential over embedded control structures. Once embedded structures were involved, they debugged the code scripts more frequently, suggesting more errors and greater difficulty.

In addition, a study conducted by Cetin (2015) indicated that novice learners experienced different stages when learning nested loops. In the early stage, students tend to execute each command within a loop explicitly. As they advanced to the late stage, students can conceptualize all commands within a loop as a single function or procedure, thereby eliminating the need for step-by-step execution to obtain an output. Furthermore, given the difficulty in learning embedded coding structures, teaching or learning strategies have been proposed to help novice learners (Cetin, 2020; Yamashita et al., 2016). However, these strategies are not yet well grounded in theory due to the limited understanding of the cognitive mechanism underlying embedded coding structures processing.

The current study

Based on hierarchical complexity theory (Commons, 2007; Commons et al., 1998) and previous studies in other domains (e.g., Makuuchi et al., 2009; Winkler et al., 2018), this study aimed to investigate the cognitive mechanism underlying the comprehension of embedded coding structures among novice learners. Two experiments were conducted in which participants were required to perform a code comprehension task that incorporated both embedded and sequential conditions, each with a different number of commands inserted. Consistent with previous studies (Brünken et al., 2010; Paas et al., 2003), the cognitive load generated in each condition was measured based on task performance, as indexed by accuracy and response times. Furthermore, we measured individual working memory capacity using a behavioral task that targets the components of maintenance and updating. In each experiment, we used these measures to test the three hypotheses derived from the predictions in the Theoretical Framework.

H1 Novice learners would exhibit slower responses and lower accuracy in embedded versus sequential conditions, as well as in the conditions with more versus fewer inserted commands.

H2 Working memory capacity would be negatively related to the differences in response times and accuracy between embedded and sequential conditions, but unrelated to the differences in response times and accuracy caused by the increasing inserted commands.

H3 The differences in response times and accuracy between embedded and sequential conditions would be more significant when there were more inserted commands.

Experiment 1

Method

Participants

Experiment 1 involved a total of 73 participants (mean age = 21.570 years, SD = 1.930, 49 females) who were recruited from Zhejiang University, which is a top-ranked comprehensive university in China (https://www.topuniversities.com/university-rankings/world-university-rankings/2023). Among 71 participants who reported their coding experience, 31 students had never learned coding, and the others had more or less coding learning experience (two students: < 1 month, ten: 1–3 months, eleven: 3–6 months, four: 6–12 months, ten: > 12 months). Finally, 58 participants were included in statistical analyses after excluding 15 participants (i.e., two students did not report coding experience, ten students learned coding longer than 12 months, and three students failed to pass the practice). The effect of prior coding experience has been examined in Additional file 1. All participants signed the informed consent form before participating in the experiment and were reimbursed for their time and travel. This study was approved by the Research Ethics Committee of Zhejiang University.

Code comprehension task

In this task, we created two experimental conditions by organizing two repeat blocks in an embedded or sequential form. Except for this difference, the two conditions were exactly matched in other dimensions (e.g., the number of inserted commands and repeat blocks). As shown in Fig. 2, the sequential structures were composed of two adjacent repeat blocks, with one or two inserted commands placed outside the repeat blocks. In contrast, the embedded structures were designed to nest one repeat block within another, with one or two commands inserted between the outer and inner repeats (Fig. 2). Each command (i.e., "Transform color", "Transform shape", or "Number ± 1, 2, 3") with a size of 2.8 cm × 0.5 cm was displayed to instruct the transformation in the color (i.e., red, blue, yellow, green, and purple), shape (i.e., alternation of color parts), or number (i.e., 1–15) dimension.

Corresponding to the commands within each code snippet displayed on the left side of a 14-in. screen with a resolution of 1920 × 1080 pixels, an initial circle and 14–16 transformed circles, each with an area of 4 cm^2, were displayed on the right side of the screen (Fig. 2). These transformed circles were arranged into 4 rows. Among them, there were 3 or 4 probing errors, suggesting that the changed dimension (i.e., color, shape or number) in the new circle was inconsistent with that instructed by the command. For example, when the command was "Transform color", the transformation applied to the new circle occurred in the number dimension. The adjacent circles were designed to differ only in one dimension to avoid obvious errors that could be detected without processing commands. Additionally, probing errors were not presented consecutively. Participants

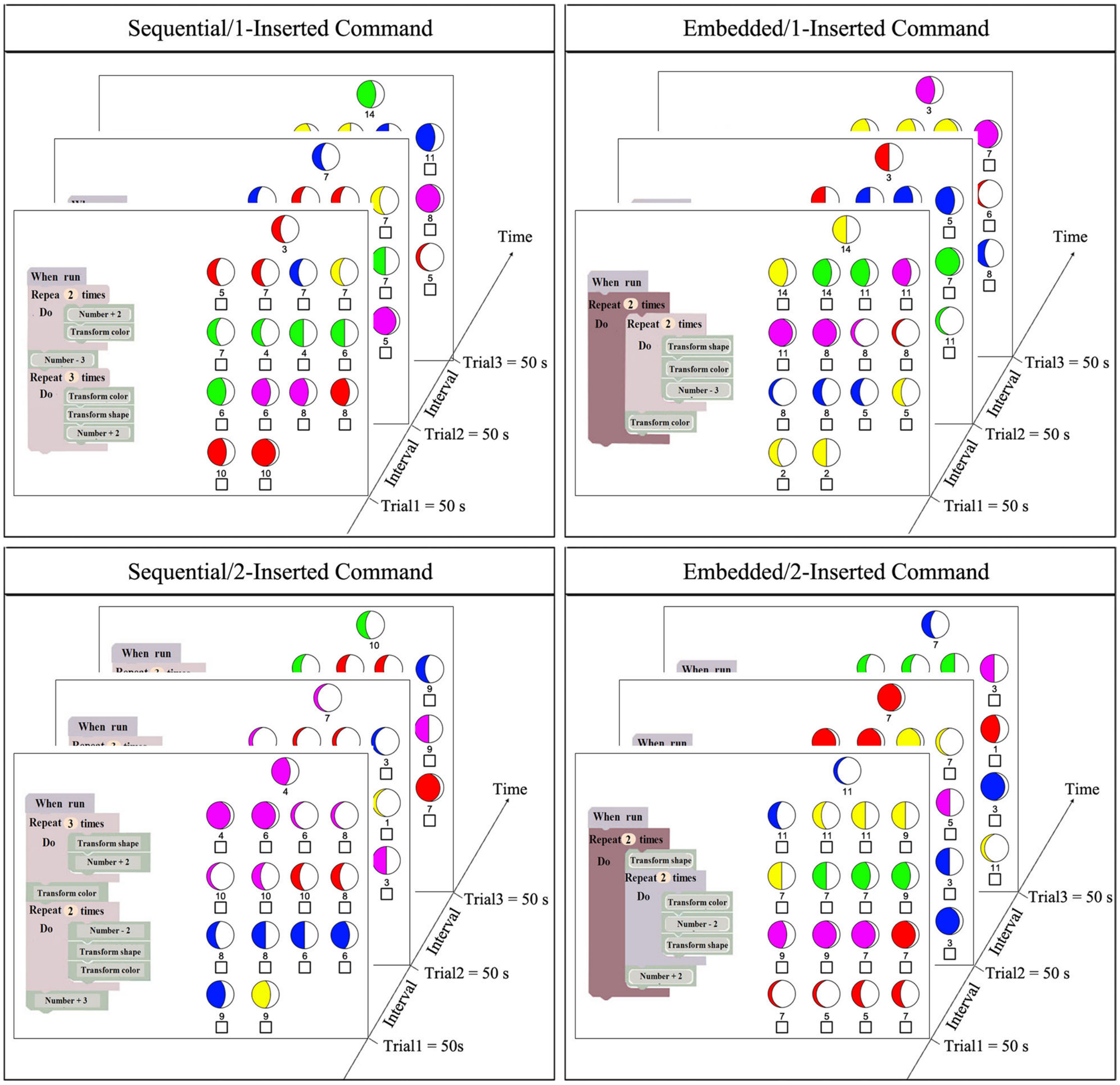


Fig. 2 Code comprehension task in Experiment 1

were instructed to identify all probe errors by checking the boxes underneath the circles.

