International Research Journal of
SCIENCE, TECHNOLOGY, EDUCATION, AND MANAGEMENT
P-ISSN: 2799-063X — E-ISSN: 2799-0648
Volume 4, No. 3 | September 2024

Improving computational thinking through flipped classroom: A case study on K-12 programming course in Macao

Wan Chong Choi[1], Iek Chong Choi[2], Chi In Chang[3]
[1]*Department of Computer Science, Illinois Institute of Technology, USA*
[2]*School of Education, City University of Macau, Macao SAR, China*
[3]*Department of Psychology, Golden Gate University, USA*
Corresponding email: wchoi8@hawk.iit.edu

ABSTRACT

In the twenty-first century, programming courses have become integral to primary school education worldwide, emphasizing the importance of computational thinking. However, research on computational thinking within programming courses has primarily focused on text-based programming languages, particularly at the secondary and higher education levels. There has been a lack of research on the influence of block-based programming languages on computational thinking in primary school. To fill this research gap, our study investigated the impact of the flipped classroom approach combined with block-based programming tools on developing computational thinking in primary school students in Macao. Over nine weeks, 20 third-grade students engaged with mBlock and Codey Rocky programmable hardware, with pretest and posttest assessments using the Programming Computational Thinking Scale (PCTS). Results indicated significant improvements in Computational Concepts and Computational Practices, though changes in Computational Perspectives were not statistically significant. Moreover, the Pearson correlation tests showed strong positive correlations among the dimensions of computational thinking and programming achievement. The findings demonstrated the flipped classroom approach, which emphasizing student-centered learning and active participation, effectively enhanced computational thinking skills. This study provided valuable insights for educators and curriculum developers aiming to improve programming education and computational thinking in primary schools, highlighting the potential of innovative teaching methods to meet twenty-first century educational needs. Using block-based programming tools combined with the flipped classroom approach offers a promising avenue for developing comprehensive computational thinking skills in young learners, ensuring they are better prepared for future technological challenges.

ARTICLE INFO

Received : July 28, 2024
Revised : Aug. 29, 2024
Accepted : Sept. 25, 2024

KEYWORDS

Block-based programming language, Computational thinking, Flipped classroom, K-12 programming course, Programming education

Suggested Citation (APA Style 7th Edition):

Choi, W.C., Choi, I.C., & Chang, C.I. (2024). Improving computational thinking through flipped classroom: A case study on K-12 programming course in Macao. *International Research Journal of Science, Technology, Education, and Management*, 4(3), 95-109. https://doi.org/10.5281/zenodo.13859011

INTRODUCTION

In today's educational landscape, teaching programming at both the primary and secondary school levels is increasingly recognized as crucial, stressing the significance of computational thinking for effective problem-solving and proficient use of technology (Chen et al., 2018; Wing, 2006). Initiatives like Estonia's ProgeTiger program, the U.K.'s updated computer science curriculum (Gove, 2014), and the U.S. government's investments in programming education reflected this educational shift (Cai & Wang, 2016). Asian regions, including Japan (Ohashi et al., 2018), China (China Ministry of Education, 2016), and Macao (Macao DSEDJ, 2024) also made notable progress in incorporating programming education.

Despite these efforts, challenges persisted in effectively developing computational thinking skills. Traditional teaching methods might not adequately translate programming concepts into practical applications (Tsou, 2014). Research on computational thinking within programming courses has primarily focused on text-based programming languages, particularly at the secondary and higher education levels (Lye & Koh, 2014). Research exploring the influence of block-based programming languages on primary school students' computational thinking is limited. (Fan et al., 2018). Additionally, the rapid evolution of educational technology and teaching methodologies calls for continuous evaluation and adaptation to ensure that educational practices meet current and future needs.

Addressing this research gap, our study contributed by exploring the flipped classroom model and block-based programming tools as a potential solution to enhance computational thinking in primary school programming courses. The flipped classroom, emphasizing student-centered learning, aligns with contemporary educational needs and focuses on higher-level cognitive processes (Anderson et al., 2001; Bergmann & Sams, 2012, 2014). Block-based programming tools, particularly, are examined for their ability to facilitate computational thinking by emphasizing higher-level cognitive skills (Grover & Pea, 2013) and potentially improving learning attitudes in primary education. This approach offers valuable insights and recommendations for implementing flipped teaching in block-based programming courses tailored for primary schools. This innovative combination aims to not only improve conceptual understanding but also to make learning more engaging and effective for young learners. Given this context, the study was directed by two research questions:

1) What is the influence of implementing the flipped classroom in primary school programming courses on students' development of computational thinking?
2) What is the correlation between the different dimensions of computational thinking and programming achievement?

LITERATURE REVIEW

Flipped Classroom

The flipped classroom concept was first introduced in the United States in 2007 and marked a significant change in teaching methods. Aaron Sams and Jonathan Bergmann (2012) developed it to address issues like poor weather and long distances that affected student attendance. They began by creating and sharing video lessons online, which formed the basis of what we now know as the flipped classroom.

This innovative approach reversed the traditional educational model: students engaged with instructional content at home, often through videos, and then spent classroom time on interactive activities, discussions, and problem-solving. This model promoted independent learning and allowed for a more profound understanding through collaborative in-class tasks (Hwang, 2016). It transformed the teacher's role from a lecturer to a facilitator, focusing on supporting individual learning needs and styles (Yen, 2018).

