Understanding Block-based Code with Preservice Mathematics Teachers

Elvia R. Ruiz Ledezma[1], Fermín Acosta Magallanes[2] and Alma R. Villagómez Zavala[3]

[1] *Instituto Politécnico Nacional, CECyT 11, Av. de los maestros 217, Casco de Santo Tomás, Mexico City, Mexico*
[2] *Instituto Politécnico Nacional, UPIITA, Av. Instituto Politécnico Nacional 2580, Ticomán, Mexico City, Mexico*
[3] *Escuela Normal Superior de México, Hacienda de Sotelo 201, El Rosario, Mexico City, Mexico*

Abstract

The present study shows the difficulties and achievements presented by a group of students in the fifth semester of the degree in Teaching and Learning of Mathematics in Secondary Education of the Escuela Normal Superior de México, in Mexico City, on the final design project. of digital materials with programming in Scratch, the analysis was carried out within the framework of computational thinking for science in its cognitive processes and the proposed scientific activity.

Keywords

Computational thinking, Scratch, STEM

1. Introduction

Education in science, technology, engineering, and mathematics (STEM) has become an increasingly important educational perspective, gaining great attention around the world [1], with the objective: to train new talents with 21st century skills, developing the computational, critical, and creative thinking [2]. Therefore, coding has been proposed as an integral part of STEM Education, allowing interdisciplinary connections in relation to computational thinking (CT) to address problems in everyday life and face challenges. Thus block-based coding languages, such as Scratch, have become popular due to the use of drag code commands, simplifying the text syntax used in other programs [3]. However, in a first approach, subjects have difficulty processing nested structures, that is, placing one control structure inside another, as well as algorithm sequences and the interaction between blocks of code, because not all students They are at the same level of handling a computer equipment.

Our work aims to show the difficulties and achievements presented by a group of students in the 5th semester of the bachelor's degree in teaching and Learning of Mathematics in Secondary Education of the Escuela Normal Superior de Mexico, when covering the study program "Mathematics in Science and Technology" (MST), having as its final project the design of digital materials with programming in Scratch.

This paper is divided into five sections in addition to this space where we show: the theoretical perspective, the interpretation of the study program, the methodological process, the most relevant findings, conclusions, and references.

CITIE 2023: International Congress on Education and Technology in Sciences, December 04–06, 2023, Zacatecas, Mexico

ruizelvia@hotmail.com (E.R. Ruiz); ferminacosta66@hotmail.com (F. Acosta); amyy_0214@hotmail.com (A. R. Villagómez)

0000-0001-9528-4146 (E. R. Ruiz); 0000-0003-1471-5376 (F. Acosta); 0000-0003-0021-8977 (A. R. Villagómez)

2. Theoretical framework

The cognitive processes employed in interactions with computational tools are an important aspect of computational thinking as originally conceptualized by Papert and are crucial to CT-S. That is, preparing students for the increasingly computational nature of science. It is essential to develop students' abilities to think about the functionality and position of computational tools within an activity [4].

Cuny, Snyder and Wing, defined [5], "Computational thinking is the thinking processes involved in formulating problems and their solutions, so that the solutions are represented in a form that can be carried out effectively by an information processing agent" (p. 1).

2.1. Computational thinking for science framework

Our research through the conceptual framework that describes CT-S takes an evidence-centered approach, a model of cognition [6], to identify the types of cognitive processes that are already being leveraged in classrooms that reflect CT-S. Such a framework will allow us to (a) delineate subconstructs that specify the cognitive processes characteristic of CT-S and (b) operationalize the subconstructs of CT-S to develop performance tasks that can elicit CT-S. That is, beyond knowing what activities are likely to engage students in CT-S, a testable model of how those practices engage students in CT-S is necessary. Additionally, identifying which of those activities' students are likely to participate in.