The task was programmed using E-prime 3.0 software (https://www.eprime.info/). Overall, the task used a 2 (Structure: sequential versus embedded) × 2 (Inserted Command: 1 versus 2) experimental design that yielded 4 types of blocks. Each block included 8 trials. The code snippet in each trial was randomly selected from 16 predesigned code snippets. Participants were asked to respond as fast and accurately as possible within 50 s. After executing all commands in a trial, participants just clicked the "Next" button to proceed to the next trial. The cognitive load in each condition was quantified by the average accuracy of all trials and the average response times of trials with no error.

The experimental procedure for the code comprehension task consisted of three phases. During the learning phase, participants watched an instructional video where the instructor illustrated the execution order of code commands within different structures and provided an example to explain the task. Then, participants practiced the code comprehension task. In total, they practiced

8 trials with 2 in each condition. Only when the practice accuracy was greater than 80% would participants enter the formal testing. Finally, participants performed the formal test with short breaks between different conditions.

Working memory task

We used the *n*-back task to measure working memory because this task specifically measures the maintenance and selective updating of information in mind (Gajewski et al., 2018; Rac-Lubashevsky & Kessler, 2016). In this task, a series of letters (i.e., A, B, C, D, E, F, G, and H) were presented one by one on the screen (Braver et al., 1997). Participants were asked to memorize the letters and press the space bar when the current letter was the same as the first one (i.e., $n=0$) or the one presented n trials ago (e.g., $n=3$), as shown in Fig. 3. After learning these rules, participants performed the practice test. Only when they correctly identified more than 3 out of 6 target letters with no more than 2 nontarget letters inaccurately responded at each level would they enter the formal test. In the formal test, the 0-back level contained 10 blocks with 11 trials in each block, whereas the 3-back level contained 10 blocks with 12 trials in each block. The percentage of targets that needed responses in each block was 27.2% for the 0-back level and 25% for the 3-back level. Each trial lasted 2000 ms with a stimulus duration of 500 ms and an interval between stimuli of 1500 ms.

Individuals' working memory capacity was indexed by the difference in discriminability (*d*-prime) and response times between 0- and 3-back conditions. For each condition, *d*-prime was calculated using the following formula (Haatveit et al., 2010): $d\text{-prime} = Z\text{ (HIT)} - Z\text{ (FA)}$. The HIT refers to the proportion of targets that are correctly identified, whereas the false alarm (FA) is the proportion of nontargets that are incorrectly identified as targets. Additionally, the mean response times of correctly identified targets were calculated for each condition. Finally, working memory performance was calculated using the difference in *d*-prime (3-back minus 0-back) and mean response times (0-back minus 3-back), which was separately represented as $\text{WM}_{\text{dprime}}$ and WM_{rt} below. In this study, greater $\text{WM}_{\text{dprime}}$ and WM_{rt} indicated better working memory capacity.

Statistical analyses

To compare the difference in cognitive load between conditions, a series of linear mixed-effects models were built with performance in the code comprehension task (i.e., accuracy or response times) as the dependent variable and with Structure (sequential versus embedded) and Inserted Command (1 versus 2) as independent variables. The basic models only contained the main effects of Structure and Inserted Command. Such basic models were further compared with the new ones that contained both the main effects and the Structure × Inserted Command interaction. The new models could be chosen only if the Akaike's Information Criterion (AIC) value decreased more than 2 compared to the basic models. For all models, the Satterthwaite approximation was adopted to estimate the degrees of freedom. If there was a significant Structure × Inserted Command interaction, further analyses were conducted to interpret this interaction.

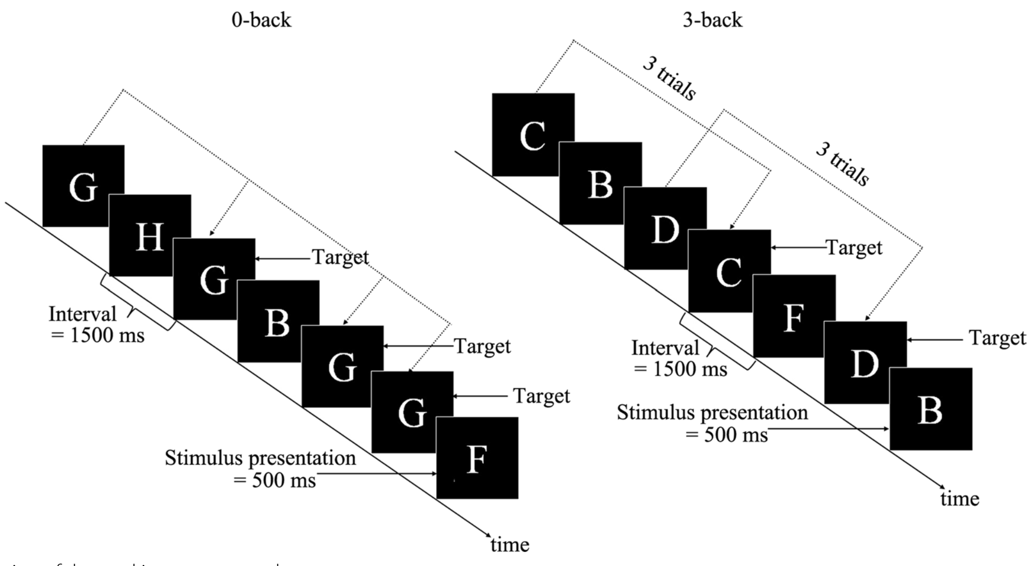


Fig. 3 Illustration of the working memory task

Additionally, to test the relations between working memory and cognitive load, we constructed separate models for working memory as indexed by discriminability (WM_{dprime}) and response times (WM_{rt}) (see the details about these models in Additional file 5). Specifically, the basic models contained the main effects of Structure, Inserted Command, and Working Memory and the Structure × Inserted Command interaction. In addition to these variables, the new models further included the Structure × Working Memory and Inserted Command × Working Memory interactions. We selected the basic or new models based on the AIC value mentioned above. Further analyses were conducted if there was any significant interaction involving working memory. The significance level for results involving working memory was adjusted to 0.025 to reduce the Type I error rates.

In the above analyses, the subject factor was included in all models to test the random intercept effect. If this effect was not significant, the subject factor was removed, and the analyses were continued using fixed-effects models.

Results

Cognitive load in the code comprehension task

For accuracy, we selected the basic model that contained only the main effects of Structure and Inserted Command. However, there was no significant difference in accuracy between embedded and sequential conditions as well as between conditions with 1- versus 2-inserted commands ($ps > 0.379$). For response times, the selected model included the main effects of Structure and Inserted Command as well as the Structure × Inserted Command interaction. The results indicated that both the main effects and the interaction were significant (Structure: $\beta = -2.271$, SE = 0.733, t (169) = − 3.100, $p < 0.01$; Inserted Command: $\beta = -3.308$, SE = 0.733, t (169) = − 4.515, $p < 0.001$; Interaction: $\beta = 3.680$, SE = 1.033, t (169) = 3.561, $p < 0.001$). Such main effects suggested that responses were faster in sequential versus embedded conditions, as well as in conditions with 1- versus 2-inserted commands. *Then,* we conducted further analyses to interpret the significant interaction. The results indicated that when there were two inserted commands, responses were faster in sequential versus embedded conditions ($\beta = -2.214$, SE = 0.642, t (56) = − 3.447, $p < 0.01$), but this difference was reversed when there was one inserted command ($\beta = 1.421$, SE = 0.711, t (57) = 2.000, $p = 0.05$). *Additionally*, when there were embedded relations between control structures, the responses were slower in the conditions with 2- versus 1-inserted command ($\beta = -3.277$, SE = 0.920, t (112) = − 3.562, $p < 0.01$). In contrast, such condition difference was not significant when control structures were organized sequentially (Fig. 4).