A critical distinction between traditional teaching and the flipped classroom was the focus of each approach. Traditional teaching was often teacher-centered, concentrating on lower-level cognitive processes like Remembering

and Understanding. In contrast, the flipped classroom was student-centered, highlights advanced thinking abilities such as Applying, Analyzing, Evaluating, and Creating, as illustrated in Table 1. This aligned with the upper tiers of the Revised Bloom's Taxonomy (Anderson et al., 2001), fostering improved computational thinking (Wang et al., 2015). This change in focus is consistent with the objectives of today's education, which increasingly encourages critical thinking and problem-solving abilities above rote learning.

Table 1. Traditional Teaching Vs. Flipped Classroom

Comparison	Traditional Teaching	Flipped Classroom
Teaching Mode	Teacher-centred	Student-centred
Revised Bloom's Taxonomy	Remember, Understand	Apply, Analyze, Evaluate, Create
Learning Level	Lower level	Higher level

The flipped classroom model may potentially enhance computational thinking abilities, especially in K-12 programming programs. Traditional methods, being more teacher-focused, tended to lower student engagement in computational thinking. Conversely, with its student-centric approach, the flipped classroom encouraged more complex in-class problem-solving, aligning with the "Low Floor, High Ceiling" principle in selecting computational tools. Block-based programming, for instance, simplified the learning curve, allowing students to concentrate on higher-level conceptual thinking rather than the intricacies of coding syntax (Brennan & Resnick, 2012; Grover & Pea, 2013).

In conclusion, the flipped classroom approach, emphasizing student-centered learning and advanced cognitive skills, might offer a compelling alternative to traditional teaching methods.

Computational Thinking

Computational thinking, a concept introduced by Jeannette M. Wing (2006), became fundamental in education, comparable to essential skills like language and mathematics. It's characterized as a systematic approach to solving problems involving breaking down problems, identifying rules, abstraction, and algorithm design (Phillips, 2009).

Brennan and Resnick (2012) presented an influential framework for computational thinking which developed by The Massachusetts Institute of Technology (MIT). This framework includes aspects like loops, sequences, problem-solving, and understanding programming ideas, and is structured in three dimensions: computational concepts, computational practices, and computational perspectives. This framework not only guides curriculum development but also provides a clear structure for assessing student progress in computational thinking.

In primary school programming courses, computational thinking emphasized the process of programming—analyzing, developing, and testing ideas rather than just completing code. This approach stressed problem-solving and perseverance (Liu, 2017). According to the K-12 Computer Science Framework Steering Committee (2016), acquiring knowledge in computer science effectively cultivated these skills. Becker et al. (2017) highlighted programming as a crucial tool for developing computational thinking, bridging the gap between abstract concepts and concrete thinking processes (Bienkowski et al., 2015). Studies have shown the benefits of applying block-based programming tools like Blockly or Scratch in elementary coding education.

Moreover, a research (Choi, 2022) that employed Code.org for block-based programming education revealed notable enhancements in students' computational thinking and attitudes toward learning. This illustrated the potential efficacy of thoroughly designed programming curriculums. Moreover, research on the educational game CodeCombat demonstrated its effectiveness in reducing cognitive load during programming education. This suggested that interactive platforms could simplify complex topics while increasing student engagement in learning to program (Choi & Choi, 2024).

In summary, computational thinking became vital in education with its structured approach to problem-solving. MIT's framework was widely adopted, and programming became essential in enhancing computational thinking skills.

MATERIALS AND METHODS

Design of Research

This study adopted a single-group pretest-posttest design and was structured using a quasi-experimental methodology (Büyüköztürk et al., 2008). The flipped classroom approach's influence was evaluated based on the variance in the Programming Computational Thinking Scale (PCTS) scores obtained before and after the intervention. Details of the study's methodology are outlined in Table 2.

Table 2. Design of Research

N	Pre-Test	Teaching Experiment	Post-Test
20	O_1	X	O_2

O_1: Initial assessment using the PCTS pre-test before implementing flipped teaching.
X: Application of the flipped classroom model.
O_2: Subsequent assessment using the PCTS post-test following the flipped classroom intervention.

Participants

The participant group comprised 20 grade three students aged between 8 and 9 years.

Teaching Materials

The instructional program extends over nine weeks, with two lessons per week for eighteen lessons in total. Each lesson lasts for 40 minutes. The mBlock used in this study, developed by Makeblock in 2013, is a teaching tool for block-based programming. It features a user-friendly "drag-and-drop" interface, supports student account management, and provides resources for teaching programming.

The mBlock software can control hardware and robots to enhance teaching versatility and student engagement. Codey Rocky, depicted in Figure 1, integrates hardware with software, offering a playful and creative way for students to learn programming. Using mBlock, Codey Rocky introduces students to advanced technologies.

