Enhancing algorithmic thinking skills for application development: a methodological approach in programming education

Feruz Jamoliddinovich Tokhirov
Navoi State University

Abstract: This paper presents a methodology for developing students' algorithmic thinking in application creation. In the digital age, algorithmic thinking is vital for computer science and IT students. The approach focuses on systematic algorithm design, implementation, and improvement, with emphasis on real-life natural processes and best practices in pedagogy and IT education. The experiment was conducted in three stages and analyzed using the Student-Fisher criterion.

Keywords: algorithmic thinking, application development, programming education, computational thinking, problem-solving, block diagrams

Introduction. In our country, specialized courses now include computer graphics, web design, databases, programming languages, networking technologies, information security, as well as software development and technical support [1].

Our observations show that many future IT professionals tend to focus on areas such as computer graphics, web design, and software maintenance, while giving less attention to programming languages. A key factor behind this tendency is the insufficient stimulation of students' motivation and interest in programming, combined with the incomplete delivery of the subject's essence. As a result, significant challenges remain in the teaching of programming languages [2,3].

Fostering algorithmic thinking in future IT specialists is vital for application development. Defined as "the structured sequence of actions inherent in programming" [4,5], it reflects intellectual, creative, figurative, and logical reasoning. By applying planning skills, students better outline their steps for effective problem-solving. Since "every process relies on algorithmic thinking" [7], it represents "a system of cognitive operations and strategies directed toward problem-solving, where the outcome is the construction of an algorithm" [8].

Such a mode of thinking is distinguished by its formality, logic, and accuracy, as well as by the capacity to translate abstract concepts into a sequence of clear instructions. Through the gradual execution of these steps, problems are resolved. At the same time, this approach forms the fundamental basis for mastering programming technologies [9].

Cultivating algorithmic thinking among pupils and students plays a crucial role in programming practice. Through this process, learners acquire the skills and competencies necessary to grasp the core of a programming problem and translate it effectively into program code [10].

In the current system of continuous education, considerable emphasis is placed on teaching pupils and students to algorithmize various problems, both in general secondary schools and higher education institutions. However, our observations reveal that many learners struggle to design an algorithm for a given task and to convert it into programming code. Consequently, their interest in programming diminishes, and their algorithmic thinking fails to fully emerge and develop.

One of the key factors contributing to these difficulties lies in the lack of emphasis on logical consistency when teaching students how to construct algorithms within the continuous education system. In addition, the majority of assigned tasks are concentrated on algorithmizing mathematical problems, which limits the scope of students' algorithmic development.

Methodology. Programming involves multiple stages, with problem modeling and algorithm formulation being the most difficult [11]. Textbooks usually explain linear, branching, and iterative problems through examples [12], but often provide ready-made solutions for complex tasks without showing their construction. This leads students to overestimate their ability to design algorithms, resulting in difficulties when faced with new problems. Such challenges highlight the insufficient development of algorithmic thinking and the urgent need to improve its teaching methodology.

To boost students' interest and competence in programming, teaching should start with algorithmic problem-solving based on real-life and technical processes. This approach enhances engagement and develops skills in designing complex algorithms.

Incorporating natural processes into teaching offers a meaningful and engaging context for introducing algorithmic concepts. Such "natural algorithms" often model complex systems and emergent behaviors, thereby making abstract computational principles more concrete and accessible for learners. By highlighting this real-world relevance, students' motivation and interest in studying algorithms can be significantly enhanced [12,13].

Therefore, within the framework of this research, particular emphasis is placed on this matter, and it is recommended that students in higher education institutions be introduced to the following questions when studying algorithms. Our investigations confirmed that presenting material through block diagrams proves to be effective in guiding students to construct algorithms for various problems. The following issues can serve as illustrative examples.

Example 1.Algorithm for picking ripe fruit and sorting it into three different sizes.

