computational algorithms, but on examples of algorithms control of real and virtual moving objects" [2]. This thesis - mastering the basics of programming by developing programs to control moving objects - can be achieved with an emphasis on real objects and their digital counterparts, or with an emphasis on purely imaginary objects represented by sprites on a two-dimensional screen.

The latter approach, an emphasis on sprite management, is implemented in the famous Scratch [3] and Scratch Junior [4; 5] systems and other systems based on the Blockly platform [6]. An approach with an emphasis on controlling real robots and their screen counterparts, implemented in popular robotics kits like Primo toys [7]; Matatalab [8], KIBO-robot [9], LEGO Education MINDSTORMS EV3 [10].

B. Methods for Representing and Generating Programming Code

An important characteristic of a learning programming environment is the method it provides for the presentation and generation of program code. The full-text method of code creation, inherited from professional programming systems, is not suitable for beginners of any age, forcing them to spend time entering program text character by character, searching for and correcting syntax errors. In the world, two approaches have been proposed that allow one to get away from the full-text representation of the program and the character-by-character input of its text: icon-base programming and block-based programming.

Comment. Below we will use the term *pictogram-based* as a synonym for *icon-based*.

C. Pictogram-based Methods Representing and Generating Programming Code

In pictogram-based programming, the user represents the program with one or more lines of pictograms. Pictograms (icons) are atomic, indivisible objects selected by the user from some predefined set and placed in the program in the required order. This approach is used in many popular systems such as the Lightbot game [11] and the Scratch Junior environment [5]. The pictographic approach has a number of advantages, both appreciated and underestimated. The fully appreciated advantages include the ability to create programs in the material world by mechanical manipulations with tangible objects. When a program is represented in pictograms (icons), it can be created not on the tablet screen, but by manually laying out material carriers of pictograms. These tangible carriers of pictograms can be blocks with electrical connectors placed in the recesses of a special electronic device such as, for example, in the robotic set The Cubetto Playset [7]. Or passive cards with images recognized by special devices included in the robotic kit as in the Matatalab kit [8].

An underestimated advantage of the pictographic approach is the ability to learn with it a full set of structured programming constructs: sequences, loops, conditionals, functions (subroutines) and, finally, counters. When learning the basics of structured programming, one should use not so much isolated structured programming constructs as their commonly used combinations: a conditional statement inside a while loop, nested loops, calling functions inside a function, etc. Therefore, to study structured programming constructs, we need simple intuitive means to set in the pictographic program of nested block structures. Below we will talk about our approach to defining such structures.

D. Block-based Methods Representing and Generating Programming Code

The second approach is a block representation of the program and the use of a syntactically oriented editor that allows you to easily, quickly and accurately add not symbols or words to the program, but ready-made syntactically correct control structures and blocks. This approach, described, for example, in [5],[12] and [13], turned out to be very successful and is widely used today. However, with the accumulation of experience in using the block method of creating programs for different audiences, it became clear that the transition from "child-oriented" block programming with graphical input of program code to "adult", professional programming with input of textual information from the keyboard is not easy both technologically and psychologically.Ways to overcome the difficulties of transition from block programming to textual programming are the subject of works [14], [15] and [16].

Technologically, this transition is difficult, since a) changes the visual form of the image of the program code, b) changes the way of entering the program code, c) sets the task of understanding and correcting syntax errors, which was completely or largely automatically solved by the block programming environment. Psychologically, the problem lies in the fact that children from a certain age, on the one hand, perceive block programs as "toys" and strive to master a "real" style of programming based on text, and on the other hand, they are afraid that real programming is much more difficult than toy programming and will be for them too complicated [16].

The choice of methods for representing and generating program code in the initial programming course and the number of lessons conducted using one method or another depend on the objectives of the course.

II. RUSSIAN EXPERIENCE. MANDATORY REQUIREMENTS FOR THE RESULTS OF MASTERING PROGRAMMING BY 9TH GRADE GRADUATES IN ACCORDANCE WITH THE RUSSIAN EDUCATIONAL STANDARD

In accordance with the legislation of the Russian Federation on education, compulsory, with the exception of cases provided for by law, is the acquisition of primary education (grades 1-4, age 7-11) and the acquisition of Basic General Education (russian acronim OOO - Основное Общее Образование) (grades 5-9).

By law, 80% of the content of education is determined by federal educational standards. The current standards for preschool education and primary education do not provide for compulsory study of programming. The 2021 revision of the Basic General Education Standard requires compulsory study of programming at a basic level and also standardizes possible advanced study. Here is a literal translation of the two clauses of the standard that define mandatory programming skills at a basic level [17, section 45.5.3.]

45.5.3.6) *the ability to compose, manually and on a computer, simple algorithms for managing robots (Turtle,*

Draftsman); create and debug programs in one of the programming languages (Python, C++, Pascal, Java, C#, School Algorithmic Language) that implement simple algorithms for processing numerical data using loops and branches; the ability to split tasks into subtasks, use constants, variables and expressions of various types (numeric, logical, symbolic); analyze the proposed algorithm, determine what results are possible for a given set of initial values;

45.5.3.7) *the ability to write in the studied programming language algorithms for checking the divisibility of one integer to another, checking a natural number for simplicity, separating digits from a natural number, searching for maxima, minima, the sum of a numerical sequence;*

A. Fulfillment of most of the requirements 45.5.3.6 and 45.5.3.7 of the standard is possible using only pictogram-based and block-based programming environments

The School Algorithmic Language, included above in the list of 6 programming languages, is a Pascal-like programming language with Russian-language vocabulary, proposed by Academician A.P. Ershov in 1985 with the introduction of a new subject in Russian schools, "The Basics of Computer Science and Computer Technology." Thus, work in any of the 5 professional programming languages allowed by the standard, as well as work in The School Algorithmic Language, requires the use of a full-text programming environment - the use of a block programming environment is not provided for by the standard. However, an analysis of the concepts of the mandatory basic level part of the standard shows that most of these concepts relate to structured programming and working with simple variables and do not require working with complex data structures with direct acces. Therefore, most of the concepts prescribed by the standard can be successfully studied in block and even iconic programming environments. For example, solving the most difficult programming problem for the 9th grade final exam in computer science in Russia in 2021 uses only conditions and WHILE LOOP constructions and therefore can be easily implemented in a programming environment based on PiktoMir icons. (Fig. 1)