The framework is a table of four rows and three columns, which creates 12 cells (Table 1). The rows represent four categories of science activity (data collection, data processing, modeling, and problem solving) where computational tools are likely to be leveraged in science learning. The columns represent three interactions with computational tools (Reflective use, Design, and Evaluation of computational tools) that involve the cognitive processes characteristic of computational thinking [4]. Each cell within the frame represents the CT-S as the intersection of a row with a column. That is, every time a person participates in a science learning experience that can be categorized by one or more of the cells of the framework, they are engaging in computational thinking for science.

Table 1
The computational thinking for science framework

	Cognitive Processes		
Science Activity	Reflective Use	Design	Evaluation
Data Collection	How can symbology be used to design the diagram for effective coding?	How can diagram instructions be translated with block structures?	The pseudocode allows for efficient block allocation. Under what conditions does it work or under what conditions does it fail?
Data Processing	What software features can help identify the relationship between scripts?	What would be needed from software to help find patterns related to specialized commands?	The software is being used to create a visualization of structures. What are the possibilities or limitations that help communicate them?
Modeling	How can you use the three sections of the software to predict the actions in it?	What rules need to be included in the organization of sequences?	Which aspects of this digital model accurately reflect the selection, and which do not?

Problem-Solving	How can screen reading software be used to ensure that the procedure is communicated correctly?	How can you create an algorithm to evaluate the claims and reasoning of the respective arguments?	How should I test it and how will I know I've tested it enough?

3. Interpretation of the study program

The MST program aims to ensure that heuristic experiences foster creativity and ingenuity, with an approach where digital technology is closely linked to science and favors the creation of development projects. of materials and activities that contribute to the generation of mathematical knowledge, the strengthening of mathematical skills and thus the development of professional and disciplinary competencies.

As well as the development of creativity and technological innovation skills, both in the management of electronic devices and their programming. The contents of the MST program are organized into two blocks. In one block, it is expected to recover the historical understanding of the emergence of mathematics as a response to a need for communication, systematization, and modeling. In two block the link between science, mathematics and technology is emphasized, from Scratch programming and the development of didactic, technological, and digital materials.

4. The methodological process

Eleven normal students from the fifth semester of the bachelor's degree in teaching and Learning of Mathematics in Secondary Education in Mexico City participated (Figure 1).

Our research is a descriptive explanatory work that describes structured situations that provide a sense of understanding of the phenomenon referred to [7].

Three phases were considered, according to the CT-S framework, adapted to the use of programming in Scratch.

- Reflective use of programming.
- Designing with a programming language.
- Evaluating programming with a computational tool.

4.1. Reflective use of programming

We started with the design of the algorithm, using a structured programming technique that also served for programming, helping to document the programs. There are three ways to represent algorithms: written, graphical and auxiliary. The MST study program proposes the graphic form of diagrams. To design them, certain symbols or figures are used that represent an action within the procedure. The symbols are joined with arrows called flow lines that indicate the order in which the steps must be executed and the pseudocodes that are the descriptions or instructions of the flow chart. For programs, variables are needed, which can be numeric or characters.