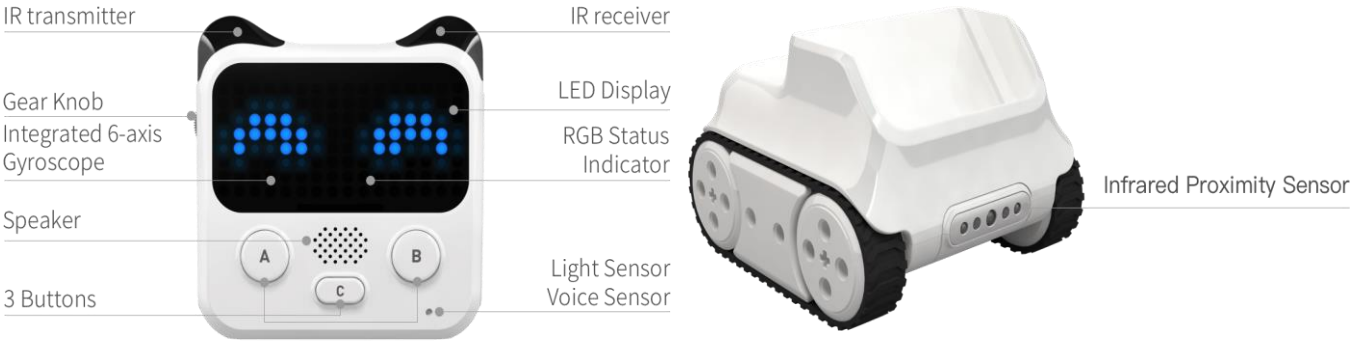


Fig. 1. Component configuration of Codey-Rocky

For example, students programmed Codey-Rocky (Figure 2) to follow a circular line (Figure 3) by pressing button A in a lesson using its light sensor. This task taught the advanced programming concepts (Events, Conditions, and Loops) and sensor use, making learning interactive and demonstrative.

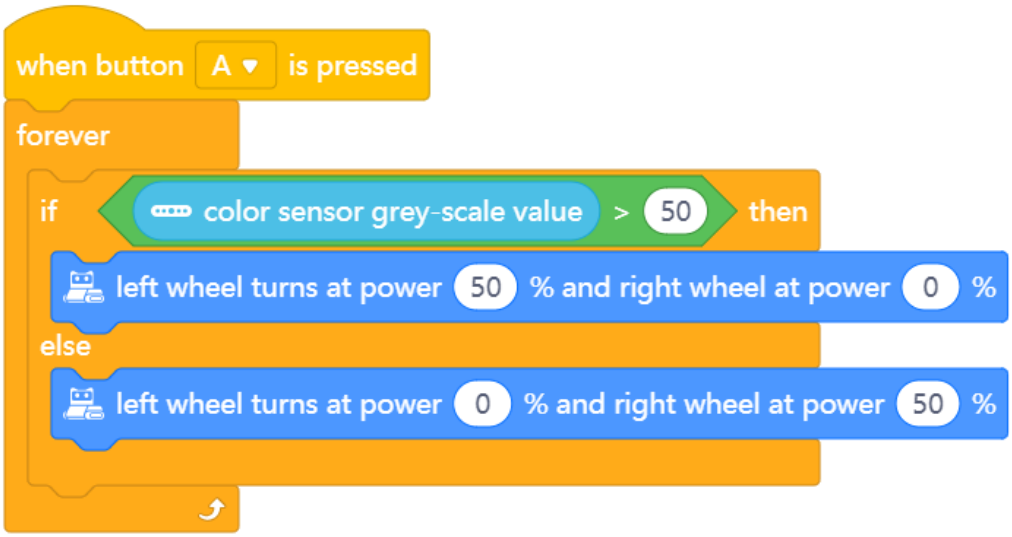


Fig. 2. The programming code written by the student

Fig. 3. Line tracked by Codey-Rocky

Flipped Classroom Activities

In order to encourage students to participate in active learning before, during, and after class, this study adopted the flipped classroom approach. Before class, students engaged in self-study, using resources such as e-books and flipped videos to familiarize themselves with the upcoming topics in eClass PowerLesson2 platform (learning management system). One critical factor for flipped classroom success was the students' ability to complete self-study before classes. As pointed out by Yeh (2015), fostering the habit of self-study was not only the responsibility of teachers but also a challenge they had to address during the flipped classroom implementation.

Before starting the research, the researcher stressed the importance of self-study and encouraged students to develop this habit. Before each lesson, the students had one week to study the e-books and flipped videos available on the eClass PowerLesson2 platform. To ensure understanding, they were required to complete three multiple-choice questions (Figure 4). Teachers could check the correctness of these responses to verify if students had adequately engaged with the e-books or flipped videos.