Relations between cognitive load and working memory

Since the main effects of Structure and Inserted Command on accuracy were not significant, response times were selected to test the relations between cognitive load and working memory. In addition, the results indicated that when there was one inserted command, responses were faster in embedded versus sequential conditions, which was inconsistent with our hypothesis. Therefore, we only examined the relations of working memory to the difference in cognitive load between embedded and

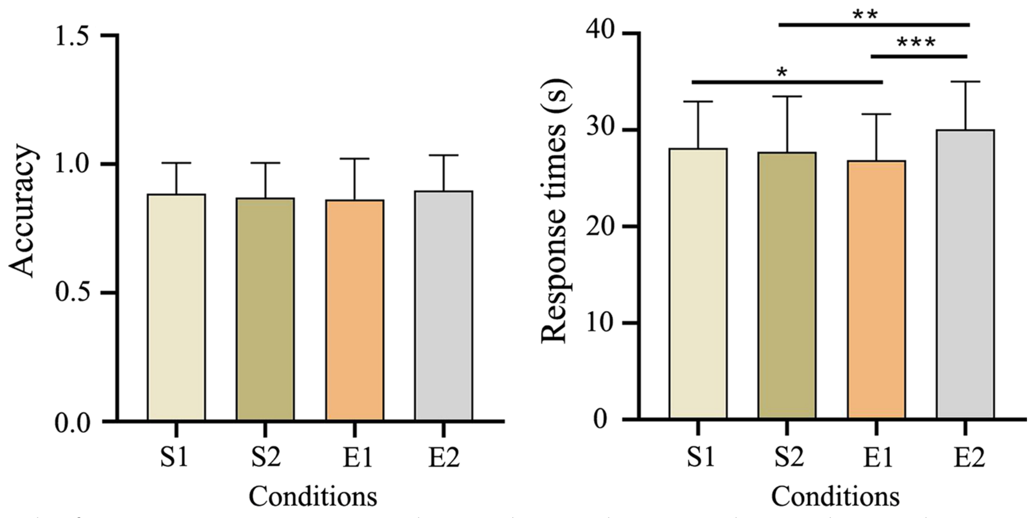


Fig. 4 Behavioral performance in Experiment 1. S1: Sequential/1-Inserted Command; S2: Sequential/2-Inserted Command; E1: Embedded/1-Inserted Command; E2: Embedded/2-Inserted Command. Error bars represent standard deviation

sequential conditions when there were two inserted commands. Accordingly, we compared the basic model that only included the main effects of Structure and Working Memory to the new model that included both the main effects and the Structure × Working Memory interaction. The new model was then selected, but there were no significant results involving working memory (*ps* > 0.401).

Similarity, only in embedded condition, the difference in response times between conditions with 1- versus 2-inserted commands was significant. Therefore, we examined the relations of working memory to the difference in cognitive load induced by different numbers of inserted commands in embedded condition. We compared the basic model that included the main effects of Inserted Command and Working Memory to the new model that also included the Inserted Command × Working Memory interaction. With the new model selected, we did not find significant results involving working memory (*ps* > 0.144).

Interim discussion

Consistent with our hypotheses, the findings indicated that when there were two commands inserted between repeat control structures, participants responded slower in embedded versus sequential conditions, suggesting greater cognitive load. However, when there was one inserted command, responses were slower in sequential versus embedded conditions. The findings might be associated with the use of cognitive strategies. Specifically, as reported by participants, when there was only one inserted command within embedded structure, all commands in the repeat control block could be easily memorized as a *chunk* containing information about the transformed dimensions and their execution orders. *For example*, in embedded condition with 1-inserted command (Fig. 2), after executing the commands (i.e., "Transform shape", "Transform color", and "Number-3") within the inner control structure for the first time, the three commands and related information were memorized, and their subsequent execution did not require a shift in attention to the code snippet on the left side of the screen. In contrast, in sequential condition, the commands contained in the upper and lower repeat blocks were different. Therefore, if the chunking strategy was ever used, participants had to chunk the two repeat blocks separately, which might contribute to the longer response times in sequential than in embedded conditions.

In addition, the using of chunking strategy may be affected by the number of inserted commands. With more commands inserted into embedded condition, greater challenge might be imposed on the working memory system. Previous studies have indicated that such a challenge may prevent the use of chunking strategy or reduce the benefit of using this strategy (Janssen & Brumby, 2010; Schorr et al., 2003). Accordingly, we speculated that the effect of the chunking strategy might be compromised when there were two commands inserted between control structures in embedded condition. Such speculation was supported by the results of this study, which indicated that when there were two inserted commands, the comprehension of embedded structures led to slower responses than sequential structures.

Furthermore, there was no significant relation between working memory and the difference in response times between embedded and sequential conditions. One possible reason may be that listing all transformed circles corresponding to the commands on the screen reduced the cognitive load associated with the mental representation of embedded relations. For example, in embedded condition with 2-inserted commands (Fig. 2), the execution of each external repeat included eight commands that were exactly represented by two rows of circles. According to the number of circles that had been processed, participants could easily count the times the inner and outer loops had been repeated. Therefore, we conjectured that the organization of circles concretized the hierarchical representation of embedded relations and reduced the cognitive cost induced by switching between inner and outer repeats. Another possible reason for the absence of a relation between working memory and cognitive load might be related to the use of different colors to distinguish inner and outer repeat control structures only in embedded condition. This color difference may help participants build the representation of hierarchical relations, which may reduce the cognitive load generated from processing the code commands in embedded condition.

To summarize, the design of the code comprehension task in Experiment 1 might lead to the unexpected use of cognitive strategies and the decrease in cognitive load associated with the processing of hierarchical relations in embedded condition. Therefore, we further conducted Experiment 2, in which the code comprehension task was redesigned to exclude the possible impacts of strategy use and stimuli presentation on cognitive processing (see details below).

