Co-funded by the European Union

Algorithms for Drawing

María Zapata Cáceres, Estefanía Martín Barroso

Universidad Rey Juan Carlos

maria.zapata@urjc.es, estefania.martin@urjc.es

Summary

This educational activity is designed for Primary Education students and introduces the concept of algorithms through the step-by-step construction of instructions to recreate a drawing. In a face-to-face classroom environment without electronic devices, students work in small groups to write sequences of instructions using natural language with certain constraints.

Each group receives a drawing that the other groups cannot see and must write the necessary steps for another group to reproduce it. They then exchange the instructions and simulate their execution: one student acts as the "computer" and another as the "compiler", following the instructions to the letter. Errors are analyzed, and students reflect on the importance of precision in writing algorithms.

The main objective of the activity is to improve understanding of instruction sequencing, error detection and correction, and the development of Computational Thinking skills. In addition, it promotes collaborative learning and effective communication, helping students structure logical thinking in an accessible and playful way.

Introduction

Computational Thinking (CT) is an essential skill in today's education—not only for programming but also for developing logical reasoning, problem-solving, and structured thinking (Wing, 2006). One of the foundations of CT is the ability to decompose problems and design solutions in the form of algorithms, that is, ordered sets of steps to solve a specific task.

Early childhood and primary education are key stages to introduce these concepts in an intuitive and playful way, allowing students to develop an early understanding of programming logic without the need for electronic devices or advanced programming languages.

Numerous studies have shown that unplugged approaches—activities conducted without computers or other devices—facilitate the understanding of fundamental computer science concepts (Resnick et al., 2009; Tedre & Denning, 2016). Thus, current approaches to CT education highlight the importance of integrating manipulative and collaborative activities into Primary Education (Grover & Pea, 2013). As Resnick et al. (2009) indicate, unplugged learning makes computational concepts more accessible by removing the technological barrier, thereby fostering a more intuitive understanding of algorithms and error debugging.

The activity proposed in this guide follows this methodological line, using a hands-on approach. By writing precise instructions for others to reproduce a drawing, students internalize key programming concepts such as sequential logic, the importance of syntax, and error correction during algorithm execution.

The operational framework by Brennan & Resnick (2012) identifies three core dimensions of CT: computational concepts (such as sequencing and iteration), computational practices (including debugging and abstraction), and computational perspectives (such as expression and connection with other learning). These elements are embedded in the activity, as students must structure clear instructions, debug their algorithms, and reflect on their effectiveness during execution.

Moreover, the activity fosters collaborative work, as students must communicate effectively to write understandable instructions and to identify and correct possible errors in the algorithms. This approach helps develop transversal skills such as problem-solving, logical thinking, and communication, which are essential in any academic discipline and in the workplace.

Activity Design

This educational activity is aimed at students in early childhood and primary education within a face-to-face classroom setting. It is structured in several phases that progressively explore the concept of algorithms and the importance of precision in writing instructions. The activity includes the following phases:

- **Phase 1. Preliminary explanation:** The activity begins with a short introduction to what an algorithm is and how it appears in everyday life. Simple examples are presented, such as tying shoelaces or making a sandwich, to highlight the importance of sequencing steps.
- **Phase 2. Writing the algorithm:** Students are divided into groups of two to four. Each team receives a drawing (see Figure 1) that the other groups cannot see and is tasked with writing a detailed set of instructions so that another group can reproduce the drawing without seeing it. To ensure algorithm clarity, restrictions are placed on the language used, defining allowed and forbidden words. Additionally, depending on the educational level, algorithmic structures such as sequence, selection (decision), or iteration (loop or repetition) may be introduced.
- **Phase 3. Executing the algorithm:** Once the groups have written their algorithms, they exchange them and begin the execution phase. One student plays the role of "computer" and another the "compiler." The compiler reads the instructions aloud, while the computer follows them literally to recreate the drawing. During execution, the rest of the group observes and analyzes possible errors or ambiguities in the instructions. At higher levels, students may be asked to identify different types of errors.
- **Phase 4. Algorithm evaluation:** After the execution, the recreated drawings are compared with the originals, and a guided discussion follows in which students reflect on the accuracy of the algorithms and the errors encountered. They are encouraged to identify improvements and rewrite the instructions to make them clearer and more effective.