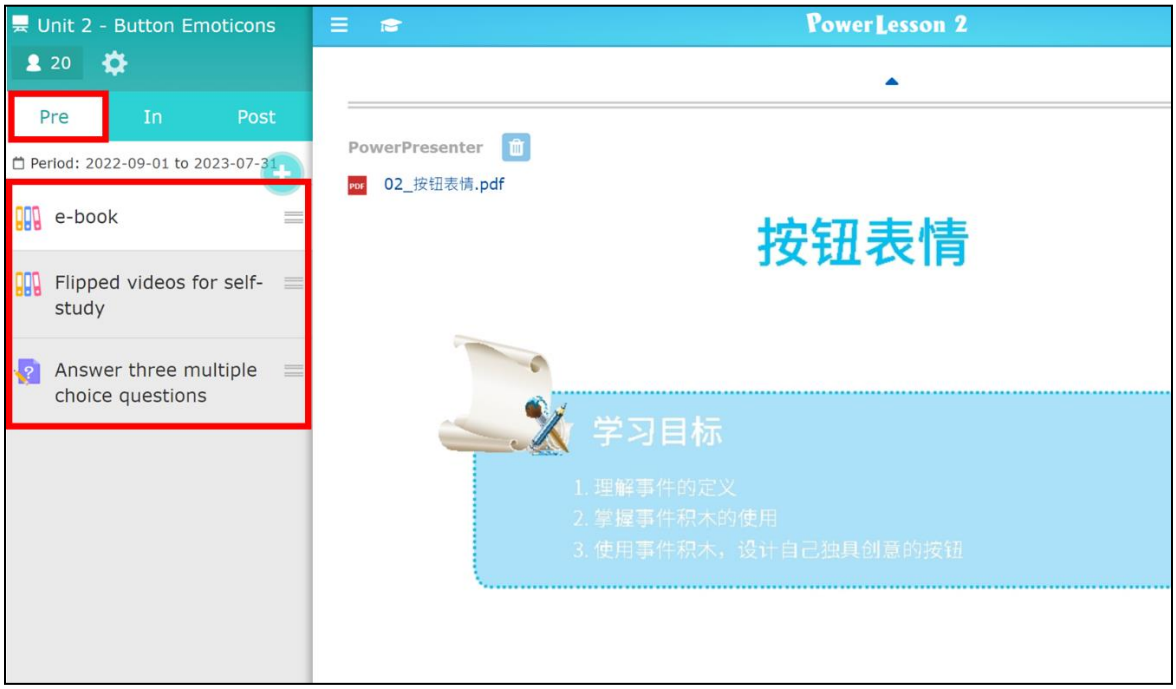


Fig. 4. Pre-class self-study materials in PowerLesson2

Fig. 5. In-class programming tasks in PowerLesson2

During class, the teaching approach was enriched with various interactive activities. Intriguing topics were introduced to make complex concepts more relatable and engaging. To facilitate this, the researcher split each week's

programming tasks into several basic and advanced tasks, placing them in the "In Class" section of the PowerLesson2 platform (Figure 5). In the classes, students were first instructed to watch the flipped videos related to the tasks, then proceed to complete the associated programming exercises, and upload their completed program files to the PowerLesson2 platform. Teachers could download these program files, observe and analyze the completeness of the student's work, and thus assess whether the students had understood the programming design concepts and fully understood the class content. This structured approach provided a clear framework for assessing learning outcomes and understanding students' grasp of the material. Additionally, the researcher interviewed students to gain insights into their perceptions of the task division and overall classroom experience.

Moreover, using programmable hardware such as Codey-Rocky provided a hands-on learning experience, enhancing the practical understanding of programming concepts. Teamwork and collaboration were emphasized to improve problem-solving skills and deepen understanding through group dynamics. Presentation sessions played a crucial role, offering students opportunities for self-expression and feedback, which helped assess their knowledge of the material.

After class, the learning process continued as students were encouraged to revisit the lesson content through flipped videos. This post-class engagement allowed for reinforcement and a more thorough comprehension of the subjects taught, ensuring a comprehensive and practical learning journey in the flipped classroom model.

Programming Computational Thinking Scale (PCTS)

The PCTS, developed by Wu (2021) and informed by Zhong et al.'s (2016) study, was employed in this research for its relevance to programming education. The scale, which adopts a five-point Likert format and aligns with MIT's computational thinking framework, encompasses 21 items distributed among three areas: Computational Practices, Computational Concepts, and Computational Perspectives, with eight, eight, and five questions in each category, respectively. The scale has shown strong reliability, with a Cronbach's Alpha of 0.947.

Programming Achievement Test

The programming achievement test evaluated students' programming skills through their submissions, which were divided into two sections. The first part consisted of 25 multiple-choice questions, each worth two points, totalling 50 marks, assessing computational concepts. The second part was a programming task worth 50 marks, where students had to design and write a program. This test was administered at the experiment's end to evaluate students' computational thinking development.

Data Analysis

SPSS was utilized, specifically for conducting paired samples t-tests to assess potential variations in computational thinking abilities between the pre-test and post-test. Additionally, the Pearson correlation test was employed to investigate relationships within the three dimensions of the PCTS and achievement test.

Moreover, we employed Python's Pandas for data management, and Seaborn and Matplotlib for visualizations. Seaborn generated boxplots for t-test results, elucidating score distributions. Heatmaps from Pearson correlation tests visually demonstrated correlations between PCTS and achievement test scores, enhancing the accessibility and comprehensiveness of our findings.

RESULTS

The Paired Samples t-test result of PCTS

The study used a paired samples t-test to assess the impact of a flipped classroom on students' computational thinking skills, as measured by the PCTS.

Table 3. Result of Paired Samples T-test for PCTS

Dimension	Test	Mean	N	Sd	df	t	p
Computational Concepts	Pre-test	3.59	20	0.55	19	-5.32	0.000
	Post-test	4.32	20	0.53			
Computational Practices	Pre-test	3.61	20	0.56	19	-3.23	0.004
	Post-test	4.14	20	0.65			
Computational Perspectives	Pre-test	3.72	20	0.68	19	-1.11	0.281
	Post-test	3.92	20	0.63			
Total	Pre-test	3.63	20	0.55	19	-3.66	0.002
	Post-test	4.16	20	0.56			

The results, summarized in Table 3, revealed significant improvements in two dimensions: Computational Concepts and Computational Practices. Specifically, the average scores for Computational Concepts increased from 3.59 to 4.32 ($t = -5.32$, $p < 0.05$), and for Computational Practices, from 3.61 to 4.14 ($t = -3.23$, $p < 0.05$). These findings suggest that the flipped classroom model positively influenced these aspects of computational thinking.

However, the change in Computational Perspectives was not statistically significant, with a slight increase in the mean score from 3.72 to 3.92 ($t = -1.11$, $p > 0.05$). Student interviews provided insights into this result. Some students reported improved language skills and the ability to discuss their work due to class presentations, while others desired more opportunities for practice to enhance confidence and presentation abilities. This feedback implies that the limited occasions for students to present in class might have contributed to the slight improvement in Computational Perspectives scores.

