

DOI: 10.22363/2312-8631-2025-22-1-76-88

EDN: TDYBYR

UDC 378.1

Research article / Научная статья

## Developing structural component of computational thinking using the algorithmic primitives method

Irina V. Bazhenova<sup>1</sup>, Margarita M. Klunnikova<sup>1</sup>, Nikolay I. Pak<sup>2</sup><sup>1</sup>*Siberian Federal University, Krasnoyarsk, Russian Federation*<sup>2</sup>*Krasnoyarsk State Pedagogical University named after V.P. Astafyev, Krasnoyarsk, Russian Federation*apkad@yandex.ru

**Abstract.** *Problem statement.* A modern specialist necessary quality is structural thinking, a skill with which a person is able to decompose a complex task into subtasks and create integral structures from a set of elements. The goal of the study is to substantiate the algorithmic primitives method to create a methodology for the development of a structural component of students' computational thinking in the cluster of disciplines "Programming – Numerical Methods – Information Technologies in Education". *Methodology.* The algorithmic primitives method is based on introduction of the concept "algorithmic primitive" understood as a template for an algorithm for solving elementary problems, from the set of which algorithms for solving complex problems can be built. Creation of the primitive is carried out with the use of mental schemes of subject area. Such an approach allows to automate practically all stages of training and to create e-learning tools. *Results.* The algorithmic primitives method for solving problems of various levels of complexity in the cluster of disciplines "Programming – Numerical Methods – Information Technologies in Education" is justified and implemented into educational practice. The training database of algorithmic primitives for e-courses in these disciplines has been created. *Conclusion.* The method of algorithmic primitives significantly facilitates teaching students to solve problems and contributes to the development of structural component of computational thinking.

**Keywords:** algorithmic primitive, structural thinking, mental scheme, cluster of disciplines

**Author's contribution.** The authors contributed equally to this article.

**Conflicts of interest.** The authors declare that there is no conflict of interest.

**Funding.** The research was carried out with the support of Krasnoyarsk Regional Fund of Science and Technology Support within the framework of the project No. 2021012106985 "Formation and development of students' computational thinking based on automated and cognitive learning tools".

---

© Bazhenova I.V., Klunnikova M.M., Pak N.I., 2025




This work is licensed under a Creative Commons Attribution 4.0 International License  
<https://creativecommons.org/licenses/by-nc/4.0/legalcode>

Article history: received 16 July 2024; revised 28 August 2024; accepted 14 September 2024.

**For citation:** Bazhenova IV, Klunnikova MM, Pak NI. Developing structural component of computational thinking using the algorithmic primitives method. *RUDN Journal of Informatization in Education*. 2025;22(1):76–88. <http://doi.org/10.22363/2312-8631-2025-22-1-76-88>

## Развитие структурного компонента вычислительного мышления с использованием метода алгоритмических примитивов

И.В. Баженова<sup>1</sup>, М.М. Клунникова<sup>1</sup>, Н.И. Пак<sup>2</sup>

<sup>1</sup>Сибирский федеральный университет, Красноярск, Российская Федерация  
<sup>2</sup>Красноярский государственный педагогический университет им. В.П. Астафьева, Красноярск, Российская Федерация  
apkad@yandex.ru

**Аннотация.** *Постановка проблемы.* Необходимым качеством современного специалиста является структурное мышление – навык, с помощью которого человек способен проводить декомпозицию сложной задачи на подзадачи и создавать целостные структуры из набора элементов. Цель исследования заключается в обосновании метода алгоритмических примитивов для создания методики развития структурного компонента вычислительного мышления студентов в кластере дисциплин «Программирование – Численные методы – Информационные технологии в образовании». *Методология.* Метод алгоритмических примитивов основан на введении понятия «алгоритмический примитив», под которым понимается шаблон алгоритма решения элементарных задач, из совокупности которых можно строить алгоритмы для решения сложных задач. Создание примитива осуществляется с использованием ментальных схем предметной области. Такой подход позволяет автоматизировать практически все этапы обучения и создавать электронные средства обучения. *Результаты.* Обоснован и внедрен в учебный процесс метод алгоритмических примитивов для решения задач различного уровня сложности в кластере дисциплин «Программирование – Численные методы – Информационные технологии в образовании». Создана учебная база алгоритмических примитивов для электронных курсов по данным дисциплинам. *Заключение.* Метод алгоритмических примитивов существенно облегчает работу преподавателя по обучению студентов решению задач и способствует развитию у них структурного компонента вычислительного мышления.

