sustainability

MDPI

Article

Developing Eighth-Grade Students' Computational Thinking with Critical Reflection

Zhenzhen He [1], Xuemei Wu [1], Qiyun Wang [2] and Changqin Huang [3,*]

[1] School of Information Technology in Education, South China Normal University, Guangzhou 510631, China; zzhe@m.scnu.edu.cn (Z.H.); wuxuemei@m.scnu.edu.cn (X.W.)
[2] National Institute of Education, Nanyang Technological University, Singapore 639798, Singapore; qiyun.wang@nie.edu.sg
[3] Key Laboratory of Intelligent Education Technology and Application of Zhejiang Province, Zhejiang Normal University, Jinhua 321004, China
* Correspondence: cqhuang@zju.edu.cn; Tel.: +86-139-2505-9968

Abstract: As computer science has become a vital power in facilitating the rapid and sustainable development of various fields, equipping everyone with computational thinking (CT) has been recognized as one of the core pillars supporting the sustainable development of individuals and our digital world. However, it remains challenging for secondary school students to assimilate CT. Recently, critical reflection has been proposed as a useful metacognitive strategy for regulating students' thinking to solve current and future problems. In this study, a quasi-experiment was conducted to investigate the role of critical reflection in advancing eighth-grade students' CT. The participants were 95 eighth-grade students, comprising an experimental group (n = 49) and a control group (n = 46). The students' CT was evaluated based on their learning performance in computational concepts, computational practices, and computational perspectives. The results showed that critical reflection, compared with traditional instruction from teachers, could significantly advance eighth-grade students' CT. Interestingly, the two groups showed significantly different learning performance in computational practices during the learning process. Furthermore, interaction with peers and instructors played an essential role in helping students engage as active agents in critical reflection. The results of this study emphasize the need to develop students' CT by practicing critical reflection in eighth-grade education.

Keywords: computational thinking; critical reflection; secondary school students; interactions

Citation: He, Z.; Wu, X.; Wang, Q.; Huang, C. Developing Eighth-Grade Students' Computational Thinking with Critical Reflection. *Sustainability* **2021**, *13*, 11192. https://doi.org/10.3390/su132011192

Academic Editor: Eila Jeronen

Received: 3 September 2021
Accepted: 4 October 2021
Published: 11 October 2021

1. Introduction

As Goal 4 of the Sustainable Development Goals highlights, providing everyone with quality education is an indispensable pillar to support the sustainable development of our world [1]. In line with the Sustainable Development Goals, computational thinking (CT) has been globally integrated into K–12 education to build a solid foundation for the future success of individual students and the sustainable development of the world [2,3], as CT has been recognized as an essential skill for everyone to solve problems effectively in our computer science-driven world [4–7].

Meanwhile, considerable efforts have been made to develop secondary school students' CT. For example, the Computer Science Teachers Association and the International Society for Technology in Education (CSTA & ISTE) suggested the steps to develop K–12 students' CT, including identifying problems, organizing and analyzing data, representing the data, developing automated solutions, implementing and evaluating the optimal solution, and generalizing and transforming solutions [8]. Moreover, the three-dimensional framework of CT has been widely adopted to develop and evaluate students' CT from the dimensions of computational concepts, computational practices, and computational perspectives [9]. This integrated framework has attracted much attention on pedagogical research, because it

not only emphasizes the conceptual knowledge and practical skills of CT, but also attaches importance to the social attributes of CT [10,11].

However, it remains a difficult task for novices to develop their CT from the three dimensions of computational concepts, computational practices, and computational perspectives [3,12]. To address this issue, researchers and educators have made great efforts in two aspects: teaching tools and learning strategies. For instance, easy-to-learn programming environments and educational robots have been used to help students use CT to simulate the problem-solving process, which can improve students' interest in and understanding of computational concepts and computational practices [2,13,14]. Meanwhile, many learning strategies have been employed to develop CT with secondary school students. For example, collaboration with others has been confirmed as an efficient way to help students develop computational perspectives on themselves and their relationships with others [15]. Motivated by the three-dimensional framework of CT and the learning theory of constructionism [9,16], the design-based learning approach has been frequently used for developing secondary school students' CT [10]. By engaging students in multiple cycles of design, evaluation, and redesign to create a digital work, this learning approach enables students to understand computational concepts and computational practices [17,18].

Nevertheless, many secondary school students are still unable to generalize computational concepts and computational practices to solve everyday problems, which means that these students have a superficial and unsustainable understanding of CT that cannot support them in solving future problems [9,18]. Moreover, educational researchers stress that more attention needs to be paid to developing secondary school students' computational perspectives [8,19]. Therefore, it is vital to seek strategies to further secondary school students' CT.

Recently, critical reflection has been suggested as an effective metacognitive strategy that can help students evaluate and regulate their thinking to solve current and future problems [20,21]. Critical reflection has great potential to help students gain a deeper understanding and sustainable use of CT, because CT is exactly a particular set of problem-solving abilities, while critical reflection could improve students' problem-solving abilities by rethinking and refining one's own problem-solving process [5,22–24]. Moreover, generalizing solutions and forming computational perspectives are essential parts of CT, while critical reflection features prominently in promoting the generalization of solutions and the formation of new perspectives [8,25]. Besides, some studies have shown that critical reflection had a positive effect on promoting university students' CT [26,27]. However, whether critical reflection can be used to improve secondary school students' CT is unclear.

Thus, this study conducted a quasi-experiment to explore the role of critical reflection in promoting CT in secondary school students.

2. Literature Review

This study reviews four areas in the literature: (1) defining CT; (2) defining critical reflection; (3) integrating CT into secondary school education; and (4) learning CT through critical reflection.

2.1. Defining CT

CT has been widely regarded as an essential ability to solve problems by applying basic knowledge of computer science in technological societies [2,28]. Initially, CT was defined as using the fundamental concepts of computer science to solve problems, design systems, and understand human behaviors [5].

Meanwhile, some operational definitions of CT have been proposed. For example, the CSTA & ISTE put forward that CT is a particular set of problem-solving skills that includes identifying the problem, organizing and analyzing data, representing the data, developing automated solutions, implementing and evaluating the optimal solution, and generalizing and transforming solutions [8]. Adding to this, Brennan and Resnick proposed the three-dimensional framework of CT. This framework argues that students' CT

should be developed and evaluated in three dimensions: (1) computational concepts (the concepts that students often use in programming, such as sequences, loops, and events); (2) computational practices (the practices that students develop when they engage with computational concepts, such as iterating, debugging, and abstracting); and (3) computational perspectives (the perspectives students form about themselves and about the world, such as recognizing that computation is a medium of creation, realizing the power of working with others, and feeling empowered to ask questions) [9]. This integrated framework has been used frequently to cultivate K–12 students' CT, because it not only emphasizes the conceptual knowledge and practical skills of CT but also attaches importance to the social attribute of CT [10,11].

To sum up, we draw three conclusions. First, CT is an important ability to solve problems. Second, generalizing solutions is an integral part of learning CT. Third, students need not only to learn computational concepts and computational practices but also to develop computational perspectives.

2.2. Defining Critical Reflection

In the context of thinking, critical reflection is defined as a process of thinking about the conditions and the effects of what a person is doing or has done [29]. It reveals the influence and function of thought and action on people [30]. To achieve critical reflection, students need to complete the following steps. First, students need to examine noticeable details of the problem-solving process. Then, they are required to judge the reasons for their decisions from different perspectives. Following this, they re-cast prior experiences and knowledge into other contexts. Finally, they should develop new plans for solving similar problems in the future [31–33]. Therefore, researchers believe that a statement can be measured as critical reflection only when it shifts from a description of events to a critical report that analyzes, integrates and reconstructs experiences, and ultimately produces a new perspective [34,35].

In the domain of education, the theory of experiential learning recognizes critical reflection as an essential metacognitive strategy for acquiring meaningful learning outcomes from specific experiences [20,21]. Researchers believe that metacognition refers to thinking about one's own thinking, which is a crucial component in controlling and regulating one's thinking, especially problem-solving [20,24,36]. If problem-solvers can be aware of their cognition and can use this awareness to control and regulate their problem-solving process, they will have a better chance of success [37].

2.3. Integrating CT into Secondary School Education

In the past years, CT has been integrated into secondary school education to build a solid foundation for students' future success and to support the sustainable development of our computer science-driven world [2,3,6]. Consequently, various learning strategies have been used to develop students' CT. In general, there are two types of learning strategies for developing CT with secondary school students.

