Australian Educational Computing, 2019, 34(1).

Algorithmization and Programming Teaching Methodology in the course of Computer Science of Secondary School

Nurzhamal Oshanova
Abai Kazakh National Pedagogical University

Gulzat Anuarbekova
Abai Kazakh National Pedagogical University

Shirinkyz Shekerbekova
Abai Kazakh National Pedagogical University

Gylnar Arynova
Abai Kazakh National Pedagogical University

In this paper, the researchers have tried to solve one of the didactic tasks of an educational institution - the formation of a student's thinking, the development of their algorithmic style of thinking and intelligence. Since an important component of human intellectual development is precisely algorithmic thinking, learning to solve standard algorithmic problems is the primary goal of school education at various levels of computer science. Creating game moments that can be applied when solving problems from the "Basics of Algorithmization and Programming" section is aimed at motivating and increasing the efficiency of work in computer science lessons, as well as improving the quality of students' skills based on their cognitive interest. The use of ready-made and developed game moments when solving problems on the use of a structured array data type allows for a detailed analysis of algorithms in steps explaining the work of teams by the students themselves. Continued use of properly designed algorithms contributes to making students work focused and motivated.

Introduction

Engineers and computer scientists should have the skills to develop and optimize algorithms and experiments are one way to learn how to write optimized algorithms - to measure the temporal and spatial characteristics of a problem through an implementation of the algorithm. Using textbooks or similar non-interactive resources, one can easily study the data structures of basic algorithms, but to solve a problem effectively, it is useful to choose an algorithm and implement it fully. Developing and optimizing new algorithms for problem-solving is even more difficult when it comes to real-world problems.

Competitions on algorithmic problem-solving (Combéfis & le Clément, 2012) provide feedback to students to support their learning (Combéfis, Van den Schrieck, & Nootens, 2013) and developing learning algorithms is important not just for school students but university students as well (Combéfis & Wautelet, 2014).

García-Mateos and Fernández-Alemán (2008) explain how using programming skills can make courses more interesting and increase student motivation. In this article, the authors concluded that they are developing their own tasks in a competition style designed for educational purposes. Evaluation of the management of programming competitions by students (Leal & Silva, 2003) shows that the improvement of skills in the development and optimization of algorithms in programming should be introduced at an early stage of student development (Leal & Silva, 2008; Ribeiro & Guerreiro, 2008).

The importance of context in the study of computer science and technology has been investigated (Booth, 2001). The author sets out the theoretical foundations and considerations for the transition to a highly constructive pedagogy in higher education, which corresponds to the method proposed in this article.

The diverse and entertaining forms of learning conducted by studying through the use of interactive multimedia and electronic materials have a positive effect on the students' specific knowledge and skills (Urquiza-Fuentes & Velázquez-Iturbide, 2009; Jurinová, 2013; Jurinová, 2015; Avancena, Nishihara & Kondo; Herout, 2016). Students can memorize the subject better and be more motivated to learn the topic and improve their knowledge. Fundamentals of algorithms are the main subject for students of IT and various applications are created in order to facilitate the process of understanding and acquiring the necessary knowledge and skills of students on IT (Jurinová, 2015).

Materials and Methods

Introducing the concept of algorithm

Algorithmic thinking helps to develop the skills of programming in students. The study of algorithmization is the main focus, and when teaching this course, the teacher was especially careful when teaching algorithmization as this is the most important step in the computer science course.

In the standard of the basic course of education in computer science and ICT, the main content on algorithmization is carried out through the following concepts:

- Algorithm, methods of recording algorithms, properties of the algorithm;
- Executors of algorithms (purpose, environment, mode of operation, the system of commands);
- Computer is a formal executor of algorithms;
- Basic algorithmic constructions (following, branching, repetition);
- Splitting tasks into subtasks and auxiliary algorithm;
- Algorithms for working with values (data type, data input, and output) (Kurland, Pea, Clement, & Mawby, 1986).

The concept of an algorithm begins with the study of algorithmization. The algorithm is a mathematical concept and therefore cannot be defined through other, simpler concepts. In school textbooks, the definition of the algorithm is very diverse.

Based on practical examples from the life of students, the course “Algorithmization and Programming” is a practical orientation. The main executor at the initial moment of studying the topic can be a person. In the role of executors of simple algorithms should be the students themselves. The main characteristic of the executor, from the point of view of management, is the Executive Command System (ECS). To solve problems with the ECS, students can be given an algorithm that they will not be able to perform at first. After that, should follow the consolidation of the studied concept through the implementation of tasks for the definition of the ECS with various executors.