**Ключевые слова:** алгоритмический примитив, структурное мышление, ментальная схема, кластер дисциплин

**Вклад авторов.** Все авторы внесли равный вклад в подготовку публикации.

**Заявление о конфликте интересов.** Авторы заявляют об отсутствии конфликта интересов.

**Финансирование.** Исследование выполнено при поддержке Красноярского краевого фонда поддержки научной и научно-технической деятельности в рамках

реализации проекта № 2021012106985 «Формирование и развитие вычислительного мышления обучаемых на основе автоматизированных и когнитивных средств обучения».

**История статьи:** поступила в редакцию 16 июля 2024 г.; доработана после рецензирования 28 августа 2024 г.; принята к публикации 14 сентября 2024 г.

**Для цитирования:** *Bazhenova I.V., Klunnikova M.M., Pak N.I.* Developing structural component of computational thinking using the algorithmic primitives method // Вестник Российского университета дружбы народов. Серия: Информатизация образования. 2025. Т. 22. № 1. С. 76–88. <http://doi.org/10.22363/2312-8631-2025-22-1-76-88>

**Problem statement.** The world around us appears to us as a whole in which its individual parts are distinguished. The fundamental knowledge of matter and the ordering of its chaos make the “part – whole” construction the most important in the structural and systemic construction of the general world picture and human behavior in it. Currently, due to the digitalization of society, there are new requirements for modern specialists for high-tech areas of the economy. For example, they need skills related to dividing a common task into subtasks; planning the stages and timing of their activities; searching for necessary and relevant information; understanding sequential, parallel and nondeterministic (intuitive) actions. Such skills in a person are to a greater extent provided by his / her structural thinking. Structural thinking is the ability to identify connections between objects and the ways they interact with each other. Structural thinking views entities as being part of a larger whole [1].

The person structural thinking should be formed in childhood and continuously developed at school and university, so the person learns to see “the particular in the whole” and “the whole in the particular”. One of the ways to develop structural thinking is the Barbara Minto pyramid method, used in communications – business correspondence and speaking, consulting, and in many other fields [2].

Many teachers use techniques for developing structural thinking in their educational practice, for example [3–5]. But there is no generally recognized methodology for the systematic development of structural thinking in the disciplines training process. This applies, among other things, to teaching mathematics and programming, which require advanced structural thinking. Perhaps the exception is some construction and technology disciplines, including art and graphics, where graphical primitives (parts) are explicitly used, allowing the construction of complex objects (whole). Indeed, the use of elementary template structures in the form of object or conceptual primitives for building a complex project structure or solving a problem fully implements the principle of “part – whole”.

Structural thinking can be considered as a component of computational thinking, a necessary skill of a modern specialist [6]. In this regard, it is of interest to model teaching methods for disciplines based on the training primitives

database development used to solve complex problems. The article goal is to justify and develop the algorithmic primitives method to create the methodology for the development students' structural thinking. The methodology is used in the cluster of disciplines "Programming – Numerical Methods – Information Technologies in Education", which is discussed in detail by the authors in [7].

Research objectives:

- to identify the essence of structural component of computational thinking and analyze ways of its development in the subject teaching of students;
- to develop examples of algorithmic primitives to solve problems in the courses "Programming" and "Numerical Methods";
- to develop the algorithmic primitives method for teaching students to solve computational-algorithmic problems;
- to outline the frame of methodology for development of structural thinking of students in the cluster of disciplines.

**Methodology.** In March 2006, University of Pittsburgh professor Jeannette Wing coined the concept "computational thinking". From Wing's point of view [8], computational thinking is a universally applicable approach and a set of skills that modern humans use to solve problems that arise in all areas of human activity using computer technology. Later, clarifying the definition, she identified two important educational aspects that consider computational thinking as a thought process independent of technology, and as a special way of solving problems involving a person, a computer, or a combination of them.

Wing's concept has sparked widespread discussion among educational scientists around the world about the nature of computational thinking and its implications for education in general. The most cited definition is proposed by specialists from the Royal Academy of Engineering in Great Britain [9]: "Computational thinking is the process of recognizing aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes".

The Association of Teachers of Computer Science and the International Society for Technology in Education (CSTA & ISTE) has formulated a definition that highlights the practical operations that make up computational thinking [10]:

- formulating problems in a way that enables us to use a computer and other tools to help solve them;
- logically organizing and analyzing data;
- representing data through abstractions such as models and simulations;
- automating solutions through algorithmic thinking (a series of ordered steps);
- identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources;
- generalizing and transferring this problem-solving process to a wide variety of problems.