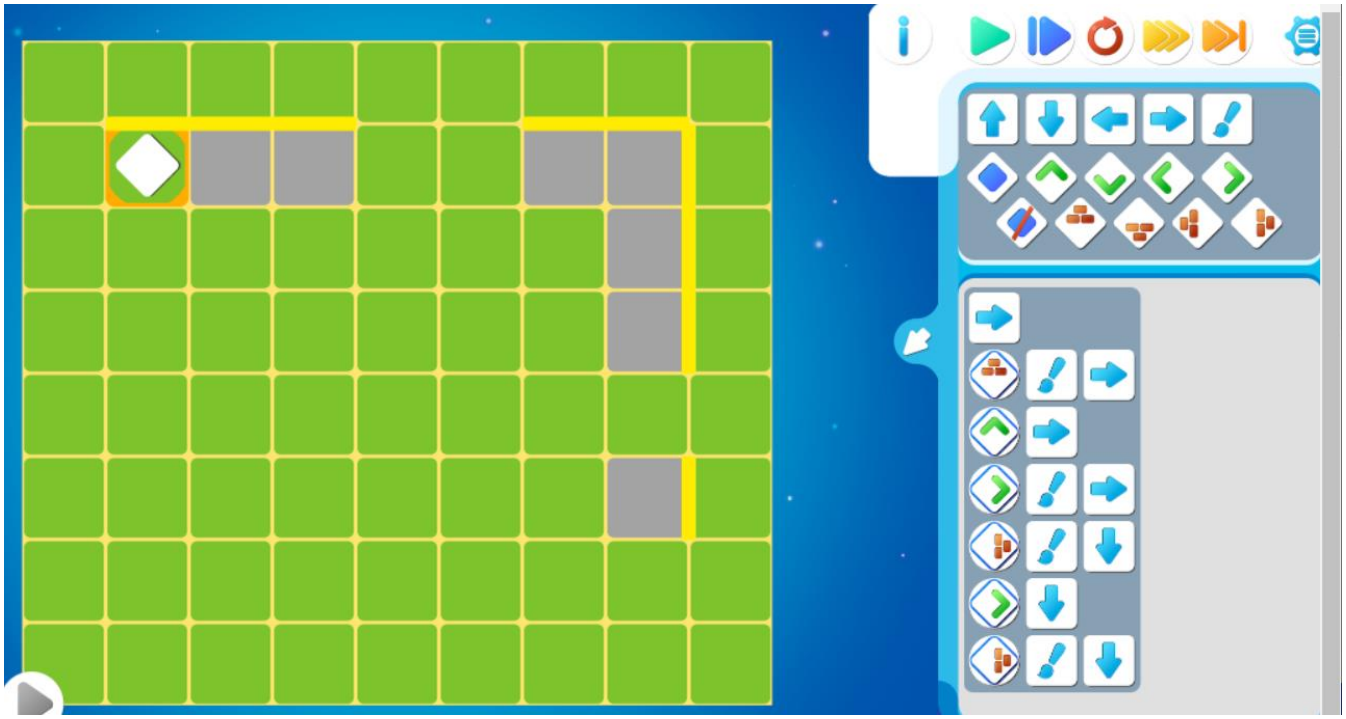

Fig. 1 The robot, shown in the form of a diamond, must walk along the wall, consisting of one horizontal section, in which one pass was made and one vertical section, in which a second pass was made, and paint over the cells adjacent to the wall. The length of the wall sections and passageways is unknown.

Thus, the basic programming concepts described in the above standard can be equally successfully mastered by 9th grade graduates and primary school graduates using pictogram-based and block-based programming environments. The requirement to master a text-based programming environment in one of the 6 languages listed in the standard should be considered as a reasonable additional requirement that can be easily fulfilled in the final part of the course, provided that the language and text-based programming environment are close to a block environment.

B. Why the fulfillment of the requirements of the standard is impossible without the use of pictogram-based and block-based programming environments

The minimum core of structured programming concepts formulated in the standard should be mastered volens nolens in any conceivable introductory programming course. Just as learning arithmetic requires solving at least several hundred exercises, and learning to write requires writing hundreds of lines, learning to code requires completing tasks of composing and debugging several hundred simple programs. Therefore, a significant share of efforts in the development of the ABC of Coding course should be aimed at achieving the following goal: to create conditions in which each

student independently performs several hundred programs with a minimum of time and effort during training. tasks. Achieving this goal is impossible if we restrict ourselves to full-text programming environments. Not a single such system gives a teacher the opportunity to organize a student's solution to hundreds of simple problems in a dozen lessons. Full-text programming environments are focused on solving more complex problems and, no matter how convenient they are for the user, they do not provide adequate student performance in solving simple exercises. The desired performance at the beginning of the course can be achieved using pictogram-based and block-based programming environments. Some data to support this claim will be given below.

III. Draft Introductory Programming Course for Primary School

Over the past 10 years, the Department of Educational Informatics of SRISA RAS has been developing methods and tools for initial teaching programming to preschoolers, schoolchildren, students of natural science and pedagogical universities. Employees of the department work in Moscow State University, Moscow State Pedagogical University, State University of Maanagement, kindergartens and schools in Moscow, supervise experiments in teaching programming to preschoolers in the city of Surgut, Samara region and Tyumen region. Every year we work with several thousand students in dozens of educational organizations. The accumulated experience has shown that the trajectory of teaching programming, which ultimately leads to the acquisition of skills in a full-text "adult" programming environment, must go through the stages of pictogram (iconic) and block programming and the greatest difficulties arise at the junction of the stages. These difficulties are also noted by other researchers [18].

As a result of many years of experimentation, we have developed a three-stage introductory programming course project for primary school students that uses a family of three educational software environments based on pictogram-based, block-based and full-text approaches. Our decision to include practice full-text programming in the course is not obvious. There are arguments for and against this decision [16]. Our choice was partly due to external reasons - the desire to fully comply with the spirit and letter of Russian educational standards, but mainly - the desire to facilitate the transition to traditional programming for those course graduates who later choose advanced learning programming. In accordance with world and domestic experience, the basics of programming are first set out using examples of programs that control various virtual and real robots, and only at the final stages do various tasks of processing text and numerical information begin to be solved. As students age, the methods available to them for representing code and developing code change.

A. *First stage*

The basic set of control structures of programming languages is mastered by preschoolers and primary school students in free educational programming environment called PiktoMir [19; 20], by compiling programs for controlling real or virtual toy robots. Work on the PiktoMir environment began 10 years ago and the current version of the system takes into account ten years of operating experience in a real educational process. Programs in this environment have a non-textual representation. Programming is done by dragging and dropping icons onto the touch screen or manually laying out tangible media of icons, some tangible objects such as magnetic cards, wooden cubes, etc., on a table or on a magnetic board, followed by photographing the configuration of the icons with the child's tablet and recognizing the structure of the program. In none of these modes PiktoMir does not allow the child to make a syntax error. The language supports subroutines with a predefined one-letter name, branching and repetition constructs.

As already mentioned, nested block structures are supported in PictoMir. Attempts to define block structures by introducing block-terminating icons, such as in [9], have the disadvantage that syntax errors are possible (see Fig. 2)

Fig. 2 Nested loops in KIBO environment. The inner loop REPEAT 2 TIMES is nested in the outer loop REPEAT 4 TIMES.