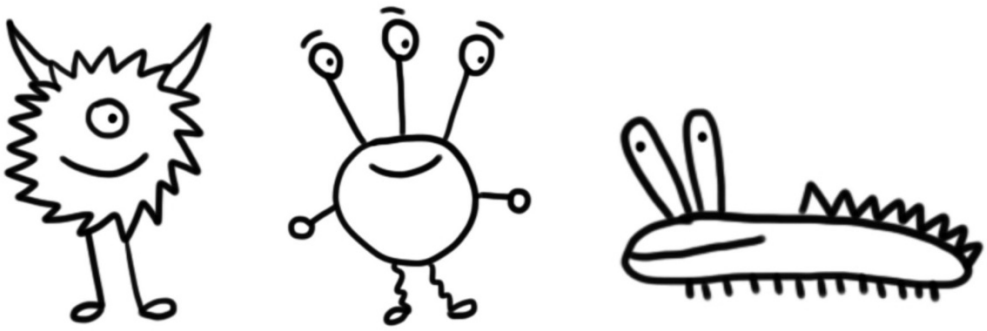

Figure 1. Sample drawings for the activity

Implementation of the Activity

The activity was carried out in three 4th-grade primary school classes at CEIP Pedro Duque (Madrid, Spain), with a total of 70 students participating in groups of up to four. Each session lasted one hour per class and took place during the week of February 11th, to commemorate the International Day of Women and Girls in Science.

Figure 2. Phase 2 of the activity: Writing the algorithm.

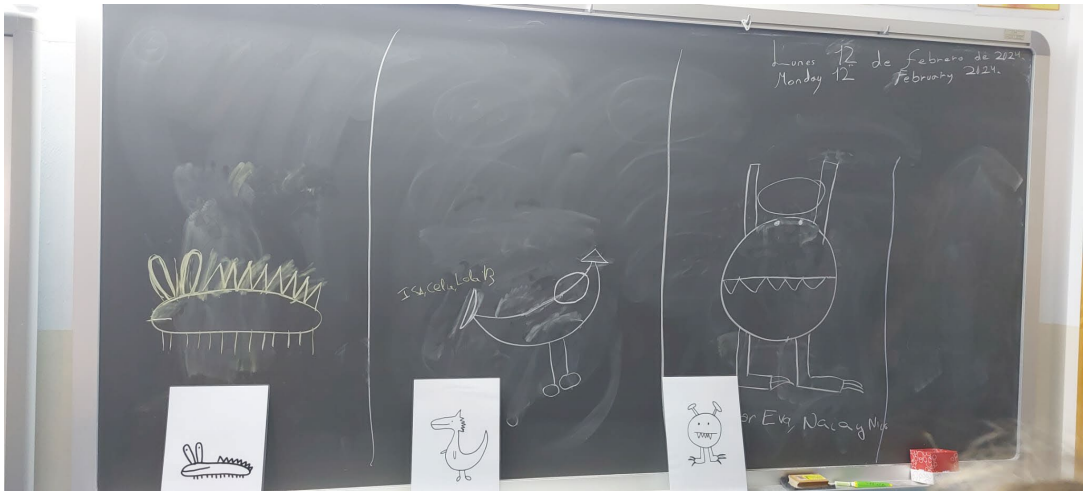

Figure 3. Result of Phase 3: Executing the algorithm and comparing the reproduced drawings with the original. This step leads into Phase 4: Evaluation.

Results

The outcomes of the activity demonstrated that students were capable of structuring instructions in a sequential and precise manner. As the activity progressed, improvements were observed in the formulation of algorithms and in the ability to identify and fix errors.

Students showed a high level of engagement and collaboration, communicating effectively with their peers to construct clear and understandable instructions. During the execution phase, difficulties in interpreting imprecise instructions became evident, prompting valuable reflections on the importance of clarity when writing algorithms.

