Fostering Programming Interest Through Block Coding in Young Learners

Dorjpalam Tserendejid[1]*, Khajidmaa Otgonbaatar[2], Magsarjav Bataa[3]
Department for Education Policy, Mongolian National Institute for Educational Research, Ulaanbaatar, Mongolia[1]
Academic Secretary, Mongolian National Institute for Educational Research, Ulaanbaatar, Mongolia[2]
Department of Information and Computer Sciences, National University of Mongolia, Ulaanbaatar, Mongolia[3]

***Abstract*—Coding has become a fundamental skill for cultivating computational thinking in young learners. This study aims to examine the impact of coding education on programming interest among fifth-grade students in Mongolia with no prior coding experience. A 16-week pilot course using block-based coding activities on Code.org involved 195 5th-grade students across three schools. A pre-test/post-test design compared an experimental group ($N_{experimental}$=95) with a control group ($N_{control}$=100). Results indicate significant improvements in coding skills (Cohen's d = 0.84) and computational thinking, with average programming skill ratings increasing from 5.19 to 7.07 out of 10. Qualitative data show heightened engagement, perceived creativity, and motivation, with students describing coding as gamified activities alongside an increase in self-rated programming interest. Moreover, post-intervention interviews reveal emerging awareness of gender inclusivity in programming. These findings suggest that structured block-based coding can effectively enhance computational thinking and interest in programming among young learners, providing evidence to support early integration of programming in primary education curricula.**

***Keywords*—Computational thinking; problem-solving; block-based coding; coding education; programming interest**

I. Introduction

Today, coding has become a fundamental skill driven by technological advancements across all aspects of learning. The concept of computational thinking can be viewed as a compilation of problem-solving competencies that are intrinsically linked to the tenets of computer science [1]. There are both economic and developmental motivations for commencing computer science education in early childhood. Research demonstrates that early educational interventions in programming lead to reduced costs and more sustainable outcomes than those initiated later in a child's academic journey [2], [3]. With the emergence of the sixth information technology revolution, defined by artificial intelligence, the ability to code has become a vital competitive edge for today's workforce. Young students are enthusiastic about quickly learning basic digital skills to adapt to the evolving information society [4]. Additionally, [5] highlighted that computational thinking is an essential skill set that everyone should be motivated to acquire and apply in their everyday lives. Moreover, [6] proposed that Computational Thinking (CT) has existed since the 1950s, originally termed 'algorithmic thinking', which involves employing a systematic and precise sequence of steps to address a problem, utilizing a computer when suitable for this purpose.

Through the digital landscape, computational thinking (CT) and coding have become critical foundational skills, now recognized as essential parts of national curricula in many education systems. CT refers to mental processes such as breaking down problems, recognizing patterns, designing algorithms, and abstracting solutions in ways that can be automated using technology [5]. Coding serves as a practical application of these skills, allowing learners to express ideas through structured, programmable instructions, promoting problem-solving, logical reasoning, and digital fluency [7], [8].

Although there is no consensus among academics on a definitive interpretation of computational thinking (CT), Wing describes it as the process of problem-solving, system design, and the concept of human behavior, drawing from the essential principles of Computer Science. She notes that computational thinking is not merely programming but rather involves a broader scope that includes thinking at multiple levels of abstraction [5]. In addition, [5] describes computational thinking as a form of analytical thought that seeks to understand human behavior, offering a structure on essential concepts such as generating system designs, addressing problems, and processing information. On one hand, studies have shown that instruction in computational thinking (CT) fosters a deeper understanding that programming is fundamentally about addressing problems, rather than simply coding, and it can bolster female students' perceptions and self-assurance in programming [9]. It is noteworthy that CT serves as an early indicator and predictor of academic success, as the CT scores have shown a strong correlation with overall academic performance [10].

Several countries, including Finland, the United Kingdom, South Korea, and the United States, have adopted CT and coding in primary and secondary education as part of broader digital literacy and STEM agendas [8], [11]. Research consistently shows that introducing coding at an early age supports cognitive development, including executive function, critical thinking, and conceptual reasoning [11], [12]. Beyond cognitive gains, coding activities, especially when delivered through visual programming platforms, mobile apps, or educational robotics, enhance learner motivation, engagement, collaboration, and creativity [11], [13]. Importantly, CT is not limited to computer science alone. It supports learning across diverse subjects, including mathematics, science, language arts, and the arts, and promotes interdisciplinary problem-solving [14], [15]. These cross-cutting benefits have led to a global

*Corresponding author

shift toward embedding coding and CT within general education rather than treating them as niche or elective skills.

The Mongolian education context, [16], [17] explicitly calls for enhanced digital competencies, improved access to STEM education, and inclusive, future-oriented learning pathways. Integrating coding and CT into primary education supports these goals by preparing students with relevant skills for digital society and addressing urban–rural disparities in access to quality education. This study responds to the need for localized evidence by examining the effects of structured coding activities on the development of computational thinking in fifth-grade students in Mongolia. Through a pre-test/post-test design, the study analyzes both the statistical significance and practical impact of coding instruction, contributing to curriculum reform and global dialogue on 21st-century education.