We manage to avoid this by using indentation to indicate the end of a block, as in Python. Please note that the computing power of modern mass tablets is so great that programs assembled from images, similar to the program shown in Fig. 3, can be recognized by the neural network module in a split second without the help of any QR codes, which cannot be deciphered by a child.

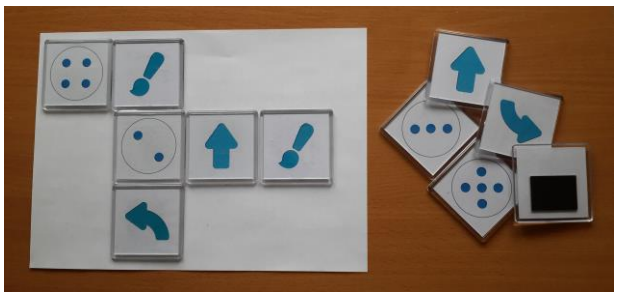

Fig. 3 Nested loops in PiktoMir environment. The inner loop REPEAT 2 TIMES is nested in the outer loop REPEAT 4 TIMES.

The generalization of the semantics of horizontal indentation used in Python, which we have invented, allows us to consider it syntactically correct and execute absolutely any program laid out from square icons in the cells of an imaginary rectangular table. This semantics will be formally described in a separate publication. A meaningful examples of using indentation to define nested block structures can be found in Fig. 3 and Fig. 9.

The current version of PiktoMir allows you to control one real and five virtual robots (see Fig. 4). An example program for the Vertun robot is shown in Fig. 6. Our experience has shown that preschoolers from 6 years old and first graders easily master the process of drawing up programs for these robots and are able to independently compose 120-150 programs per year, provided that one half-hour lesson is held weekly (see Results). It is worth mentioning that 50% of the time of each lesson is devoted to various unplugged

activities, so the screen time is about a quarter of an hour. (see Results).

This first stage of learning described in detail in the articles [21; 22] and can be carried out in kindergarten or in the early years of school.

Most of PiktoMir's robots are united by a space legend. According to this legend, robots perform some work on passenger and cargo mobile space platforms orbiting the planet. For example, the Vertun robot repairs the cracked plates of the space platform after takeoff by painting them with fire-resistant paint.

PiktoMir also supports a collaborative programming process. While carrying out a common task, two children - each at his workplace on his tablet - compose two programs to control two robots operating in a common environment. When robots work together, they have to do some common work.

The completed course of textless, pictographic programming takes about a year and a half and allows all children, without exception, to master the basic constructions of sequential programming, techniques for drawing up and debugging programs, and gain experience in cooperative solution of programming tasks. In [23] and [24], we explain why, in our opinion, such introductory course of textless programming should be included in the compulsory curriculum of kindergartens and primary schools.

Fig. 4 Five virtual robots in PiktoMir and PiktoMir-K environments

The didactic unit in PiktoMir is the Game, which consists of about ten methodically related programming tasks. The implementation of these tasks is carried out in the form of passing the levels of the game. The game is played during one lesson, homework is not assigned. At this stage, in a comfortable pictogram-based programming environment, the standard control structures of procedural programming languages are systematically and deeply mastered and with their help a corpus of about one and a half hundred problems is solved to compose control algorithms with feedback. The main problem of this stage is to avoid the child's loss of motivation locally, during the lesson, or globally, during the course after one or two dozen lessons. The local problem is mitigated by providing the child at each level of the Game with some program template that the child will have to fill (Fig. 5 and Fig 6). Thanks to the templates, the child quickly passes the first levels of the Game proposed in this lesson and strives to pass the following levels of the Game.

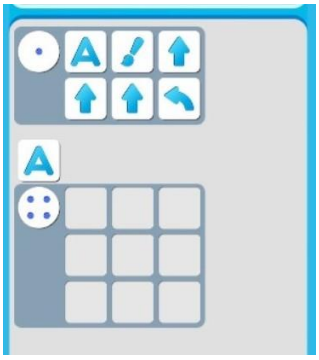

Fig. 5 An example of a partially filled program template

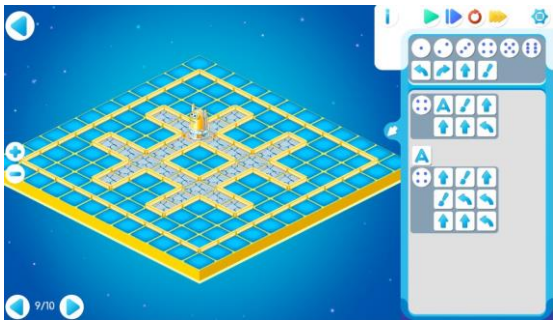

Fig. 6 An example program in the PiktoMir environment, created using given template

The global problem is countered by the alternation of different types of activities: teamwork with a real robot, co-programming in pairs, individual or team competitions. It is also important to have a large number of attractive graphically virtual robots. . Acquaintance with each of them is carried out in the form of imitation of the behavior of this robot by children. This is facilitated by the presence of plush toys that accurately represent screen robots (see Fig. 7.).

Fig. 7 Plush toys Vertun, Dvigun and Tyagun and toy robot Polzun

But the most important thing is the attractiveness and accessibility for the child of the tasks offered at each lesson, ensuring success for each child in each lesson. The tasks in the PiktoMir system are quite interesting and diverse. For example, although there are no variables in the PiktoMir system, the virtual device “Magic Jug with Stones” allows you to visually simulate the counter and solve problems such as "reach the obstacle and return to the starting point." During the transition to block programming, this Magic Jug will help children quickly learn the concept of an integer variable (see Fig. 10 below). The pictographic form of the program presentation is also good in that it allows you to compactly show on one screen, without rolling, several dozen commands.

PiktoMir allows you to execute programs continuously and step by step. Linear sections of the program can be created as protocols for remote control of the robot with command-by-command haulage (the so-called Piggy bank mode). Thus, already at the first stage of training, children get acquainted with the methods of debugging.

B. Second stage

From our point of view, the most important property of a block programming environment for novice students and for a teacher leading a large group of novice students is not clarity and graphical appeal, but the lack of the ability to make a syntax error; the main task of the stage of transition from pictogram representation to block-text is to develop skills for quick perception of the structure of the program from its text, these skills can be effectively developed only if the semantics of control structures of structured programming is fully understood by the learner and external hindrances to this perception due to the appearance of syntactic errors are completely absent. Therefore, at the next stage, the development of programs is carried out in the PiktoMir-K block environment with a hybrid graphical-textual representation of the program, close to the traditional textual representation of the program, but with non-textual input of the program code through dragging and dropping the graphically represented blocks. Unlike traditional block environments, in our PiktoMir-K environment, when a block is released at the right place in the program, its graphical representation turns into a textual one, slightly supplemented with graphical markings. As already mentioned, the most important property of this environment, like many other block environments, is its syntactic safety - in these environments, a student, in principle, cannot make a syntax error.