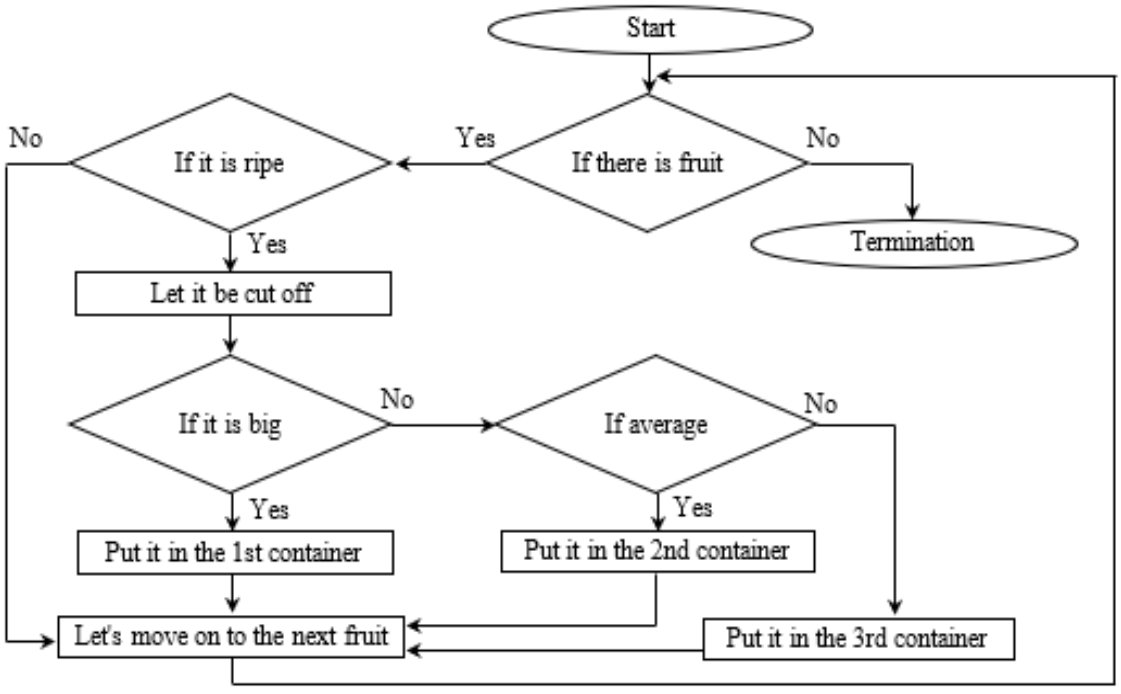

Figure 1. Fruit sorting algorithm block diagram.

Example 2. Algorithm for moving from address A to address B in a car.