Different authors identify a wide range of skills related to the development of computational thinking, the key ones are problem decomposition, pattern recognition, abstraction, and algorithmization. In fact, pattern recognition and the ability to divide complex problems into simpler ones constitute structural thinking, which allows us to distinguish structural thinking as one of the components of computational thinking. At the same time, structural thinking can be considered as an independent type of thinking, which is widely used in a variety of fields, including everyday life [11].

Mathematics education is closely related to the development of computational and structural thinking skills. M. Gronow et al. rely on the CRIG pedagogical framework, the components of which are Connections, Recognising patterns, Identifying similarities and differences, and Generalising and Reasoning, as a tool for developing structural thinking [12]. Mason et al. consider structural thinking as knowledge and use of concepts and procedures in solving mathematical problems [13]. H.Y. Durak and M. Saritepeci in their study showed that to a greater extent the development of computational thinking skills depends on “thinking styles, academic success in mathematical class, attitude against mathematical class” [14].

The process of structural thinking consists of generation of ideas and structuring itself, i. e. first there is a process of data collection, and then there is an analysis of the data.

In the article [15], the authors propose to consider the decomposition of problems into subproblems as a process consisting of the following stages:

- categorization of potential elements: identifying the basic elements and defining the relationships between the elements;
- choosing a strategy for the chosen decomposition: analysis of means and goals, bottom-up analysis method, multivariate statistical analysis, etc. [16];
- iterative evaluation of usefulness of a particular decomposition.

By highlighting this process, it is possible to analyze which types of categorization or strategies experts use and how to develop the students' ability to think structurally in a more targeted way.

In our opinion, the most promising is the pyramid principle developed by Barbara Minto. The essence of the method is to divide the problem into parts in such a way that the pyramid top is the main question, and the next levels include disjoint ideas that create the entire possible range of solutions to the problem. Further, each idea is detailed until a specific solution is built. If the person is not proficient in the topic, the pyramid uses a decision tree based on arguments and statements that answer the question “How?”. If the person is an expert in the field, a hypothesis tree is built, in which case the problem is to prove or disprove the hypothesis, i. e. to answer the question “Why is it so?” or “Why is it necessary?”. Usually these two approaches are combined.

Let us introduce the concept “algorithmic primitive”, which we will understand as a template of an algorithm for solving elementary problems, from the set of which we can build algorithms for complex problems.

From this definition follows the hierarchical-network structure of the system of such primitives and the expediency of distinguishing basic and composite ones among them.

It should be specified that, for example, in programming, algorithmic primitives are not identical to the basic algorithmic constructions (sequence, branching, loop). In most cases, a primitive is expressed through an algorithmic construct or a combination of them. Let us consider a set of basic algorithmic primitives for solving some typical algorithmic and computational problems:

1. Organization of the counter of variables or objects. In algorithmic language, it is written as  $i := i + 1$ . At the same time, in programming languages where there is an increment operation (for example, C/C++), the counter is implemented through an increment.

2. Exchange of values of two variables using a buffer variable.

3. Checking the multiplicity of a number, for example, the evenness of a number.

4. Summing a numeric sequence without using an array.

5. The product of numerical sequence elements.

6. Finding the maximum / minimum in a sequence of numbers / objects.

7. Iteration of array elements using a loop with a counter.

8. Using the flag – a boolean / integer variable for the case of non-fulfillment / fulfillment of a given condition in the problem statement. For example, you need to write all negative elements of the original array to another array. If they are absent, changing the state of the flag will allow you to avoid incorrect actions and print the corresponding message.

An example of a composite algorithmic primitive can be the problem of summing even array elements (superposition of primitives 7, 3, 4, 8).

Using basic and composite algorithmic primitives it is possible to build real algorithms for solving computational problems and data processing problems, for example, realization of array sorting methods, numerical methods for solving systems of linear and nonlinear equations, optimization problems, etc. In this connection, it seems reasonable to create groups (complexes) of basic and composite primitives to solve certain classes of problems.

Unlike the method of structural programming, where the control structures are basic algorithmic constructions (sequence, branching, loop), and on the basis of which algorithms for solving problems are constructed by superposition, in the proposed method the control structures are algorithmic primitives. At the same time, they facilitate the initial planning of the structural construction of problem-solving algorithm first on the usual language level, then in the program code.