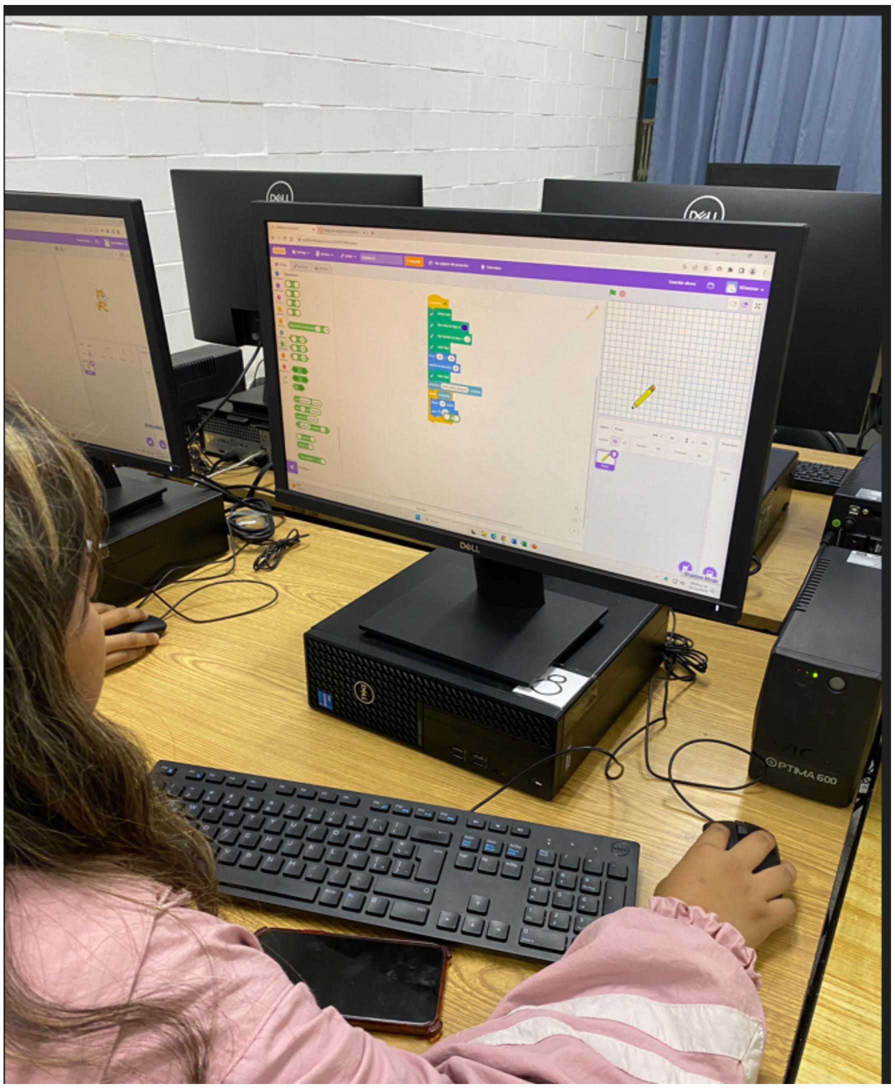


Figure 1. Students working on their project

4.2. Designing with a programming language

Learning Scratch offers students in training great possibilities due to its applications, it allows them to promote creativity and thinking skills, learning basic concepts of computing and mathematics, as well as promoting interpersonal communication and a sense of collaboration. This program is extremely intuitive, without requiring memorization of commands for its mastery. It is available for various operating systems. The Scratch screen is divided into three sections. On the right the commands that make up the routines of this language are added one by one, in the center the commands that each of the ten categories of actions have been presented and on the left the block of commands is executed through a click. Actions are linked by means of labels or blocks. In this phase, the student began his design with the use of a flowchart and pseudocode since it is crucial to plan and visualize the flow of actions in Scratch programming and represent the logical sequences of commands to be executed in a concrete way. Basic exercises were then carried out to help in understanding the use of the blocks, thus the difficulty was increased according to their progress.

4.3. Evaluating programming with a computational tool

We included the generation of educational programs and considered that the project was finished, that it had an optimized design, that it had been completely developed and that the minimum functionality tests were covered with the analysis of system requirements, the evaluation was based on rubrics (Table 2).

4.4. The application scenario

Every time a subject participates in a science learning experience that can be categorized by one or more of the cells of the framework, he or she is engaging in computational thinking for science. For each cell in Table 1, a question is provided that a subject would likely need to engage in CT-S to answer successfully. As an example, the student who answers the question in the upper left corner of the frame will need to engage in the reflective use of programming tools: diagram and pseudocode, to work toward data collection. Initially you will use the elements of the diagram and the specific variables in the approach to a problem. You can do this by participating in the reflective use of the diagram while interacting with its elements, until you form a mental model of the functionality of programming. As the student continues to interact with this tool, their discoveries reinforce, revise, or complement their developing mental model. Once the student has a working mental model, they can use it toward the use of commands contained in Scratch's categories of actions, linked by blocks. Design can occur throughout an iterative creation process in which the subject has to repeatedly update and modify his or her mental model of the functionality of the computational tool in relation to a sequence of commands in each number of cycles.