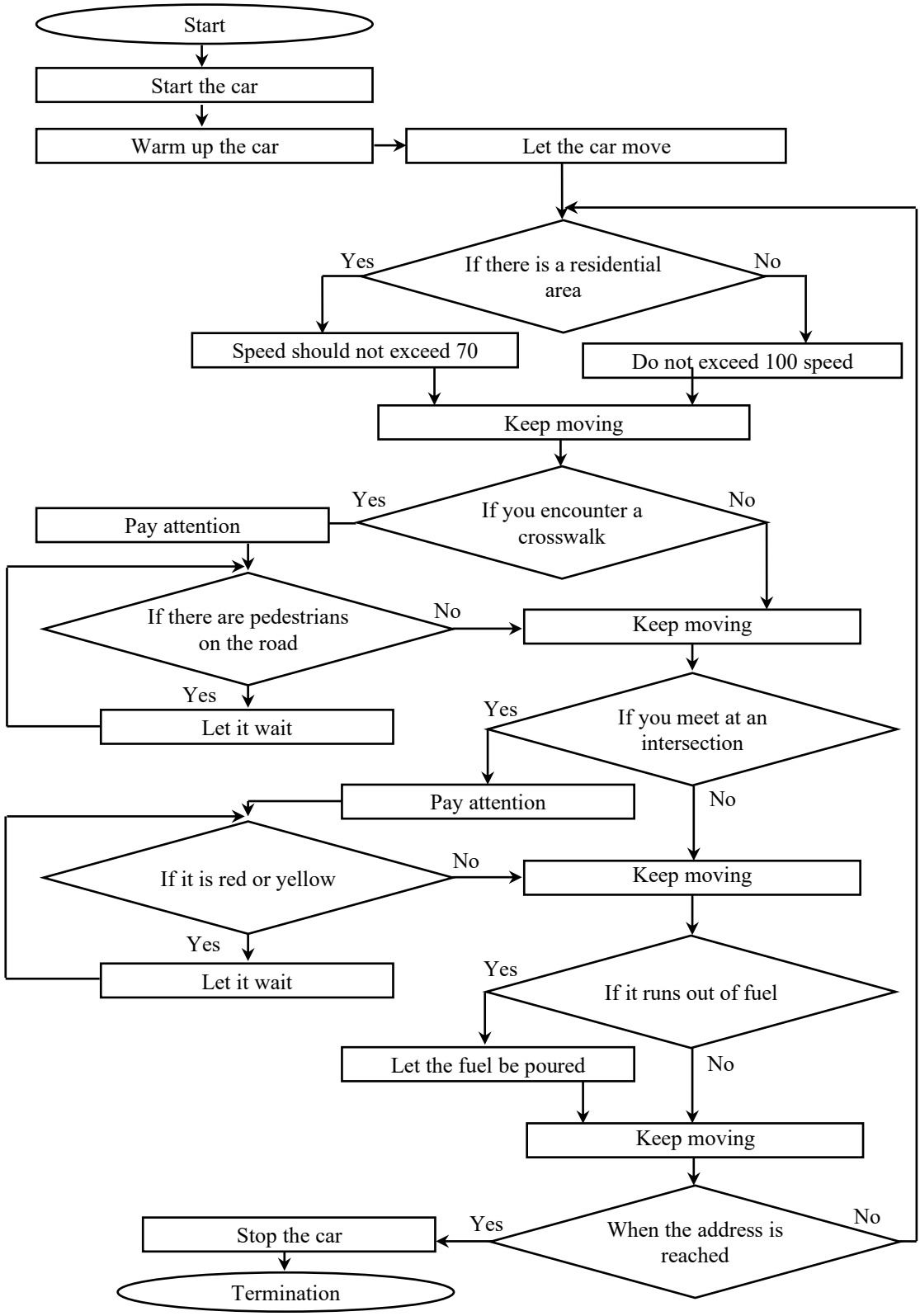

Figure 2. Block diagram of the car's motion algorithm.