Programming in general and block programming in particular is an excellent tool for developing problem solving skills, but when mastering new interfaces, the meaningful difficulty of the problems being solved should be reduced or eliminated altogether. When starting to master the block programming environment, at first, you should avoid examples or tasks that have additional complexity and are designed to develop general problem-solving skills, rather than specific skills in working in a new environment. That is why at the beginning of the second stage, children receive familiar tasks for managing familiar robots. Program input is carried out using familiar command and control structures icons. The only difference is the textual, not the pictographic, representation of the program. The program in the PiktoMir-K environment is depicted on a subset of so called *the school algorithmic language* widely known in Russia, implemented in the full-text programming system KuMir. The school algorithmic language and the Kumir system are used in many Russian school textbooks [25-31]. Having created a program in the PiktoMir-K system, the child can see how the same program would look in the KuMir environments, which will further facilitate the transition to the third stage (see Fig.8). However, it should be noted that the visual appearance of the program in the PiktoMir K block system is very close to the appearance of

the program in the KuMir full-text system. In this matter, our point of view is close to the point of view of the developers of the WoofJS software environment, half-jokingly-half-seriously expressed in [32]:
"The marketing solution for Scratch is counter-intuitive, but not impossible: tell kids Scratch is hard. This means removing all of the bright, primary colors, and childish cartoons. Replace them with harder, more standard adult colors and photos. Eschew rounded blocks for ones with hard edges. … Basically make Scratch look more like what kids imagine adult coding to look like." In [23] it is explained that the popularity of KuMir in Russia is largely due to the fact that this environment has the Russian vocabulary of control structures and allows names of variables and subroutines with spaces. This was the reason for using *the school algorithmic language* in both block and full-text programming environments. The transition from PiktoMir to PiktoMir-K is psychologically comfortable for children. Several lessons are devoted to working with robots from PiktoMir and re-solving in a new environment the problems that were solved in the previous step. The child can, by choice, carry out solutions of these problems in the PiktoMir-K environment in the pictogram-based or in the block-based representation of the program, switching, if desired, between these representations. *The main goal of this stage is to acquire solid skills in reading programs in text form.* As already mentioned, the program in the PiktoMir-K system is depicted on a subset of the school algorithmic language. Accelerated acquisition of reading and comprehension skills in programs in this language is helped by superimposed on the text of the program a laconic graphical form of representation of a block structure using nested rectangles (see Fig. 8). Note that some pseudo-graphic representation of nested block structures is preserved in the Kumir environment as well.

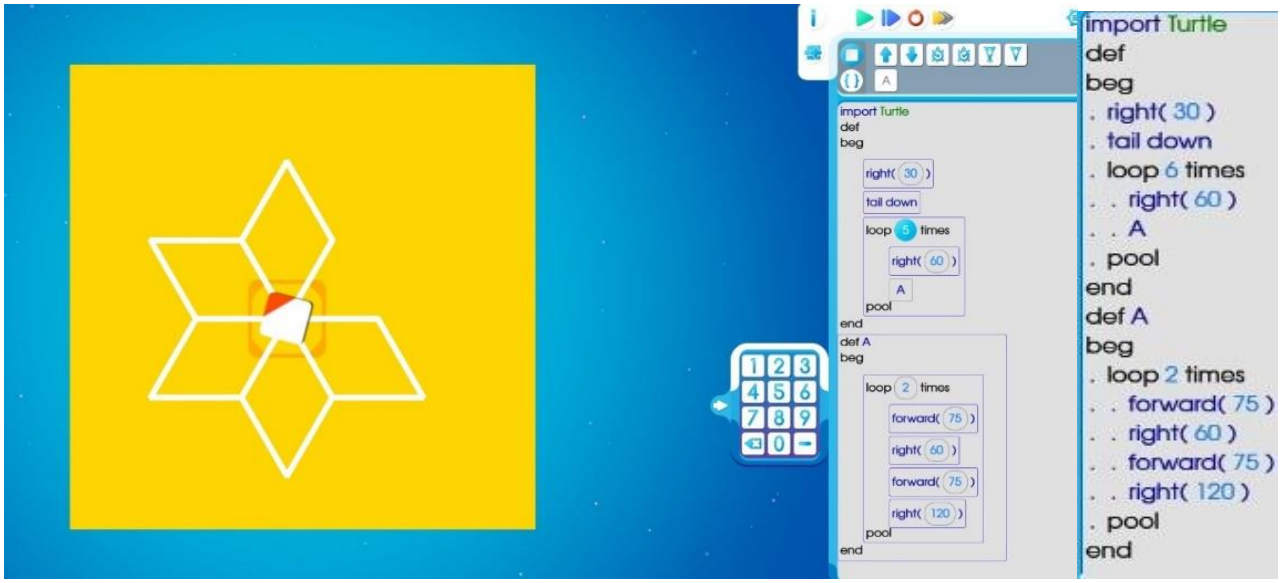

Fig. 8 A program that controls a turtle in the PiktoMir-K environment. The block representation of the program in the PictoMir-K environment and the full-text representation in the Kumir environment are shown.

So, the very important thing is that when creating a simple program (without variables), a child can see how the same program would look in PiktoMir systems, and when creating more complex programs, a child can, looking ahead, see how the created program will look like in the KuMir system. Fig. 9 shows the representation of the same program in the PiktoMir, PiktoMir-K and KuMir systems. Note that in KuMir, the editor automatically concatenates the beginning and end of each control structure with a vertical column of dot symbols to better visualize the nested block structure. A simple comparison of these three views reveals the reason for the effectiveness of work in the PictoMir environment.