The first and most popular is student-centered learning. Motivated by constructionism [16], researchers have used the design-based learning strategy to develop CT in eighth-grade students [17]. Students self-explored the applications of CT by engaging in multiple cycles of design, evaluation, and redesign to make computer games about science topics iteratively in a programming environment. The results showed that design-based learning activities enable students to master computational concepts and computational practices. Along similar lines, a pedagogical strategy based on agile software engineering methods has been adopted to develop secondary school students' CT [14]. This strategy allows students to explore, iterate, and experience computational practices in different settings. The results showed that students' CT was enhanced by exploring these multidisciplinary activities. Moreover, collaborative learning strategies have been integrated into student-centered learning activities. A three-year research project claimed that co-

designing mobile games about social change could advance secondary school students' understanding of computational concepts and computational practices [18].

The second is teacher-directed learning. For instance, Saritepeci conducted a study that explored the effect of completing programming tasks on CT development with ninth-grade students [15]. The teacher introduced computational concepts and sample problem-solving activities to the students. Then, the teacher directed the students to collaborate on computational practices. The results showed that the students' CT had been significantly improved, while interacting with others had a positive effect on developing these students' computational perspectives about themselves and their relationships with others. Moreover, teacher-directed learning has also been used to develop interdisciplinary CT in ninth-grade students [38]. In this pedagogical process, the teacher demonstrated examples, guided the students to discuss, and provided the students with a series of pre-designed problems. As a result, the students' CT and programming skills were improved.

Overall, gaining computational experiences through self-practice or observation has been a main learning activity in cultivating CT with secondary school students. These cognitive experiences not only help students understand computational concepts and computational practices, but also shape students' computational perspectives [9].

Meanwhile, many studies have been conducted to explore the possible means of assessing CT with students. In general, there are four ways of assessing CT. The first popular way is works analysis. For example, a visualization tool named Scrape (http://happyanalyzing.com/ accessed on 9 October 2021) has been developed and used to automatically present the computational concepts used within Scratch projects [9]. Moreover, there are researchers who analyze works manually from the four aspects of content, creativity, artistry, and technology to evaluate students' learning performance in computational concepts and computational practices [11]. The second way is using traditional quizzes, such as multiple-choice items, to evaluate students' learning performance in computational concepts and computational practices [39]. The third way is conducting artifact-based interviews or self-reports to assess computational concepts, computational practices, and computational perspectives [9]. The fourth way is using scales. For instance, a five-point Likert scale was developed by Korkmaz et al. [40]. Computational thinkers use this scale to evaluate their creativity, algorithmic thinking, cooperation, critical thinking, and problem solving.

However, generalizing and transferring CT to other contexts remains a challenge for secondary school students [9,18]. In addition, more attention should be paid to the ways of developing K–12 students' computational perspectives [8,19].

2.4. Learning CT through Critical Reflection

In recent years, some researchers have discussed the relationship between CT and critical reflection. They argued that CT is exactly a particular set of problem-solving abilities, while critical reflection could improve students' problem-solving abilities [5,22–24]. Moreover, generalizing solutions and forming computational perspectives are essential parts of CT, while critical reflection features prominently in promoting the generalization of solutions and the formation of new perspectives [8,25].

In view of these arguments, critical reflection has been employed as a metacognitive strategy for developing CT-related knowledge in higher education. For instance, university students working in programming and multimedia systems development were required to blog about their critical reflections on their learning process. The students were prompted to examine what problems they have encountered, how they solved the problem, why the solution worked or not, and what they should do next time. The results showed that these activities could help the participants develop computational practices, such as finding problems and making revisions [27]. Moreover, Kwon and Jonassen's study showed that critical reflection had positive effects on helping university students understand computational concepts and complete computational practices in the domain of programming [23]. In their study, participants were asked to explain and critically judge their decisions. Similarly,

Miller et al. integrated critical reflection into university students' CT learning activities [26]. The students were provided with critical reflection prompts to think about their CT learning activities at a more abstract level. The results confirmed that critical reflection enabled the students to transfer CT to more general problem-solving contexts.

However, whether and how critical reflection can improve CT with secondary school students remains to be explored. At present, only noncritical reflection has been integrated in the CT learning activities of secondary school students. These noncritical reflections stated the noticeable details of the problem-solving process but did not evaluate the solutions. For instance, Zhong et al. stated that reflective card, which was designed for students to report their noticeable errors after they finished computational practices, was a useful tool for helping teachers discover students' learning barriers [11]. Similarly, reflective journals have been adopted to reveal secondary school students' perspectives on CT training activities and problems encountered [18]. In particular, the students were assigned to individually report the successes and difficulties they faced, as well as the activities they enjoyed and disliked. Although these noncritical reflections benefit teachers in detecting students' immediate and present learning difficulties, they have a trivial effect on activating and enhancing metacognition, which is essential for controlling and regulating learners' thinking [24,41].

In light of the above findings, this study integrated critical reflection in the learning process of secondary school students to help them gain an in-depth understanding of computational concepts and computational practices and develop computational perspectives. Specifically, the research questions of this study are as follows:

1. Do students who engage in critical reflection have better learning performance in CT than those who do not?
2. What are the participants' perceptions of engaging in critical reflection?

3. Method

We conducted a quasi-experiment in a secondary school. The participants' learning performance and perceptions of the learning activity were documented and analyzed.

3.1. Participants

The participants were 95 eighth-grade students aged 13 to 15 at a secondary school in China. Two classes were randomly selected from 14 eighth-grade classes and randomly assigned to an experimental group and a control group. Before the experiment, all of the participants had basic computer literacy, but no programming or CT learning experience, and no knowledge of computational concepts or computational practices. Excluding those participants who were absent from some lessons or tests because of sick leave and so on, there were 43 effective participants in the experimental group and 41 effective participants in the control group.

In this study, the two classes were taught by the same teacher. The teacher had four years of teaching experience at the secondary school level and one year of programming experience. In particular, the teacher had two years of experience teaching CT in this secondary school. Before the experiment, we gave the teacher a training session in teaching CT and critical reflection. The training session was divided into two stages. In the first stage, the teacher learned about the resources provided. These resources included: (1) What are CT and critical reflection? (2) Why are CT and critical reflection necessary? (3) How to teach students CT and critical reflection? In the second stage, the teacher, along with a professor who studied CT and a professor who studied critical reflection, spent a week testing the training effect. The teacher was required to provide a critical reflection-integrated CT lesson plan for secondary school students. Then, the professors gave comments and suggestions. Following this, the lesson plan was discussed until the two professors and the teacher reached agreement. During the discussion, the teacher was asked about her design rationale. For example, what are the key steps in critical reflection? How are these steps arranged in this lesson plan? Finally, the teacher conducted a trial lesson

based on the lesson plan and the professors provided feedback to enhance the teacher's learning outcomes.

3.2. Course Setting

In this study, the setting for all learning activities was an information and computer science course that lasted 13 weeks. This course was compulsory for all eighth-grade students and it was held weekly. Each lesson lasted 40 min during the school day. Each student was provided with a computer in the school's computer lab.

According to the syllabus for this course, students should master basic computational concepts, computational practices, and they should be able to apply CT to create a digital work and to solve problems. Table 1 shows the content that students needed to learn, which was designed based on the three-dimensional framework of CT [9]. More specifically, the programming tool selected was Small Basic (https://smallbasic-publicwebsite.azurewebsites.net, accessed on 9 October 2021). Thirteen examples of how to create a digital work using Small Basic were also provided to the students for reference (as listed in Table 2).

Table 1. The three-dimensional framework of CT used in this experiment.

Dimension	Subset	Description
Computational concepts (the concepts that students often use in programming)	Sequences	Identifying steps to solve a problem
	Loops	Executing a sequence multiple times
	Parallelism	Making things happen at the same time
	Events	One thing giving rise to another thing
	Conditionals	Making decisions according to different conditions
	Operators	Used in mathematical and logical expressions
	Variables	Used to hold one or more values
	Subroutine	Used in programs wherever the particular task should be performed
Computational practices (the practices that students develop when they engage with the computational concepts)	Experimenting and iterating	Developing a little, then giving it a try, then developing more
	Debugging and testing	Finding and trying to solve problems
	Reusing and remixing	Making something based on existing projects or ideas
	Modularizing and abstracting	Exploring connections between the whole and the parts
Computational perspectives (the perspectives students form about themselves and about the world)	Expressing	Recognizing that computation is a medium of creation
	Connecting	Realizing the power of working with others
	Questioning	Feeling empowered to ask questions

Table 2. Examples list.