Experiment 2

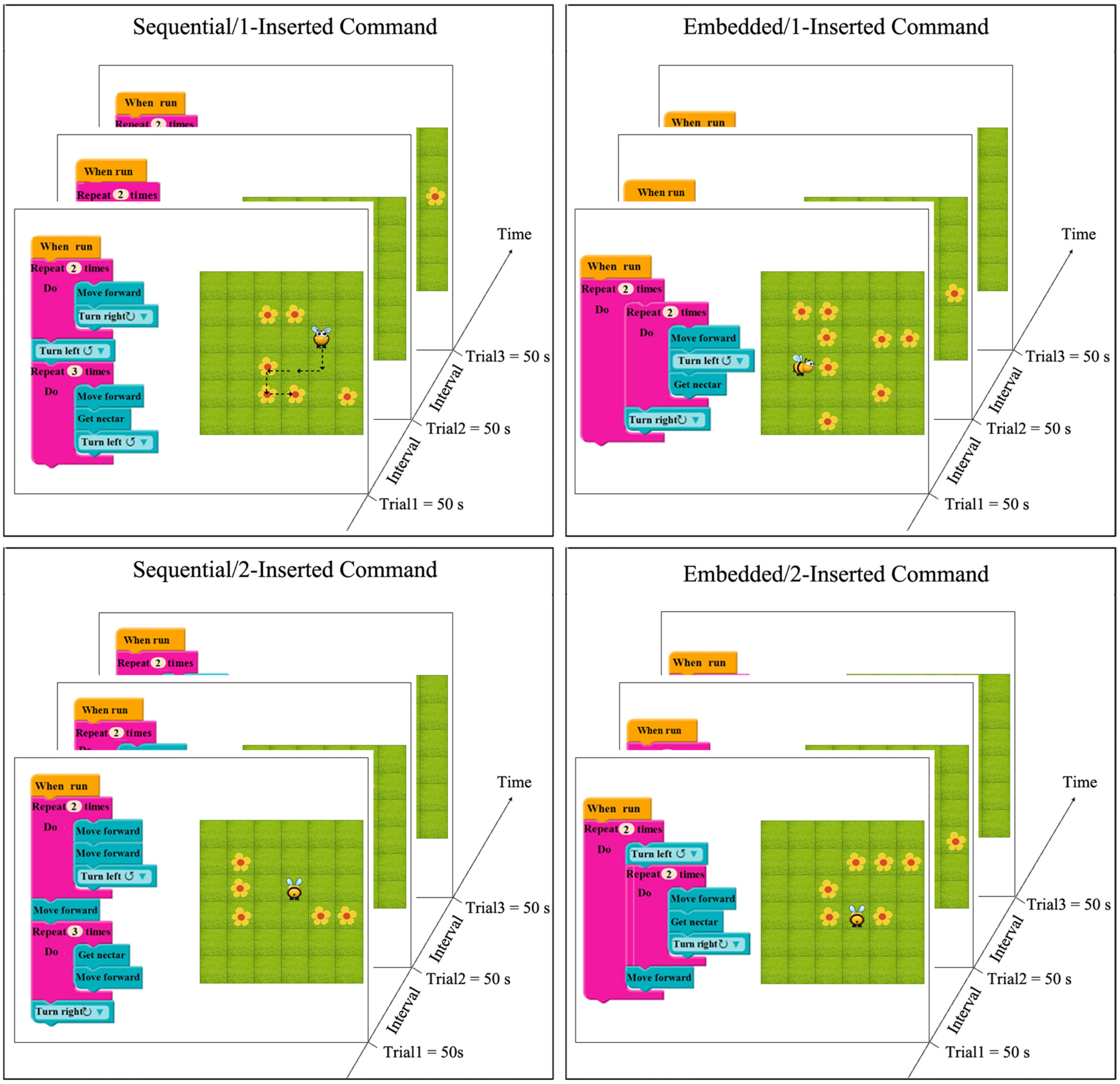
Method

Participants

A total of 79 college students (mean age = 21.67, SD = 1.97, 55 female) participated in Experiment 2, including 29 participants from Experiment 1 and 50 newly recruited participants.The influence of participant resue have been examined in Additional file 2. All participants performed the *n*-back task in Experiment 1 or 2. Additionally, among 75 participants with reported coding experience, 38 students had never learned coding, and the others had learning experience shorter than one year (three students: < 1 month, ten: 1–3 months, eleven: 3–6 months, four: 6–12 months, nine: > 12 months). Finally, 66 participants were included in the analyses, with 13 excluded (i.e., four students did not report coding experience, and nine students learned coding for longer than 12 months).

Code comprehension task

As discussed above, the design of the code comprehension task in Experiment 1 might induce confusions that we did not expect. Therefore, we redesigned the task in Experiment 2. Consistent with the task in Experiment 1, the code snippet in the new task also assembled two repeat control blocks in an embedded or sequential manner. We also inserted one or two commands outside the repeat block in sequential condition and between the inner and outer repeat blocks in embedded condition (Fig. 5). Overall, 4 types of code snippets were classified


Sequential/1-Inserted Command
Embedded/1-Inserted Command
Sequential/2-Inserted Command
Embedded/2-Inserted Command
Time
Trial3 = 50 s
Interval
Trial2 = 50 s
Interval
Trial1 = 50 s

Sequential/1-Inserted Command:
```
When run
Repeat 2 times
  Do Move forward
     Turn right
Turn left
Repeat 3 times
  Do Move forward
     Get nectar
     Turn left
```

Embedded/1-Inserted Command:
```
When run
Repeat 2 times
  Do Repeat 2 times
       Do Move forward
          Turn left
          Get nectar
     Turn right
```

Sequential/2-Inserted Command:
```
When run
Repeat 2 times
  Do Move forward
     Move forward
     Turn left
Move forward
Repeat 3 times
  Do Get nectar
     Move forward
Turn right
```

Embedded/2-Inserted Command:
```
When run
Repeat 2 times
  Do Turn left
     Repeat 2 times
       Do Move forward
          Get nectar
          Turn right
     Move forward
```

Fig. 5 Code comprehension task in Experiment 2

based on the Structure (sequential versus embedded) and Inserted Command (1 versus 2). Contrary to the task in Experiment 1, the new task required participants to imagine the moving route of a bee on a map based on their comprehension of code commands. Therefore, participants need to maintain and update the location and direction of the bee in their minds all the time. Additionally, the embedded and sequential conditions were matched in the use of colors. These changes in task design were expected to exclude the possible influence of strategy use and stimuli presentation, as mentioned in the "Discussion" section of Experiment 1.

The new code comprehension task was programmed using Opensesame software (https://osdoc.cogsci.nl/) with a 2 (Structure: sequential versus embedded) × 2 (Inserted Command: 1 versus 2) experimental design. Each condition contained 8 trials involving different code snippets that were randomly selected from 16 predesigned code snippets.

The experimental procedure was the same as that in Experiment 1. On the left side of the screen, a code snippet containing 14 to 16 commands was presented, each command had a height of 0.5 cm and a width between 1.4 and 1.8 cm. The commands "Move forward", "Turn left", "Turn right", and "Get nectar" were designed with reference to the code.org platform (https://code.org/) (Fig. 5). The number of different types of commands was well-matched (see Additional file 6) across experimental conditions, and the order of commands was different between conditions to prevent the learning effect.

The actions instructed by code commands were executed on a grid map presented on the right side of the screen (Fig. 5). The map had 6×6 grids, each of which was 1.6 cm × 1.6 cm in size. A bee and some nectars were presented on the map. The forward direction of the bee was always in line with the orientation of its head. In addition to the nectars in the moving route of the bee, three or four nectars were placed in the grids outside the moving route to prevent guessing.

As participants performed the task, the bee stayed in its initial grid all the time. Participants had to imagine the moving route of the bee and click on each grid to perform the actions as instructed by commands. *Specifically*, when the command was "Move forward", participants had to click on the grid that the bee would move to; when the command was "Turn left", "Turn right", or "Get nectar", participants had to click on the current grid again. The position and time of each click made by participants were recorded. In each trial, participants were asked to respond as quickly and accurately as possible within 50 s.

These recordings of clicks were used to calculate accuracy. *First*, we created MATLAB scripts to reconstruct the moving routes of the bee based on these clicks. *Then*, we manually counted the proportion of correct grids that the bee passed on each route to represent the accuracy of each trial (see Additional file 4 for examples). *Finally*, for each condition, the accuracy was computed as the mean accuracy of all trials, and the response times were averaged across all trials in which the bee had finally reached the correct destination. Two raters were trained by an expert in a group meeting to determine whether correct destinations had been reached based on the reconstructed routes. After reaching good reliability for 160 trials (alpha coefficient = 0.972), each rater independently evaluated the data from half of the participants. Approximately two weeks later, the raters re-evaluated 480 trials to calculate the intrarater reliability. The two raters were consistent with their own ratings according to Cohen's kappa: rater 1 = 0.991; rater 2 = 0.997.