Additionally, the activity contributed to the development of critical thinking, as students analyzed and corrected their algorithms autonomously by comparing the resulting drawings with the original ones. Younger students, in particular, encountered challenges in structuring detailed instructions, but with peer support, they succeeded in improving their algorithms through successive iterations.

From a computational thinking perspective, this activity explicitly addressed core skills such as algorithm design and evaluation, which are essential in both programming and problem-solving. Through writing and executing algorithms, students developed an

intuitive understanding of the importance of order and precision in sequencing instructions.

Furthermore, the activity fostered key CT skills such as **generalization**, as students identified common patterns in algorithm construction and applied similar principles to different drawings. **Problem decomposition** was also practiced, requiring students to break down the drawing process into specific steps that others could follow systematically. Finally, **abstraction** was promoted by encouraging students to simplify visual representations into a set of comprehensible instructions.

This practical and collaborative approach helped students internalize fundamental programming concepts without the need for electronic devices, enhancing their ability to structure logical solutions and progressively debug errors.

Conclusions

The activity *"Drawing Algorithms"* has proven to be an effective strategy for introducing Computational Thinking (CT) in early childhood and primary education. Through a hands-on and collaborative methodology, students developed key skills in structuring instructions, identifying errors, and communicating effectively.

The unplugged approach allowed students to grasp basic programming principles without the need for electronic devices, facilitating intuitive and meaningful learning. The playful and participatory nature of the activity motivated students to explore abstract concepts in a practical way, fostering their interest in logical and computational thinking.

In future implementations, progressive levels of difficulty could be incorporated, or interdisciplinary themes could be integrated—for example, using algorithms to solve math problems or tell interactive stories. Overall, the activity provides a solid foundation for teaching programming at early ages and for developing essential skills in the digital age.

Funding

The materials used for this activity were funded by the Erasmus+ CoTEDI project, which is financed by the European Union under Key Action 2023-1-NL01-KA220-SCH-000152037 – OID E10207981.

Acknowledgements

We would like to thank the management team and the fourth-grade teachers of CEIP Pedro Duque for their involvement and support in carrying out this activity.

Referencias bibliográficas

Brennan, K. & Resnick, M. (2012) New frameworks for studying computational thinking. Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vol. 1, Vancouver, 13-17 April 2012, pp. 1-25. Disponible en: https://bit.ly/4bzeVe1.

Grover, S. & Pea, R. (2013) Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42(1), pp. 38-43. DOI: 10.3102/0013189X12463051.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … & Kafai, Y. (2009). Scratch: Programming for all. Communications of the ACM, 52(11), pp. 60-67. DOI: 10.1145/1592761.1592779.

Tedre, M. & Denning, P. J. (2016) The long quest for computational thinking. Proceedings of the 16th Koli Calling International Conference on Computing Education Research, pp. 120-129. DOI: 10.1145/2999541.2999542.

Wing, J. M. (2006) Computational thinking. Communications of the ACM, 49(3), pp. 33-35. DOI: 10.1145/1118178.1118215.

CoTEDI – Activity sheet

Title	Algorithms for Drawing
Author	María Zapata Cáceres y Estefanía Martín Barroso
Educational Center	CEIP Pedro Duque
Status	Completed
Start Date	February 12, 2024
End Date	February 12, 2024
Target Group	
Target Age Range	9-10 year-olds
Educational Level	4th Grade, Primary Education
Number of Students Involved	70
Educational Context	In-person class
Diversity	Students of all genders participated
Resources Required	
WiFi Connection	No
Devices	No
Tangible materials	▪ Paper sheets ▪ Pencils or markers ▪ Blackboard or digital board ▪ Pre-designed drawings for the activity
Descripción de la actividad	
Activity Description	Students learn about algorithm logic by writing sequences of instructions to recreate a drawing. The activity takes place in a non-digital classroom setting,

following an unplugged approach to facilitate understanding of key CT concepts.

Phases of the activity:

Phase 1. Initial Explanation:

- Brief introduction on what an algorithm is and its presence in daily life. Simple examples are provided.
- Objectives:
  - Explain the importance of sequencing instructions for problem solving.
  - Relate algorithms to daily tasks.