Lesson	Sample	Computational Concepts Illustrated
Lesson 1	Paint a regular triangle	Operators, Sequences
Lesson 2	Paint a color regular pentagon	Variables, Sequences
Lesson 3	Paint regular polygons	Loops, Variables
Lesson 4	Cumulative sum	Loops, Conditionals, Variables
Lesson 5	Make a calculator	Conditionals, Loops
Lesson 6	Make a comparison	Conditionals, Loops
Lesson 7	Draw concentric circles	Conditionals, Loops, Parallelism
Lesson 8	Draw the Olympic rings	Variables, Operators, Sequences
Lesson 9	Paint colorful chains	Subroutine, Loops, Parallelism
Lesson 10	Record mouse track	Events, Conditionals, Subroutine
Lesson 11	The moving balloons	Events, Conditionals, Loops, Operators
Lesson 12	Balloon shooter	Events, Conditionals, Loops, Operators
Lesson 13	Tiny fishing master	Events, Conditionals, Loops, Operators

In this study, the students in the control group followed the teacher's original teaching plan, which was designed as suggested in literature: (1) computational experience was essential [3,10,12]; (2) collaborating with others was allowed [9,15]; and (3) iterative design was considered [17,18]. In each lesson, the teacher began by giving a representative example to illustrate new computational concepts, computational practices, and programming

skills. Then, the teacher provided the students with supporting materials and learning objectives. After gaining a preliminary understanding of the new knowledge, the students were given time to practice with the sample and explore different applications of the new knowledge. The students could ask each other for help. Meanwhile, the teacher walked around to provide learning assistance and keep track of the students' problems and learning performance. The practices and explorations were used to help students gain a better understanding of conceptual knowledge and acquire practical skills [42]. Moreover, creating digital works and interacting with peers could help the students develop computational perspectives, such as regarding computation as creating and recognizing the power of creating with others [9]. Near the end of each lesson, the teacher gave a summary of the students' learning performance to enhance their learning outcomes. In addition, all students were required to submit their works and concise reflection reports at the end of each lesson. The three prompts to help the students in the control group complete their concise reflection were: Have you accomplished at least one work? What was the hardest part for you? Which class activity has helped you the most?

To answer the research questions of this study, we redesigned the learning activities for the experimental group. Instead of giving a summary directly and assigning students to write the concise reflection reports, the teacher guided the students in the experimental group to engage in critical reflection at the end of each lesson.

Due to the class time limit, the teacher randomly selected about two students in each lesson. Each selected student was required to share learning experiences and reflections by presenting an oral report. As previous research suggested, prompts and critical feedback from others can stimulate critical reflection [43,44]. Therefore, the teacher was required to prompt (see Appendix A for examples of prompts) the selected students to report the following progressive reflection topics: What was the most difficult problem or important issue? How was the problem solved? What computational concepts or computational practices were used in this solution? Why did the problem occur and why did the solution work? Answering these questions could enhance the students' ability to find problems and abstract, which are important computational practices. Previous research has shown that secondary school students struggle to complete critical reflection independently, and interacting with the entire class is the most effective strategy for helping them [43,45]. Thus, all the students were then prompted by the teacher to collaborate on critical reflection by discussing the following. First, they openly shared perspectives on the problem and solution so that different perspectives could be considered by everyone, and so that underlying assumptions could be challenged (e.g., whether the computational knowledge was appropriate to solve the problems, and whether there were other possible solutions and why). Second, in order to generalize the computational knowledge, the students were then prompted to think of the computational knowledge at a more abstract level (e.g., what similar problems could they solve with this solution). By linking to other experiences or existing knowledge structures, students could gain a broader perspective and expand understanding of computational knowledge. With the aim of guiding the students to use the solutions flexibly in appropriate situations, the teacher then prompted them to summarize the advantages and disadvantages of these solutions, and to think about how they would handle similar challenges in the future.

3.3. Procedure

Before the experiment, we obtained the participants' consensus on data collection. Besides, the background questionnaire was used to collect the students' personal information and CT-related learning experience. Figure 1 shows the next steps of the experiment.

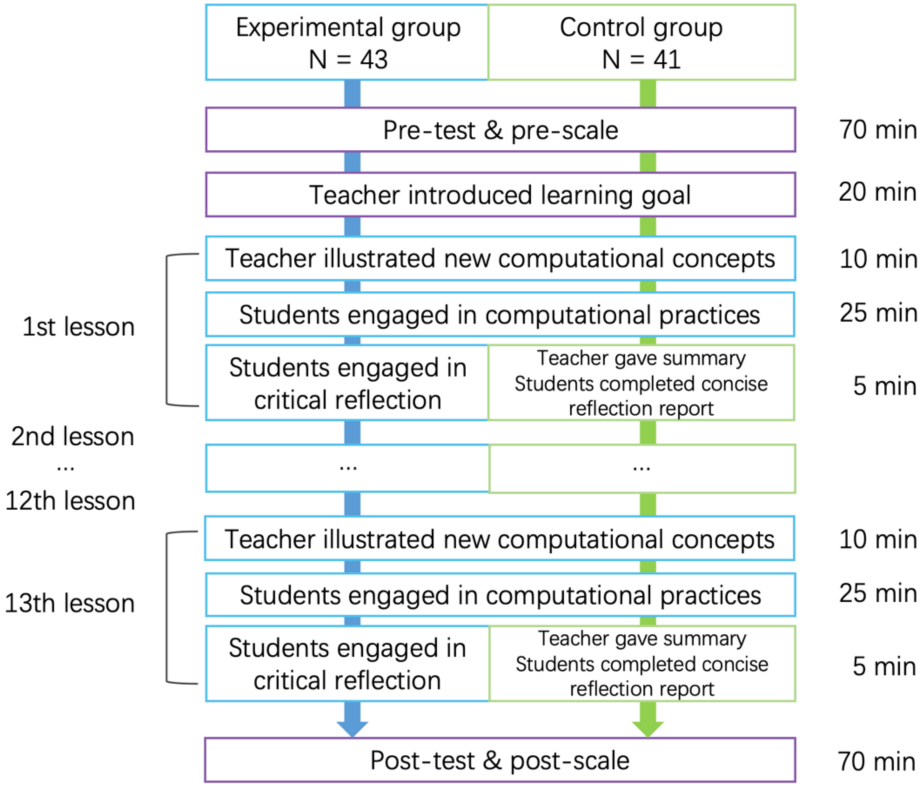


Figure 1. The experimental procedure.

First, every student was required to complete a test of computer operation and a scale for measuring computational perspectives individually. The reasons for this requirement are: (1) The background questionnaire results showed that there was no difference between the experimental group and the control group in terms of computational concepts and computational practices. Before the experiment, none of the students had knowledge about computational concepts or computational practices. (2) In the learning environment where programming was the primary learning approach, the ability to operate computers could influence the students' learning performance in computational practices. In this study, creating digital works by programming was the main learning activity and evaluation method, which was also an important and common way to learn and evaluate computational concepts and computational practices. Since the ability to operate computers could affect students' performance in programming, we could regard this ability as one factor that influences students' learning performance in computational practices. Thus, the test of computer operation was carried out. (3) Students might have computational perspectives, even if they have no knowledge about programming and CT. For example, they might have recognized the power of creating with others. Thus, the students were required to complete the scale for measuring computational perspectives. After the students had finished the pre-test and the pre-scale, the teacher introduced the learning objectives to all the students.

The experiment comprised 13 lessons. In each lesson, the teacher guided the students under the aforementioned course setting. Taking the third lesson as an example, the teacher used the task of painting regular polygons as an example to illustrate the computational concepts (i.e., loops and variables). The teacher taught the students about the usefulness and limitations of loop statements. She also re-coded the program of painting a regular triangle (i.e., the example from the first lesson) with a loop statement to enhance the students' understanding. Following this, the teacher provided the students with supporting materials and learning objectives. Then, all the students were given 25 min to participate in computational practices. During the whole time period, each student was required to create a digital work using their newly acquired computational concepts. Meanwhile, the students discussed with others and revised their works. To create a digital work, the students must

understand the computational concepts and complete a series of computational practices. For instance, they should abstract the main points from their ideas first. Then, they had to experiment and test. Moreover, they needed to reuse previous works when necessary. As suggested by the literature, these practices could help students gain a better understanding of computational concepts and enable them to become more familiar with computational practices [17,18]. Furthermore, creating different digital works could help students develop a computational perspective regarding computational tools as a vehicle for creation [9]. In addition, working with classmates helps the students develop computational perspectives, such as recognizing the power of creating with others [15].

At the end of each lesson, for the control group, the teacher gave a summary of the students' learning performance to enhance their learning outcomes. Following this, all the students completed a concise reflection report (as stated in Section 3.2) individually.

