
# FORMATION OF ALGORITHMIC THINKING IN PRIMARY IN PRIMARY SCHOOL STUDENTS THROUGH VISUAL PROGRAMMING ENVIRONMENTS

**Madiyeva Charos Dovutovna**
Senior Lecturer at the Tashkent University of Applied Sciences
https://doi.org/10.5281/zenodo.21056138

**Annotation:** This research focuses on the development of algorithmic skills in primary school students through the use of programming tools. The study investigates the effectiveness of interactive and visual programming techniques such as ScratchJr and physical activities in fostering algorithmic thinking. It highlights how these methods help students understand the sequence and logic of algorithms, promoting critical thinking and problem-solving skills. The findings demonstrate that game-based learning and real-life applications can significantly enhance young learners' cognitive development and interest in programming.

**Keywords:** Algorithmic thinking, primary education, programming tools, ScratchJr, interactive learning, game-based learning, cognitive development, problem-solving.

**Introduction**

The development of algorithmic thinking in schoolchildren plays a crucial role in shaping their intellectual and cognitive abilities. Algorithmic thinking refers to the ability to structure and solve problems systematically and logically, which is an essential skill for navigating the modern, technology-driven world. This research aims to explore the effectiveness of interactive practices and visualization techniques in teaching algorithmic thinking and programming to 1st to 4th grade students. Various experimental activities were designed and implemented to engage students in learning algorithmic concepts and programming basics through play and visual tools. The purpose of this study is to determine the best methods to foster interest and understanding of programming and algorithms in young learners.

**Methodology**

The research was carried out with a group of 60 schoolchildren from 1st to 4$^{th}$ grades, divided into control and experimental groups. Initial knowledge levels of the students were assessed, and the activities were conducted based on a specially designed methodology. A series of interactive lessons and experiments were conducted using visualization tools and game-based learning approaches to introduce the concept of algorithms and programming.

**Experimental Activities**

**1. "Commands Are Executed" Game**

Objective: To introduce students to the concept of algorithms and the importance of the sequence of instructions.

Tools: Cards with pre-written commands (e.g., "Move forward 3 steps," "Turn right").

Process: One student acted as a "robot," while others gave them commands written on cards. The robot followed the commands in a specific order. Students had to ensure the sequence of commands was correct for the robot to perform tasks properly.

Result: Students understood the importance of command order in an algorithm and experienced learning through physical activity.

**2. ScratchJr Visual Programming**

Objective: To introduce basic programming concepts to young learners using a visual interface.

Tools: Tablets or computers, ScratchJr software.

Process: Students created a simple algorithm where a cat character moved towards a target by following a series of blocks (e.g., "Move forward," "Turn left," "Turn right").

Result: Students developed logical thinking by independently creating and testing algorithms. The simplicity of the ScratchJr interface made it easy for younger students to grasp programming concepts.

**3. Colorful Cards for Algorithm Creation**

Objective: To understand the sequence and logic of algorithms.

Tools: Colored cards (red, green, yellow, blue), paper, and pens.

Process: Each card was associated with a specific action (e.g., red for "Move forward," green for "Turn right"). Students were asked to arrange the cards in a sequence to navigate a space.

Result: Students grasped the concept of sequential instructions and began to understand the logic behind algorithms. The activity also promoted teamwork and collaborative problem-solving.

**4. Real-Life Algorithms**

Objective: To connect algorithmic thinking with everyday life.

Tools: Paper and markers.

Process: Students were asked to write down the sequence of actions they take in the morning before school (e.g., wash face, have breakfast, wear clothes). They then compared their algorithms with classmates.

Result: Students learned to approach daily routines systematically, developing a logical way of thinking through real-life applications of algorithms.

**DISCUSSION**

The findings of this study demonstrate that visual programming environments constitute an effective pedagogical tool for developing algorithmic thinking among primary school students. The implementation of block-based programming platforms enabled learners to understand fundamental computational concepts without being hindered by the syntactic complexity of traditional programming languages. This supports the assumption that algorithmic thinking can be successfully cultivated during the early stages of formal education when learning activities are designed according to children's cognitive development characteristics.