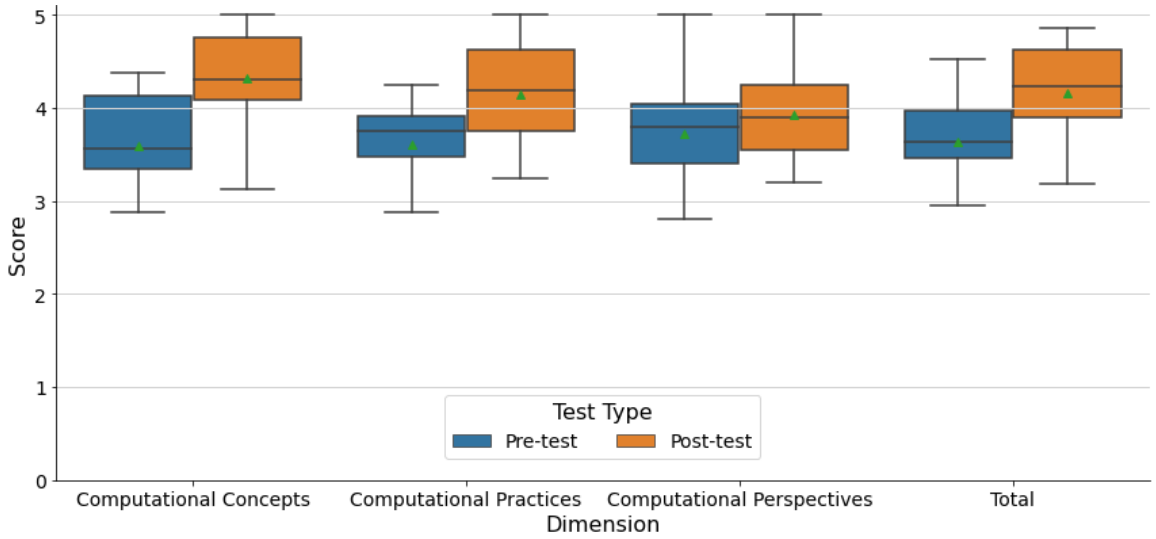


Fig 6. Combined boxplot of Pretest-Posttest scores

Figure 6 presents a combined boxplot of the pretest-posttest scores, visually supporting the statistical findings. The post-test scores exhibit an upward shift in the median and a narrowing of the interquartile ranges for Computational Concepts and Computational Practices, which indicates an improvement in these areas. However, the boxplots for Computational Perspectives show an overlap in interquartile ranges between the pretest-posttest, reflecting the nonsignificant statistical result.

Overall, the intervention of the flipped classroom approach successfully enhanced computational concepts and practices, but it appears that additional focus on frequent, supportive presentation opportunities is needed to see a significant improvement in computational perspectives.

The Students' Achievement after Flipped Teaching

During the final session of the teaching experiment, the students completed fifty multiple-choice questions and two programming tasks, which the teacher subsequently evaluated in terms of completeness and correctness. Throughout a nine-week period, the students' performance was determined by analysing the counting results and teacher observations, specifically in mBlock with programmable hardware Codey Rocky activities, significantly surpassed their typical classroom activities. This outcome indicated their diligent efforts in acquiring and applying computational thinking concepts to programming.

Table 4. Result of Programming Achievement Test

Concept	Total	Mean	Correctness Rate
Sequences	10	9.10	91.0%
Parallelism	10	8.40	84.0%
Events	10	8.65	86.5%
Conditions	20	15.70	78.5%
Data	15	12.71	84.7%
Operators	15	13.29	88.6%
Loops	20	15.80	79.0%
Total	100	83.65	83.7%

Table 4 presents the results. It is evident from these that the vast majority of students responded to the Sequence questions with a correctness rate of 91%. Furthermore, a significant proportion of students answered correctly to questions related to Parallelism (84%), Events (86.5%), Data (84.7%), and Operators (88.6%), with correctness rates exceeding 80%. However, students exhibited a relatively lower level of proficiency in Conditions (78.5%) and Loops (79%). This was because these concepts are characterized by abstraction, complex syntax, and advanced problem-solving, making them more challenging for students to comprehend. As such, further practice and instruction may be required to facilitate a more thorough comprehension of these concepts (Conditions and Loops).

The correlation between the PCTS and Achievement Test

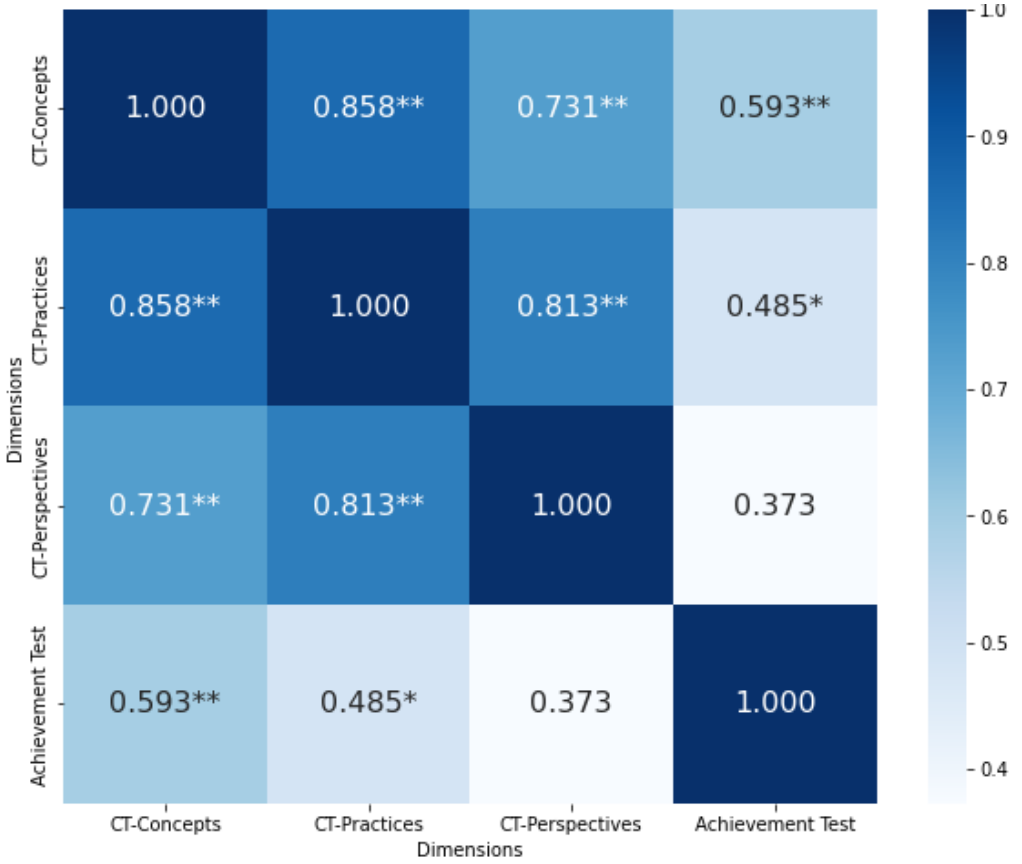
The research used the Pearson correlation test to examine the correlations between the three dimensions of the PCTS and the programming achievement test. The results, presented in a heatmap in Figure 7, revealed significant positive correlations among these dimensions.