The effectiveness is the second important property, which is reflected in the definition of the algorithm. Effectiveness is the execution of the algorithm that will be obtained from the desired result and must be completed in a finite number of steps. Here, the step refers to the execution of a separate command. In this case, this property reflects the situation when the algorithm “loops” and does not give a result. Such an algorithm is useless, and students should learn to distinguish between these algorithms (Yoo J., Yoo S., Seo, Don, & Pettey, n.d.).

Discreteness is the third property of the algorithm. Discreteness of the command of the algorithm is performed sequentially, with exact fixation of the moments of the completion of each command and the start of the next one. The main requirement of discreteness is the sequential execution of commands (Burton, 2010).

The mass character is the fourth property when for solving a particular given problem, the algorithm must once be applied to any specific formulation. When solving this problem in regards to the initial data, the mass character is the universality of the algorithm.

Another property is a personal computer, which is an automatic executor. The students themselves can call such automatic executors: robots, automatic machines, automatic washing machines, microwave ovens, and so on.

After all the properties of the algorithm are considered, they should be fixed through the execution of tasks. For this it is useful to consider with the students' several tasks of the following content:

1. The algorithm is given, formally execute it;
2. Determine the executor and the command system for this type of work;
3. According to the given command system to build an algorithm;
4. Determine the required set of source data for solving the problem.

Teaching of constructing methods of algorithms on training executors

An important goal of the algorithmization section is that students master the structural method of constructing algorithms. A didactic tool in algorithmization is the educational implementers of algorithms. The advantages of these executors are clarity for the student of the tasks to be solved, visualization of the work process during the execution of the program. The didactic principle of visualization is one of the most important in the process of any training.

In order to make it easy and convenient for students to work with training executors, they must meet the following requirements:

- There must be an executor working “in the situation”;
- The executor must imitate the process of controlling some real object (turtle, robot, etc.);
- In the executive command system must be all structural control commands (branches, cycles);
- The executor can use auxiliary algorithms (procedures).

By studying the work of the performer of algorithms, the teacher should give his characteristics, which are called the architecture of the executor. They are:

- Environment in which the executor works;
- Its mode of operation;
- ECS;
- The data he works with.

It is desirable to organize training in programming in the course of solving problems, selected in a specially constructed sequence, which meets the following didactic principles:

- Gradual complication of the tasks being solved, i.e. from simple to complex;
- Novelty, each task must introduce a new element of knowledge, a command, a programming technique.
- Inheritance, that is, the solution of each following problem requires the use of the knowledge gained in solving the previous.

When writing algorithms for training executors, an algorithmic language and flowcharts are used. You can get acquainted with them at one or two lessons, and then continue the study of algorithmization and flowcharts in conjunction with the construction of algorithms for training executors. This will help to study the basic algorithmic structures from the theoretical and practical side.

The main advantage of flowcharts is the visibility of the algorithm representation. It is achieved by displaying flowcharts in a standard way from top to bottom.

Algorithmic language is a textual description of the algorithm, which is close to the programming language, but as such is not, and therefore does not have a strict syntax. To structure the text of an algorithm in an algorithmic language, lowercase indents are used. In this case, the following rule is observed: all constructions of the same nesting level are written on the same vertical level (indent), and nested constructions are shifted relative to the outer right. This rule makes the visual structure of the algorithm. Therefore, it is advisable for the teacher to spend some training time on the formation of the skill of correct recording of the algorithm (Saeli, Perrenet, Jochems, & Zwaneveld, 2011).

After getting acquainted with the architecture of the executor and the methods of recording the algorithms, one should proceed to solve the problems corresponding to the above didactic principles. In this case, only practical work on training executors helps to master the construction of algorithms.

At practical lessons the following types of tasks are used:

- Development of linear algorithms;
- Development and use of supporting algorithms;
- Development of cyclic algorithms;
- Development of branched algorithms;
- Usage of the method of sequential detail in the development of complex algorithms.

Of course, the first tasks should be in a linear structure, for example, in a training executor to draw a letter.

When analyzing the given task, it is desirable to draw the attention of students to two circumstances:

• Management of the executor to achieve the goal will occur without feedback. In this case, the algorithm will have a linear structure.