For the experimental group, the students engaged in critical reflection. Prompted by the teacher, the students reflected critically on their experiences and knowledge. Below, an example of critical reflection on the computational concept of parallelism from lesson 11 is described:

Teacher: Just as student A [all student names are anonymous] stated, his assumption is that parallelism is an efficient problem-solving strategy, and this is why he and student B decided to debug his program separately. What do you think about this? For example, what do you think is good about this, or what might not go so well?

Student C: I think it is a good idea. Doing different things at the same time saves time.

Student D: Parallelism is a really good computational idea. However, I prefer to work together, because when we work together, we can discuss about what are the possible problems. It is more targeted.

Teacher: Exactly. There are always many ways to solve a given problem. Does anyone have any other ideas?

Student E: It looks like they can discuss how to solve the bug first. I mean, they can split the work. For example, student A checks the first half and student B checks the second half. Then, they go through their share. Would that not be more effective?

Critical reflection on the solutions from different perspectives could help the students develop a comprehensive perspective on themselves and their initial works [46]. By re-casting prior experiences and knowledge into other contexts, critical reflection could enable students to come up with more problem-solving methods [31,32]. Thus, the students were then prompted to link parallelism to other everyday scenarios.

Student F: On my way to school, I memorize English words while walking.

Student G: I went shopping with my mother in the supermarket yesterday. There were too many people in the checkout line. Then, I waited in line while my mother picked out things so we could save time.

To complete critical reflection, students need to not only generalize about the solution, but also consider its conditions and limitations [29]. Consequently, the teacher instructed the students to think about and summarize the conditions and limitations of parallelism.

Student H: Yes. I agree with you. But my mother may not agree, because she does not think it is safe. If I am standing in line alone, she is afraid I will be kidnapped by bad guys. But if she goes out [shopping] with my dad, they would do what you do.

Teacher: Parallelism is a good computational idea. You also recommended many cases. Just as student E suggested, parallelism works better when combined with other strategies. On the other hand, student H also pointed out that parallelism is not available in some cases. Now, we come to the conclusion that not all situations are suitable for parallelism, and similarly, we can combine this idea with other strategies to solve problems more efficiently. Right?

All students: Yes.

Finally, the students were required to develop new plans for solving similar problems in the future [34].

The teacher: So, student A, what do you think of your solution now?

Student A: Frankly, I did not think about parallelism that much. Student B and I just think it works. My classmates' suggestions are very enlightening.

The teacher: What would you do when you run into similar problems in the future?

Student A: I think I will notice the constraints first. Just like when we write programs, we use the statement of "if . . . else . . . ". I also have to consider different situations when dealing with everyday problems.

Teacher: Exactly!

After the experiment, all the students individually completed the post-test and post-scale.

3.4. Instruments

To collect data regarding CT cultivation, the following instruments were used:

The background questionnaire: This questionnaire had three fill-in-the-blank items aiming to collect the participants' demographic characteristics, two single-choice items and one fill-in-the-blank item aiming to collect the participants' programming experience, and three single-choice items aiming to collect the participants' CT learning experience.

Test of basic knowledge of computer operation: This test was developed by a professor and four experienced teachers who had at least four years of experience teaching information technology courses in secondary schools. In order to ensure the content validity of this test, the items were developed referring to the method mentioned in the literature [40,47]. Firstly, the four experienced teachers developed their own items individually. Secondly, the items were discussed and revised in terms of clarity, accuracy, and content relevance. Following this, the items were added to the item pool. Then, the professor and one of the four teachers selected items from the pool separately. After this, the selected items were compared and reselected until agreement was reached. Finally, all the selected items were examined by the authors and the teacher participating in this study. Overall, this test covered two themes: (1) Common operations, such as creating a new folder and copying text. In this study, these operations were commonly used by the participants who learned CT primarily by making programming works. (2) Skills to solve common problems in using a computer. Specifically, this test had 25 multiple-choice items, 10 yes-or-no items, and 20 fill-in-the-blank items to evaluate the students' basic knowledge of computer operation, with a perfect score of 100. The Cronbach's α value of the test was 0.82, implying the high reliability of the test. Sample items for this test are presented in Appendix B.

Test of computational concepts and computational practices: Inspired by the CT evaluation methods used by Grover [39] and the characteristics of Small Basic, we developed this test to evaluate the students' knowledge of computational concepts and computational practices. Moreover, the development of this test followed the method commonly used in prior studies to ensure the content validity [40,47]. The same procedure for ensuring content validity was used as previously mentioned. Specifically, the test comprised 25 multiple-choice items, 10 yes-or-no items, and 20 fill-in-the-blank items, with a perfect score of 100. These items examined students' understanding of computational concepts and computational practices both in Small Basic and their daily lives. The Cronbach's α value of the test was 0.83, implying the high reliability of the test. Appendix C lists some sample items for this test.

Scale of computational perspectives: This scale was developed based on the computational thinking scale proposed by Korkmaz et al. [40]. The scale comprised 29 items in five dimensions, including eight items for creativity, six items for algorithmic thinking, four items for cooperation, five items for critical thinking, and six items for problem solving. The items were scored on a five-point Likert scale. To validate the scale in this study, the items were translated from English into Chinese by three English language teachers who are bilingual and have at least four years of teaching experience in secondary schools. As suggested by Alharbi [47], the scale was firstly translated into Chinese by two of the three teachers. Then, the items in Chinese were translated back to English by the third one. Their translations were discussed and revised amongst themselves and two professors.

Finally, the items were pretested for appropriateness among 25 secondary school students (not involved in the experimental group or the control group in this study). Based on the collected feedback, some statements were slightly modified to resolve any linguistic ambiguities. In this study, the Cronbach's α value of the scale was 0.92, and the Cronbach's α values of the five dimensions were 0.73, 0.80, 0.87, 0.76, and 0.83, respectively. The Cronbach's α values imply the high reliability of this scale.

Works: The performability of the work and the types of computational concepts used are important indicators of students' CT [9,11]. Thus, each work in this study was assessed with a grade according to the degree of the work's performability, accuracy, and creativity. Table 3 shows the grades and corresponding descriptions and examples. Guided by this rubric, all the works were explored and analyzed by the same teacher.

Table 3. Rubric for grading the students' works.

Grade	Description	Example
Performability	An executable program	A simple program without syntax errors
Accuracy	A program that covers specified concepts	An unfinished program that was coded to draw a square with a loop statement
Creativity	A meaningful program that differs from the sample (provided and explained by the teacher)	An unfinished program that was coded to output the same sentence ten times
PA	An executable program that is identical to the sample	An executable program that was coded to draw a square with a loop statement
PC	An executable program that is different from the sample	An executable program for drawing blue pentagram, but it did not use a loop statement
AC	A program that integrates specified concepts with other knowledge	An unfinished program that was coded to draw a red regular octagon with a loop statement
PAC	An executable program that integrates specified concepts with other knowledge	An executable program that uses the loop statement to draw a blue square with gradually thickening lines
None	A program with a few meaningless statements	Only one statement, that is "Turtle.Move (100)"

Examples were taken from the students' works in lesson three (the computational concept illustrated were loops and variables), except for the "Creative", "PC" and "None" because no work received this grade. "PA" stands for integrating Performability and Accuracy. "PC" stands for integrating Performability and Creativity. "AC" stands for integrating Accuracy and Creativity. "PAC" stands for integrating Performability, Accuracy, and Creativity.

In addition, open-ended interviews with the teacher were conducted every two weeks. The interview questions mainly focused on her perceptions with regard to involving the students in critical reflection and suggestions for improving CT training activities. Each of the students was interviewed after the experiment to obtain the students' opinions about the overall learning activities, such as their perceptions of engaging in critical reflection.

4. Results

4.1. Students' Learning Performance in CT

Before the experiment, the two groups had no significant difference in CT proficiency. First, the background questionnaire results showed that there was no difference between the two groups in terms of computational concepts and computational practices before the experiment. Second, the average scores of the experimental group and the control group on the pre-test of computer operation were 72.14 and 73.54, respectively. The results of an independent sample t-test ($t = -0.59$, $p > 0.05$) indicated that the two groups had an equivalent knowledge of computer operation before the experiment. It implied that the two groups have an equivalent operational ability to accomplish the CT-focused programming tasks designed in this experiment. Third, the average scores of the experimental group and the control group on the pre-scale of computational perspectives were 96.47 and 97.54, respectively. The results of an independent sample *t*-test ($t = -0.32$, $p > 0.05$) showed that there were no significant differences between the two groups in relation to computational perspectives before this experiment.

After the experiment, the learning performance in CT of the experimental group was significantly better than that of the control group. The detailed results are as follows.