Let's consider the example of building an algorithm for sorting a one-dimensional numeric array of 10 elements in descending order. For problems of this class, we will form a set of primitives:

1) counter;

2) input of array elements;

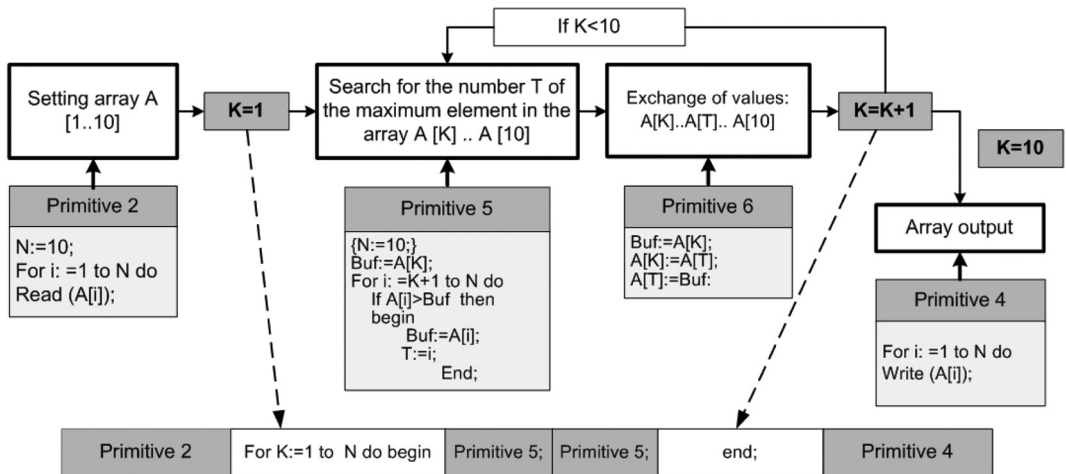


- 3) initialization of array’s elements with random numbers;
- 4) output of array’s elements on the display;
- 5) finding the maximum element and its number among the array’s elements of the unordered part of the array;
- 6) exchange of values of two variables.

A verbal structured solution to the problem may look like this: first, we find the maximum element and put it in the place of the first element (exchange them), then we find the maximum element in the array, starting from the second element, and replace the second with the maximum, then we repeat this procedure until the last element:

1. We initialize the required array  $A[1..10]$  with boundaries from 1 to 10 (primitive 2 or 3).
2. Set counter = 1.
3. Find the maximum element in the array from  $A[K]$  to  $A[10]$ (primitive 5, gives the number of the maximum element  $T$ ).
4. Exchange of values  $A[K]$  and  $A[T]$  (primitive 6).
5. Counter  $K = K + 1$  (primitive 1).
6. Repeat steps 3), 4), 5), until  $K < 10$ .
7. We display the final array (primitive 4).

A possible mental scheme for solving this problem is presented in Figure 1.



**Figure 1.** Sorting an array as mental scheme

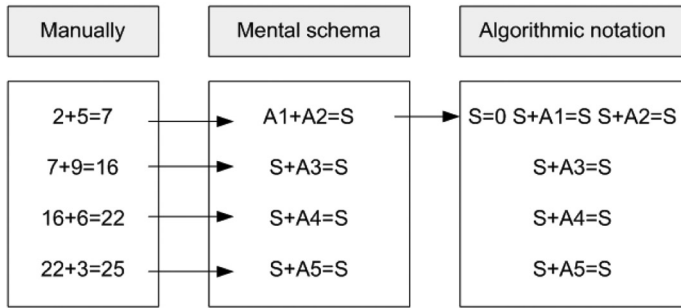
Source: created by Irina V. Bazhenova, Margarita M. Klunnikova, Nikolay I. Pak.

From the diagram shown in Figure 1 it is easy to compose the final algorithm for solving the problem, which should be clear to a beginner in programming.

**Results and discussion.** In the cluster of disciplines “Programming – Numerical Methods – Information Technologies in Education” it is possible to identify the content lines, for which sets of algorithmic primitives were created.