• The algorithm depends not only on the stated goal (the desired result), but also on the original state of the contractor. The state of the training executor is determined by its location on the field and its orientation. And the result of the algorithm is not only the drawing (the main goal) but also the final state of the executor.

The following tasks should help to develop and use supporting algorithms. In such cases, the following task is usually considered: develop an algorithm for drawing the number “282828”.

Solving this problem, you can do the following: invite students to write an algorithm using the same means. Such a task, obviously, will not arouse the enthusiasm of the students, since the principle is already clear to them, and writing a long such algorithm is rather boring. In this situation, an independent “discovery” of the idea of an auxiliary algorithm by students is possible. Drawing attention to the fact that in the figure there are figures “2” and “8” three

times, students can come up with the idea of a separate description of the algorithms for drawing these numbers, then students need to give the concept of an auxiliary algorithm and explain how its description and usage is conducted.

Skills of using auxiliary algorithms must be developed in students as early as possible, using examples of linear algorithms. The most important method of algorithms and programming is the decomposition of the problem, i.e. the selection of some simpler subtasks in the original problem. Algorithms for solving such subtasks are called auxiliary algorithms, and the programs that implement them are called subprograms (procedures). Thus, the solution to the problem is divided into several algorithms such as the main algorithm and auxiliary algorithms. Usually, in the main algorithm, there is a repeated appeal to the auxiliary algorithm (Bell, Alexander, Freeman, & Grimley, 2009).

The following is a consideration of the cyclic structures of the algorithm. For their development, you must first theoretically prepare students. It is necessary to examine and analyze in detail the cyclic algorithms with the help of flowcharts and a verbal algorithmic language. And only then go on to practice, otherwise, children cannot assimilate the cycles, and act on examples, without thinking about the content of the task.

An example of such tasks for cycles is the task of developing an algorithm for drawing a horizontal line drawn from edge to edge of a field. This task introduces the following new elements to the given topic:

- Board with feedback;
- Structural command of the cycle.

The feedback is that before performing each step, the condition “is there an edge ahead?” is checked. In the case of truth, i.e. the answer is positive, then a step is taken, otherwise, the execution of the cycle is terminated.

The team of a cycle is a structural team in contrast to the simple commands such as “step”, “turn”, “jump”. The structural command includes several actions such as checking the condition, executing the loop body, which, in turn, may consist of several commands.

Finally, the study of basic algorithmic structures ends with branching. Here you can offer the following task to depict an ornament consisting of squares located along the edge of the field. Using the example of this task, the technique of sequential detailing is once again demonstrated. Unlike previous programs, it uses two steps of detail, since the ROW procedure contains an appeal to the procedure of the next level - SQUARE.

Methods of learning programming languages are well developed. Programming languages are divided into two large groups such as machine-oriented (Autocode, Assembler) and high-level languages. The first group of languages is used by a very small number of professional-level programmers for specific purposes. Most programmers are currently using high-level languages.

Results and Discussion

Programming in the course of Computer Science and ICT

At the beginning of the study, the topic should focus on the definition of the program, and then programming. Programming is a computer science section that studies the development of computer software and other technical systems. In a narrow sense, programming means the process of developing a program in one of the programming languages. The development of tools for system software and programming systems is called system programming. Creation of applied computer programs is called application programming. By the same principle, programmers are divided into system and application programmers.

Education programming is desirable to organize in various high-level languages, for example, Pascal. Such a language is focused on a structural programming methodology.

To learn Pascal, the program "Pascal ABC" is used, which greatly simplifies the process of typing and editing a program. It can be problematic for pupils to switch from flowcharts and an algorithmic language to Pascal programming right away since they do not look like Pascal programs. There is also a difficult syntax to which students cannot get used to (setting brackets and commas in the right places). In order to avoid these difficulties, programming in an algorithmic language can be studied. You can use the program "Kumir". This program allows you to write in an algorithmic language, and also includes graphic executors. Therefore, it will not be difficult for students to move from a programme preparation for executors to programming.

In the program "Kumir" in the algorithmic language, you can study the entire course of programming. Since there is a work with values, logical operations, choice operator, cycles, work with strings in it. One of the main advantages of the program "Kumir" is the use of an algorithmic language for writing programs. This simplifies the explanation process. Students can simply read written for the example program and see what it will do. Which of course is impossible when learning other programming languages, since they are based on English.