The strongest correlation was observed between Computational Concepts and Computational Practices, registering at 0.858**, indicating a robust positive relationship between understanding computational concepts and their practical implementation. Another significant correlation was seen between Computational Practices and Computational Perspectives, scored at 0.813**, suggesting that students proficient in computational problem-solving often develop a more comprehensive Perspective on computational thinking.

Additionally, the correlation between Computational Concepts and Computational Perspectives was significant at 0.731**, although it was slightly less pronounced than the others.

Regarding the achievement test, its strongest correlation was with Computational Concepts at 0.593**, followed by Computational Practices at 0.485*, and the weakest correlation was with Computational Perspectives at 0.373. This indicates that performance in the achievement test is more closely linked to understanding and applying concepts than having a broader computational thinking perspective.

In conclusion, the study highlights a robust positive interrelation among the PCTS dimensions and their collective impact on computational thinking achievement. It suggests that enhancing one dimension of computational thinking could positively influence other areas, improving student achievement in this field.



* Correlation is significant at the 0.05 level
** Correlation is significant at the 0.01 level

Fig 7. Pearson correlation heatmap between the PCTS and achievement test

DISCUSSION

RQ1) What is the influence of implementing the flipped classroom in primary school programming courses on students' development of computational thinking?

Applying the flipped classroom approach into practice in primary school programming courses has demonstrated notable improvements in students' computational thinking, particularly in the areas of Computational Concepts and Computational Practices. The study's results revealed significant gains in these dimensions, as evidenced by the post-test scores compared to the pre-test scores.

Students showed a marked increase in their understanding of computational concepts, with average scores rising from 3.59 to 4.32. This suggests that the flipped classroom model, which allows students to engage with instructional content at their own pace before class, effectively supports their grasp of foundational programming concepts. The pre-class videos and e-books allowed students to familiarize themselves with the material, making classroom time more efficient and focused on interactive problem-solving and practical applications.

The improvement in Computational Practices scores, from 3.61 to 4.14, indicates that students benefited from the hands-on activities and collaborative tasks conducted during class. By applying their pre-learned knowledge in practical, interactive settings, students could deepen their understanding and enhance their problem-solving skills. Programmable hardware like Codey Rocky also played a crucial role in this process, providing tangible and engaging ways for students to experiment with programming concepts.

While there was an improvement in Computational Perspectives, it was not statistically significant. This aspect of computational thinking involves a broader understanding of the role and impact of computing in various contexts and the ability to articulate and present computational ideas. The slight increase from 3.72 to 3.92 suggests that while students improved in this area, the limited opportunities for presentations and discussions may have hindered more substantial progress. Interviews with students indicated a desire for more practice in presenting their work, highlighting the need for additional focus on communication and presentation skills within the flipped classroom framework.

The overall positive impact of the flipped classroom model on computational thinking can be attributed to its student-centered approach, which aligns with contemporary educational theories emphasizing active learning and higher-order cognitive skills. The model's effectiveness in enhancing computational concepts and practices underscores its potential as a valuable pedagogical tool in primary education.

RQ2) What is the correlation between the different dimensions of computational thinking and programming achievement?

The study found significant positive correlations between the different dimensions of computational thinking and programming achievement. These correlations suggest that improvements in one dimension of computational thinking are likely to be associated with enhancements in others and with overall programming performance.

The strongest correlation was observed between Computational Concepts and Computational Practices (0.858**). This robust relationship indicates that a solid understanding of computational concepts directly supports practical problem-solving abilities. Students who excel in grasping fundamental programming principles are better equipped to apply these concepts in real-world scenarios, leading to more effective computational practices.

There was also a significant correlation between Computational Practices and Computational Perspectives (0.813**), suggesting that students proficient in practical computational tasks tend to develop a broader understanding of computing's role and can articulate their ideas more effectively. This interconnection highlights the importance of integrating practice-based learning with opportunities for reflection and discussion to foster a comprehensive computational thinking skill set.

The programming achievement test showed the strongest correlation with Computational Concepts (0.593**), followed by Computational Practices (0.485*) and Computational Perspectives (0.373). These findings indicate that students' performance on the achievement test is more closely linked to their understanding and application of computational concepts than to their broader perspectives on computing. This emphasizes the need for a balanced approach in teaching programming, where foundational knowledge is reinforced through practical application and contextual understanding.