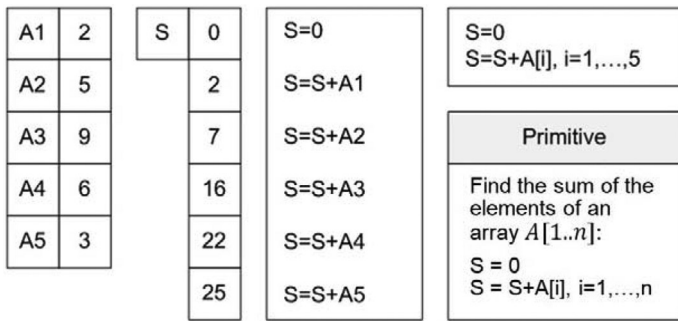
Consider a possible description of the primitive “Sum of array elements”.

Let  $A[1] = 2, A[2] = 5, A[3] = 9, A[4] = 6, A[5] = 3$ . Find the sum of these elements. Figure 2 shows the representation of this algorithm in different notations.



**Figure 2.** Visualization and structuring of “Sum of array elements” algorithm  
 Source: created by Irina V. Bazhenova, Margarita M. Klunnikova, Nikolay I. Pak.

Figure 3 shows the resulting primitive “Sum of array elements”.



**Figure 3.** Training element for the primitive “Sum of array elements”  
 Source: created by Irina V. Bazhenova, Margarita M. Klunnikova, Nikolay I. Pak.

We create a similar script for each primitive. Thus, we get a reference book of algorithmic primitives with a mental visualization of their meaning.

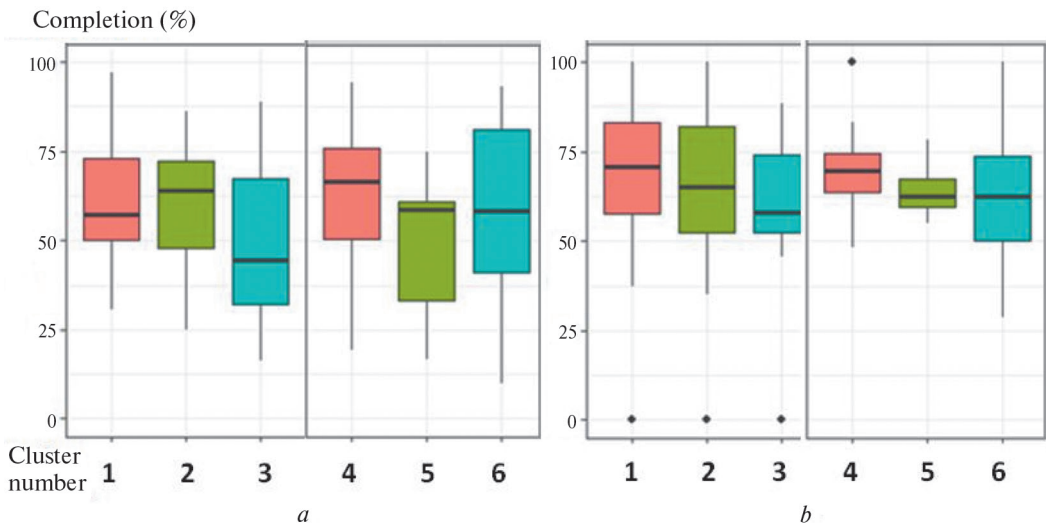
Let’s consider examples of primitives for problems related to numerical methods.

1. Finding a sequence of numbers by the formula:  $A_{n+1} = f(A_n), n = 0..K$ 
  - a) introduce  $A_0$ ;
  - b) find  $A_1 = f(A_0)$ ;
  - c) output  $A_1$  (next element of the sequence);
  - d) assign  $A_0 = A_1$ ;
  - e) repeat b), c), d)  $K$  times.
2. Finding a sequence of numbers by the formula:  $A_0 = \text{const}; A_{n+1} = f(A_n), n = 0, 1, \dots$  until  $|A_{p+1} - A_p| < \text{eps}$ , where  $\text{eps}$  is a given number
  - a) introduce  $A_0$  and  $\text{eps}$ ;
  - b) find  $A_1 = f(A_0)$ ;
  - c) output  $A_1$  (next element of the sequence);
  - d) assign  $A_0 = A_1$ ;
  - e) repeat b), c), d) until  $|A_1 - A_0| < \text{eps}$ .