At the stage of studying graphic executors, it is advisable to introduce students to the "Kumir" environment. At the beginning of the acquaintance, it is only necessary to briefly describe the components of the system, noting that they will be considered in more detail as the topic progresses.

Students should be informed that creating a program consists of three stages such as writing a program, debugging a program, and executing a program. The programming system allows you to do this in a more productive way through the use of special tools and ready-made developments of parts and blocks of the program.

Such following components as the environment, operation modes, command system, data can be allocated in any of the programming systems. Students should be briefed on them.

The modes of operation of the programming system are usually:

- Program editing mode;
- Compilation mode of the program text;
- Execution mode;

- Mode of working with files;
- Help mode;
- Debugging conditions.

By presenting the material, students should pay special attention to the fact that in each mode of operation a certain system of commands is used. For the programming system, the data are the files with the texts of the programs containing the initial and final information for the task.

In edit mode, the built-in editor is usually used, which allows you to write the text of the program. The text can also be prepared in any test editor and work with the students writing program skills.

In compilation mode, the program is translated into machine code. In this case, the program is collected from various blocks, modules, usually taken from the programming system library. As a result of compilation, an object file is created, which is a part of the program in machine language with the necessary external links and links.

In the execution mode, the program received after the broadcast is executed. Usually, the interpreter (which is one or another type of translator) directly executes the program itself in a high-level programming language.

In the file mode, the usual operations are performed such as save the file, read the information from the file into RAM, assign a name to the file and display the contents of the editor window on the screen, and then print.

In the help mode, the programmer can get a hint on the screen, both on how to work with the system and on the programming language.

The debugging conditions are always implemented in modern programming systems. In this mode, you can trace the program (display the results of each command), step-by-step program performance, track changes in certain values, find and correct errors.

The reinforcement of the theoretical material of this topic is recommended for practical training on writing short programs in the programming system “Kumir”.

After students get acquainted with the programming environment, you can proceed to the development of programs.

Training in programming should be conducted on examples of typical tasks with a gradual complication of the structure of algorithms. On the basis of the algorithmic structure, they can be divided into classes:

- Linear algorithms: calculations by formulas, all sorts of variable value transfers;
- Branching algorithms: search for the highest or lowest values from several data; sorting two or three values; dialogue with branches;

- Cyclic algorithms: calculation of the amounts and products of numeric arrays, cyclic data entry with sequential processing (Robins, Rountree, & Rountree, 2003).

Thus, the basic training in the field of computer science regarding the sections of algorithmization and programming includes the range of issues discussed above, and the development of educational material provides students with such opportunities as:

- Understand (based on the analysis of examples) the meaning of the concept of an algorithm, study the properties of the algorithm and consider the possibility of automating human activity during the execution of algorithms;
- To master the basic algorithmic constructions (cycle, branching, procedure) and apply them to construct algorithms and solve educational problems;
- Get an idea of the "library of algorithms", learn how to use these algorithms to build more complex algorithms;
- Get an idea of one of the programming languages (or training algorithmic language) and use it to write algorithms for solving simple problems.

Upon completion of the study of the topic, students should:

- Understand the essence of the concept of an algorithm, know its basic properties, use them on examples of specific algorithms;
- Know the possibilities of automating human activity when applying algorithms;
- Know the basic algorithmic structures and be able to apply them to the construction of algorithms;
- Know the possibility of applying the executor to solve a specific task on the command system, to develop and execute on the computer an algorithm for a training artist (such as "turtles", "robots", etc.)
- Write an algorithm for solving a simple problem in a training algorithmic language (or programming language);
- Have an idea of the variable as a room for ECM;
- Have an idea of the array as a combination of the same type of data;
- Know the nature of the parameter change during the execution of the cycle;
- Understand the process of executing programs containing calls to subroutines;
- Know the parameters of the variable (name, type, value);
- Know the standard functions, the rules for determining their user;
- Know the rules of writing and the order of execution of logical expressions;
- Know the procedure for defining subroutines and referring to them;
- Know the order of the description of data arrays;
- Be able to describe the processes in the job of the value of a variable using an assignment operator;
- Be able to describe the formats of the simplest operators that provide keyboard input and character output to the screen;
- Be able to describe the formats of standard functions, types of arguments, types of values; define user functions, use them in expressions;
- Be able to describe the conditional operator and the algorithm of its execution in full and incomplete variants and write down simple branching algorithms in the form of programs;

- Be able to describe the format of the operators of the organization of cycles and write simple cyclic algorithms in the form of programs;
- Be able to describe the forms of graphical operators and use these operators to create simple images;
- Be able to organize the input and output of an array of data, as well as distinguish between the index and the value of the array;
- Possess simple debugging techniques of programs.