Phase 2. Writing the Algorithm:

- Students are divided into groups of 2 to 4. Each team receives a drawing that others cannot see and must write a set of detailed instructions for another group to reproduce it without seeing it.
- Language restrictions are set, defining allowed and forbidden words. Depending on the educational level, algorithmic structures such as sequence, selection (decision), or iteration (loop) may be introduced.
- Objectives:
  - Work on CT skills: Algorithms, decomposition, and abstraction.
  - Teach structuring of instructions in logical steps.
  - Introduce basic concepts of sequencing, selection, and repetition.

Phase 3. Executing the Algorithm:

- After writing, groups exchange instructions. One student acts as a 'computer' and another as a 'compiler'.
- The compiler reads the instructions aloud and the computer follows them to recreate the drawing.
- Other students observe and analyze errors or ambiguities. Older students can identify types of errors.
- Objectives:

	• Work on the CT skill of evaluation • Understand how instructions and algorithms are executed • Evaluate the written algorithms • Improve oral and written clarity **Phase 4. Evaluating the Algorithm:** - Compare the resulting drawings to the originals. - Reflect on the importance of clarity in writing algorithms.
Estimated Time Required	1 hour per class
Subjects	The activity takes an interdisciplinary approach combining: • **CT and computer science**: Introduction to algorithms, practical applications, and execution on computers. • **Math:** Development of logical reasoning and sequential steps. • **Language:** Clear and precise language use for instructions. • **Art:** Connection between algorithms and visual representation through drawing.
Key Concepts	▪ Algorithms and instruction sequences ▪ Importance of clear instruction writing ▪ Strategies to debug and improve algorithms ▪ Teamwork and effective communication
Plugged / Unplugged	Unplugged
Type of activity	Tangible materials
Individual / Collaborative	This activity is designed as a collaborative learning experience involving group work to improve participation, communication, and teamwork skills. Each team has 2 to 4 students, self-organizing task distribution.
Creativity Level	High
Technology Level	None

CT Skills Addressed	▪ Algorithms: Creating sequential instructions to reach a goal ▪ Evaluation: Analyzing outcomes compared to the intended goal ▪ Generalization: Identifying patterns and applying similar structures ▪ Decomposition: Breaking down drawings into individual steps ▪ Abstraction: Simplifying problems with clear instructions
Implementation Guide	
Guía de realización	**Phase 1. Preliminary Explanation** **Introduction** • Explanation of what an algorithm is and how it applies to everyday life. • Practical examples using daily tasks such as washing hands or making a sandwich. **Distribution of Materials** • Handing out sheets of paper and pencils to each group. • Assigning a secret drawing to each team. **Phase 2. Writing the Algorithm** **Drafting the Algorithm** • Language constraints are introduced by specifying allowed and forbidden words. • Each group writes a detailed set of instructions to recreate their assigned drawing. **Phase 3. Executing the Algorithm** **Algorithm Exchange and Execution** • Groups exchange their written instructions with another team. • One student plays the role of the “computer” and another the “compiler.” • The compiler reads the instructions aloud while the computer follows them literally to reproduce the drawing on the board. **Phase 4. Algorithm Evaluation** **Comparison and Error Analysis** • The recreated drawings are compared to the original versions.

	• Discussion on potential improvements in the clarity of the algorithm. **Final Reflection** • Identification of the most frequent errors and strategies to avoid them. • Reflection on the importance of clarity and precision when writing instructions.
Teacher Training	No prior programming experience required. The activity can be implemented by any teacher with basic CT knowledge.
Inclusion	
Adaptations for Special Needs Students	▪ Use pictograms to represent algorithm steps ▪ Provide oral instead of written explanations for students with literacy difficulties
Learning Adaptations	▪ Allow simpler structures for younger students ▪ Visual support using guided examples
Additional Details	
▪ **Activity Extension:** Progressive difficulty levels can be introduced, or additional elements such as decisions and loops can be incorporated into the construction of the algorithm. ▪ **Cross-curricular Integration:** The activity can be connected with mathematics to strengthen logical reasoning, or with social studies to explore applications in other subject areas.	