The algorithmic primitives method was used in design of the e-course “Programming in C / C++” for 1st year students in “Applied Mathematics and



Information Science” at Siberian Federal University. The variable educational content of the e-course was based on two factors: the psychotype of information perception, determined by the leading perceptual modality and the students’ learning style. At the beginning of the training, Visuals, Auditories, Kinesthetics and Digitals were identified using a survey. The students’ learning styles in the programming training were defined as “theorists” who prefer the presentation of educational material from a formal description of the programming language to implementation and code examples; and “practitioners” who learn from examples of ready-made programs. The e-course content was developed for six student models. The content for the “Digital – Theorist” model was based on a step-by-step method of explaining the material, which provides a structured algorithm for solving the problem with a detailed decomposition of the solution into elementary operations, i.e. algorithmic primitives. The success of this technique is demonstrated by the analysis of results of the e-course after the 1st semester, which was conducted from 2020 to 2023. For the e-course learning analytics, cluster analysis was applied, taking into account type of perception, educational style and results of all types of activity on the e-course. 6 clusters were identified, each of which was dominated by some type of perception and learning style. Figure 4 in the form of a “Box with a mustache” diagram shows the students educational results in the entrance and final tests by clusters.



**Figure 4.** A box with a mustache diagram for the results of: input testing (**a**), final testing (**b**)

Source: created by Irina V. Bazhenova, Margarita M. Klunnikova, Nikolay I. Pak.

As can be seen from the diagram, the most successful were the students of clusters No. 1 (the predominant model “Digital – Theorist”) and No. 2 (the predominant model “Visual – Practitioner”) (for this student model, flowcharts were used in the e-course content).

The algorithmic primitives database created during the design of programming e-course can later be used when students study the course “Numerical Methods” to structure algorithms for solving numerical problems. And in the

course “ICT in Education”, the presented algorithmic primitives method allows you to create e-learning tools for programming and numerical methods with an emphasis on visualization and interactivity.

**Conclusion.** Problem solving by the algorithmic primitives method seems to become meta-programming, has a propaedeutic character for teaching computational and algorithmic problem solving. The proposed method has high didactic qualities – it provides gradual formation of students’ algorithmic skills from understanding and solving simple problems, to understanding and composing complex algorithms in a structured form. In this case the structural manipulation of algorithmic primitives contributes to the development of the structural component of computational thinking. To strengthen the considered factor, it is necessary to use mental schemes for visualization of algorithmic problem solving, to simulate the process of algorithm execution and to give examples of solution recording in some programming languages.

It is advisable to involve the students themselves in the development of a system of algorithmic primitives. This achieves the following goals:

Students practice problem decomposition.

They learn to analyze and compare solutions to similar problems by discovering solution patterns.

They learn to synthesize the solution of new problems based on familiar algorithmic primitives.

These skills relate to operational definition of computational thinking and traditional understanding of structural thinking, which will significantly improve their level of development. Algorithmic primitives can be presented (and expressed by students themselves) in any form convenient for them. A variety of representations of algorithmic primitives will contribute to the understanding of complex educational material related to the subject area of programming and mathematics.

Based on the system of algorithmic primitives one can create a database of problems of different levels of complexity, the introduction of which in the information and educational environment of the university will automate the process of teaching programming, increase the availability of education, and support effective teaching of this discipline.

Thus, the proposed method of algorithmic primitives significantly facilitates the work of a teacher to teach students to solve problems and contributes to the development of their structural component of computational thinking.

## References

- [1] Shannon N, Frischherz B. Structural Thinking. In: *Metathinking. Management for Professionals*. Cham: Springer; 2020. p. 23–27. [https://doi.org/10.1007/978-3-030-41064-3\\_3](https://doi.org/10.1007/978-3-030-41064-3_3)
- [2] Minto B. *The Minto pyramid principle: Golden rules for thinking, business writing, and speaking*. Trans. from English by Yurchik II, Yurchik YuI. 9th ed. Moscow: Mann, Ivanov and Ferber; 2018. (In Russ.)