In order to statistically control for differences that might be attributable to the learning performance of the students in computational concepts and computational practices, we adopted an analysis of covariance (ANCOVA) to analyze the post-test scores of the two groups by including the students' basic knowledge of computer operation as a covariate. First, the homogeneity test ($F = 0.00$, $p > 0.05$) was passed. Then, ANCOVA was carried out. As shown in Table 4, the adjusted means and standard error of the experimental group were 76.95 and 1.35, respectively. The results of ANCOVA also showed that the experimental group had a significantly better learning achievement than the control group ($F(1,81) = 4.04$, $p < 0.05$, partial $\eta^2 = 0.05$). There was a desired effect in the real educational context between the experimental group and the control group, as indicated by a partial η^2 value of 0.05 [48]. Consequently, it could be concluded that engaging in critical reflection was more effective at helping students master computational concepts and computational practices than simply receiving a summary from the teacher.

Table 4. ANCOVA of the post-test for the two groups' scores in computational concepts and practices.

Group	N	Mean	SD	Adjusted Mean	SE	F	Partial η^2
Experimental group	43	76.56	10.49	76.95	1.35	4.04 *	0.05
Control group	41	73.46	11.12	73.05	1.39		

* Significant at $p \leq 0.05$.

Moreover, we graded the works according to their performability, accuracy, and creativity. Each work was assigned to one of the eight grades (as shown in Table 3 presented in Section 3.4). Then, percentages of each grade were also calculated. For instance, the experimental group had a total of 43 works in the tenth week, among which five works were graded as PAC. Thus, the percentage of PAC works in the tenth week for the experimental group was 12%. Figure 2 shows the learning performance of the two groups in different weeks. Overall, most works were graded as PA, implying that most of the students could complete the sample practices. Interestingly, we found that the two groups performed significantly differently during the learning process. Firstly, the experimental group had more works graded as AC and PAC than the control group. That is, the experimental group was more adept at synthesizing different computational concepts, which indicated that they had a better generalization of these computational concepts. Secondly, the control group had more non-performable works than the experimental group, especially in the middle stage (i.e., the fifth week to the ninth week).

The data on the students' computational perspectives were also analyzed. The assumption of homogeneity was tenable ($F = 1.24$, $p > 0.05$). Then, one-way ANCOVA (shown in Table 5) was conducted to preclude the effects of students' prior computational perspectives. The results indicated that students in the experimental group had a significantly better learning achievement than students in the control group ($F(1,81) = 4.61$, $p < 0.05$, partial $\eta^2 = 0.05$). A desired effect in a real educational context was also found between the two groups, as indicated by a partial η^2 value of 0.05 [48]. This result implied that critical reflection significantly stimulated the formation of these eighth-grade students' computational perspectives.

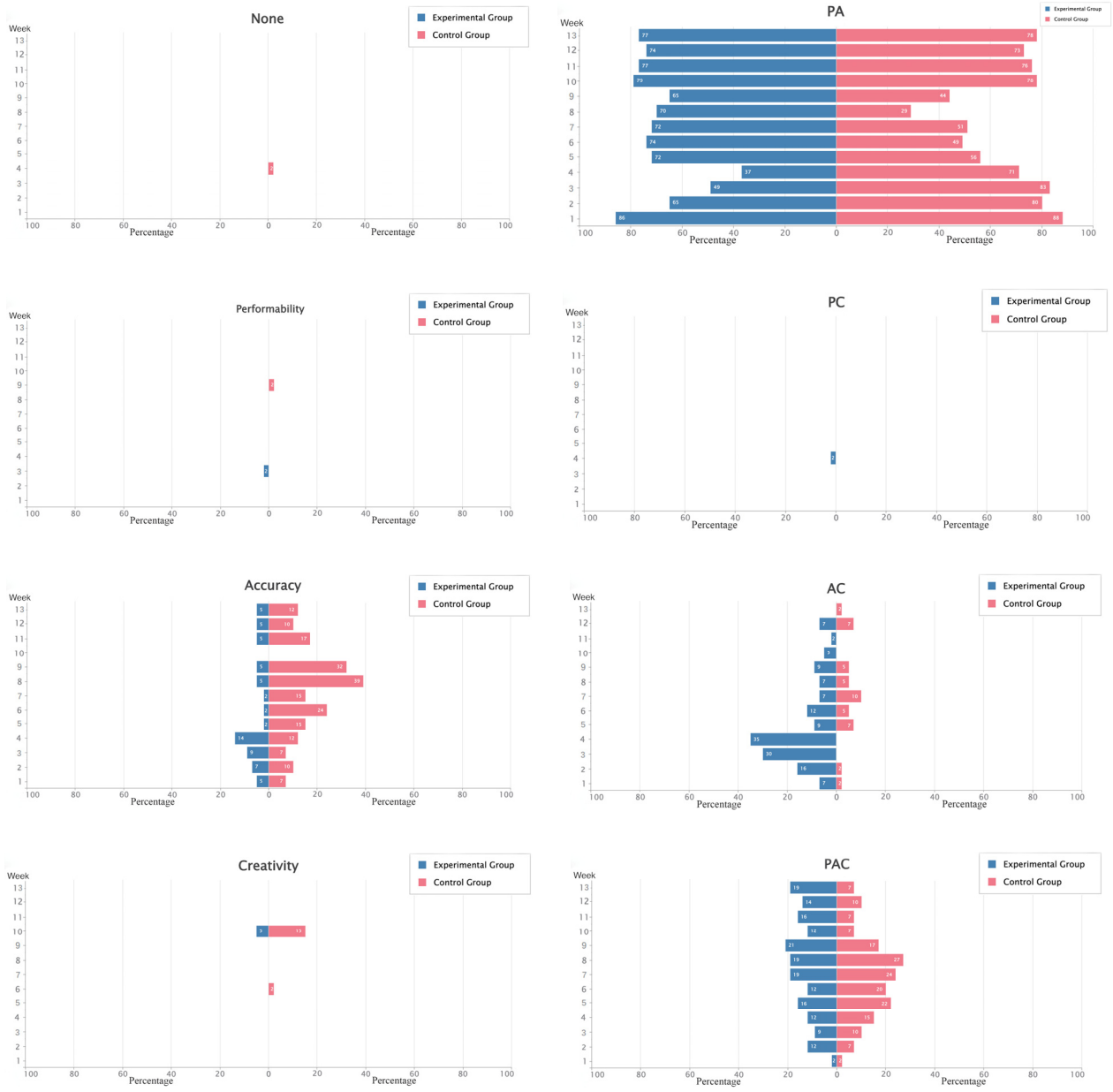


Figure 2. Performance of completing computational practices using the specified computational concepts.

Table 5. ANCOVA of the post-scale for the two groups' scores in computational perspectives.

Group	N	Mean	SD	Adjusted Mean	SE	F	Partial η^2
Experimental group	43	108.05	9.80	108.12	1.74	4.61 *	0.05
Control group	41	102.85	13.11	102.78	1.78		

* Significant at $p \leq 0.05$.

The interview results also confirmed the effectiveness of critical reflection in developing students' computational perspectives. In particular, 37 (86%) of the students in the experimental group stated that engaging in critical reflection with their classmates enabled them to recognize the power of cooperation. Moreover, 35 (81%) of the students in the experimental group expressed that critical reflection helped them to find shortcomings, which in turn gave them the confidence and ability to question and redesign computational works. By contrast, 30 (73%) of the students in the control group acknowledged that there was no discernible change in their perspectives on cooperation and computational works.

4.2. Participants' Perceptions of Critical Reflection

The teacher who was involved in the experiment was interviewed seven times during the study. All of the teacher's comments were recorded chronologically in a text document for content analysis. The teacher yielded 76 comments on critical reflection, while the experimental group made 43 comments. The comments were analyzed according to the three-layer coding procedure [49]. First, the coders read all of the comments several times and created open codes. Then, they refined and clustered all the codes into primary themes. Finally, the themes were further checked and compared to obtain core themes.

While the teacher liked using critical reflection to enhance eighth-grade students' CT, she also pointed out the problems that needed attention. She indicated that critical reflection enabled her to discern students' learning difficulties and improve their learning initiative. For instance, she stated, "critical reflection is helpful for me to gain insight into the students' problems and their problem-solving strategies" and "The thing I appreciate most with critical reflection is keeping the students actively involved". On the other hand, if the students in the experimental group were not prompted by the teacher, they did not actively engage in critical reflection, especially in the first few weeks of this experiment. The teacher said that "the students (in the experimental group) were highly dependent on reflection prompts, possibly because they had not memorized the steps of critical reflection".