Let us give an example of the calculation of a problem based on national peculiarities. For example, the Task of three batyrs and the Black Man. Batyrs Alpamys, Kobylandy, and Yer Targyn fought with several opponents. Each made 3 hits, as a result, all the opponents ran away. The greatest blow was made by Alpamys batyr - 7 blows, the smallest Targyn batyr - 3 blows. Construct a program that calculates the number of numbers.

First of all, we can create a flowchart f[illegible]lating.

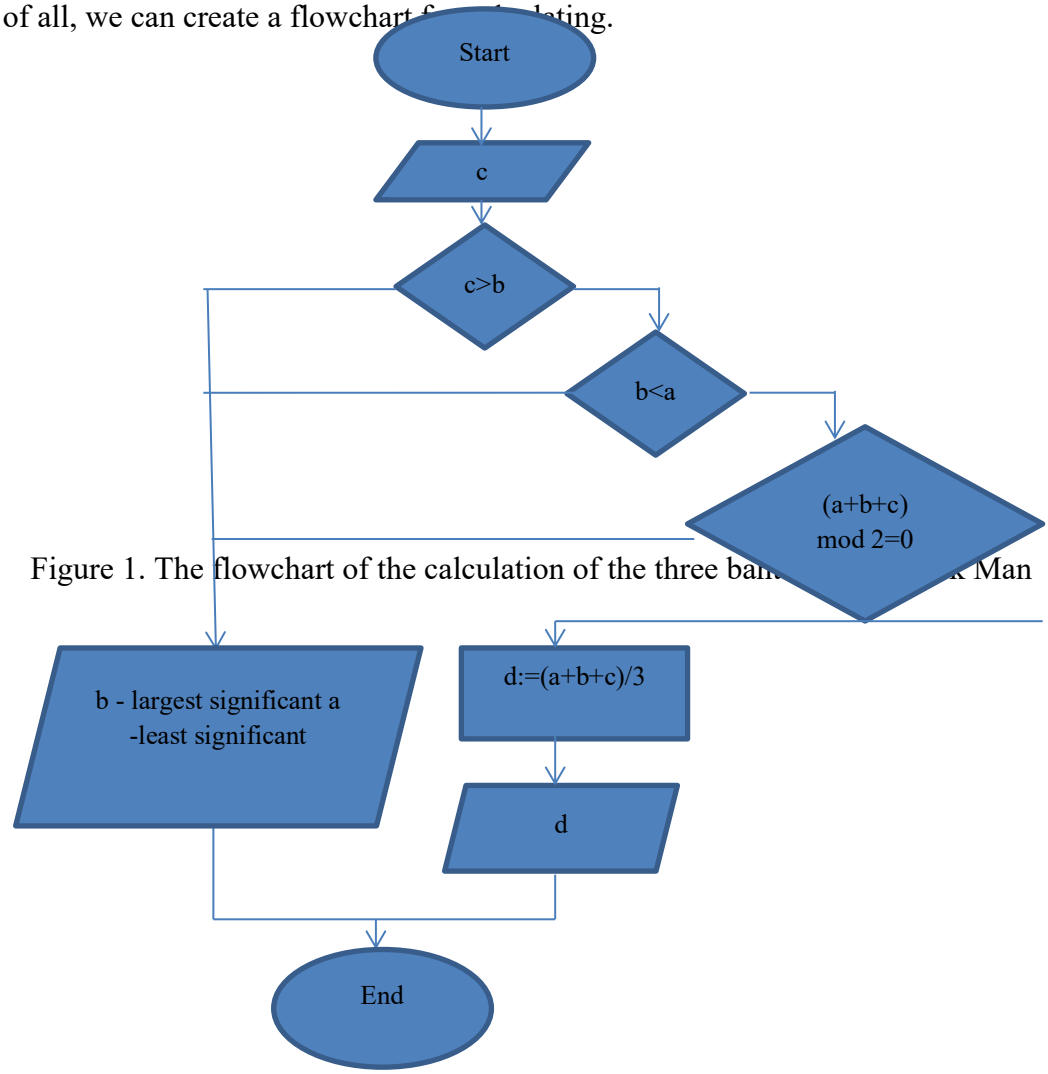