- [3] Gronow M. Noticing structural thinking through the CRIG framework of mathematical structure. In: *Proceedings of the 43rd Annual Conference of the Mathematics Education Research Group of Australasia, 5–8 July 2021, Singapore*. Adelaide: MERGA; 2021. p. 211–218.
- [4] Kupchinskaya MA, Yudalevich NV. Formation of structural thinking among students-managers in the DBMS study. *Business Education in the Knowledge Economy*. 2018;2:42–46. (In Russ.)
- [5] Barkhatova DA, Grushentseva DS. Diagnostics of structural thinking of bachelors of pedagogical education as a universal pedagogical competence. *Open Education*. 2024;28(1):54–60. (In Russ.) <https://doi.org/10.21686/1818-4243-2024-1-54-60>
- [6] Klunnikova MM, Bazhenova IV, Pak NI, Kirgizova EV. Developing students computational thinking with a recursive polydisciplinary approach. *Journal of Physics: Conference Series*. 2020;1691:012190. <https://doi.org/10.1088/1742-6596/1691/1/012190>
- [7] Bazhenova IV, Klunnikova MM, Pak NI. School-university cluster of disciplines developing the calculative-algorithmic component of computational thinking. *Informatics and Education*. 2021;3:42–49. (In Russ.) <https://doi.org/10.32517/0234-0453-2021-36-3-42-49>
- [8] Wing JM. Computational thinking. *Communications of the ACM*. 2006;49(3):33–35.
- [9] Furber S. *Shut down or restart? The way forward for computing in UK schools*. The Royal Society Education Section; 2012. <https://royalsociety.org/~media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf>
- [10] Lyon JA, Magana AJ. Computational thinking in higher education: A review of the literature. *Computer Applications in Engineering Education*. 2020;28(5):1174–1189. <https://doi.org/10.1002/cae.22295>
- [11] Pak NI. A Mental approach to digital transformation of education. *Open Education*. 2021;25(5):4–14. (In Russ.) <https://doi.org/10.21686/1818-4243-2021-5-4-14>
- [12] Gronow M, Mulligan J, Cavanagh M. Teachers’ understanding and use of mathematical structure. *Mathematics Education Research Journal*. 2022;34:215–240. <https://doi.org/10.1007/s13394-020-00342-x>
- [13] Mason J, Stephens M, Watson A. Appreciating mathematical structure for all. *Mathematics Education Research Journal*. 2009;21(2):10–32. <https://doi.org/10.1007/BF03217543>
- [14] Durak HY, Saritepeci M. Analysis of the relation between computational thinking skills and various variables with the structural equation model. *Computers & Education*. 2018;116:191–202.
- [15] Rich P, Egan G, Ellsworth J. A framework for decomposition in computational thinking. In: *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, 15–17 July 2019, Aberdeen*. New York: Association for Computing Machinery; 2019. p. 416–421. <https://doi.org/10.1145/3304221.3319793>
- [16] Simon G. *The sciences of the artificial*. Trans. from English by Nappelbaum EL. 2nd ed. Moscow: Editorial URSS; 2004. (In Russ.)

### Список литературы

- [1] Shannon N., Frischherz B. Structural Thinking // Metathinking. Management for Professionals. Cham: Springer, 2020. P. 23–27. [https://doi.org/10.1007/978-3-030-41064-3\\_3](https://doi.org/10.1007/978-3-030-41064-3_3)
- [2] Минто Б. Принцип пирамиды Минто: Золотые правила мышления, делового письма и устных выступлений / пер. с англ. И.И. Юрчик, Ю.И. Юрчик. 9-е изд. М.: Манн, Иванов и Фербер, 2018. 272 с.