Critical reflection was also generally accepted by the students in the experimental group. 38 (88%) of the students in the experimental group noted that critical reflection helped them develop CT. Firstly, they stated critical reflection helped them generalize computational concepts. A student stated that "our teacher organized us to engage in critical reflection and discussion, which helped us connect textbook knowledge with practice and apply knowledge to solve similar problems in different contexts". Secondly, they indicated critical reflection had a positive effect on their computational practices. When talking about this, one student said that "peers' evaluations (presented in the critical reflection process) enabled me to realize my shortcomings, and this made me want to further modify my programs and even change my way of thinking". Thirdly, the interview results showed that critical reflection helped the students form computational perspectives. Some of the students said the following: (1) "although I was never the one who was selected to give an oral reflection report, I gained various problem-solving strategies from my classmates," which implies that the student realized the power of learning with others; (2) "I prefer to ask questions during our critical reflection," which implies that the student felt empowered to ask questions; and (3) "when we reflected, I got a lot of ideas, which made me think that we can create a lot of interesting works with Small Basic", which implies that the student recognized computation as a medium of creation.

5. Discussion

5.1. Effect of Critical Reflection on Advancing Students' CT

In order to identify the effect of critical reflection on enhancing eighth-grade students' CT, a quasi-experiment was conducted. The learning activity of the control group ended with the teacher's summary, whereas the experimental group engaged in critical reflection to further explore their computational experiences. The results confirmed that critical reflection is more effective at advancing students' CT. This is consistent with the finding that critical reflection is more conducive to stimulating deeper learning than direct feedback from a teacher [50].

With respect to computational concepts and computational practices, the experimental group made significantly better learning achievements. These results were observed in both the post-test scores and the learning process. As the CT-oriented practices became increasingly complicated, the control group had more non-performable works than the experimental group, especially in the middle stage of the semester. On the other hand, the experimental group had more works graded as AC and PAC than the control group. In particular, the experimental group had nearly three times as many PAC works as the control group in the last stage. These creative works conveyed that the experimental

group could better generalize these computational concepts and computational practices to sustainably use them to create new works and solve problems, while the control group's understanding of these computational concepts and computational practices was relatively superficial and narrow. The different formative learning performance of the two groups in this study was consistent with the conclusion of other studies: critical reflection can support the development of computational practices in the domain of programming and transfer CT to more general problem-solving contexts [23,26,27]. There are three possible reasons for these results. Firstly, critical reflection can heighten students' internal attention that facilitates the efficient integration of knowledge and idea generation [51]. Secondly, critical reflection can benefit students' follow-up actions to solve similar problems [46]. In this study, the students in the experimental group explained that critical reflection helped them discover the powers and limitations of computational ideas and their way of thinking. Moreover, various suggestions raised during critical reflection were beneficial to testing and debugging in subsequent computational practices. The interview results also showed that many students in the control group were not efficient at debugging. When they encountered problems, the students in the control group felt frustrated and did not know what to do, while the students in the experimental group were more confident and more likely to seek help from others. As a consequence, the students in the experimental group could effectively complete their works within the time limit. Thirdly, critical reflection-integrated programming education encourages students to review what they have learned, which contributes to converting new knowledge into long-term memory [52].

In terms of computational perspectives, the experimental group also had significantly higher test scores than the control group. This indicated that critical reflection significantly facilitated the development of these eighth-grade students' computational perspectives. Prior research has shown that CT-oriented programming practices enable students to master computational concepts and practical skills, but that programming is insufficient to cause a shift in perspectives [18]. Nevertheless, critical reflection provides students with opportunities for re-examining specific computational experiences from different perspectives, as well as for building a bridge between specific computational experiences and overall experiences. Thus, many studies have used critical reflection to not only help the students develop a comprehensive perspective on themselves and their initial works, but also enable them to come up with alternative problem-solving methods [32,46]. Similarly, in this study, the students in the experimental group were prompted to evaluate and question specific problems and solutions from different perspectives, as well as to link CT to other scenarios. These peer-evaluations and generalizations stimulated the students' thinking and ideas, and ultimately benefited them by giving them a sense that they can use computational tools for creation [53]. When the experimental group engaged in critical reflection together, they received various suggestions from others. The interviews confirmed that critical reflection equipped the students with ideas for optimizing their original works and creating new applications for CT. This enhanced their appreciation of learning with others. On the other hand, peers' judgements challenged the students' original assumptions and habitual ways of solving problems. Every student had to put up with and try to accept different points of view. This helped the students correct the flaws in their habitual assumptions and assimilate new ideas, which enabled them to ascertain that they had achieved personal development and gave them a sense of accomplishment. Moreover, sharing and arguing with peers could reinforce the students' roles as learning facilitators and improve their academic self-efficacy [52]. As a result, the experimental group were more confident to ask questions.

Compared with the experimental group, the control group was more of a receiver of information after completing the computational practices. The ideas and perspectives they had access to were mainly from the teacher. There was a lack of collision of different ideas that triggers their metacognition of the CT learning process and alters their perspectives.

5.2. Perceptions of Integrating Critical Reflection into CT Education

In this study, both the teacher and students recognized the role of critical reflection in developing the eighth-grade students' CT. The interview results showed that critical reflection was helpful for the eighth-grade students to generalize computational concepts, engage in computational practices, and form computational perspectives. While CT-oriented programming practices enabled students to apply computational concepts and acquire practical computational skills, critical reflection further helped the students develop various appropriate problem-solving strategies associated with computational concepts and computational practices [15,23,26,27]. When the students in the experimental group discussed with their peers to complete critical reflection on their works, they obtained different ideas of applying CT to solve problems from peers and then generated new problem-solving strategies. This helped them form computational perspectives (i.e., realizing the power of learning with others, feeling empowered to ask questions, and recognizing computation as a medium of creation). Consistent with previous findings, the interview results confirmed that reflection reports could benefit the teacher by helping her to identify the learning difficulties faced by the students [11,18]. When the teacher carried out CT teaching activities following the conventional teaching approach, she obtained students' learning difficulties through her own observations and students' reports. Limited by teachers' energy and students' initiative, it was difficult for teachers to find students' learning difficulties. Besides, critical reflection used in this study has the effect of improving the eighth-grade students' learning initiative, which is superior to noncritical reflection used in other studies on developing secondary school students' CT. As explained by the students in the experimental group, when they participated in critical reflection, they were prompted by their teachers and given many helpful suggestions by their classmates, which made them more willing to express their ideas and more confident to improve their works.

Furthermore, this study found that interaction with others was crucial for secondary school students to engage in critical reflection. On the one hand, this study confirmed that interaction with peers can help young learners reflect critically and come up with more alternative plans [54,55]. The different opinions of their peers led the students to re-examine their own learning process and perspectives. On the other hand, it was notable that students' participation in critical reflection was particularly dependent on the teacher's prompting. If the teacher did not actively give prompts, the students had difficulty engaging in critical reflection and even lost focus, which was especially true in the first few weeks of the experiment. This was possibly because the eighth-grade students were still not well informed about the steps of critical reflection. These findings imply that completing critical reflection independently is still a challenge for secondary school students.

The findings of this study have two practical implications. First, this study verified the effectiveness of critical reflection at improving eighth-grade students' CT. Hence, we suggest that instructors should integrate critical reflection into the CT learning process for secondary school students. Second, this study revealed that interaction played a positive role in facilitating secondary school students' critical reflection. In particular, we noticed that these young students were in need of guidance from their teacher when conducting critical reflection. Thus, we suggest that instructors should design multiple interaction strategies and provide appropriate support to broaden and deepen the secondary school students' critical reflection on the application of CT.

5.3. Limitations and Future Research

In this study, there were more than 40 students in each class, but only one teacher. It was therefore difficult for the teacher to capture all of the students' real-time learning performance in a naturalistic classroom setting. Future research should apply behavior-monitoring tools and thinking visualization tools to capture and analyze students' CT performance. Moreover, future research should adopt more strategies to stimulate students to participate actively. For example, paired programming and peer assessment can be inte-

grated into learning activities [10,11], insofar as students noted that their peers' suggestions led them to come up with various problem-solving strategies.

6. Conclusions

In present and future society, equipping everyone with CT is recognized as an important component of sustainable educational development goals. In order to develop students' computational thinking more effectively, this study proposed the use of critical reflection to promote secondary school students' understanding and application of CT. The quasi-experiment conducted in this study revealed that critical reflection effectively improved the learning achievement of eighth-grade students with regard to computational concepts, computational practices, and computational perspectives. Moreover, the two groups showed significantly different learning performance during the learning process. The interview results confirmed that critical reflection helped eighth-grade students learn from their experiences. In addition, we found that interaction with others plays an essential role in stimulating critical reflection in students.