II. Literature Review

A. Importance of Coding Education

Literature defines coding as the process by which computers transform human-understandable operations into executable algorithms or sequences of commands [18]. It has been proposed that coding can encourage self-directed learning. Additionally, it supports literacy and mathematical communication, as well as metacognitive abilities, especially when children are introduced to coding at an early age [19]. Furthermore, learning to code equips students with a novel symbolic language that enables them to develop programs and projects, thereby establishing a positive link between coding skills and creative thinking [20].

Introducing children to coding, an essential skill for digital literacy and the language of today's world, should occur in early developmental stages of children [21]. The practice of coding also enables analysis, problem-solving, and concept development by formulating algorithms that break down problems and representing these algorithms in a programming language. In this way, coding serves as a medium for children to express their ideas and perspectives [22]. Programming also offers children opportunities to "interact" with computers and allows them to learn new symbolic languages [7], [20]. Additionally, 21st-century skills are essential for individuals, with coding being considered a fundamental skill. Coding should be included among the competencies that every student ought to acquire during their educational journey [23].

On the other hand, the concept of 'coding without computers' relates to educational experiences where students engage in lessons known as 'unplugged'. In these lessons, the emphasis is not on algorithmic reasoning, computer interactions, or database management. Instead, students often use pencils and paper or physical manipulatives. These thoughtfully crafted kinesthetic activities allow students to understand intricate concepts through cards, games, and puzzles. Such experiences are pertinent to their lives and promote the learning of fundamental information technology principles [24]. In addition, numerous studies noted that graphical environments can support cognitive growth in children and are more effective for teaching programming to younger age groups [25], [26].

B. Unpacking Interest in Creative Programming

In terms of creative learning, constructivism is recognized as a significant concept in today's educational sphere. Yet only a small fraction of classrooms genuinely encourages students to approach their schoolwork as a creative act of knowledge construction. The goal of these initiatives is to create a classroom environment that functions as a community for building knowledge [27]. As the integrated mode is recognized as a more imaginative processing style, the authors advised that computer programming instructors should prioritize software design, which utilizes an integrated mode [28]. Moreover, a variety of computer programs are accessible. To enhance the cultivation of creative thought, these programs should function as tools for cognitive processes. We must exercise caution against providing a haphazard assortment of programs or tools that lack internal coherence [29].

Engaging in creative coding requires diverse problem-solving competencies that leverage computer algorithms, often referred to as "computational thinking" [5]. On the one hand, Computational thinking (CT) refers to a method of problem-solving that encompasses the formulation of problems; the logical organization and analysis of data; the representation of data via abstractions; the identification and automation of solutions through algorithmic reasoning; the analysis and implementation of potential solutions; and the generalization and transfer of the problem-solving process [30], [31].

III. Methodology

A mixed-methods parallel design was employed, integrating quantitative and qualitative approaches with equal weighting. The structural workflow of this mixed-methods parallel design is visually illustrated in Fig. 1.

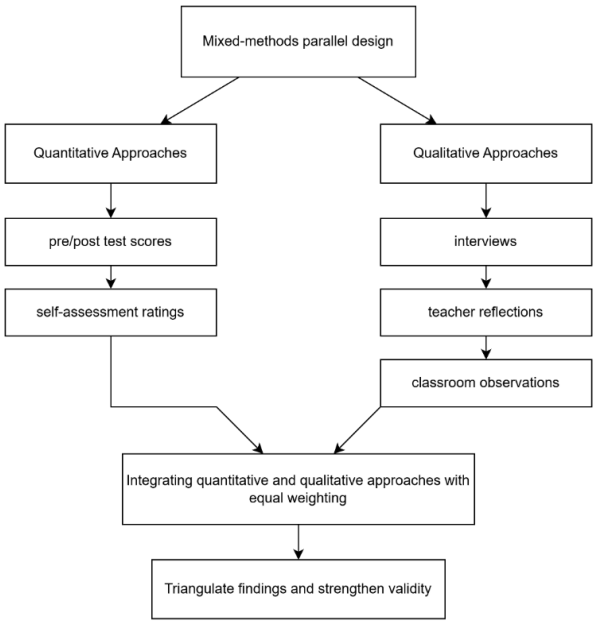

Fig. 1. Mixed-methods parallel design of the study.

This design allowed the combination of statistical analysis and thematic interpretation to triangulate findings and strengthen validity [32]. Quantitative data included pre/post

test scores and self-assessment ratings, while qualitative data were derived from interviews, teacher reflections, and classroom observations. Qualitative data from interviews, classroom observations, and teacher reflections were collected concurrently to enrich interpretation and triangulate findings. For the qualitative phase, we employed a purposive stratified sampling method to select 15 participants from the experimental group ($n_{experimental}$=95). This method was chosen to ensure a diverse representation of gender, geographic location (urban vs. rural), and varying levels of coding progression observed during the 16-week intervention. Existing classroom groups were used without random assignment due to practical school-level constraints.

Both methods were implemented concurrently and analyzed independently before being merged in a comprehensive interpretation. This approach corroborates quantitative data with rich descriptive insights from classroom practices and interviews, etc.