Example 3. Algorithm of blood circulation in the human body

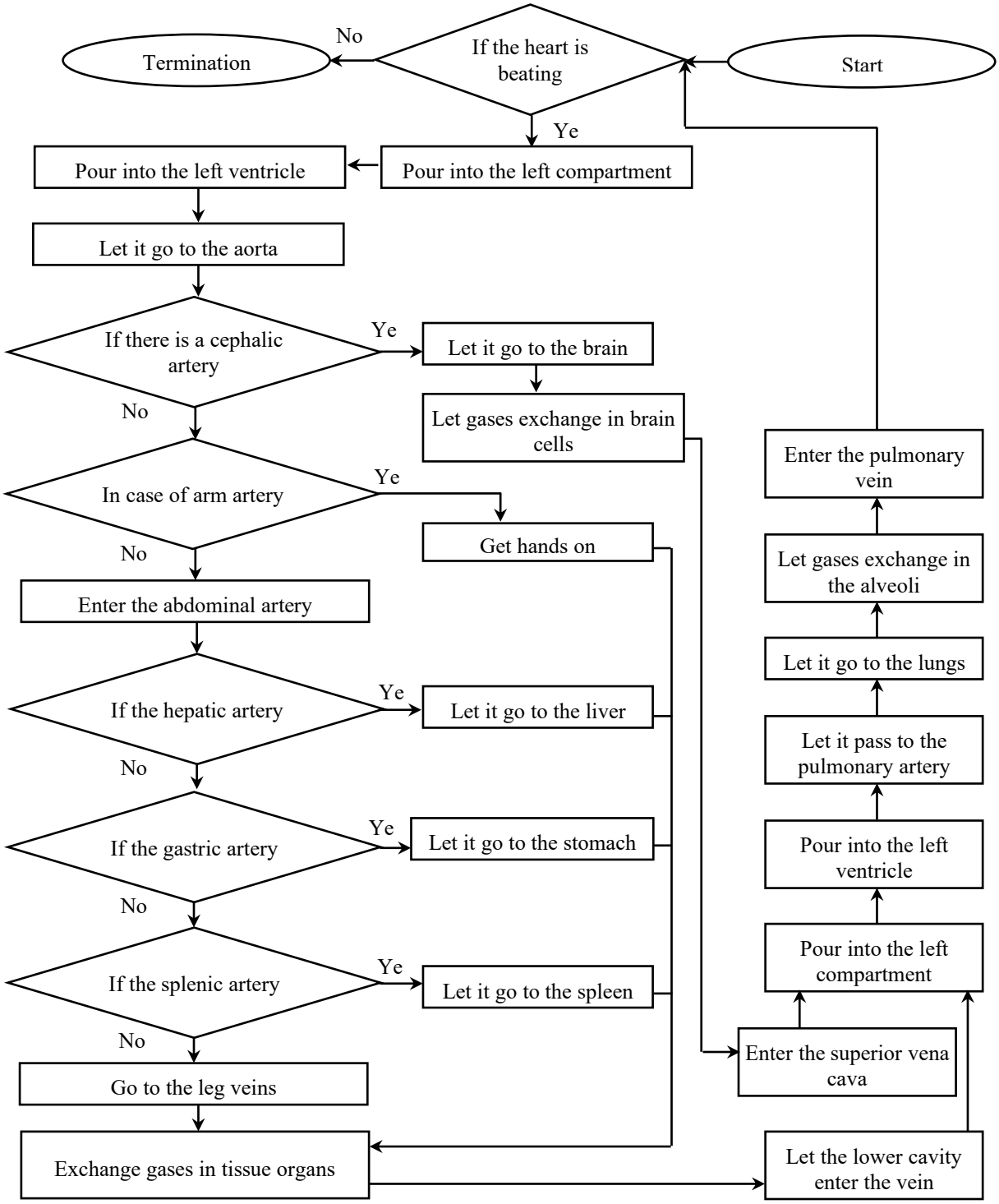

Figure 3.Algorithm of blood circulation in the human body.

The findings indicate that employing problems of this nature contributes to the development of students' algorithmic thinking. Moreover, the use of tools and platforms such as Crocodile ICT, Draw.io, Cacoo.com, and programforyou.ru has proven effective for teaching block diagrams related to these problems.

Through the use of the Crocodile ICT program, students are able to visualize algorithms for different examples and tasks in the form of block diagrams. This tool also enables the modeling of various processes, the representation of linear, branching, and iterative algorithms in diagrammatic form, as well as the monitoring and analysis of these algorithms.