Author Contributions: Conceptualization, Z.H. and C.H.; Data curation, Z.H., X.W. and Q.W.; Formal analysis, Z.H., X.W. and Q.W.; Funding acquisition, C.H.; Investigation, Z.H. and X.W.; Methodology, Z.H. and X.W.; Validation, X.W.; Writing–original draft, Z.H. and C.H.; Writing–review and editing, X.W., Q.W. and C.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Humanities and Social Sciences Planning Fund of the Ministry of Education, grant number 18YJA880027.

Institutional Review Board Statement: The study was conducted according to the guidelines of the Declaration of Helsinki. Ethical review and approval were waived because permission to conduct this government-funded project in China was granted.

Data Availability Statement: The data presented in this study are available on reasonable request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Examples of critical reflection prompts used in the experimental group are listed as follows:

1. What was the hardest/the most impressive part of this lesson for you?
2. Have you solved your problem? If yes, how? If not, why not?
3. Can you explain why you solved the problem in this way?
4. Can you tell us why you made such a decision?
5. Why did you solve the problem/address the issue this way?
6. How do you feel about it?
7. Which computational ideas did you draw upon to make your decision?
8. Could any of you say something critical/positive about the problem-solving process reported by XXX (i.e., the student selected for the oral report)?
9. How would you evaluate/judge the validity of XXX's solution?
10. Are there any other problems in your life that can be solved in the same way?
11. What might happen if we integrated other ideas into this method/if we used this method to solve other problems?
12. Do any of you have other methods to deal with this situation?
13. What would you do differently if you encountered this problem?
14. Can you suggest an alternative solution?
15. What are other ways to deal with this kind of situation?
16. How can this solution be combined with other ideas to create a better/more complete solution?

17. Did the knowledge/suggestions/judgements of others cause you to reconsider your initial reaction or assumptions about the problem/issue?
18. What conclusions can be drawn from these solutions/issues?
19. What can be extended from these particular solutions/issues?
20. Would your classmates' comments affect your solution to the problem, either now or in the future?
21. What would you do if you encountered similar problems in the future?
22. Have your classmates' suggestions changed you? If yes, how?
23. What perspectives or beliefs have changed for you? Provide examples.
24. Could you describe what you have learned from your classmates' ideas?

Appendix B

Table A1. Sample items for the test of basic knowledge of computer operation.

Subset	Sample Items	Relevance to CT
Multiple-choice items	M1. In The Windows operating system, a software can be opened in the following ways: ( ) A. Left-click the software icon and select Open B. Right-click the software icon and select Open C. Double-click the software icon with the left mouse button D. Double-click the software icon with the right mouse button M2. In The Windows operating system, which of the following statements about filenames are correct: ( ) A. Chinese characters are allowed for file names B. Multiple dot separators are allowed for file names C. Space are allowed for file names D. Any character is allowed for file names	The students' abilities to use computers play an important role in completing computational practices in this study. For instance, students with poor computer skills were more likely to be unable to complete computational practices using Small Basic in the given amount of time.
Yes-or-no items	Y1. You can have two identical files in the same folder. ( ) Y2. The software will be shut down when its window is minimized. ( )	
Fill-in-the-blank items	F1. If you find an extra typo in front of the cursor while typing, you can press the ( ) key to delete it. F2. To enter the "*" above the numeric key "8", you must first hold down the ( ) key and then hold down the numeric key.	

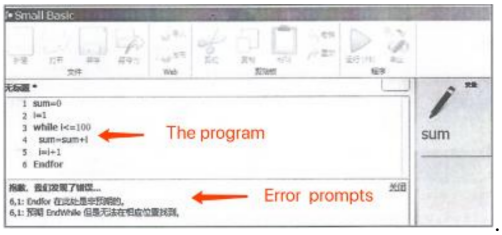
Since all the computers used in this experiment were installed Windows operating system, the test items were also subject to the operation in the Windows operating system. "( )" stands for "Please write your answers in brackets so that teachers can find your answers quickly and accurately".

Appendix C

Table A2. Sample items for the Test of Computational Concepts and Computational Practices.

Subset	Sample Items	Relevance to CT
Multiple-choice items	M1. What are the three basic structures of programming? ( ) A. Sequences B. Conditionals C. Loops D. Events	This item examines students' understanding of computational concepts in programming.
	M2. Which of the following situations requires a conditional statement? ( ) A. If it doesn't rain tomorrow, we will go to the amusement park. B. Only after you finish your homework can you play games. C. We will be late if we do not leave now. D. We should study hard.	This item examines students' understanding of computational concept (Conditionals) in everyday problem solving.

Table A2. *Cont.*

Subset	Sample Items	Relevance to CT
Yes-or-no items	Y1. When writing a program with many repeated statements, you can use conditional statements to make the program concise. ( )	This item examines students' understanding of computational concepts (Conditionals and Loops) in programming.
	Y2. We need to take different conditions into consideration when making plans, which is a parallel approach. ( )	This item examines students' understanding of computational concepts (Conditionals and Parallelism) in everyday problem solving.
Fill-in-the-blank items	Lily wrote a program to calculate "1 + 2 + 3 + . . . + 100" using Small Basic. As shown in the following figure, it did not work and showed error prompts:  . F1. From the figure above, we can see that the error was in line ( ). F2. If we want to help Lily correct the mistake in the program, we should change line ( ) of the program to ( ).	These items examine students' understanding of computational practices (Debugging and testing) in programming.

"( )" stands for "Please write your answers in brackets so that teachers can find your answers quickly and accurately".

References

1. UNESCO. Education for Sustainable Development Goals. Available online: https://en.unesco.org/gem-report/sdg-goal-4 (accessed on 20 August 2021).
2. Ardito, G.; Czerkawski, B.; Scollins, L. Learning computational thinking together: Effects of gender differences in collaborative middle school robotics program. *TechTrends* **2020**, *64*, 373–387. [CrossRef]
3. Hsu, T.; Chang, S.; Hung, Y. How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Comput. Educ.* **2018**, *126*, 296–310. [CrossRef]
4. Yuan, Y.; Liu, C.; Kuang, S. An innovative and interactive teaching model for cultivating talent's digital literacy in decision making, sustainability, and computational thinking. *Sustainability* **2021**, *13*, 5117. [CrossRef]
5. Wing, J.M. Computational thinking. *Commun. ACM* **2006**, *49*, 33–35. [CrossRef]
6. Jong, M.S.; Geng, J.; Chai, C.S.; Lin, P. Development and predictive validity of the computational thinking disposition questionnaire. *Sustainability* **2020**, *12*, 4459. [CrossRef]
7. Wu, S.Y.; Su, Y.S. Visual programming environments and computational thinking performance of fifth- and sixth-grade students. *J. Educ. Comput. Res.* **2021**, *59*, 1075–1092. [CrossRef]
8. CSTA & ISTE. Operational Definition of Computational Thinking for K-12 Education. Available online: https://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/CompThinkingFlyer.pdf (accessed on 15 August 2020).
9. Brennan, K.; Resnick, M. New frameworks for studying and assessing the development of computational thinking. In Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, BC, Canada, 13–17 April 2012; p. 25.
10. Lye, S.Y.; Koh, J.H.L. Review on teaching and learning of computational thinking through programming: What is next for K−12? *Comput. Hum. Behav.* **2014**, *41*, 51–61. [CrossRef]
11. Zhong, B.; Wang, Q.; Chen, J.; Li, Y. An exploration of three-dimensional integrated assessment for computational thinking. *J. Educ. Comput. Res.* **2016**, *53*, 562–590. [CrossRef]
12. Kalelioğlu, F. A new way of teaching programming skills to K-12 students: Code.org. *Comput. Hum. Behav.* **2015**, *52*, 200–210. [CrossRef]
13. Chang, C.K. Effects of using Alice and Scratch in an introductory programming course for corrective instruction. *J. Educ. Comput. Res.* **2014**, *51*, 185–204. [CrossRef]
14. Fronza, I.; Ioini, N.E.; Corral, L. Teaching computational thinking using agile software engineering methods: A framework for middle schools. *ACM Trans. Comput. Educ.* **2017**, *17*, 1–28. [CrossRef]
15. Saritepeci, M. Developing computational thinking skills of high school students: Design-based learning activities and programming tasks. *Asia-Pac. Educ. Res.* **2020**, *29*, 35–54. [CrossRef]
16. Papert, S.A. *Mindstorms. Children, Computers and Powerful Ideas*; Basic Books: New York, NY, USA, 1980.