To participate in the assessment, students must know what the digital model should do in different settings to determine if it is a complete and accurate model. To do this, they investigate of their efficiency running the program. Once this evaluation was completed, students would be able to determine how well it works. Therefore, each student will have built a mental model of the possibilities and limitations of the functionality of the computational tool and how it could be used in their activity, so students are involved in CT-S.

In phase one for the reflective use of a computational tool in programming with Scratch, the evidence shown by the students in relation to the representation of the algorithms in graphic and written format with flowcharts and pseudocode was reviewed (Figure 2), under the criteria that were considered, namely, the problem statement and the project design by 30% and 70% respectively.

For phase two, the students, using their own criteria, manage to solve the rules, using the program's sentences. At this stage, the coding of the project logic implemented in Scratch is prioritized because it reflects the application of the knowledge provided by the class work. In this phase, documentation and integration formed another segment to evaluate and provide feedback, 65% of the group of students obtained excellent performance and 60% improved compared to the previous phase, demonstrating that the feedback allowed them to improve their performance.

In phase three, the implementation of the system, where prior knowledge and the user interface are involved, the results indicate that 11.2% of the students did not complete the project, while 88.8% achieved a finished project with a good design, development and tested (Figure 3). Thus, 80% of the group managed to integrate prior knowledge and adequately complete the program in contrast to the 20% who had problems integrating the information.

Table 2
Rubric established to evaluate the development of the project based on the CT-C framework

Aspects/ %assigned	Excellent	Good	Regular	Insufficient
Data Collection 20%	(20-16%) Can use symbology in the design of the diagram, translating the instructions into	(15-11%) Can use symbology in the design of the diagram, translating the	(10-6%) Can use symbology in the design of the diagram without translating the	(5-0%) Cannot use symbology in diagram design.

	block structures and modifying them according to the needs raised.	instructions into block structures.	instructions into block structures.	
Data Processing	(20-16%)	(15-11%)	(10-6%)	(5-0%)
20%	Can identify the relationship between specialized scripts to find related patterns and modify them according to your needs.	Can identify the relationship between specialized scripts to find related patterns.	Can identify the relationship between specialized scripts without finding related patterns	Cannot identify the relationship between specialized scripts.
Modeling	(30-21%)	(20-16%)	(15-11%)	(10-0%)
30%	Can use all three sections of the software, predicting the actions in it with the use of rules for sequential organization and modifying according to the stated needs.	Can use all three sections of the software, predicting the actions in it with the use of rules for sequential organization.	Can use all three sections of the software, predicting actions in it without considering the use of rules for sequential organization	Cannot use all three sections of the software, predicting the actions in it.
Problem-Solving	(30-21%)	(20-16%)	(15-11%)	(10-0%)
30%	Can use screen reading software to ensure you communicate the procedure correctly, creating an algorithm to evaluate the respective statements and reasoning, modifying according to the needs raised.	Can use screen reading software to ensure you communicate the procedure correctly, creating an algorithm to evaluate the respective statements and reasoning.	Can use screen reading software to ensure you communicate the procedure correctly, without creating an algorithm to evaluate the respective statements and reasoning.	Cannot use screen reading software to ensure you communicate the procedure correctly.