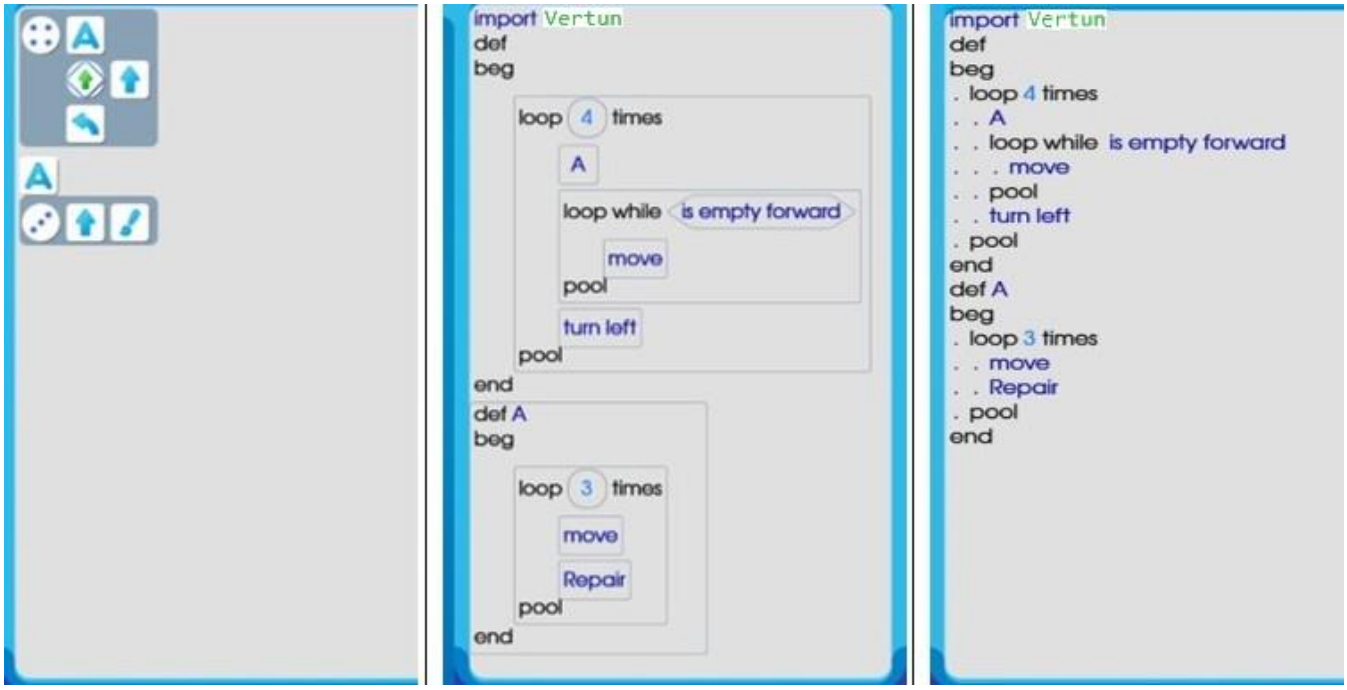

Fig. 9 How the same program looks in three programming environments

Representation of the program in the PiktoMir environment requires the use of 3 lines and 9 characters, while in PiktoMir-K and Kumir, 5 times more lines and 10 times more characters are required. Only after obtaining fluent, stable skills of recognition and understanding of the block structure of a program with nested blocks, the development of new, in comparison with PiktoMir, system capabilities begin.

The central new concept is the concept of a named integer variable, a value assignment construct, and a way of entering an arithmetic expression that makes it impossible to make a syntax error. An integer variable is introduced to replace the Magic Jug with Stones already mastered by children and is explained using its commands (see Fig. 10).

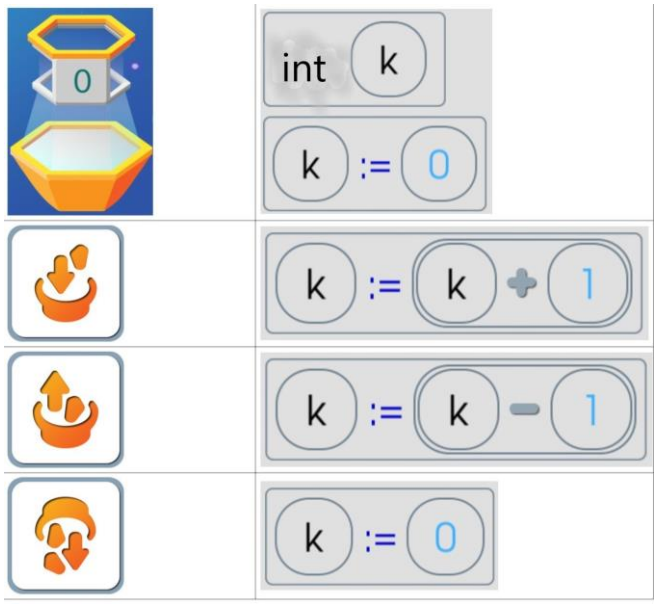

Fig. 10 Representation of the "counter" commands in the PiktoMir and PiktoMir-K environments

Further, at the second stage, robots are mastered, the commands of which have numerical parameters and the technique of error-free input of arithmetic expressions is further mastered. In a limited form in the PiktoMir-K environment, it is possible to solve the tasks of processing data arrays. The fact is that in the PiktoMir-K and KuMir environments there is a moving device called the Robot, which, according to legend, inspired by the Chernobyl events, can measure the temperature and radiation of the cell in which it is located.

The temperatures of cells in a rectilinear row of cells along which the Robot moves form a numerical sequence. This allows you to solve the problem provided for by the above Russian educational standard, namely to educate "*the ability to write in the studied programming language algorithms for searching for maxima, minima, the sum of a numerical sequence*". In a visual form, these tasks turn into tasks of creating a program that finds the maximum, minimum and average temperature of cells of a given row in the Robot field.

Our experience shows that a schoolchild can complete up to a dozen tasks in the PiktoMir-K environment in an hour's lesson. The submission of tasks and the registration of the results of their implementation can be carried out under the control of a networked digital educational environment external to the PiktoMir-K programming environment.

C. Third stage

When the complexity of the problems being solved and the size of the program reach a certain threshold level, the textual representation of the program and character-by-character input of the program code from the keyboard become more efficient than the block representation, since it makes it easier for the programmer to transform the program from one correct state to another along the path passing through syntactically incorrect states. In theory, this could allow the course authors to visually demonstrate to the novice programmer the feasibility of switching to full-text programming environments. In practice, however, this

threshold level of complexity is not reached in a short introductory programming course. All the problems to be solved are so simple that the transition to the usual block-based programming environment to a more complex full-text programming environment does not give the student any visible advantages. The student will only need to believe in the need to master the full-text environment. For this reason, special care must be taken to provide a user-friendly interface for a text-based programming environment.

At the final stage, the programming language and learning environment are expanded to a full-text educational Pascal-like digital platform KuMir, that allows working with complex text and numerical data structures: arrays and strings. In terms of the style of work, this platform does not differ much from 5 production platforms, listed in the Russian standard and prepares students who decide to study computer science and programming at an advanced level to transition to a production style of programming in languages such as Python, Java Script, or C. When using the KuMir system in elementary school, it is important that this system uses keywords and names written in Russian and has a developed system for diagnosing syntax errors.

In KuMir, as in any professional full-text programming environment, there are tools that make it easier to enter a program in text form. But, basically, the input of the program, as in professional programming systems, is carried out in KuMir character by character. At this stage, students first encounter the concept of a syntax error, and special measures have been taken in KuMir to minimize the costs of mastering of methods for correcting syntax errors. KuMir is equipped with an advanced system for detecting syntax errors, including establishing any cross-references errors.