17. Tucker-Raymond, E.; Puttick, G.; Cassidy, M.; Harteveld, C.; Troiano, G.M. "I Broke Your Game!": Critique among middle schoolers designing computer games about climate change. *Int. J. STEM Educ.* **2019**, 6. [CrossRef]
18. Thomas, J.O.; Rankin, Y.; Minor, R.; Sun, L. Exploring the difficulties African-American middle school girls face enacting computational algorithmic thinking over three years while designing games for social change. *Comput. Support. Coop. Work* **2017**, *26*, 389–421. [CrossRef]
19. Yaşar, O. A new perspective on computational thinking. *Commun. ACM* **2018**, *61*, 33–39. [CrossRef]
20. Colbert, C.Y.; Graham, L.; West, C.; White, B.A.; Arroliga, A.C.; Myers, J.D.; Ogden, P.E.; Archer, J.; Mohammad, Z.T.A.; Clark, J. Teaching metacognitive skills: Helping your physician trainees in the quest to 'know what they don't know'. *Am. J. Med.* **2015**, *128*, 318–324. [CrossRef]
21. Medina, M.S.; Castleberry, A.N.; Persky, A.M. Strategies for improving learner metacognition in health professional education. *Am. J. Pharm. Educ.* **2017**, *81*, 78. [CrossRef]
22. Boud, D.; Keogh, R.; Walker, D. *Reflection: Turning Experience into Learning*, 1st ed.; Routledge: Oxfordshire, UK, 1985.
23. Kwon, K.; Jonassen, D.H. The influence of reflective self-explanations on problem-solving performance. *J. Educ. Comput. Res.* **2011**, *44*, 247–263. [CrossRef]
24. Lee, C.B.; Koh, N.K.; Cai, X.L.; Quek, C.L. Children's use of meta-cognition in solving everyday problems: Children's monetary decision-making. *Aust. J. Educ.* **2012**, *56*, 22–39. [CrossRef]
25. Howie, P.; Bagnall, R. A beautiful metaphor: Transformative learning theory. *Int. J. Lifelong Educ.* **2013**, *32*, 816–836. [CrossRef]
26. Miller, L.D.; Soh, L.; Chiriacescu, V.; Ingraham, E.; Shell, D.F.; Hazley, M.P. Integrating computational and creative thinking to improve learning and performance in CS1. In Proceedings of the 45th ACM Technical Symposium on Computer Science (SIGCSE'2014), New York, NY, USA, 5–8 March 2014. [CrossRef]
27. Robertson, J. The educational affordances of blogs for self-directed learning. *Comput. Educ.* **2011**, *57*, 1628–1644. [CrossRef]
28. Witherspoon, E.B.; Higashi, R.M.; Schunn, C.D.; Baehr, E.C.; Shoop, R. Developing computational thinking through a virtual robotics programming curriculum. *ACM Trans. Comput. Educ.* **2017**, *18*, 1–20. [CrossRef]
29. Steier, F. *Reflexivity and methodology: An Ecological Constructionism*; Sage: Thousand Oaks, CA, USA, 1991.
30. Birch, M.; Miller, T. Inviting intimacy: The interview as therapeutic opportunity. *Int. J. Soc. Res. Methodol.* **2000**, *3*, 189–202. [CrossRef]
31. Chi, F. Turning experiences into critical reflections: Examples from Taiwanese in-service teachers. *Asia-Pac. J. Teach. Educ.* **2013**, *41*, 28–40. [CrossRef]
32. Matsuo, M. Goal orientation, critical reflection, and unlearning: An individual-level study. *Hum. Resour. Dev. Q.* **2018**, *29*, 49–66. [CrossRef]
33. Williams, B. Developing critical reflection for professional practice through problem-based learning. *J. Adv. Nurs.* **2001**, *34*, 27–34. [CrossRef]
34. Gibson, A.; Aitken, A.; Sándor, A.; Shum, S.B.; Tsingos-Lucas, C.; Knight, S. Reflective writing analytics for actionable feedback. In Proceedings of the 7th International Learning Analytics & Knowledge Conference, Vancouver, BC, Canada, 13–17 March 2017. [CrossRef]
35. Kember, D.; McKay, J.; Sinclair, K.; Wong, F.K.Y. A four-category scheme for coding and assessing the level of reflection in written work. *Assess. Eval. High. Educ.* **2008**, *33*, 369–379. [CrossRef]
36. Flavell, J.H. Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry. *Am. Psychol.* **1979**, *34*, 906–911. [CrossRef]
37. Fleming, S.M.; Dolan, R.J.; Frith, C.D. Metacognition: Computation, biology and function. *Philos. Trans. R. Soc. Lond.* **2012**, *367*, 1280–1286. [CrossRef] [PubMed]
38. Settle, A.; Franke, B.; Hansen, R.; Spaltro, F.; Jurisson, C.; Rennert-May, C.; Wildeman, B. Infusing computational thinking into the middle- and high-school curriculum. In Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education, New York, NY, USA, 3–5 July 2012. [CrossRef]
39. Grover, S. Assessing algorithmic and computational thinking in K-12: Lessons from a middle school classroom. In *Emerging Research, Practice, and Policy on Computational Thinking. Educational Communications and Technology: Issues and Innovations*; Rich, P., Hodges, C., Eds.; Springer: Cham, Switzerland, 2017. [CrossRef]
40. Korkmaz, Ö.; Çakir, R.; Özden, M.Y. A validity and reliability study of the computational thinking scales (CTS). *Comput. Hum. Behav.* **2017**, *72*, 558–569. [CrossRef]
41. Alayoub, H.W.M.; Nouby, A.M.; Amer, A.M. The effect of e-learning based on meta-cognition strategy on students' achievement and awareness of creative problems solving processes in a C++ course at Kuwait University. In Proceedings of the International Conference on e-Learning "Best Practices in Management, Design and Development of e-Courses: Standards of Excellence and Creativity", Manama, Bahrain, 7–9 May 2013. [CrossRef]
42. Jun, S.; Han, S.; Kim, S. Effect of design-based learning on improving computational thinking. *Behav. Inf. Technol.* **2017**, *36*, 43–53. [CrossRef]
43. Wang, Q.; Woo, H.L. Investigating students' critical thinking in weblogs: An exploratory study in a Singapore secondary school. *Asia Pac. Educ. Rev.* **2010**, *11*, 541–551. [CrossRef]
44. Xie, Y.; Ke, F.; Sharma, P. The effect of peer feedback for blogging on college students' reflective learning process. *Internet High. Educ.* **2008**, *11*, 18–25. [CrossRef]

45. Solhaug, T. Two configurations for accessing classroom computers: Differential impact on students' critical reflections and their empowerment. *J. Comput. Assist. Learn.* **2009**, *25*, 411–422. [CrossRef]
46. Kim, Y.H.; Min, J.; Kim, S.H.; Shin, S. Effects of a work-based critical reflection program for novice nurses. *BMC Med. Educ.* **2018**, *18*, 30. [CrossRef]
47. Alharbi, H.E. An Arabic assessment tool to measure technological pedagogical and content knowledge. *Comput. Educ.* **2019**, *142*, 103650–103658. [CrossRef]
48. Hattie, J. *Visible Learning: A Synthesis of over 800 Meta-Analyses Relating to Achievement*; Routledge: Oxfordshire, UK, 2008.
49. Strauss, A.L.; Corbin, J.M. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, 2nd ed.; SAGE Publications Inc.: Thousand Oaks, CA, USA, 1998.
50. Smith, E. Teaching critical reflection. *Teach. High. Educ.* **2011**, *16*, 211–223. [CrossRef]
51. Hao, N.; Ku, Y.; Liu, M.; Hu, Y.; Bodner, M.; Grabner, R.H.; Fink, A. Reflection enhances creativity: Beneficial effects of idea evaluation on idea generation. *Brain Cogn.* **2016**, *103*, 30–37. [CrossRef] [PubMed]
52. Song, Y. Design and implementation of reflection-based coding education: Case study of "SW and computational thinking course at H university. *J. Educ. Technol.* **2017**, *33*, 709–736. [CrossRef]
53. Turner, S.A.; Pérez-Quiñones, M.A.; Edwards, S.H. Peer review in CS2: Conceptual learning and high-level thinking. *ACM Trans. Comput. Educ.* **2018**, *18*, 1–37. [CrossRef]
54. van Lierop, M.; de Jonge, L.; Metsemakers, J.; Dolmans, D. Peer group reflection on student ratings stimulates clinical teachers to generate plans to improve their teaching. *Med. Teach.* **2018**, *40*, 302–309. [CrossRef] [PubMed]
55. Yang, S. Using blogs to enhance critical reflection and community of practice. *Educ. Technol. Soc.* **2009**, *12*, 11–21. Available online: https://www.jstor.org/stable/jeductechsoci.12.2.11 (accessed on 20 August 2020).