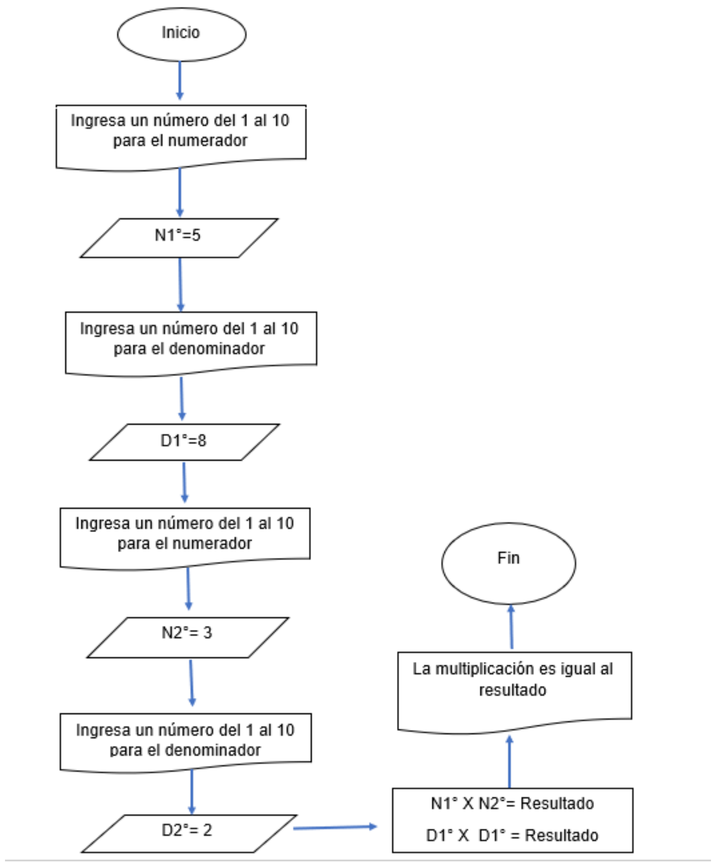
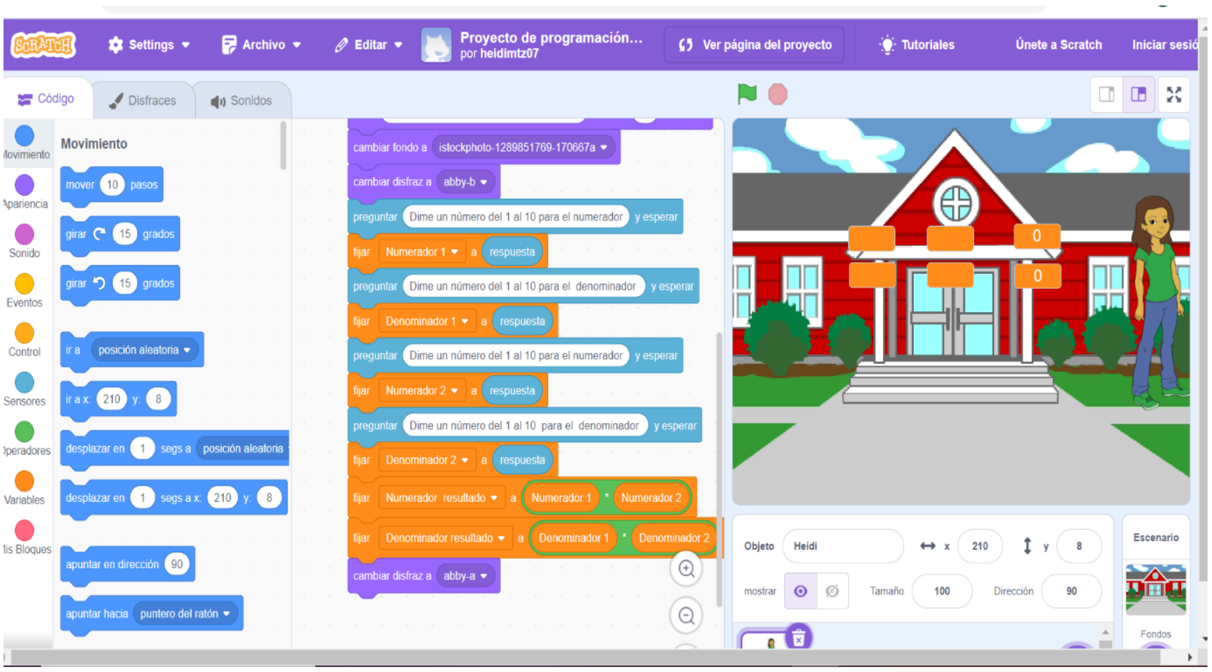