Working memory task

Working memory was measured using the same *n*-back task as described in Experiment 1.

Statistical analyses

The data analysis approaches were the same as those used in Experiment 1. In addition, to exlcude the possible influence of different number of commands between conditions, we further re-analyzed the data from embedded and sequential conditions with only 1-inserted command (see Additional file 3).

Results

Cognitive load in the code comprehension task

When accuracy as a measure of cognitive load was included as the dependent variable, the model that contained the main effects of Structure and Inserted Command had the best fit. The results indicated that the main effect of Inserted Command was significant ($\beta = 0.056$, SE = 0.010, t (188) = 5.442, $p < 0.001$), suggesting that accuracy was greater in conditions with 1- versus 2-inserted commands. However, the main effect of Structure was not significant ($\beta = 0.003$, SE = 0.010, t (186) = 0.316, $p = 0.753$). For response times, the selected model included the main effects of Structure and Inserted Command as well as the Structure × Inserted Command interaction. The results indicated that the main effects of both Structure and Inserted Command were significant (*Structure*: $\beta = -3.735$, SE = 0.623, t (181) = − 5.991, $p < 0.001$; *Inserted Command*: $\beta = -3.379$, SE = 0.619, t (181) = − 5.460, $p < 0.001$), suggesting that the responses were faster in sequential versus embedded conditions as well as in 1- versus 2- inserted commands conditions (Fig. 6). However, the interaction between Structure and Inserted Command was not significant ($\beta = 1.081$, SE = 0.867, t (181) = 1.248, $p > 0.05$), suggesting that

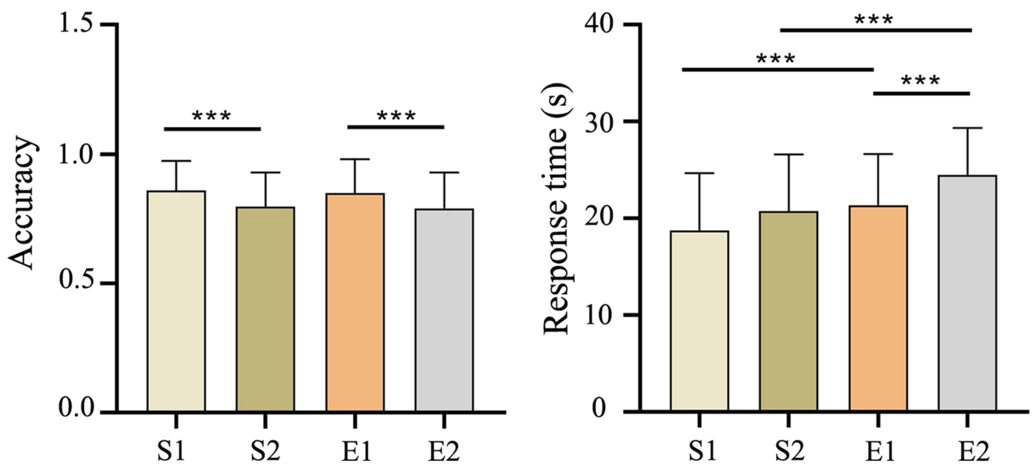


Fig. 6 Behavioral performance in Experiment 2. S1: Sequential/1-Inserted Command; S2: Sequential/2-Inserted Command; E1: Embedded/1-Inserted Command; E2: Embedded/2-Inserted Command. Error bars represent standard deviation

the extra cognitive load induced by embedded coding structures was not affected by the number of inserted commands.

Relations between cognitive load and working memory

To further examine the relations between cognitive load and working memory, we selected the response times in code comprehension task as the dependent variable since there was no significant difference in accuracy between embedded and sequential conditions. The best-fitting models included the main effects of Structure, Inserted Command, and Working Memory as well as the Structure × Inserted Command, Working Memory × Structure and Working Memory × Inserted Command interactions. The results indicated that there were significant interactions between Structure and Working Memory (WM_{dprime}: $\beta = -1.523$, $SE = 0.444$, $t(172) = -3.434$, $p < 0.01$; WM_{rt}: $\beta = -1.229$, $SE = 0.421$, $t(172) = -2.916$, $p < 0.01$).

To interpret these interactions, we tested the relations between working memory and the difference in response times between embedded and sequential conditions. The results indicated that for participants with greater working memory, responses slowed down less in embedded versus sequential conditions (Fig. 7). In contrast, there was no significant interaction between Inserted Command and Working Memory ($ps > 0.401$), suggesting that working memory was not significantly related to the change in response times caused by the increase in inserted commands (Fig. 7).

Interim discussion

Consistent with our hypotheses, the results indicated that responses were slower in embedded versus sequential conditions, suggesting that the cognitive load was greater in the former than the latter when these two conditions were well-matched in other dimensions. Such condition difference in cognitive load may be caused by the greater hierarchical complexity involved in embedded structures compared to sequential structures. Additionally, as we expected, the extra cognitive load generated by processing embedded structures was negatively related to individual working memory capacity, indicating that the amount of cognitive load induced by embedded coding structures varies significantly across individuals with different working memory capacities. In contrast, there was no significant relation between working memory and cognitive load induced by more inserted commands.

Therefore, although both embedded structures and the increase in inserted commands induced extra cognitive load, there are differences in the cognitive mechanisms underlying the processing of these two types of information. As the maintenance and manipulation of information within hierarchical structures relies upon the working memory system (Glahn et al., 2000; Stukken et al., 2016), we infer that processing embedded structures involves greater participation of the working memory system to maintain and update the hierarchical information in mind.

Additionally, inconsistent with our prediction, the results suggested that the difference in cognitive load between embedded and sequential conditions was not affected by the number of inserted commands. One possible reason might be that the change in inserted commands from one to two was too small to influence the processing of hierarchical relations in embedded condition. This speculation needs to be further tested by creating conditions with more commands inserted.