A. Research Context and Sampling

This study was conducted as part of a national pilot project designed to integrate computer science education into the Mongolian primary curriculum. The research focused on fifth-grade students who had no prior coding experience. The objective was to examine how structured, block-based coding activities influence computational thinking and interest in programming. The research data refers to 195 students in 5th grade from three public schools, consisting of two rural schools and one urban school, including 97 males and 98 females, who possessed no previous coding knowledge or skills and were being introduced to the subject for the first time. Teachers enrolled in the pilot lessons had limited coding experience and pedagogical confidence. However, they received pre-study training and continuous pedagogical support. Ethical approval and informed consent were obtained from all participants and their guardians before data collection.

B. Teacher Profile and Preparation

Fifteen teachers participated in the pilot implementation (see Table I). The majority were female (86.7%) and held a bachelor's degree (66.7%). Before classroom implementation, all teachers attended an 8-hour professional development workshop focused on Code.org pedagogy, computational thinking concepts, and strategies for facilitating creative, student-centered learning. Teachers were provided with lesson plans, teaching aids, and access to a monitoring platform to track student progress and ensure consistent delivery across schools.

C. Instruments and Outcome Measures

To evaluate the intervention, two distinct quantitative assessment tools were deployed at baseline (pre-test) and post-intervention (post-test).

1) Computational thinking performance assessment: This 10-point instrument was adapted from the verified Code.org Course C diagnostic metrics and validated primary school block-coding tasks established by relevant literature. It comprises 10 items (scored 0 or 1 point each, yielding a 0–10 composite scale) that assess five core computational thinking sub-constructs: algorithmic design (2 items), pattern recognition (2 items), loops/repetition logic (2 items), abstraction (2 items), and debugging logic (2 items). A pilot testing phase involving an independent cohort of fifth-grade students (N = 25) yielded a Cronbach's alpha of $\alpha = 0.81$, indicating robust internal consistency. Construct validity was confirmed through evaluation by an expert panel comprising two educational technology professors and one senior computer science teacher.

2) Self-rated programming interest and ability scales: Student perceptions were collected through two standalone 10-point Likert-type single items ("Rate your interest in learning computer programming" and "Rate your current ability to write code/solve puzzles", where 1 = Very Low and 10 = Very High).

TABLE I. THE DEMOGRAPHIC CHARACTERISTICS OF TEACHERS

Variables	Values	Frequency	Percentage (%)
Gender	Female	13	86.7%
	Male	2	13.3%
Educational attainment	Bachelor's degree	10	66.7%
	Master's degree	5	33.3%
Age bracket	25–35	8	53.3%
	36–45	3	20.0%
	46–55	4	26.7%
Professional commitment	2–5 years	3	20.0%
	6–10 years	6	40.0%
	11–20 years	2	13.3%
	21–30 years	2	13.3%
	31–40 years	2	13.3%
Experience in teaching coding	Yes	0	0.0%
	Somewhat	5	33.3%
	No	10	66.7%

D. Qualitative Data Collection and Analysis Rigor

The qualitative strand utilized semi-structured interview protocols alongside structured classroom observations. Interviews were conducted in the native Mongolian language by the primary research team to eliminate communication barriers. All recorded audio files were transcribed and translated into English using a back-translation protocol. The text was analyzed using inductive thematic analysis. Two researchers independently coded the transcripts to identify repeated semantic patterns regarding student enjoyment, confidence, and equity. Inter-rater reliability was verified using Cohen's Kappa *(k)*, yielding an initial agreement of $k = 0.83$. Minor coding discrepancies were resolved through iterative team consensus meetings. Final frequencies (e.g., 11 out of 15 participants noting high enjoyment) represent systematic code distribution across the entire sample rather than illustrative anecdotes. Table II presents the demographic data and backgrounds of participating students.

TABLE II. DEMOGRAPHIC DATA AND BACKGROUNDS OF PARTICIPATING STUDENTS.

Demographic and Personal Information	Values	Frequency	Percentage (%)
Gender of students	Male	97	49.7%
	Female	98	50.3%
Student age	10 years old	189	96.9%
	11 years old and above	6	3.1%
Previous coding experience	Yes	5	2.6%
	No	190	97.4%
Parents work in IT-related jobs	Yes	5	2.6%
	No	190	97.4%
Awareness of platforms	Khan Academy	3	1.5%
	Scratch	1	0.5%
	Code.org	2	1.0%

E. Course Design and Delivery

The 16-week pilot course comprised 16 lessons, each lasting approximately 40 minutes. The structure of each lesson followed a consistent pattern:

1) Introduction (10 minutes): Presentation of the lesson goal and demonstration of a simple concept.

2) Guided coding practice (20 minutes): Students completed block-based coding challenges on Code.org using computers or tablets and extra coding tasks.

3) Reflection and discussion (10 minutes): Students discussed solutions, errors, and strategies to reinforce computational thinking and creativity.

The intervention combined computer-based coding and unplugged activities, such as card sequencing, paper-based tasks, and movement-based problem-solving, to reinforce algorithmic reasoning without reliance on digital tools. This blended approach allowed students to grasp core programming logic through both physical and digital representations. Throughout the intervention, teachers maintained reflective notes, and researchers conducted classroom observations to record behavioral engagement and problem-solving interactions.