Figure 1. The flowchart of the calculation of the three ba[illegible] Man

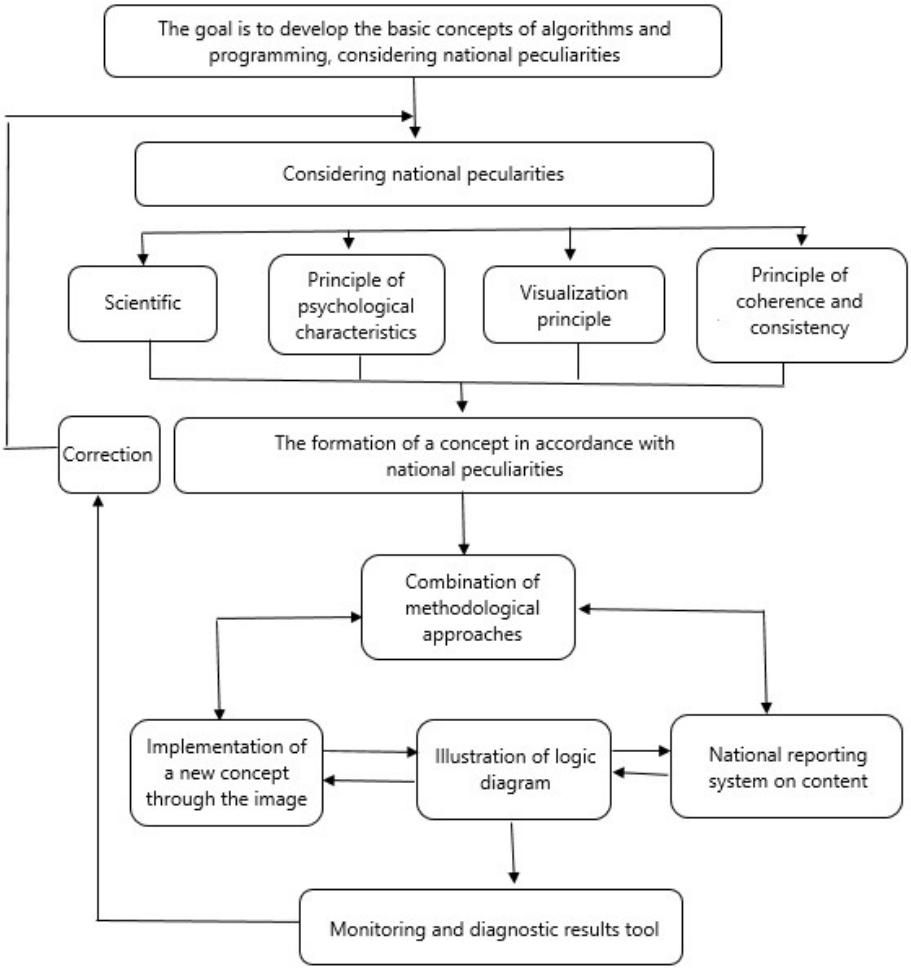

Figure 2. Structural and logical modeling of the basic concepts of algorithms and programming, considering national peculiarities.

The result of the program is shown in Figure 3.

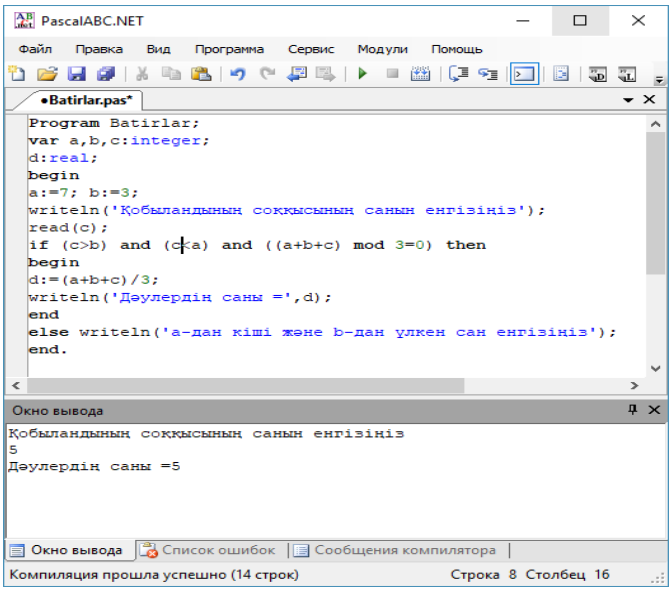

Figure 3. The program counts the number of Black People

As result, researchers consider it expedient to use a system of tasks based on national characteristics in teaching algorithms and programming in high school.