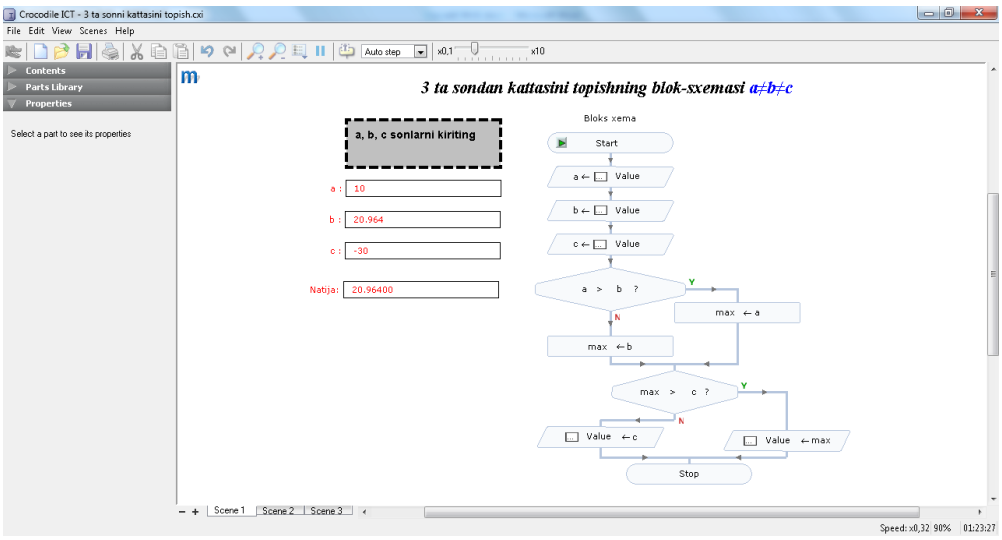

Figure 4. Block diagram prepared in Crocodile ICT program.

Block diagrams created in Crocodile ICT can be demonstrated in an animated format, where various characters are capable of performing more than 40 different actions. Such visual and interactive presentations enhance students' engagement during instruction, particularly in learning methods of constructing and explaining algorithms [6].

Additionally, this program provides a visual representation of the sequence of actions within a block diagram. It also allows for the identification of errors: if a mistake occurs in the order of operations, the program halts execution and generates an error notification.

Similar opportunities are offered by platforms such as lusidshart.som, gliffy.som, wireflow.som, textograrho.som and Google Drawings. These tools can be effectively used to spark students' interest in algorithms and programming while simultaneously fostering the development of their creative thinking.

Results. As part of the research, pilot tests were conducted in order to determine the effectiveness of the methodology designed to develop students' algorithmic thinking regarding the creation of applications. The success of experimental work shows the need to take into account its organizational and pedagogical aspects in this process. Therefore, special attention was paid to these aspects. Experimental work was conducted in 2023 among students of the Navoi State Pedagogical Institute in the field of "Mathematics and Informatics".

A total of 124 students were recruited for the experimental and control groups. Experimental work was carried out in three stages: emphasis; formative; the finisher. At the critical stage of the experimental work, students were interviewed and observed about the main features of the informational educational environment.

In the formative stage, trainings were conducted for the experimental group based on the proposed informational educational environment, and the following criteria were developed to evaluate the efficiency of students' learning: motivational; cognitive; technological; creative.

At the final stage, a mathematical-statistical analysis was performed based on the Student-Fisher criterion in order to check the reliability of the results of students in the experimental and control groups.

Appropriate mean values for samples using this criterion $\overline{X} = \frac{1}{n}\sum_{i=1}^{4} n_i X_i$, dispersion coefficients $D_n = \sum_{i=1}^{3} \frac{n_i (x_i - \bar{X})^2}{n-1}$, mean squared deviations $\tau_n = \sqrt{D_n}$, variation indicators $\delta_n = \frac{\tau_n}{X}$, reliable deviations of estimation $\Delta_n = t_{kH} \cdot \frac{D_n}{\sqrt{n}}$, and in determining the mastery indicators $P = \frac{\vec{X}}{3} \cdot 100\% - \frac{\vec{Y}}{3} \cdot 100\%$ formulas were used. According to the calculation result, it was found that the average mastery rate of the experimental group was higher than that of the control group, that is, it increased by 11%.

Conclusions. The study demonstrates that applying algorithms based on natural processes in education effectively enhances students' algorithmic knowledge, skills, and problem-solving abilities, while also fostering greater engagement and motivation, which are crucial for successful learning.