The 16-week educational intervention was designed to isolate, emphasize, and evaluate block-based programming skills by mapping the digital environments of Code.org Course C and Course D (2024 editions) directly into the core coding chapters of the official Mongolian 5th-grade Computer Science textbook (used only in curriculum pilot). While the textbook introduces physical computational hardware and foundational robotics elements in its introductory units, this pilot study intentionally focused exclusively on the algorithmic logic and programming modules to systematically evaluate progress in students' computational thinking over the course.

The curriculum structure was split into four interconnected 4-week thematic blocks. To ensure a blended learning approach, every theoretical programming paradigm introduced in the national textbook was immediately applied and tested through matching interactive, gamified puzzles on Code.org. The specific chronological layout, matching local textbook units, practical student block operations, and targeted computational thinking dimensions are comprehensively outlined in Table III.

TABLE III. SYSTEMATIZED 16-WEEK EXPERIMENTAL PROGRAMMING TIMELINE MAPPED TO MONGOLIAN 5TH GRADE TEXTBOOK MODULES

Phase / Timeline	Textbook Unit & Chapter Alignment	Corresponding Code.org Modules	Practical Student Programming Tasks & Core Operations	Targeted Computational Thinking (CT) Competency
Weeks 1–4	(Unit 2: Characters and Movement - Lessons 2.1, 2.2)	Course C: Lessons 1–4: Digital Citizenship & Sequencing in Maze	Students constructed explicit sequential command chains to move sprites autonomously. Handled event-driven logic by mapping interactions to physical action blocks (e.g., programming character speech outputs on clicking events).	Algorithmic Thinking & Decomposition: Breaking single overarching motion routes into individual directional vectors.
Weeks 5–8	(Unit 2: Optimization - Lesson: Sequencing & Loops)	Course C: Lessons 5–8: Loops in Laurel & Artist	Students identified geometric repetitions in design files. Programmed nested "Repeat" parameters to construct complex grid arrays and multi-sided shapes without writing redundant lines of individual code.	Pattern Recognition & Code Efficiency: Utilizing structural patterns to minimize command redundancy via loops.
Weeks 9–12	(Unit 3: Variables & Game Design - Lessons 3.1–3.3)	Course D: Lessons 9–11: Conditionals & Scores in Bee/Farmer	Students initialized dynamic parameters to track game actions (e.g., / score variables). Built condition checks ("If/Else") forcing characters to gather assets only when strict environmental variables were met.	Conditional Logic & Data Tracking: Designing rule-based branching paths and tracking fluid states via data variables.
Weeks 13–16	(Unit 4: Functions & Optimization - Lessons 4.1, 4.2)	Course D: Lessons 12–14: Functions in Minecraft & End Project	Students audited broken scripts to isolate hidden logical failures (Debugging). Wrapped complex, recurring code instructions into distinct named macro modules to assemble an original capstone project.	Abstraction and Debugging: Encapsulating complex sub-routines into clean functions; diagnosing system failures methodically.

IV. RESULTS

A. Coding Experiment Outcomes

The box plot (Fig. 2) illustrates the changes in score distribution after the intervention, emphasizing trends that extend beyond mere mean differences. In the experimental group, the post-test scores show a higher median and a reduced interquartile range, indicating enhanced consistency in learning outcomes. This trend is consistent with previous studies indicating that coding activities can improve not only individual computational thinking (CT) skills but also foster more equitable performance among students [33], [15].

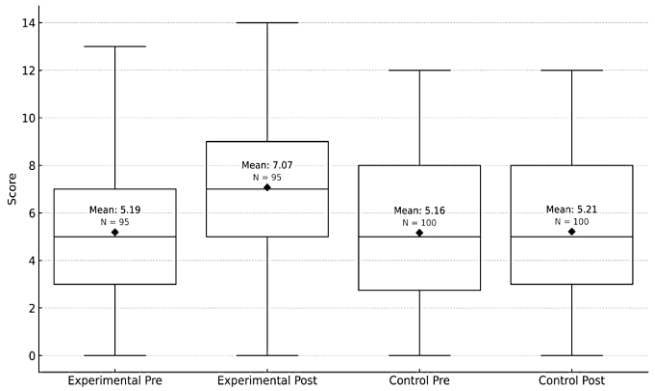

Fig. 2. Box plot of pre- and post-intervention scores in experimental and control groups.

In contrast, the control group demonstrates only a slight upward shift in the mean, with overlapping interquartile ranges both before and after the test. This suggests that general instruction alone may yield minimal or uneven learning gains in CT, consistent with studies indicating that explicit, structured computational tasks are more effective than implicit exposure [34], [35]. The absence of extreme outliers in both groups further supports the reliability of the intervention's effect. Significantly, the experimental group's decrease in variability suggests that the coding program was more effective in assisting lower-performing students - a pattern linked to well-structured CT interventions [36].

These findings support the interpretation that coding activities, when systematically implemented, lead to both higher and more consistent CT performance among primary school learners.

B. Coding Ability Progression (Hypothesis 1)

To evaluate the progression of students' coding capabilities following the 16-week intervention, quantitative tracking metrics were analyzed. This section details the statistical shifts in objective computing performance alongside subjective self-efficacy developments across both cohorts, as highlighted in the metrics below. The detailed post-intervention evaluation metrics and assessment score breakdowns are summarized in Fig. 3.