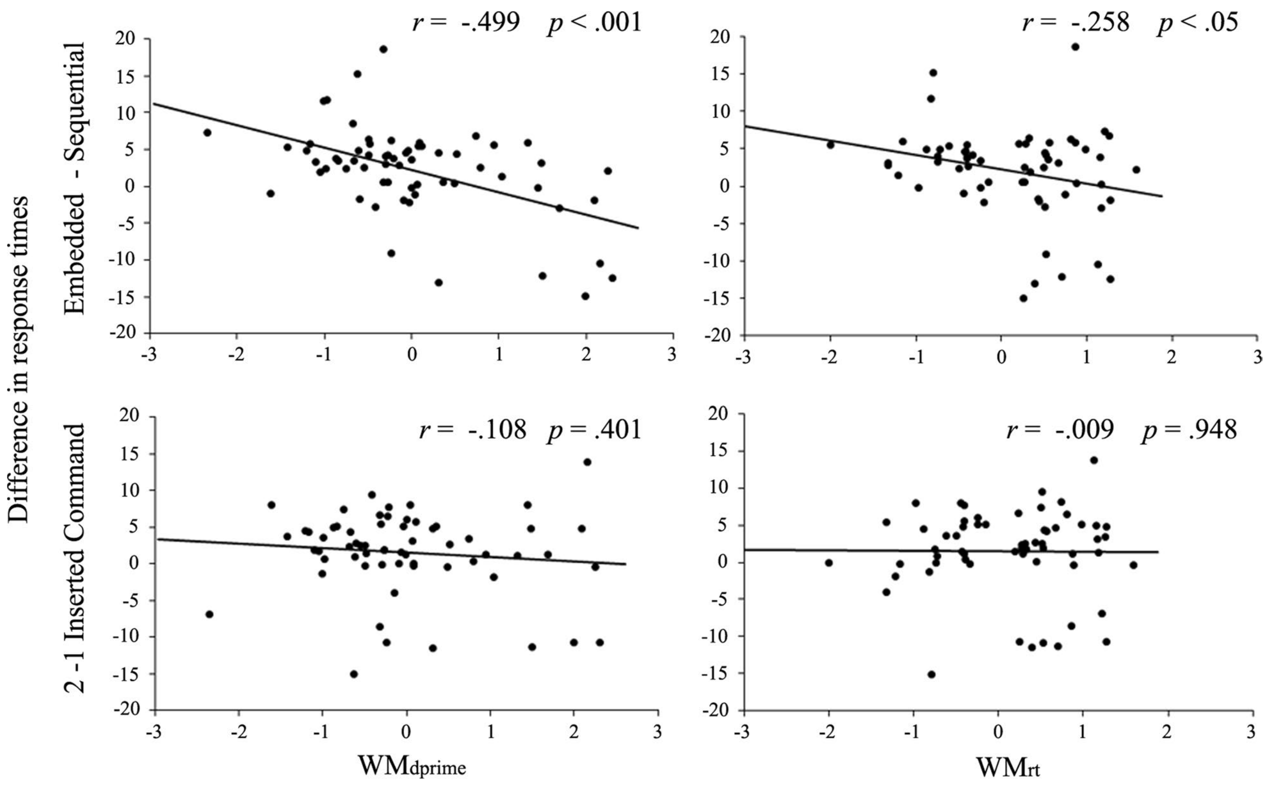


Fig. 7 Relations of working memory to the difference in cognitive load across different conditions. WM_{dprime} and WM_{rt} separately refer to working memory capacity as indexed by *d*-prime and response times

General discussion

The difficulty in processing embedded coding structures has been repeatedly observed in the coding learning of novice learners. Based on hierarchical complexity theory, we predicted that such difficulty is related to the complex hierarchies involved in embedded coding structures. To test such hypotheses, we developed a well-controlled experimental paradigm to measure and compare the difference in cognitive load generated from processing the codes within embedded and sequential structures. As these two conditions were well-matched in all dimensions except hierarchical complexity, the greater cognitive load in embedded versus sequential conditions can be attributed to the difficulty in processing complex hierarchical information.

In addition, the extra cognitive load induced by embedded structures was significantly related to working memory capacity, supporting the hypotheses that processing embedded structures may exert great demands on the working memory system. In other words, individuals with different working memory capacities vary in the ability to process embedded coding structures. These findings suggest that the difficulty in processing embedded coding structures for novice learners may arise from the maintenance and manipulation of complex hierarchical information, which imposes a great burden on the working memory system. Understanding such cognitive mechanisms provides important educational implications for helping novices learn and use embedded coding structures.

First, to reduce the cognitive load induced by complex hierarchies involved in embedded coding structures, well-designed cognitive strategies or scaffolds should be used to support the processing of embedded coding structures. For example, providing reminders during the transition between inner and outer control structures can reduce the cognitive demands on maintaining and manipulating hierarchical information in mind. In addition, as discussed in Experiment 1, using chunking strategies in certain situations can reduce the cognitive efforts required to process complex hierarchical structures.

Second, it is important to integrate strategies in appropriate ways at different learning stages. For example, teaching strategies can be incorporated to help novice learners with the hierarchical processing of embedded structures. Specifically, teachers can use a visual display, such as Fig. 1, to show novice learners the hierarchical relations involved in embedded coding structures and help students track the execution of code commands. However, visualization strategies should also be gradually reduced or cancelled to foster the development of students' ability to process hierarchical information. In addition, this study found that the cognitive load induced by embedded structures was significantly related to individual working memory capacity, which varied substantially across individuals (e.g., Jarrold & Towse, 2006; Luck & Vogel, 2013). Therefore, it is important to consider individual differences when teaching novices to learn embedded coding structures.

Third, hierarchical decomposition and organization can be used to help students create embedded coding structures. CT emphasizes developing students' ability to use coding concepts (e.g., embedded structures) and skills to solve problems. However, students often struggle with identifying which code commands should be placed within the inner or outer control structures when creating embedded structures to solve problems (e.g., Mladenović et al., 2018). To address this challenge, it is essential to educate students on breaking down complex problems into smaller subproblems and organizing them hierarchically. This approach enables the effective representation of code commands at different hierarchical levels within embedded coding structures.

Strengths and limitations

This study has limitations. *First*, the differences in some visual properties, such as visual length and color use, between embedded and sequential conditions as well as between different types of code commands may interfere with the findings of this study. Future studies can use other techniques or control experiments to test whether or how these properties influence the findings. *Additionally*, the code comprehension task used in Experiment 2 was designed to have good ecological validity, but such a design may prevent us from carrying out command-level analyses to explore whether processing embedded coding structures is generally more challenging compared to sequential structures or if the challenge is specific to certain commands. Future studies can use alternative research techniques (e.g., neuroimaging methods) to explore the cognitive load associated with each command within embedded structures. *Moreover*, this study only included adults with few coding experience. Recently, wide attention has been given to coding education for children and adolescents (e.g., Angeli & Giannakos, 2020; Lye & Koh, 2014), whose ability to maintain and update information in working memory was significantly lower than that of adults (Jarrold & Towse, 2006; Kharitonova et al., 2015). Therefore, future studies should examine whether there are age differences in the processing of embedded coding structures. Despite these limitations, this study also has strengths. Specifically, to our knowledge, this is the first study to quantify the difficulty in processing embedded coding structures through the measurement of cognitive load induced by hierarchical complexity and to examine its relation with individual working memory capacity.

Conclusion

To conclude, this study designed a well-controlled experimental paradigm to measure the cognitive load induced by the comprehension of block-based codes within embedded versus sequential structures. The results showed that novice learners generated greater cognitive load when processing embedded structures compared to sequential structures. In addition, although the increase in inserted commands also induced additional cognitive load, only the extra cognitive load induced by embedded coding structures was significantly related to individual working memory capacity. These findings suggest that processing codes within embedded structures exerts great demands on the working memory system to maintain and manipulate hierarchical information. This cognitive mechanism can be applied to guide the design of instructional strategies in coding education and inspire its integration with educational practices in other STEM fields.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s40594-023-00432-9.

Additional file 1. Testing whether prior coding experiences affected behavioral performance.

Additional file 2. Testing whether reusing participants affected the results of Experiment 2.

Additional file 3. Re-analyzing data from conditions with 1 inserted command in Experiment 2.

Additional file 4. Rating examples from Experiment 2.

Additional file 5. The AIC values for all basic and comparison models.

Additional file 6. Number of each command type within 16 predesigned code snippets in each condition.

Acknowledgements
We thank all participants for their contributions.

Author contributions
XH and FG were major contributors in study design and administration, data analyses, and manuscript preparation. ZX and MG contributed to data collection and analyses. YH was a major contributor in data analyses and writing. All authors read and approved the final manuscript.

Funding
This work was supported by the National Natural Science Foundation of China (General Program: 62077042; 81971245), Fundamental Research Funds for the Central Universities, the MOE (Ministry of Education in China) Project of Humanities and Social Sciences (20YJA190002), and Zhejiang University Education Foundation Global Partnership Fund.