Inicio
Ingresa un número del 1 al 10 para el numerador
N1°=5
Ingresa un número del 1 al 10 para el denominador
D1°=8
Ingresa un número del 1 al 10 para el numerador
N2°= 3
Ingresa un número del 1 al 10 para el denominador
D2°= 2
N1° X N2°= Resultado
D1° X D1° = Resultado
La multiplicación es igual al resultado
Fin

Figure 2. Example of a program in graphic form, prepared by a student



```
cambiar fondo a istockphoto-1289851769-170667a
cambiar disfraz a abby-b
preguntar Dime un número del 1 al 10 para el numerador y esperar
fijar Numerador 1 a respuesta
preguntar Dime un número del 1 al 10 para el  denominador y esperar
fijar Denominador 1 a respuesta
preguntar Dime un número del 1 al 10 para el numerador y esperar
fijar Numerador 2 a respuesta
preguntar Dime un número del 1 al 10  para el  denominador y esperar
fijar Denominador 2 a respuesta
fijar Numerador  resultado a Numerador 1 * Numerador 2
fijar Denominador resultado a Denominador 1 * Denominador 2
cambiar disfraz a abby-a
```

Figure 3. Example of a program carried out by one of the students

5. Discussion

The students began their reflective use, manipulating the commands of the Scratch program. By pressing certain blocks and observing the results of their actions, they subsequently reflected on their manipulations and began to form a mental model of the functionality of this language. As the students continued to interact, their discoveries reinforced, revised, or complemented their developing mental model. We were able to observe that reflective use is bidirectional in the transfer of information. Once the student had a mental model, he was able to use language more intentionally to process the algorithms previously contained in the flowchart, so that he could analyze to learn how to translate them.

Reflective use can also occur when the student begins his activity with an incomplete or inaccurate mental model of the functionality of a program, allowing him to re-investigate and modify his mental model already with the use of the programming language, testing with the computational tool, choosing the commands that lead you to the design of your program.

6. Conclusions

The CT-S model and its cognitive framework allowed us to review the bidirectional interactions between the student and the programming design of the prepared materials.

In phase two, one of the problems that arose is that not everyone has a good understanding of mathematical logic to be able to visualize the logical sequences of the algorithm.

The most complicated procedures to teach were loops, conditionals, and manipulation between variables, because they are abstract procedures.

Using diagrams is crucial to plan and visualize the flow of actions in Scratch programming, these represent the logical sequences of commands to be executed in a concrete way to understand.

At first, it is common to make mistakes such as not connecting blocks correctly, forgetting the loop blocks or necessary conditions, not fully understanding how Scratch variables and events work.

It is important to start with basic exercises that help understand the use of the blocks and increase the difficulty according to the progress of the group.

References

[1] Denning, P. J. Computational thinking in science. American Scientist, 2017, (1), 13–17. https://doi.org/10.1511/2017.124.13

[2] Guzdial, M. Learner-centered design of computing education: Research on computing for everyone. Synthesis Lectures on Human-Centered Informatics, 2015, 8(6), 1–165. https://doi.org/10.2200/S00684ED1V01Y201511HCI033

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

[4] Hurt, T., Greenwald, E., Allan, S. The computational thinking for science (CT-S) framework: operationalizing CT-S for K–12science education researchers and educators. 2023. https://doi.org/10.1186/s40594-022-00391-7

[5] Cuny, J., Snyder, L., & Wing, J. M. Demystifying computational thinking for non-computer scientists. 2010, Unpublished manuscript referenced in https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdfD.

[6] Brown, N. J., & Wilson, M. A model of cognition: The missing cornerstone of assessment. Educational Psychology Review, 2011, 23(2), 221–234. https://doi.org/10.1007/s10648-011-9161-z

[7] Hernández, R. Fernández, C., y Baptista, L. Fundamentos de metodología de la investigación. 2007. México: McGraw-Hill.