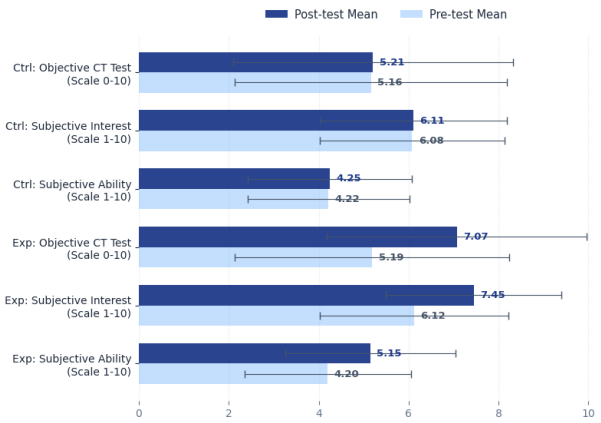

Fig. 3. Assessment score metrics

C. Coding Experiment Outcomes

Before the intervention, baseline equivalence between the groups was statistically verified. A two-sample independent t-test conducted on the pre-test scores showed a statistically insignificant difference between the experimental group (M = 5.19, SD = 2.98) and the control group (M = 5.16, SD = 3.02; t (195) = 0.07, p = 0.944), validating their baseline comparability.

Following the 16-week intervention, a paired sample t-test was conducted to examine intra-group progress from pre-test to post-test. Because zero student attritions occurred during the course, the degrees of freedom (*df*) correspond directly to n – 1 for each group. For the experimental group (n = 95), computational performance scores increased significantly from M = 5.19 (SD = 2.98) to M = 7.07 (SD = 3.14), yielding t (94) = -8.13, $p < 0.001$. The standardized effect size was computed using the paired-samples standardized mean difference formula:

$$d = \frac{M_{post} - M_{pre}}{SD_{difference}}$$

where, M_{post} and M_{pre} represent the post-test and pre-test means, respectively, and $SD_{difference}$ represents the standard deviation of the difference scores.

These findings indicate that structured block-based coding activities significantly improved students' computational thinking and coding performance. To verify the specific areas of student improvement behind these mean differences, an item-level analysis was conducted. As explicitly mapped across the curriculum topics in Fig. 4, the experimental pilot group showed a substantial accuracy advantage over the control class group, particularly in advanced block logic tasks.

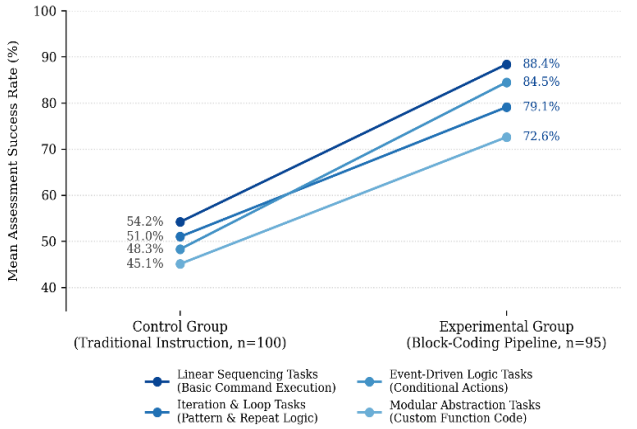

Fig. 4. Explicit performance accuracy tracking across core coding tasks, comparing the mean success rates of the Experimental Pilot Group (n = 95) and the Traditional Class Group (n = 100).

In contrast, the control group ($n_{control}$ = 100) exhibited a statistically negligible change from pre-test (M = 5.16, SD = 3.02) to post-test (M = 5.21, SD = 3.11), with t (99) = 0.74, p = 0.456 and a negligible effect size (Cohen's *d* = 0.07).

D. Programming Interest Gain Scores (Post-Test – Pre-Test)

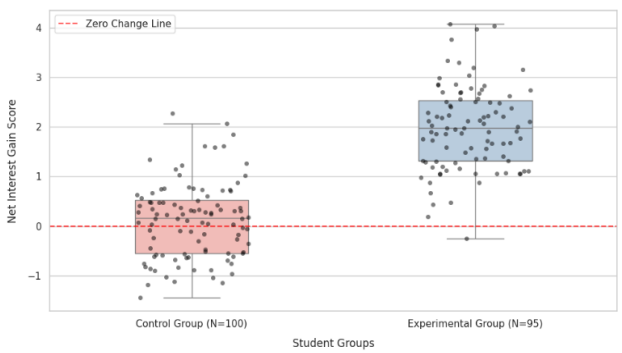

Fig. 5. Programming interest gain scores between the experimental and control groups.

To see exactly how the program affected each student, we calculated personal progress by subtracting pre-test scores from post-test scores. As shown in Fig. 5, the results are completely different. In the Control group, almost everyone's score is crowded right around the zero line, meaning regular classes did not really change their interest in coding.

In contrast, almost every student in the Experimental group shifted well above the zero line. Even the middle half of the experimental group sits completely above the control group's highest concentration. This visually confirms our strong overall effect size (Cohen's d = 0.84) and proves that the coding activities systematically increased programming interest across the entire classroom rather than shifting it via a few standout students.

E. Programming Interest and Motivation (Hypothesis 2)

Changes in student self-rated programming interest are described in the metrics below and mapped visually in Fig. 6.