Kumir graphically indicates the expected location of the error in the edited line of the program code and places a diagnostic message about the error on the continuation of the line, in which found the error. Error messages are placed in the margin of the program, at the moment when the input focus leaves the line. The control of the correctness and integrity of the program created by the trainee is carried out in parallel with its editing. When the input focus is removed from the edited line, the program is fully analyzed, and the error message, if necessary, automatically appears in the continuation of the edited line, as well as in the continuation of all lines that are affected by the error in the newly edited line. Even if there is an error, KuMir lets you run the program and see the partial results of that execution. These features of the KuMir system create a comfortable environment for students in which they can focus on the algorithmic component of the problem being solved. At the previous two stages of mastering programming, in the PiktoMir and PiktoMir-K environments, each task was checked by the programming environment, within which tasks were created. In the full-text programming environments KuMir, each task of the course can be equipped with a built-in, invisible to the learner, a checking program (created by the author of the task in the same school algorithmic language in which the student performs the task.). The didactic unit in the KuMir system is a workshop with automatic verification of the correctness of the tasks performed by the student. Since the PiktoMir-K and KuMir environments have several common virtual robots, mastering the KuMir system begins with resolving the control problems of known robots mastered at the previous two stages.

Thus, the features of working in a full-text programming system are first mastered on the fully studied algorithmic material, and only after getting used to the full-text system, the development of new capabilities of the school algorithmic language and the KuMir environment begins.

Our experience shows that in a one hour contest a student is able to complete up to a dozen easy tasks for managing robots or about half a dozen tasks for processing numerical or symbolic information minimal complexity, for example, counting the number of positive elements in a numeric array or the number of spaces in a string. The submission of tasks and the registration of the results of their implementation can be carried out under the control of a networked digital educational environment external to the KuMir programming environment.

IV. RESULTS

A. In the course of many years of experiments, we made sure that preschoolers aged 6+ and younger schoolchildren with pleasure and without difficulty perform more than a hundred tasks for compiling simple programs in an annual introductory programming course.

A1. Our experience of working with preschoolers in the city of Surgut age 6+ in the 2018-2019 academic year and the 2019-2020 academic year (6,000 children annually), described in [22], shows that preschoolers aged 6+ make up about 5 - 6 simple programs for a 15-20 minute computer part of the lesson. Thus, in an annual course of 30-35 half-hour lessons, children acquire the skills to independently compile 120-150 of the simplest programs.

A2. Article [33] provides data on the number of problems solved by preschoolers of the municipal kindergarten in the ninth week lesson of the course in December 2019. The data refer to a group of 50 preschoolers from the municipal kindergarten of the city of Surgut, 27 boys, 23 girls. This number 9 lesson under discussion was organized as a competition., therefore, the entire time of a half-hour lesson was allocated to work on tablets - 30 minutes with a 5-minute exercise break. 9 tasks were offered. More than half of the children completed all 9 assignments without any assistance from the teacher. The rest of the children completed 7 or 8 tasks out of 9, while some of the children needed the teacher's help in solving one or two tasks.

A3. In 2018-2019, we monitored the work of a private kindergarten in Moscow. Children and their parents were promised that at the end of each semester, children will be issued certificates listing their specific achievements. So throughout the year we recorded the number of tasks completed by each child in each lesson. The course consisted of 12 lessons in the fall semester and 12 lessons in the spring semester. The group consisted of 5 boys and 6 girls, the age of the youngest child at the beginning of the course was 5 years and 6 months. The fall semester was devoted to the compilation of linear programs. The number of tasks completed by children per semester varied from 55 to 71, that is, on the computer part of each lesson, each child managed to solve at least 5 problems. Similar results were achieved in the second, spring semester. Namely, in 7 lessons devoted to the REPEAT N TIMES cycle, each child

completed 35 to 40 tasks, that is, at least 5 tasks in each lesson. During 5 sessions devoted to subprograms, each child completed about 20-25 tasks. Thus, in an annual course of 24 lessons, the number of tasks completed by a child varied from 110 to 130.

A4. In the 2019-2020 academic year, we tracked the annual Algorithmics course as part of extracurricular STEM classes, conducted according to our methodology for fourth-graders in one of the Moscow schools. In a group of 22 schoolchildren, divided into two subgroups (a total of 11 boys and 11 girls), 26 lessons were held, each lasting 45 minutes. For a computer workshop on tablets, 15 minutes were allocated for each lesson. 20 workshops were held in the pictogram-based environment of PiktoMir and 6 workshops in the block-based environment of PiktoMir-K. At each workshop, each child managed to complete from 4 to 5 tasks. Thus, it turned out that the productivity of novice fourth-graders in compiling the simplest programs is commensurate with the productivity of preschoolers aged 6-7 years.

B. Successful use of our three programming environments in teaching future teachers at Moscow State Pedagogical University.

In parallel with the development of an introductory programming course for russian elementary school (grades 1 to 4) we were solving the problem of preparing teachers capable of teaching this course. For the last five years we have been teaching the annual course "Methods of Teaching Programming" at Moscow State Pedagogical University. For the first three of these five years, we used the PiktoMir system in the first quarter of the course, and the Kumir system in the rest of the course. Our experience has shown that the transition from PiktoMir to KuMir was psychologically difficult for students, and technologically required a lot of time and effort to master the minor details of the language.

In the last two years, the course "Methods of Teaching Programming" has been redesigned for students with the main specialization: elementary school teacher and additional specialization: computer science teacher in basic school (grades 5-9). In this course, we have increased the number of hours devoted to pictogram programming and started using the PiktoMir-K block programming system.

First use of the PiktoMir-K system in 2019-2020 academic year as a bridge between the two styles of programming showed encouraging results. So in the next 2020-2021 academic year we began to consistently use our three programming environments in teaching future teachers at Moscow State Pedagogical University.

In the 2020-2021 academic year, 19 students began to attend the course. Within a year, 1 student dropped out of the university for family reasons. 18 students graduated from the course, including 1 boy and 17 girls. One student was 25 years old, the rest of the students were 20-21 years old. In this course, as in many courses taught at this university, an initial questionnaire was conducted. Here are the answers to three questions of the questionnaire concerning the results of students in mastering programming in school.

Question. *In which programming language (if any) you were taught to program at school*.

Answers. Pascal - 5, KuMir - 4, did not learn - 9.

Question and **answers**. *Assess your level of programming by the time you leave school by choosing one of the most appropriate answers out of 3.*

- *did not learn to program at all* – 8 students;
- *learned to program a little* – 8 students;
- *learned to program well* – 2 students.