In summary, the study's findings highlight the importance of a holistic approach to teaching computational thinking. Enhancing one dimension can positively influence others, leading to improved programming achievement. The flipped classroom model, with its emphasis on active and student-centered learning, appears well-suited to support this integrated approach, providing a solid foundation for developing computational thinking skills in primary education.

CONCLUSION

The investigation into the flipped classroom model's effectiveness, combined with block-based programming tools in a primary school setting in Macao, has provided an insightful understanding of young learners' development in computational thinking. This study revealed the multifaceted impacts of this approach on different computational thinking aspects.

Implementing the flipped classroom model significantly enhanced students' Computational Concepts and Practices abilities. This was evident from the marked improvements in post-test scores, affirming the model's capability to foster advanced cognitive skills. These outcomes align with the model's focus on student-centered learning, promoting active involvement and a deeper understanding of complex concepts.

However, less improvement was seen in Computational Perspectives, possibly due to limited opportunities for student presentations and discussions. These activities are essential for developing broader perspectives and confidence in computational thinking. This indicates the need for more frequent and diverse student-led activities encouraging idea exploration and presentation.

The study also highlighted strong positive correlations among the dimensions of computational thinking and their collective impact on programming achievement. This interconnection suggests that a holistic approach targeting all computational thinking dimensions can lead to more comprehensive enhancements in programming skills.

Despite its benefits, the flipped classroom model presents certain challenges. One notable issue is the increased workload for teachers, who must prepare comprehensive self-study materials, such as videos and e-books, for students to engage with outside of class. This preparation requires significant time and effort, which may not be feasible for all educators, especially in schools with limited resources. To address this challenge, a collaborative approach is recommended, where teachers work in teams to develop self-study materials. Sharing resources and dividing tasks among colleagues can reduce the individual workload and lead to a richer, more diverse range of learning materials. Additionally, schools could provide training and support for teachers to develop high-quality educational content efficiently.

In terms of pedagogical implications, this study contributed to existing knowledge by demonstrating that the flipped classroom model, supported by block-based programming tools, can significantly enhance primary school students' computational thinking skills. It underscored the importance of active learning strategies and the use of appropriate technological tools to foster a deeper understanding of computational concepts and practices. This study suggested that educators seeking to improve computational thinking skills should consider integrating the flipped classroom approach into their teaching practices. It provided a valuable model for developing engaging, student-centered learning environments encouraging exploration, problem-solving, and applying computational thinking in real-world contexts.

In primary education, these findings carry significant implications. The flipped classroom, emphasizing student engagement and higher cognitive skills, represents a shift from traditional teacher-centric methods. Its effectiveness in this study indicates its potential value in the broader context of educational strategies for programming and computational thinking.

Moreover, utilizing tools for block-based programming (mBlock) and programmable hardware (Codey-Rocky) has been an effective complement to this educational approach. These tools simplify the learning process, letting students concentrate on conceptual understanding rather than coding syntax complexities. This approach fits well with the primary-level goal of establishing a solid computational thinking foundation.

LIMITATIONS AND FUTURE DIRECTIONS

While this study provided insightful observations about the flipped classroom approach’s potential for enhancing computational thinking in primary education, it is crucial to recognize certain limitations that could have influenced the results. Firstly, the research involved a relatively small sample of 20 third-grade students. This limited sample may not adequately reflect the diversity of student populations across different schools, regions, or educational settings. Consequently, the results may not be entirely applicable to broader contexts, as different student groups might have responded differently to the flipped classroom approach.

Second, the teaching experiment lasted only nine weeks, which may have been too brief to observe the long-term impact of the flipped classroom model on students' computational thinking development. A longer study period could have provided a more comprehensive view of how sustained exposure to this teaching method impacts learning outcomes over time. Short-term gains observed in computational concepts and practices might differ from long-term retention or transfer of skills.

Furthermore, the study's focus on a specific age group (8-9 years) and educational context (primary school in Macao) limited the scope of its applicability. The flipped classroom model's effectiveness might vary significantly in different age groups, such as older students who may have different learning needs and levels of cognitive development. Additionally, cultural and educational differences could have influenced how the flipped classroom approach was received and implemented, affecting its overall impact.

Another important consideration is the role of student motivation in the flipped classroom. Future research should explore how various factors, such as the type of materials used and classroom activities, impact student engagement and motivation. Understanding these factors can help optimize the flipped classroom approach for different learners.

To overcome these limitations and strengthen future research, it is suggested that future studies include larger, more diverse samples, incorporating students from different grade levels, schools, and regions. Extending the duration of the studies would allow for a more in-depth investigation into the long-term effects of the flipped classroom model on computational thinking and other related skills. Moreover, future research could explore variations in implementation strategies, including different types of flipped classroom activities and technological tools, to determine the most effective approaches for different educational contexts.

Given the promising results of this small-scale intervention, there is a need to address the scalability of the flipped teaching method for larger groups of students and different educational settings. Scaling up would involve addressing logistical challenges, such as providing access to the necessary technological resources and ensuring that teachers are adequately trained to implement this model effectively. Additionally, adapting the flipped classroom approach to different curricula and varying levels of student readiness will be crucial for maintaining its effectiveness across diverse educational environments. By conducting pilot studies in larger and more varied educational contexts, researchers can gather data on the feasibility and impact of scaling the flipped classroom model, offering valuable insights into how this innovative approach can be integrated into mainstream education on a broader scale.