- [3] *Gronow M.* Noticing structural thinking through the CRIG framework of mathematical structure // Proceedings of the 43rd Annual Conference of the Mathematics Education Research Group of Australasia, Singapore, 5–8 July 2021. Adelaide: MERGA, 2021. P. 211–218.
- [4] *Купчинская М.А., Юдалевич Н.В.* Формирование структурного мышления у студентов менеджеров при изучении СУБД // Бизнес-образование в экономике знаний. 2018. № 2. С. 42–46.
- [5] *Бархатова Д.А., Грушенцева Д.С.* Диагностика структурного мышления бакалавров педагогического образования как универсальной педагогической компетенции // Открытое образование. 2024. Т. 28. № 1. С. 54–60. <https://doi.org/10.21686/1818-4243-2024-1-54-60>
- [6] *Klunnikova M.M., Vazhenova I.V., Pak N.I., Kirgizova E.V.* Developing students computational thinking with a recursive polydisciplinary approach // Journal of Physics: Conference Series. 2020. Vol. 1691. Article no. 012190. <https://doi.org/10.1088/1742-6596/1691/1/012190>
- [7] *Баженова И.В., Клуникова М.М., Пак Н.И.* Школьно-вузовский кластер дисциплин как средство развития расчетно-алгоритмического компонента вычислительного мышления // Информатика и образование. 2021. № 3. С. 42–49. <https://doi.org/10.32517/0234-0453-2021-36-3-42-49>
- [8] *Wing J.M.* Computational thinking // Communications of the ACM. 2006. Vol. 49. Issue 3. P. 33–35.
- [9] *Furber S.* Shut down or restart? The way forward for computing in UK schools. The Royal Society Education Section, 2021. <https://royalsociety.org/~media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf>
- [10] *Lyon J.A., Magana A.J.* Computational thinking in higher education: A review of the literature // Computer Applications in Engineering Education. 2020. Vol. 28. Issue 5. P. 1174–1189. <https://doi.org/10.1002/cae.22295>
- [11] *Пак Н.И.* Ментальный подход к цифровой трансформации образования // Открытое образование. 2021. Т. 25. № 5. С. 4–14. <https://doi.org/10.21686/1818-4243-2021-5-4-14>
- [12] *Gronow M., Mulligan J., Cavanagh M.* Teachers' understanding and use of mathematical structure // Mathematics Education Research Journal. 2020. Vol. 34. P. 215–240. <https://doi.org/10.1007/s13394-020-00342-x>
- [13] *Mason J., Stephens M., Watson A.* Appreciating mathematical structure for all // Mathematics Education Research Journal. 2009. Vol. 21. Issue. 2. P. 10–32. <https://doi.org/10.1007/BF03217543>
- [14] *Durak H.Y., Saritepeci M.* Analysis of the relation between computational thinking skills and various variables with the structural equation model // Computers & Education. 2018. Vol. 116. P. 191–202.
- [15] *Rich P., Egan G., Ellsworth J.* A framework for decomposition in computational thinking // Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, Aberdeen, 15–17 July 2019. New York: Association for Computing Machinery, 2019. P. 416–421. <https://doi.org/10.1145/3304221.3319793>
- [16] *Саймон Г.* Науки об искусственном / пер. с англ. Э.Л. Наппельбаума. 2-е изд. М.: Едиториал УРСС, 2004. 144 с.

#### Bio notes:

*Irina V. Vazhenova*, Candidate of Pedagogical Sciences, Associate Professor at the Department of Computing and Information Technologies, Institute of Mathematics and Computer Science, Siberian Federal University, 79 Svobodny Prospect, Krasnoyarsk,

660042, Russian Federation. ORCID: 0000-0001-6960-0408. SPIN-code: 9208-1141. E-mail: apkad@yandex.ru

*Margarita M. Klunnikova*, Candidate of Pedagogical Sciences, Associate Professor at the Department of Computing and Information Technologies, Institute of Mathematics and Computer Science, Siberian Federal University, 79 Svobodny Prospect, Krasnoyarsk, 660042, Russian Federation. ORCID: 0000-0003-3657-1019. SPIN-code: 9927-4184. E-mail: mklunnikova@sfu-kras.ru

*Nikolay I. Pak*, Doctor of Pedagogical Sciences, Professor, Head of the Department of Informatics and Information Technology in Education, Institute of Mathematics, Physics and Informatics, Krasnoyarsk State Pedagogical University named after V.P. Astafyev, 89 Ada Lebedeva St, Krasnoyarsk, 660049, Russian Federation. ORCID: 0000-0003-4163-9436. SPIN-code: 9943-2111. E-mail: nik@kspu.ru

### **Сведения об авторах:**

*Баженова Ирина Васильевна*, кандидат педагогических наук, доцент базовой кафедры вычислительных и информационных технологий, институт математики и фундаментальной информатики, Сибирский федеральный университет, Российская Федерация, 660042, Красноярск, пр. Свободный, д. 79. ORCID: 0000-0001-6960-0408. SPIN-код: 9208-1141. E-mail: apkad@yandex.ru

*Клунникова Маргарита Михайловна*, кандидат педагогических наук, доцент базовой кафедры вычислительных и информационных технологий, институт математики и фундаментальной информатики, Сибирский федеральный университет, Российская Федерация, 660042, Красноярск, пр. Свободный, д. 79. ORCID: 0000-0003-3657-1019. SPIN-код: 9927-4184. E-mail: mklunnikova@sfu-kras.ru

*Пак Николай Инсебович*, доктор педагогических наук, профессор, заведующий кафедрой информатики и информационных технологий в образовании, институт математики, физики и информатики, Красноярский государственный педагогический университет имени В.П. Астафьева, Российская Федерация, 660049, Красноярск, ул. Ады Лебедевой, д. 89. ORCID: 0000-0003-2105-8861. SPIN-код: 9943-2111. E-mail: nik@kspu.ru