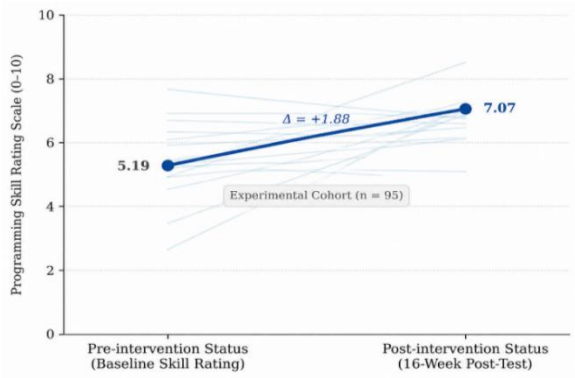

Fig. 6. Longitudinal trajectory shift mapping the baseline programming self-efficacy ratings against post-intervention evaluation metrics within the experimental cohort (n = 95).

TABLE IV. QUALITATIVE EVIDENCE OF INCREASED INTEREST IN PROGRAMMING.

Theme	Frequency	Example Student Quotes
Enjoyment of coding	11/15	"Coding is like playing a game."
Perceived creativity	9/15	"I can make easier stories and games."
Increased confidence	8/15	"I want to learn more coding after this class."

Quantitative and qualitative data jointly support Hypothesis 2. Students not only improved in self-rated programming ability (+1.88) but also developed positive attitudes toward coding, describing it as fun, creative, and motivating. The data show that students who engaged in Code.org block-based coding showed significant improvements in coding scores and increased interest in programming, as observed in interviews and classroom behavior. Qualitative responses revealed that students described coding as "fun", "creative", and "like playing games", indicating increased motivation and engagement. Together, the significant improvement in test scores and self-reported enthusiasm support Hypothesis 2. Table IV shows qualitative evidence of increased interest in programming.

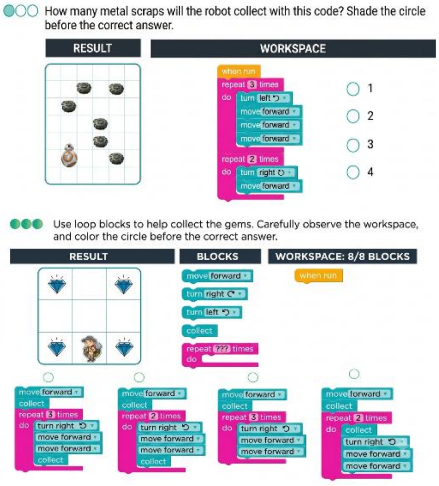

Fig. 7. Supplementary reference items from the post-intervention assessment instrument. (A) Linear grid sequencing task (Q2), (B) Iterative loop optimization task (Q9), and (C) Asynchronous event-driven matching task (Q7).

Together, the significant improvement in test scores and self-reported enthusiasm support Hypothesis 2. To visually confirm the nature of this data improvement, Fig. 7 provides supplementary reference samples of the task logic successfully parsed by the pilot cohort. The observed score improvements reflect students' growing ability to use more advanced computational thinking concepts, including loops and event-based programming structures.

F. Gender Awareness and Inclusivity (Hypothesis 3)

Although not universal, interview data indicate a shift toward gender-equitable views, providing partial support for Hypothesis 3. Female students showed increased confidence and interest, reflecting the potential of early coding instruction. Furthermore, post-intervention interviews revealed that several students recognized gender-related themes, with female students acknowledging increased confidence in coding skills and noting that "both girls and boys can be good at programming". Although initial surveys reflected limited awareness of gender-related stereotypes in computer science, the course prompted discussions about inclusivity. Teachers also reported observing a reduction in gendered perceptions of coding as an activity "more suited for boys". These findings provide partial support for Hypothesis 3: students' improved grasp of Computer Science concepts includes an emerging

awareness of gender equity and challenges to existing stereotypes, although further targeted instruction may be needed to reinforce these outcomes. Table V presents the gender-related awareness indicators (post-intervention interviews).

TABLE V. GENDER-RELATED AWARENESS INDICATORS (POST-INTERVENTION INTERVIEWS).

Indicator	Frequency (n=15 interviews)	Example Quotes
Acknowledgment that girls can code	6	"Girls can be good coders like boys."
Expression of inclusivity in coding	4	"Everyone should try coding; it's for everyone."
Recognition of existing stereotypes	2	"I thought coding was just for boys before."

V. DISCUSSION

The study demonstrates that structured, block-based coding education significantly enhances computational thinking, motivation, and further interest in programming among fifth-grade students with no prior coding experience. Quantitative results showed a large performance improvement (Cohen's d = 0.84), confirming the intervention's effectiveness and aligning with findings from [12], [13].

Qualitative responses further reveal that coding is perceived as *"fun", "creative",* and *"like playing games"*. This indicates that interactive, game-like design elements in block-based coding environments reduce cognitive barriers and foster intrinsic motivation. Such experiences transform programming from an abstract, syntax-heavy subject into an accessible and enjoyable learning process. Post-intervention discussions also revealed growing gender inclusivity, as both male and female students expressed that *"anyone can code."* Although awareness of stereotypes was not universal, the trend toward inclusivity suggests that early coding education can play a meaningful role in addressing gender disparities in STEM participation [11].