In summary, this study offered valuable insights into primary education for programming and computational thinking. It demonstrated the potential of innovative teaching methods, such as the flipped classroom and suitable technological tools, to significantly improve young learners' computational thinking skills. This research can guide future educational practices and curriculum development, aligning them with the evolving needs of 21st-century learners. By addressing these limitations and expanding the scope of research, educators and policymakers can gain a more nuanced understanding of the flipped classroom model's potential benefits and challenges, ultimately leading to more informed decisions about its integration into programming and computational thinking curricula.

ETHICAL CONSIDERATIONS

In conducting this study, we followed the ethical guidelines for research involving children (NHREC, 2016). Before initiating the research, consent was secured from the Macao primary school participating in the study. Detailed information about the research was subsequently shared with the school's administrators and teachers involved in the intervention study, from whom we obtained permission to proceed. Additionally, comprehensive information was provided to the parents, from whom explicit consent was obtained. To further protect participant privacy, we used pseudonyms instead of real names. This measure ensured participant anonymity and that no identifiable information was disclosed in this study.

REFERENCES

Anderson, L., Krathwohl, D., Airasian, P., Cruikshank, K., Mayer, R., Pintrich, P., Raths, J., & Wittrock, M. (2001). *Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Pearson.

Becker, S. A., Cummins, M., Davis, A., Freeman, A., Hall, C. G., & Ananthanarayanan, V. (2017). *NMC horizon report: 2017 higher education edition*. The New Media Consortium.

Bergmann, J., & Sams, A. (2012). *Flip your classroom: Reach every student in every class every day*. International society for technology in education.

Bergmann, J., & Sams, A. (2014). *Flipped learning: Gateway to student engagement*. International Society for Technology in Education.

Bienkowski, M., Snow, E., Rutstein, D. W., & Grover, S. (2015). Assessment design patterns for computational thinking practices in secondary computer science: A first look. *SRI International*.

Brennan, K., & Resnick, M. (2012). Using artifact-based interviews to study the development of computational thinking in interactive media design. *Annual American Educational Research Association Meeting, Vancouver, BC, Canada*, 1–25.

Büyüköztürk, Ş., Kılıç Çakmak, E., Akgün, Ö. E., Karadeniz, Ş., & Demirel, F. (2008). *Scientific research methods* (3rd ed.). Ankara: PegemA.

Cai, S., & Wang, W. (2016). The Role of Social Organizations to STEM Education in the U.S. and Enlightenment to China. *China Educational Technology*, *10*, Article 10.

Chen, P., Huang, R. H., Liang, Y., & Zhang, J. B. (2018). How to Cultivate Computational Thinking——A Review of the International Publications in the Past Decade and the International Conference on Computational Thinking Education in 2017. *Modern Distance Education Research*, *1*, 98–112.

China Ministry of Education. (2016). *The 13th Five-Year Plan for Education Informatization*. www.gov.cn/gongbao/content/2016/content_5133005.htm

Choi, W. C. (2022). The Influence of Code.org on Computational Thinking and Learning Attitude in Block-Based Programming Education. *Proceedings of the 2022 6th International Conference on Education and E-Learning*, 235–241.

Choi, W. C., & Choi, I. C. (2024). Investigating the Effect of the Serious Game CodeCombat on Cognitive Load in Python Programming Education. *2024 IEEE World Engineering Education Conference (EDUNINE)*.

Fan, W., Zhang, Y., & Li, Y. (2018). An overview of research and development of computational thinking in China and abroad. *Journal of Distant Education*, *(02)*, 3–17.

Gove, M. (2014). *Michael Gove speaks about computing and education technology*. https://www.gov.uk/government/speeches/michael-gove-speaks-about-computing-and-education-technology

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, *42*(1), 38–43.

Hwang, G. J. (2016). Definition, purpose and development of flipped teaching. In *Flipped Classroom: Theories, Strategies and Applications* (pp. 1–20). Taiwan Higher Education.

K-12 Computer Science Framework Steering Committee. (2016). *K-12 computer science framework*. ACM.

Liu, M. C. (2017). Creative Education, Computational Thinking, and Programming ~ Several Curriculum and Instructional Design Concepts from "Thinking" to "Doing." *Taiwan Education Review Monthly*, 138–140.

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.

Macao DSEDJ. (2024). *School Year School Development Plan*. https://portal.dsedj.gov.mo/webdsejspace/page/fe/index.jsp

Ohashi, Y., Kumeno, F., Yamachi, H., & Tsujimura, Y. (2018). Readiness of Japanese elementary school teachers to begin computer-programming education. *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, 807–810.

Phillips, P. (2009). Computational Thinking. *A Problem Solving Tool for Every Classroom. CSTA*.

Tsou, C. Y. (2014). *The effect on computer technology learning by means of PBL strategy to fifth grade students：Taking the learnong of Scratch program language as an example*. NTOU-National Taiwan Ocean University.

Wang, Z., Shi, Y. X., & Ma, C. (2015). *A study of flipped classroom teaching based on learning science*. Global Chinese Conference on Computers in Education, Taipei, Taiwan.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wu, P. C. (2021). *The Development of Computational Thinking Self-Efficacy Scale*. National Taichung University of Education.

Yeh, P. C. (2015). *Teach for the Future: The New Thinking of BTS Education by Yeh Ping Cheng*. Common Wealth Education Media and Publishing.

Yen, M. C. (2018). *A Meta-Analysis of the Learning Effects of Flipped Classroom on Elementary School and Junior High School Students*. Central Taiwan University of Science and Technology.

Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. *Journal of Educational Computing Research*, *53*(4), 562–590.