Availability of data and materials
The dataset used and/or analyzed during the current study is available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 2 May 2022 Accepted: 24 May 2023
Published online: 15 June 2023

References
Angeli, C., & Giannakos, M. (2020). Computational thinking education: Issues and challenges. *Computers in Human Behavior, 105*, 106185. https://doi.org/10.1016/j.chb.2019.106185
Anmarkrud, Ø., Andresen, A., & Bråten, I. (2019). Cognitive load and working memory in multimedia learning: Conceptual and measurement issues. *Educational Psychologist, 54*(2), 61–83. https://doi.org/10.1080/00461520.2018.1554484
Asenov, D., Hilliges, O., & Müller, P. (2016). The effect of richer visualizations on code comprehension. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 5040–5045). https://doi.org/10.1145/2858036.2858372
Baddeley, A. (1992). Working memory. *Science, 255*(5044), 556–559. https://doi.org/10.1126/science.1736359
Badre, D., & Nee, D. E. (2018). Frontal cortex and the hierarchical control of behavior. *Trends in Cognitive Sciences, 22*(2), 170–188. https://doi.org/10.1016/j.tics.2017.11.005
Barrett, L. F., Tugade, M. M., & Engle, R. W. (2004). Individual differences in working memory capacity and dual-process theories of the mind. *Psychological Bulletin, 130*(4), 553. https://doi.org/10.1037/0033-2909.130.4.553
Bers, M. U., González-González, C., & Armas-Torres, M. B. (2019). Coding as a playground: Promoting positive learning experiences in childhood classrooms. *Computers & Education, 138*, 130–145. https://doi.org/10.1016/j.compedu.2019.04.013
Braver, T. S., Cohen, J. D., Nystrom, L. E., Jonides, J., Smith, E. E., & Noll, D. C. (1997). A parametric study of prefrontal cortex involvement in human working memory. *NeuroImage, 5*(1), 49–62. https://doi.org/10.1006/nimg.1996.0247
Bressler, S. L., & Richter, C. G. (2015). Interareal oscillatory synchronization in top-down neocortical processing. *Current Opinion in Neurobiology, 31*, 62–66. https://doi.org/10.1016/j.conb.2014.08.010
Brünken, R., Seufert, T., & Paas, F. (2010). Measuring cognitive load. *Cognitive Load Theory*, 181–202. https://doi.org/10.1017/cbo9780511844744.011
Cetin, I. (2015). Students' understanding of loops and nested loops in computer programming: An APOS theory perspective. *Canadian Journal of Science, Mathematics and Technology Education, 15*, 155–170. https://doi.org/10.1080/14926156.2015.1014075
Cetin, I. (2020). Teaching loops concept through visualization construction. *Informatics in Education an International Journal, 19*(4), 589–609. https://doi.org/10.15388/infedu.2020.26
Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education, 95*, 202–215. https://doi.org/10.1016/j.compedu.2016.01.010
Commons, M. L. (2007). Introduction to the model of hierarchical complexity. *Behavioral Development Bulletin, 13*(1), 1. https://doi.org/10.1037/h0100493
Commons, M. L., Trudeau, E. J., Stein, S. A., Richards, F. A., & Krause, S. R. (1998). Hierarchical complexity of tasks shows the existence of developmental stages. *Developmental Review, 18*(3), 237–278. https://doi.org/10.1006/drev.1998.0467
Fitch, W. T., & Hauser, D. M. (2004). Computational constraints on syntactic processing in a nonhuman primate. *Science, 303*(5656), 377–380. https://doi.org/10.1126/science.108940
Gajewski, P. D., Hanisch, E., Falkenstein, M., Thönes, S., & Wascher, E. (2018). What does the n-Back task measure as we get older? Relations between working-memory measures and other cognitive functions across the lifespan. *Frontiers in Psychology, 9*, 2208. https://doi.org/10.3389/fpsyg.2018.02208
Ginat, D. (2004). On novice loop boundaries and range conceptions. *Computer Science Education, 14*(3), 165–181. https://doi.org/10.1080/0899340042000302709
Glahn, D. C., Cannon, T. D., Gur, R. E., Ragland, J. D., & Gur, R. C. (2000). Working memory constrains abstraction in schizophrenia. *Biological Psychiatry, 47*(1), 34–42. https://doi.org/10.1016/S0006-3223(99)00187-0
Haatveit, B. C., Sundet, K., Hugdahl, K., Ueland, T., Melle, I., & Andreassen, O. A. (2010). The validity of d prime as a working memory index: Results from the "Bergen n-back task." *Journal of Clinical and Experimental Neuropsychology, 32*(8), 871–880. https://doi.org/10.1080/13803391003596421
Hochmann, J. R., Azadpour, M., & Mehler, J. (2008). Do humans really learn AnBn artificial grammars from exemplars? *Cognitive Science, 32*(6), 1021–1036. https://doi.org/10.1080/03640210801897849
Holmes, V. M., Kennedy, A., & Murray, W. S. (1987). Syntactic structure and the garden path. *The Quarterly Journal of Experimental Psychology Section A, 39*(2), 277–293. https://doi.org/10.1080/14640748708401787
Hu, Y., Chen, C. H., & Su, C. Y. (2021). Exploring the effectiveness and moderators of block-based visual programming on student learning: A meta-analysis. *Journal of Educational Computing Research, 58*(8), 1467–1493. https://doi.org/10.1177/0735633120945935
Izu, C., Weerasinghe, A., & Pope, C. (2016). A study of code design skills in novice programmers using the SOLO taxonomy. In *Proceedings of the 2016 ACM Conference on International Computing Education Research* (pp. 251–259). https://doi.org/10.1145/2960310.2960324.
Janssen, C. P., & Brumby, D. P. (2010). Strategic adaptation to performance objectives in a dual-task setting. *Cognitive Science, 34*(8), 1548–1560. https://doi.org/10.1111/j.1551-6709.2010.01124.x
Jarrold, C., & Towse, J. N. (2006). Individual differences in working memory. *Neuroscience, 139*(1), 39–50. https://doi.org/10.1016/j.neuroscience.2005.07.002
Kelleher, C., & Hnin, W. (2019). Predicting cognitive load in future code puzzles. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1–12). https://doi.org/10.1145/3290605.3300487
Kharitonova, M., Warren, W., & Sheridan, M. A. (2015). As working memory grows: A developmental account of neural bases of working memory capacity in 5-to 8-year old children and adults. *Journal of Cognitive Neuroscience, 27*(9), 1775–1788. https://doi.org/10.1162/jocn
Koelsch, S., Rohrmeier, M., Torrecuso, R., & Jentschke, S. (2013). Processing of hierarchical syntactic structure in music. *Proceedings of the National Academy of Sciences, 110*(38), 15443–15448. https://doi.org/10.1073/pnas.1300272110
Lakretz, Y., Dehaene, S., & King, J. (2020). What limits our capacity to process nested long-range dependencies in sentence comprehension? *Entropy, 22*(4), 446. https://doi.org/10.3390/e22040446
Levelt, W. J. (2020). On empirical methodology, constraints, and hierarchy in artificial grammar learning. *Topics in Cognitive Science, 12*(3), 942–956. https://doi.org/10.1111/tops.12441
Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2020a). Computational thinking is more about thinking than computing. *Journal for STEM Education Research, 3*, 1–18. https://doi.org/10.1007/s41979-020-00030-2
Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2020b). On computational thinking and STEM education. *Journal for STEM Education Research, 3*, 147–166. https://doi.org/10.1007/s41979-020-00044-w
Liu, A. S., & Schunn, C. D. (2020). Predicting pathways to optional summer science experiences by socioeconomic status and the impact on science attitudes and skills. *International Journal of STEM Education, 7*, 1–22. https://doi.org/10.1186/s40594-020-00247-y
Luck, S. J., & Vogel, E. K. (2013). Visual working memory capacity : From psychophysics and neurobiology to individual differences. *Trends in Cognitive Sciences, 17*(8), 391–400. https://doi.org/10.1016/j.tics.2013.06.006
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. https://doi.org/10.1016/j.chb.2014.09.012