Within the context of [16], [17], these findings provide actionable evidence for integrating computational thinking and coding into the primary education curriculum. The visual, low-barrier coding tools such as Code.org can reduce urban–rural learning gaps and foster equitable digital participation.

The qualitative interview findings provide preliminary insights into student engagement and perceptions. Regarding gendered views of computing, 6 out of the 15 interviewed participants (40%) explicitly verbalized that girls and boys possess equal capability in programming tasks. While these responses indicate positive individual experiences among female students and point to potential self-confidence gains within the sample, this exploratory finding cannot be extrapolated as a sweeping attitudinal shift across the wider student population. Because baseline gender-role stereotype measures were not quantitatively tracked from the outset, these expressions are interpreted purely as emerging individual awareness among the interviewees. This highlights a critical area that requires dedicated, structured tracking in future longitudinal research.

Therefore, future research should build on this pilot by including larger, more diverse samples, longitudinal data, and teacher training components. More investigation into socio-cultural and contextual factors - such as parental influence, language barriers, and teacher digital competence - will enhance the scalability and sustainability of computer education reform.

VI. CONCLUSION

This study demonstrates that the strategic integration of block-based coding activities into primary education effectively develops foundational computational thinking, operational problem-solving, and a heightened interest in computer programming among young learners with no prior experience. Over the course of a 16-week intervention utilizing Code.org, fifth-grade students in Mongolia demonstrated statistically significant and practically substantial gains in objective coding metrics, while concurrently shifting their subjective perceptions of computer science from an abstract, syntax-heavy domain into an accessible, creative endeavor. Furthermore, the study provides preliminary evidence indicating that early exposure can foster individual awareness of gender equity in technology, helping minimize stereotypical biases before they solidify in later academic stages. Ultimately, providing structured, low-barrier visual programming pathways within primary school frameworks serves as a practical sandbox for building equitable digital competencies and offers localized empirical evidence to support ongoing nationwide computer science curriculum reforms.

However, while some promising awareness regarding gender emerged post-intervention, the results suggest that brief interventions are not a cure-all; additional targeted curricular measures are necessary to enforce long-term gender-equitable attitudes. To address the limitations of this pilot study, future research should recruit a larger, more diverse student sample spanning multiple regional backgrounds to enhance generalizability. Additionally, despite high modern exposure to artificial intelligence (AI) technologies, further longitudinal tracking is recommended to evaluate how student attitudes, computational thinking retention, and intrinsic curiosity evolve alongside national educational policies.

ACKNOWLEDGMENT

The authors wish to thank all the students, teachers, parents, and training managers who gave consent and participated in the study. The authors also thank the teachers who shared their feedback and suggestions for improving the curriculum content and inclusivity.

ETHICAL CONSIDERATIONS

The study involved minor participants (fifth-grade students). The study protocol was approved by the Research Committee of the Mongolian National Institute for Educational Research. All procedures performed were conducted in accordance with the ethical standards of the institutional research committee and the national guidelines for educational research involving minors. Written informed consent was obtained from the school principals and the students' parents or legal guardians. Additionally, verbal assent was secured before administering assessments, observations, or interviews. All

student data were completely anonymized using unique numeric identifiers, and participants were informed of their right to withdraw from the activities at any stage without academic penalty.

REFERENCES

[1] P. Curzon, J. Black, L. R. Meagher, and P. McOwan, "cs4fn. org: Enthusing students about computer science," Proceedings of Informatics Education Europe IV, pp. 73–80, 2009.

[2] F. Cunha and J. Heckman, "The technology of skill formation," *American Economic Review*, vol. 97, no. 2, pp. 31–47, 2007.

[3] J. Heckman and D. Masterov, "The productivity argument for investing in young children," *Review of Agricultural Economics*, vol. 29, no. 3, pp. 446–493, 2007.

[4] S. Popat and L. Starkey, "Learning to code or coding to learn? A systematic review," *Computers & Education*, vol. 128, pp. 365–376, 2019.

[5] J. M. Wing, "Computational thinking," Communications of the ACM, vol. 49, no. 3, pp. 33–35, Mar. 2006.

[6] P. J. Denning, "The profession of IT: The problem of computation," *Communications of the ACM*, vol. 52, no. 9, pp. 132–141, 2009.

[7] M.U. Bers, Coding as a Playground: Programming and Computational Thinking in the Early Childhood Classroom, New York, NY, USA: Routledge, 2018.

[8] M. Resnick, Lifelong Kindergarten: Cultivating Creativity through Projects, Passion, Peers, and Play. Cambridge, MA, USA: MIT Press, 2017.

[9] R. Davies, "Gender and computing: Understanding female students' attitudes and self-efficacy in programming," *Computers & Education*, vol. 50, no. 2, pp. 430–440, 2008.

[10] H. Haddad and F. Kalaani, "The predictive role of computational thinking skills on academic performance," *Education and Information Technologies*, vol. 20, no. 4, pp. 567–578, 2015.

[11] S. Papadakis, "Apps to promote computational thinking and coding skills to young-age children: A pedagogical challenge for the 21st century learners," *Educational Process: International Journal*, vol. 11, no. 1, pp. 7–13, 2022. doi: 10.22521/edupij.2022.111.1.