Conclusion

In the given paper, the researchers decided one of the didactic tasks of an educational institution - the formation of a student's thinking, the development of his algorithmic style of thinking and intelligence. Since an important component of human intellectual development is precisely algorithmic thinking, learning to solve standard algorithmic problems is the primary goal of school education at various levels of computer science.

As a result of the work, the methods of constructing and using algorithms for solving standard problems from the section “Basics of algorithms and programming” were considered on the example of training tasks for working with the structural data type array. Since a variety of ways and forms of building work in the classroom helps to simplify cumbersome and monotonous work in solving standard problems.

References

Avancena, T., Nishihara, A., & Kondo, Ch. (2015). Developing an Algorithm Learning Tool for High School Introductory Computer Science. *Education Research International*.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: school students doingreal computing without computers. *Journal of Applied Computing and Information, 13*(1), 20–29.

Burton, B. (2010). Encouraging algorithmic thinking without a computer. *Olympiads in Informatics, 4*, 3–14.

Booth, S. (2001). Learning computer science and engineering in context. *Computer Science Education*, *11*(3), 169–188.

Combéfis, S., & le Clément de Saint-Marcq, V. (2012). Teaching programming and algorithm.

Combéfis, S., Van den Schrieck, V., & Nootens, A. (2013). Growing algorithmic thinking design with Pythia, a web-based learning platform. *Olympiads in Informatics*, *6*, 31–43.

Combéfis, S., & Wautelet, J. (2014). Programming trainings and informatics teaching through online contest. *Olympiads in Informatics*, *8*, 21–34.

Diehl, S. (2007). *Software visualization: Visualizing the Structure, Behaviour, and Evolution of Software*. New York: Springer.

García-Mateos, G., & Fernández-Alemán, J. L. (2009). Make learning fun with programming contests. *Transactions on Edutainment II*, *5660*, 246–257.

Herout, L. (2016). Elektronické studijní opory v prostředí terciárního vzdělávání. Praha: powerprint. ISBN 978-80-7568-016-7.

Jurinová, J. (2013). Rozvoj kognitívnych vedomostí za využitia multimediálnej učebnej pomôcky. In: *XXVI.DIDMATTECH 2013 New Technologies in Science and Education*. Gyor: University of West Hungary.

Jurinová, J. (2015). Instruction Videos for Psychomotor Skills Development. *R&E-source: časopis pre výskum a vzdelávanie*, *4*, 129-133. ISSN 2313-1640.

Kurland, D. M., Pea, R. D., Clement, C., & Mawby, R. (1986). A Study of the Development of Programming Ability and Thinking Skills in High School Students. *Journal of Educational Computing Research, 2*(4), 429–458.

Leal, J. P., & Silva, F. (2003). Mooshak: a web-based multi-site programming contest system. *Software: Practice and Experience*, *33*(6), 567–581.

Leal, J. P., & Silva, F. (2008). Using Mooshak as a competitive learning tool. In: *Proceedings of the ACM-ICPC Competitive Learning Institute Symposium (CLIS 2008). Through interactive problems to encourage learning programming. Olympiads in Informatics, 7,* 3–13.

Ribeiro, P., & Guerreiro, P. (2008). Early introduction of competitive programming. *Olympiads in Informatics, 2*, 149–162.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education, 13*(2), 137-172.

Saeli, M., Perrenet, J., Jochems, W. M. G., & Zwaneveld, B. (2011). Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective. *Informatics in Education - An International Journal, 10*(1), 73-88.

Urquiza-Fuentes, J., & Velázquez-Iturbide, J. A. (2009). A survey of successful evaluations of program visualization and algorithm animation systems. ACM Transactions on Computing Education (TOCE). *5th Program Visualization Workshop, 9*(2), 1–21.

Yoo, J., Yoo, S., Seo, S., Don, Zh., & Pettey, C. (n.d.). Teaching Algorithm Development Skills. *Semantic Scholar*.