Question. Reconstruct the two missing slots of this 3-line program snippet to calculate the number of distinct roots of equations x (x-a) (x-b) = 0, or write a similar snippet in any programming language

```
if(a=0) then n:=1 else n:=2;
if((b<>0) and ..... then .....;
writeln('The number of roots = ', n);
```

Answers. In the allotted time of 10 minutes, only 5 students out of 18 tried to solve this problem, none of them managed to fill in correctly two slots of 6 characters each:

```
if((b<>0) and (b<>a) then n:=n+1;
```

The results of the questionnaire survey of this group reflect, in the opinion of the authors, a picture typical for Russia: half of the school graduates in Russia did not encounter programming at all at school. Those who studied programming in school are not getting enough practical skills. So, the course "Methods of Teaching Programming" was attended by students who did not have stable basic programming skills. Therefore, the main attention in this course had to be paid not to teaching methods, but to the very basics of programming.

C. We managed to develop a one-year programming course for future teachers (70 academic hours), in which students who did not have even the most basic programming skills could master the skills of working in pictogram, block and full-text programming environments. by completing at least three hundred assignments for writing simple programs. of them at least 80 programs in the full-text programming environment.

For this course, we have prepare

130 tasks in the PictoMir environment,
170 tasks in the PiktoMir-K environment and
105 tasks in the KuMir environment.

In total, 405 assignments were given in the course of 35 one and a half hour lessons. Certification required 80 percent of the tasks in each environment to be completed, and all 18 students (100%) successfully completed this task.

D. The time spent working in pictogram and block programming environments pays off when switching to a full-text programming environment.

Of the 105 tasks in the Kumir environment, the first 30 were robot control tasks that could be solved in the PictoMir-K environment, that is, they were proposed to facilitate mastering the interface of the full-text programming environment. Only 75 out of 105 tasks in the KuMir environment were aimed at working with integers, arrays and strings, traditional for introductory courses. It turned out that the introduction of the intermediate

environment PiktoMir-K increased the productivity of students in the KuMir environment when solving these traditional problems. A significant part of the students in the current academic year succeeded, during classes in the KuMir environment, not to be limited to the completion of 80% of the credit assignments, but to complete all the assignments. (Each task of the course has a built-in mechanism for automatic verification of the correctness of the solution found by the student, the number of attempts to verify the solution is not limited, and unsuccessful attempts are not penalized)

E. Conclusions and Future Works

Based on many years of work with beginners of different ages, we have come to several conclusions, outlined below.

Thesis 1. *The effectiveness of the sequential use of three programming styles in the initial course of programming is due to the nature of things. A beginner must simultaneously learn both a) the rules for presenting the program code and b) the methods of creating program code.*
Both a) and b) can be done in a simpler visual-graphic form and in a more complex symbolic-textual form. Combinations of these forms give three types of programming environments of increasing complexity to learn.

Table 1. Three types of programming environments

type of programming environment	program code representation	program code input method
icon-based	icon-graphical, pictographic	manipulations of tangible objects, drug-and-drop
block-based	mixed textual/graphical	drug-and-drop
full-text-based	plain text	keyboard

Thesis 2. *In the initial programming course for students of all ages, the study of structured programming constructs - sequence, branching, repetition, condition, subroutine, counter - and their most common combinations are most effectively carried out in an icon-based programming environment. To define nested block structures in an icon-based environment, in our opinion, some generalization of the horizontal indentation technique in the spirit of the Python language is beyond competition.*
Future work. Publication of an article on defining the semantics of horizontal indents in pictogram programming, which allows any program to be considered syntactically correct and executable.

Thesis 3. *Completion of mastering the full set of structured programming constructs and their combinations such as 'loop inside a conditional structure', 'conditional structure inside a loop', 'loop in a loop', setting one or several subroutines, calling one subroutine inside another, etc. on the material of control algorithms for the simplest real and virtual robots, possibly upon reaching the age of 7 years (in Russia this is the age of completion of the stage of preschool education). This development can be started at the age of 4-5 years.*
Future work. Together with the Institute of Educational Technologies in the city of Samara, we began a massive experiment to introduce two and three-year programming courses in the preschool education system of the Russian Federation at 250+ experimental sites. [34].

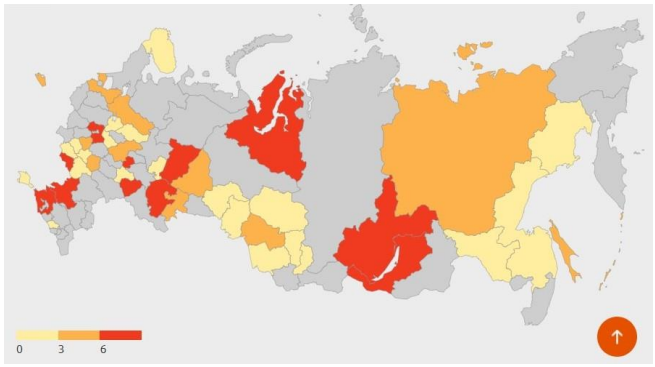

Fig. 11 Experimental sites of the "PiktoMir for Preschoolers" project on the map of Russia.

Thesis 4. An introductory programming course that meets the requirements of the Russian educational standard given in section II above can be mastered by elementary school students.
Future work. Together with the Institute of Educational Technologies in the city of Samara, we began an experiment to introduce such a course in several dozen schools in Russia.

Thesis 5. In the annual programming course for future teachers (70 academic hours), it is possible to ensure the development of the pictogram-based, block-based and full-text style of programming by solving at least three hundred tasks for compiling simple programs by each student.

The main disadvantage of the approach proposed in this article. The methodology for learning the basics of programming described in the article has the disadvantage that in the first two stages, students do not get acquainted with the methods of input-output, event control and dialog work.
Future work. a) Develop a legend that requires interactive work and event management when working with virtual robots used in PiktoMir and PiktoMir-K,
b) develop tasks that require the preparation of dialogue programs.

Acknowledgements

The authors are grateful to reviewers 1, 2, and 3 for comments and suggestions, the consideration of which led to an improvement in the quality of the article.

The work was carried out within the framework of the state assignment of Scientific Research Institute of System Analysis of the Russian Academy of Sciences (SRISA/NIISI RAS) on topic No. 0580-2021-0010, registration No. 121031300134-3).