One of the most significant outcomes of the study is that visual programming promotes the gradual acquisition of algorithmic structures such as sequencing, decomposition, pattern recognition, iteration, conditional reasoning, and debugging. These cognitive operations represent the essential components of algorithmic thinking and form the basis for future computational competence. Unlike text-based programming environments, visual programming allows pupils to manipulate graphical blocks that represent logical operations, enabling them to concentrate on solving problems rather than memorizing syntax. Consequently, students become more confident in constructing algorithms and testing various solution strategies.

The results also indicate that visual programming environments increase learners' motivation and engagement throughout the instructional process. Young learners naturally perceive programming activities as creative games rather than complex technical tasks. Interactive animations, colorful interfaces, and immediate visual feedback significantly enhance students' willingness to experiment with different algorithmic solutions. Such an educational environment encourages active participation, independent exploration, and sustained attention during classroom activities.

Another important observation concerns the relationship between visual programming and problem-solving competence. Algorithmic thinking is not limited to writing computer programs; instead, it represents a universal cognitive strategy applicable across multiple academic disciplines. During the implementation of visual programming tasks, pupils learned to analyze problems, identify key elements, divide complex tasks into manageable components, evaluate alternative solutions, and verify the correctness of their algorithms. These intellectual processes contribute to the development of higher-order thinking skills that are equally valuable in mathematics, science, language learning, and everyday decision-making.

The study further revealed that collaborative learning significantly enhances the effectiveness of visual programming instruction. When students worked in pairs or small groups, they exchanged ideas, discussed algorithmic solutions, identified programming errors collectively, and reflected upon alternative approaches. Such interaction not only strengthened computational understanding but also improved communication skills, teamwork, and peer learning. The collaborative nature of visual programming aligns well with constructivist learning theories, where knowledge is actively constructed through social interaction and shared experiences.

An important pedagogical implication emerging from the findings is the role of teachers as facilitators rather than information transmitters. Effective implementation of visual programming requires educators to design inquiry-based learning activities, guide students through problem-solving processes, encourage experimentation, and provide timely feedback. Teachers who integrate project-based learning with visual programming create opportunities for learners to apply algorithmic thinking to authentic situations, thereby increasing the practical relevance of classroom instruction.

The findings also suggest that age-appropriate visual programming platforms reduce learners' cognitive load. Since primary school pupils possess limited abstract reasoning abilities, the graphical representation of programming constructs supports conceptual understanding by connecting abstract computational ideas with concrete visual objects. This corresponds to cognitive learning theories emphasizing the importance of visual representations during children's intellectual development. The reduction of unnecessary cognitive demands allows learners to allocate more mental resources to reasoning, planning, and evaluating algorithms.

Furthermore, visual programming contributes to the formation of metacognitive skills. While designing and debugging algorithms, students continuously monitor their own thinking processes, identify logical inconsistencies, revise incorrect procedures, and evaluate the effectiveness of different strategies. These reflective activities gradually develop self-regulated learning, enabling pupils to become independent problem solvers capable of transferring algorithmic reasoning to unfamiliar contexts.

The research additionally indicates that visual programming supports differentiated instruction in heterogeneous classrooms. Because block-based programming environments allow tasks of varying complexity, teachers can adapt activities according to learners' individual abilities and learning pace. Advanced students may design sophisticated interactive projects involving variables, events, and nested conditions, while beginners focus on mastering basic sequencing and simple loops. Such flexibility ensures inclusive participation and minimizes learning disparities among pupils.

Despite these positive outcomes, several challenges should be acknowledged. Successful integration of visual programming into primary education depends on adequate technological infrastructure, reliable Internet access, availability of digital devices, and sufficient teacher professional development. In schools where these conditions are limited, the effectiveness of visual programming may be reduced. Moreover, some teachers continue to perceive programming primarily as a technical subject rather than a means of developing logical reasoning and computational thinking. Addressing these misconceptions requires systematic teacher training programs emphasizing pedagogical applications of visual programming.