Ma, X., Ding, N., Tao, Y., & Yang, Y. F. (2018). Differences in neurocognitive mechanisms underlying the processing of center-embedded and non-embedded musical structures. *Frontiers in Human Neuroscience, 12*, 425. https://doi.org/10.3389/fnhum.2018.00425

Makuuchi, M., Bahlmann, J., Anwander, A., & Friederici, A. D. (2009). Segregating the core computational faculty of human language from working memory. *Proceedings of the National Academy of Sciences, 106*(20), 8362–8367. https://doi.org/10.1073/pnas.0810928106

Meyer, L., & Friederici, A. D. (2016). Neural systems underlying the processing of complex sentences. *Neurobiology of Language*, 597–606. https://doi.org/10.1016/B978-0-12-407794-2.00048-1

Mladenović, M., Boljat, I., & Žanko, Ž. (2018). Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Education and Information Technologies, 23*, 1483–1500. https://doi.org/10.1007/s10639-017-9673-3

Murty, V. P., Sambataro, F., Radulescu, E., Altamura, M., Iudicello, J., Zoltick, B., Weinberger, D. R., Goldberg, T. E., & Mattay, V. S. (2011). Selective updating of working memory content modulates meso-cortico-striatal activity. *NeuroImage, 57*(3), 1264–1272. https://doi.org/10.1016/j.neuroimage.2011.05.006

Ntemngwa, C., & Oliver, J. S. (2018). The implementation of integrated science technology, engineering and mathematics (STEM) instruction using robotics in the middle school science classroom. *International Journal of Education in Mathematics, Science and Technology, 6*(1), 12–40. https://doi.org/10.18404/ijemst.380617

O'Reilly, R. C., & Frank, M. J. (2006). Making working memory work: A computational model of learning in the prefrontal cortex and basal ganglia. *Neural Computation, 18*(2), 283–328. https://doi.org/10.1162/089976606775093909

Opitz, B., & Friederici, A. D. (2007). Neural basis of processing sequential and hierarchical syntactic structures. *Human Brain Mapping, 28*(7), 585–592. https://doi.org/10.1002/hbm.20287

Paas, F., Tabbers, H., Gerven, P. V., & Tuovinen, J. (2003). Cognitive load measurement as a means to advance cognitive load theory. *Educational Psychologist, 38*(1), 63–71. https://doi.org/10.1207/S15326985EP3801_8

Paas, F., & Van Merriënboer, J. (1994). Instructional control of cognitive load in the training of complex cognitive tasks. *Educational Psychology Review, 6*, 351–371. https://doi.org/10.1007/BF02213420

Perruchet, P., & Rey, A. (2005). Does the mastery of center-embedded linguistic structures distinguish humans from nonhuman primates? *Psychonomic Bulletin & Review, 12*(2), 307–313. https://doi.org/10.3758/BF03196377

Poletiek, F. H., Monaghan, P., van de Velde, M., & Bocanegra, B. R. (2021). The semantics-syntax interface: Learning grammatical categories and hierarchical syntactic structure through semantics. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 47*(7), 1141. https://doi.org/10.1037/xlm0001044

Qian, Y., Hambrusch, S., Yadav, A., Gretter, S., & Li, Y. (2020). Teachers' perceptions of student misconceptions in introductory programming. *Journal of Educational Computing Research, 58*(2), 364–397. https://doi.org/10.1177/0735633119845413

Rac-Lubashevsky, R., & Kessler, Y. (2016). Decomposing the n-back task: An individual differences study using the reference-back paradigm. *Neuropsychologia, 90*, 190–199. https://doi.org/10.1016/j.neuropsychologia.2016.07.013

Schorr, T., Gerjets, P., & Scheiter, K. (2003). Analyzing effects of goal competition and task difficulty in multiple-task performance: Volitional action control within ACT-R. In *Proceedings of the Annual Meeting of the Cognitive Science Society* (Vol. 25, No. 25).

Shetreet, E., Friedmann, N., & Hadar, U. (2009). An fMRI study of syntactic layers: Sentential and lexical aspects of embedding. *NeuroImage, 48*(4), 707–716. https://doi.org/10.1016/j.neuroimage.2009.07.001

So, W. W. M. (2023). Does computation technology matter in science, technology, engineering and mathematics (STEM) projects? *Research in Science & Technological Education, 41*(1), 232–250. https://doi.org/10.1080/02635143.2021.1895099

Stukken, L., Van Rensbergen, B., Vanpaemel, W., & Storms, G. (2016). Understanding individual differences in representational abstraction: The role of working memory capacity. *Acta Psychologica, 170*, 94–102. https://doi.org/10.1016/j.actpsy.2016.06.002

Tucker-Raymond, E., Puttick, G., Cassidy, M., Harteveld, C., & Troiano, G. M. (2019). "I Broke Your Game!": Critique among middle schoolers designing computer games about climate change. *International Journal of STEM Education, 6*, 1–16. https://doi.org/10.1186/s40594-019-0194-z

Uddén, J., de Jesus Dias Martins, M., Zuidema, W., & Tecumseh Fitch, W. (2020). Hierarchical structure in sequence processing: How to measure it and determine its neural implementation. *Topics in Cognitive Science, 12*(3), 910–924. https://doi.org/10.1111/tops.12442

Wang, W., Kwatra, A., Skripchuk, J., Gomes, N., Milliken, A., Martens, C., & Price, T. (2021). Novices' learning barriers when using code examples in open-ended programming. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1* (pp. 394–400). https://doi.org/10.1145/3430665.3456370.

Wang, L., Jensen, O., Van den Brink, D., Weder, N., Schoffelen, J. M., Magyari, L., Hagoort, P., & Bastiaansen, M. (2012). Beta oscillations relate to the N400 during language comprehension. *Human Brain Mapping, 33*(12), 2898–2912. https://doi.org/10.1002/hbm.21410

Weintrop, D. (2019). Block-based programming in computer science education. *Communications of the ACM, 62*(8), 22–25. https://doi.org/10.1145/3341221

West, D. B. (2001). *Introduction to graph theory* (Vol. 2). Upper Saddle River: Prentice hall.

Wiggins, J. B., Fahid, F. M., Emerson, A., Hinckle, M., Smith, A., Boyer, K. E., Mott, B., Wiebe, E., & Lester, J. (2021). Exploring novice programmers' hint requests in an intelligent block-based coding environment. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (pp. 52–58). https://doi.org/10.1145/3408877.3432538

Winkler, M., Mueller, J. L., Friederici, A. D., & Männel, C. (2018). Infant cognition includes the potentially human-unique ability to encode embedding. *Science Advances, 4*(11), eaar8334. https://doi.org/10.1126/sciadv.aar8334

Xu, Z., Ritzhaupt, A. D., Tian, F., & Umapathy, K. (2019). Block-based versus text-based programming environments on novice student learning outcomes: A meta-analysis study. *Computer Science Education, 29*(2–3), 177–204. https://doi.org/10.1080/08993408.2019.1565233

Yamashita, K., Nagao, T., Kogure, S., Noguchi, Y., Konishi, T., & Itoh, Y. (2016). Code-reading support environment visualizing three fields and educational practice to understand nested loops. *Research and Practice in Technology Enhanced Learning, 11*(1), 1–22. https://doi.org/10.1186/s41039-016-0027-3

Ye, H., Liang, B., Ng, O. L., & Chai, C. S. (2023). Integration of computational thinking in K-12 mathematics education: A systematic review on CT-based mathematics instruction and student learning. *International Journal of STEM Education, 10*(1), 1–26. https://doi.org/10.1186/s40594-023-00396-w

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.