References

[1] Betelin, V.B., Kushnirenko A.G., Semenov A.L., Soprunov S.F. ABOUT DIGITAL LITERACY AND ENVIRONMENTS FOR ITS DEVELOPMENT. Informatics and Applications, 14(3), pp 100-107 (2020), (In Russ.) DOI:https://dx.doi.org/10.14357/19922264200414

[2] Papert, Seymour: Mindstorms: Children, Computers and Powerful Ideas.: Basic Books Inc. Publishers. New York, NY USA 252 p. (1980).
[3] Resnick, M. et al. Scratch: Programming for all. Commun. ACM 52, 11 (Nov. 2009), Pages 60-67. https://doi:10.1145/1592761.1592779
[4] Flannery, L.P., Kazakoff, E.R., Bonta, P., Silverman, B., Bers, M.U., & Resnick, M. (2013). Designing ScratchJr: Support for early childhood learning through computer programming. In Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13). ACM, New York, NY, USA, 1-10. DOI=10.1145/2485760.2485785
[5] Bers M.,U., Resnick, M., The Official ScratchJr Book: Help Your Kids Learn to Code, No Starch Press, (2015)
[6] Blockly, https://en.wikipedia.org/wiki/Blockly
[7] Meet Cubetto. URL: https://www.primotoys.com, last accessed 2021/09/27.
[8] Matatalab, URL: https://matatalab.com/en, last accessed 2021/09/27.
[9] Sullivan, A., Elkin, M., & Bers, M. U. (2015). KIBO Robot Demo: Engaging young children in programming and engineering. In Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15). ACM, Boston, MA, USA.
[10] Lego_Mindstorms_EV3, URL: https://en.wikipedia.org/wiki/Lego_Mindstorms_EV3, last accessed 2021/09/27.
[11] Lightbot. URL: https://en.wikipedia.org/wiki/Lightbot, last accessed 2021/09/27.
[12] Resnick, M. et al. Scratch: Programming for all. Commun. ACM 52, 11 (Nov. 2009), Pages 60-67. https://doi:10.1145/1592761.1592779
[13] Kelleher, C., Pausch, R., and Kiesler, S. Storytelling Alice motivates middle school girls to learn computer programming. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (2007), Pages 1455-1464. https://doi.org/10.1145/1240624.1240844
[14] Homer M., Noble J., Lessons in Combining Block-based and Textual Programming, Journal of Visual Languages and Sentient Systems (VLSS), Volume 3, 2017
DOI 10.18293/VLSS2017-007
[15] Weintrop, David & Wilensky, Uri. Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms. ACM Transactions on Computing Education. 18. 1-25. (2017). https://doi.org/10.1145/3089799
[16] Weintrop, D., To block or not to block, that is the question: students' perceptions of blocks-based programming
IDC '15: Proceedings of the 14th International Conference on Interaction Design and Children June 2015 Pages 199–208,
https://doi.org/10.1145/2771839.2771860
[17] Russian Federal State Educational Standard,
basic general education (Federal'nyy gosudarstvennyy obrazovatel'nyy standart osnovnogo obshchego obrazovaniya), (In Russ.), URL:http://publication.pravo.gov.ru/Document/View/0001202107050027, last accesed 2021/09/27.
[18] Weintrop D., Block-based Programming in Computer Science Education, Communications of the ACM, August 2019, Vol. 62 No. 8, Pages 22-25, 10.1145/3341221.
[19] Rogozhkina, Irina & Kushnirenko, Anatoli. PiktoMir: Teaching programming concepts to preschoolers with anew tutorial environment. Procedia - Social and Behavioral Sciences. 28. pp 601-605. (2011). doi:10.1016/j.sbspro.2011.11.114
[20] PiktoMir: The project starting page at the website of the Federal Scientific Center NIISI RAS,
URL: https://www.niisi.ru/piktomir, last accessed 2021/09/27.
[21] Betelin V. B., Kushnirenko A. G., Leonov A. G., Mashchenko K. A., Basic Programming Concepts as Explained for Preschoolers, pp. 245-255, Volume 15, 2021, International Journal of Education and Information Technologies (NAUN). DOI: 10.46300/9109.2021.15.25.
https://www.naun.org/main/NAUN/educationinformation/2021/a502008-024(2021).pdf
[22] Besshaposhnikov N.O., Kushnirenko A.G., Leonov A.G., Raiko M.V., Sobakinskikh O.V. Digital educational environment PiktoMir: Experience of development and mass implementation of an annual programming course for preschoolers. Informatics and education. 2020;(10):28-40. (In Russ.)
https://doi.org/10.32517/0234-0453-2020-35-10-28-40
[23] Leonov, A. & Pervin, Yu & Zaidelman, Ya. SOFTWARE EXECUTORS IN THE DIGITAL PEDAGOGICAL ENVIRONMENTS: PIKTOMIR, ROBOTLANDIA AND KUMIR. Informatics in school. pp 54-61 (2019).
https://doi.org/10.32517/2221-1993-2019-18-9-54-61.
[24] Besshaposhnikov N.O., Kushnirenko A.G., Leonov A.G., Pictomir: how and why do we teach textless programming for preschoolers, first graders and students of pedagogical universities, CEE-SECR '17: Proceedings of the 13th Central & Eastern European Software Engineering Conference in RussiaOctober 2017 Article No.: 21, Pages 1-7,
https://doi.org/10.1145/3166094.3166115
[25] Kushnirenko, A.G., Lebedev, G.V., Svoren', R.A. Osnovy informatiki i vychislitel'noj tehniki. Uchebnoe posobie dlja 10 – 11-h klassov obshheobrazovatel'nyh uchrezhdenij. Prosveshhenie. Moscow, USSR. 224 s. (In Russ.), (1990).
[26] Zvonkin, A.K., Lando, S.K., Semenov, A.L. Informatika. Algoritmika. 6 klass. Prosveshhenie. Moscow, Russian Federation, 239 s. (In Russ.), (2006).
[27] Lando, S.K., Semenov, A.L., Vjalyj, N.M. Informatika. Algoritmika. 7 klass. Prosveshhenie. Moscow, Russian Federation, 208 s. (In Russ.), (2008).
[28] Bosova, L.L., Bosova, A.Yu. Informatika. 7 klass. LBZ. Moscow, Russian Federation, 224 s. (In Russ.), (2013).
[29] Kushnirenko, A.G., Leonov A.G., Zaidelman Ya.N., Tarasova V.V., Informatika. 7 klass. Dropha. Moscow, Russian Federation, 176 s. (In Russ.), (2017).
[30] Kushnirenko, A.G., Leonov A.G., Zaidelman Ya.N., Tarasova V.V., Informatika. 8 klass. Dropha. Moscow, Russian Federation, 224 s. (In Russ.), (2017).

[31] Kushnirenko, A.G., Leonov A.G., Zaidelman Ya.N., Tarasova V.V., Informatika. 9 klass. Dropha. Moscow, Russian Federation, 232 s. (In Russ.), (2017).
[32] Krouse, S., Scratch Has a Marketing Problem, URL: https://medium.com/free-code-camp/scratch-has-a-marketing-problem-f84626bd18ef, last accessed 2021/09/27.
[33] Leonov, A.G., Raiko, M.V., Sobakinskikh, O.V., Sobyanina, N.V.The results of mastering the annual program "Algorithmics for preschool children" by the preparatory groups of the municipal preschool educational institution, Trudy SRISA RAS, 2020, vol. 10, № 5-6, pp. 195-199, (In Russ.)
[34] Experimental sites of the “PiktoMir for preschoolers and firstgraders” project on the map of Russia. URL: https://inott.ru/projects/piktomir/uchastniki-doshkolnoe-obrazovanie/ , last accessed 2021/09/27.