Another limitation concerns instructional time. The formation of algorithmic thinking is a gradual developmental process that cannot be achieved through isolated programming lessons. Sustainable educational outcomes require long-term curriculum integration where algorithmic concepts are continuously reinforced across different subjects. Interdisciplinary learning activities integrating mathematics, science, language, art, and technology may further strengthen students' computational competencies while maintaining meaningful educational contexts.

The discussion also highlights the broader educational significance of algorithmic thinking within the framework of twenty-first-century competencies. Digital transformation increasingly demands individuals capable of analyzing information systematically, designing efficient procedures, solving complex problems, and adapting to rapidly changing technological environments. Consequently, algorithmic thinking should not be regarded solely as preparation for future programmers but as an essential component of general education that supports lifelong learning and digital citizenship.

Overall, the findings confirm that visual programming environments provide an effective methodological foundation for developing algorithmic thinking among primary school students. Their pedagogical value extends beyond programming instruction by fostering logical reasoning, creativity, collaboration, critical thinking, metacognitive awareness, and independent learning. Future research may investigate longitudinal effects of early visual programming education, compare different visual programming platforms, explore artificial intelligence-supported programming environments, and examine how algorithmic thinking developed during primary education influences students' later academic achievement in STEM disciplines.

**Results and Discussion**

The experiments demonstrated that interactive and game-based learning methods significantly enhanced students' understanding of algorithms and programming. By using tools like ScratchJr, Ozobot, and physical activity-based learning, students were able to visualize and better understand the abstract concepts of algorithms. These methods fostered students'

interest in programming and provided a foundation for further learning in the field of computer science.

1-4 grade students benefited from learning algorithms and programming through games and visual experiments. They developed critical thinking skills, understood the importance of logical sequencing in problem-solving, and were able to engage actively with the content. The use of real-life scenarios, such as daily routines, helped students relate abstract concepts to their everyday lives, making learning more accessible and relevant.

The work of Uzbek scholars such as Sh. I. Kholmuradov[1] and R. T. Usmanov [2], who emphasize the importance of problem-solving skills in the development of algorithmic thinking, aligns with the results of this study. The experiments conducted in this research complement these theoretical insights and reinforce the notion that integrating game-based and visual tools in teaching can significantly improve young learners' cognitive skills in algorithmic thinking.

**Conclusion**

The study confirms that introducing algorithmic thinking at an early age through interactive methods and visual programming tools, such as ScratchJr and Ozobot, is an effective approach. These methods not only enhance students' understanding of algorithms but also foster interest in programming. The integration of physical activities and real-life applications into the learning process further strengthens students' cognitive skills and prepares them for future challenges in the digital world. This research provides valuable insights into the development of algorithmic thinking in primary education and suggests that such practices should be incorporated into standard teaching methodologies to cultivate a generation of problem solvers and critical thinkers.

Uzbek scholars such as M. A. Safarov[3] and T. A. Mirzayev have also demonstrated in their works that integrating algorithmic thinking into primary education is critical to fostering future innovation in technology fields. The inclusion of visual programming tools and interactive learning methods can play a pivotal role in bridging the gap between theory and practice.

## References:

1.Kholmuradov, Sh. I., "The Importance of Algorithmic Thinking in Primary Education," Uzbekistan Journal of Educational Research, 2019.
2.Usmanov, R. T., "Game-Based Learning in the Development of Cognitive Skills," Journal of Educational Technology, 2021.
3.Safarov, M. A., Mirzayev, T. A., "The Role of Visual Programming Tools in Primary Education," Journal of Computer Science Education, 2020.
4.ScratchJr, "A Visual Programming Language for Young Children," MIT Media Lab, 2018.
5.Ozobot, "Interactive Robots for Teaching Algorithmic Thinking," Ozobot Educational Series, 2020.