[12] Ş. Metin, M. Başaran, and D. Kalyenci, "Examining coding skills of five-year-old children," *Pedagogical Research*, vol. 8, no. 2, Article em0154, 2023.

[13] A. Strawhacker and M. U. Bers, "What they learn when they learn coding: Investigating cognitive domains and computer programming knowledge in young children," *Educational Technology Research & Development*, vol. 67, no. 3, pp. 541–575, Jun. 2019.

[14] C. Tofel-Grehl, K. A. Searle, and B. L. MacDonald, "Lighting up history: Integrating mathematics and computational thinking in the science classroom," *Science Scope*, vol. 44, no. 5, pp. 64-71, 2021.

[15] K. A. Mills, J. Cope, L. Scholes, and L. Rowe, "Coding and computational thinking across the curriculum: A review of educational outcomes," *Review of Educational Research*, vol. 94, 2024. DOI: 10.3102/00346543241241327.

[16] Government of Mongolia, *Vision-2050 Long-Term Development Policy of Mongolia*, Ulaanbaatar, Mongolia, 2020.

[17] Ministry of Education and Science of Mongolia, *Education Sector Medium-Term Development Plan (2021–2030)*, Ulaanbaatar, Mongolia, Dec. 2020.

[18] P. Van Roy and S. Haridi, *Concepts, Techniques, and Models of Computer Programming*, Cambridge, MA, USA: MIT Press, 2004.

[19] D. H. Clements, "Curriculum research: Toward a framework for examining early mathematics learning," Journal for Research in Mathematics Education, vol. 33, no. 2, pp. 111–136, 2002.

[20] F. Wang, Y. Li, and X. Zhang, "The impact of coding on students' creative thinking and problem-solving skills," *Computers & Education*, vol. 170, p. 104220, 2021.

[21] G. Futschek and R. Moschitz, "Algorithmic thinking: The key for understanding computer science," in *Proc. 5th Baltic Sea Conf. Comput. Educ. Res.*, Koli, Finland, 2010, pp. 159–168. doi: 10.1145/1867651.1867677.

[22] M. U. Bers, Blocks to Robots: Learning with Technology in the Early Childhood Classroom, New York, NY, USA: Teachers College Press, 2008.

[23] J. Voogt, N. Fisser, R. Good, P. Mishra, and K. Yadav, "Computational thinking in compulsory education: Towards an agenda for research and practice," *Education and Information Technologies*, vol. 20, no. 4, pp. 715–728, 2015.

[24] F. Kalelioğlu and O. Keskinkılıç, "A study on the effects of unplugged programming activities on computational thinking skills of young learners," *Journal of Educational Computing Research*, vol. 55, no. 7, pp. 909–933, 2017.

[25] F. Kalelioğlu, "The effects of visual programming environments on young learners' computational thinking skills," *Computers & Education*, vol. 87, pp. 1–12, 2015.

[26] S. Zhang and J. Nouri, "Teaching programming in primary schools: Effectiveness of graphical programming environments," *Education and Information Technologies*, vol. 24, no. 2, pp. 1231–1249, 2019.

[27] M. Scardamalia and C. Bereiter, Higher Levels of Knowledge: Teachers' Roles in a Knowledge-Building Community, New York, NY, USA: Routledge, 1992.

[28] J. Chesson, R. McBride, and T. Cartwright, "Integrated mode processing in computer programming: Implications for software design education," *Journal of Educational Computing Research*, vol. 8, no. 3, pp. 323–338, 1992.

[29] D. H. Clements and B. K. Nastasi, "Supporting creativity in computer-based learning environments," *Journal of Educational Computing Research*, vol. 8, no. 4, pp. 451–468, 1992.

[30] V. Barr, C. Harrison, and L. Conery, "Computational thinking: A digital age skill for everyone," *Learning & Leading with Technology*, vol. 38, no. 6, pp. 20–23, 2011.

[31] A. Ioannidou, J. Sampson, and M. Karagiannidis, "Computational thinking: Educational perspectives and practice," *ACM SIGCSE Bulletin*, vol. 43, no. 1, pp. 322–326, 2011.

[32] J. W. Creswell and V. L. Plano Clark, *Designing and Conducting Mixed Methods Research*, 3rd ed., Thousand Oaks, CA, USA: SAGE Publications, 2017.

[33] S. Grover and R. Pea, "Computational thinking in K–12: A review of the state of the field," *Educational Researcher*, vol. 42, no. 1, pp. 38–43, 2013.

[34] A. Fessakis, K. Gousiou, and P. Matsinas, "Children's computational thinking and programming — Unplugged and plugged-in activities," *Computers & Education*, vol. 69, pp. 418-427, 2013.

[35] T. Soffer and A. Cohen, "Students' engagement characteristics predict success and completion of online courses," *Journal of Computer Assisted Learning*, vol. 35, no. 3, pp. 378–389, 2019. DOI: 10.1111/jcal.12340.

[36] Á. Sáez-López, E., Aguaded, and A. Martín-Márquez, "Computational thinking interventions in K–12 classrooms: Effects on student learning and equity," Computers in Human Behavior, vol. 107, pp. 105637, 2020.