Using the recommended programs and platforms enables students to: develop skills in expressing algorithms in words and block diagrams; apply algorithms to different problem types (linear, branching, recurring); strengthen abilities in algorithmizing natural and technical processes; enhance creativity in selecting effective algorithms; and improve skills in analyzing and revising flawed algorithms.

In conclusion, it is effective to connect the given issues to life processes in developing students' thinking about algorithms. Teaching algorithms of natural processes is a powerful pedagogical approach in developing students' algorithmic thinking. It combines the strengths of contextual learning, visualization, interaction, problem solving, and cognitive development to promote a deep and intuitive understanding of computing principles. On the basis of these, it will be possible to increase students' interest in algorithmization and to teach algorithmization of various complex problems. As a result, it is possible to increase students' interest in programming and creating applications.

References

1. Tokhirov F. J. Problems of Developing Students' Algorithmic Thinking about Programming //ONLINE-CONFERENCES" PLATFORM. – 2021. – С. 169-170.

2. Jamoliddinovich T. F. Algorithmic Thinking of Students in Program using Electronic Learning Resources Principles in Development //Kresna Social Science and Humanities Research. – 2022. – Т. 3. – С. 93-94.

3. Jamoliddinovich T. F. Methodology of developing algorithmic thinking of students on programming in higher educational institutions //Berlin Studies Transnational Journal of Science and Humanities. – 2022. – Т. 2. – №. 1.5 Pedagogical sciences.

4. Tokhirov F. et al. Methodology for developing students' algorithmic thinking about creating applications //AIP Conference Proceedings. – AIP Publishing LLC, 2025. – Т. 3268. – №. 1. – С. 070016.

5. David Weintrop. 2019. Block-based programming in computer science education. Commun. ACM 62, 8 (August 2019), 22–25. https://doi.org/10.1145/3341221

6. Otakulova Durdona Rahmonovna. (2024). Methodology For Organizing Independent Education Of Students Of Higher Educational Institutions In Subjects Related To Computer Graphics. Educational Administration: Theory and Practice, 30(5), 168–173. https://doi.org/10.53555/kuey.v30i5.1252

7. Friday Joseph Agbo, Solomon Sunday Oyelere, Jarkko Suhonen, and Sunday Adewumi. 2019. A Systematic Review of Computational Thinking Approach for Programming Education in Higher Education Institutions. In Proceedings of the 19th Koli Calling International Conference on Computing Education Research (Koli Calling '19). Association for Computing Machinery, New York, NY, USA, Article 12, 1–10. https://doi.org/10.1145/3364510.3364521

8. Ana M Pinto-Llorente, Sonia Casillas Martín, Marcos Cabezas González, and Francisco José García-Peñalvo. 2016. Developing computational thinking via the visual programming tool: lego education WeDo. In Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM '16). Association for Computing Machinery, New York, NY, USA, 45–50. https://doi.org/10.1145/3012430.3012495

9. Mirsanov U.M. Requirements for Creating Electronic Informational and Educational Resources on Subjects of Mathematical Cycle in Global Internet //www. auris-verlag. de.–2017.

10. Mirsanov U.M., Ravshanova G. REQUIREMENTS FOR THE DESIGN OF TEACHING AIDS IN THE SUBJECT OF PROGRAMMING LANGUAGES //International Journal of Engineering Mathematics (Online). – 2021. – Т. 3. – №. 1.

11. Kanaki K, Kalogiannakis M. Assessing Algorithmic Thinking Skills in Relation to Age in Early Childhood STEM Education. Education Sciences. 2022; 12(6):380. https://doi.org/10.3390/educsci12060380

12. Futschek G. Algorithmic thinking: the key for understanding computer science //International conference on informatics in secondary schools-evolution and perspectives. – Berlin, Heidelberg : Springer Berlin Heidelberg, 2006. – С. 159-168.

13. Dybvig R.K. The Scheme programming language. – Mit Press, 2009.