INTERNATIONAL JOURNAL OF EDUCATION AND INFORMATION TECHNOLOGIES
DOI: 10.46300/9109.2021.15.25 Volume 15, 2021

Basic programming concepts as explained for preschoolers

Betelin V.B.[1], Kushnirenko A.G.[1], Leonov A.G.[1,2,3], Mashchenko K.A.[1]

[1] Federal Research Center "Scientific Research Institute for System Analysis of the Russian Academy of Sciences," 36-1 Nakhimovsky Prosp., Moscow 117218, Russian Federation
[2] Moscow State University, 1 Leninskiye Gory, Main Building., Moscow 119991, Russian Federation
[3] Moscow State Pedagogical University, 1/1 Malaya Pirogovskaya Ulitsa, Moscow 119435, Russian Federation

Received: August 6, 2021. Revised: September 3, 2021. Accepted: September 6, 2021.
Published: September 9, 2021.

Abstract—The development of information technologies has formed a socio-economic request to reduce the age at which children can be introduced to programming. After 6 years' efforts, the authors managed to develop and on a large scale introduce a one-year programming course for preschoolers, which is built on a metaphor of programmed control. During the course development, a set of basic programming concepts was selected and specified to be mastered by preschoolers aged 6+ in the activity-game form. This set of concepts goes back to Seymour Papert's ideas about teaching programming by writing programs that control moving objects using intuitive sets of commands. The main feature of the proposed system of concepts, allow at the initial stage of training to demonstrate and assimilate all the elements of the concept in the real, not in the virtual world. In the picture of the World, which we explain and demonstrate to children, only one function remains at the computer - the execution of the program it has memorized. Everything else happens in the real world. In the real world, an environment is created in which a real robot will move. In the real world, a program is created from tangible objects, which will then be "shown" to the computer so that it stores it in its memory and can then execute it at the command of a human. In the real world, obeying the signals of the computer, the robot performs the work stipulated by the program. This allows you to begin acquaintance with the programming of children from 4 years old, without working individually or collectively with electronic screens, which in today's Russia is prohibited by federal medical authorities in the educational process of children under 5 years old. The course is built on the text-free pictographic programming system PiktoMir developed by Russian Academy of Sciences. The methodological content of the course allows each preschooler to gain experience in the development and debugging of 120-150 simple programs by the end of the course. The final part of the article discusses the authors' plans for the development of a three-year textless programming course, methodologically and instrumentally connected with the primary school programming course.

Keywords—computer science, robot, program, tangible object, computer, programming language, preschooler, kindergarten, pictogram, PiktoMir, Scratch Junior

I. INTRODUCTION

LEGISLATIVE bodies of Russia at the federal level put forward proposals to reduce the age of introduction of children to computer science and programming at the level of preschool education [1]. This initiative reflects the global and European trends towards a decrease in the age at which children become familiar with programming from the secondary school level to primary school and from the elementary school level to preschool [2; 3]. Since these trends are now close to practical implementation, it is not surprising that they are beginning to be discussed and resolved at the national level, not only in Russia, but also in other countries. In the UK, teaching computer science underwent a significant transformation a few years ago and, to some extent, has become compulsory for all students aged 5 and over. Some efforts are also being made in Germany in this direction [4]. A number of experts express a categorical point of view, for example, "European countries harm their primary and secondary school students, both educationally and economically, by not offering them education in the basics of computer science."[4] The thesis that "programming is the new literacy" began to be shared by an increasing number of educational theorists and practitioners in the early 21st century, and by the end of the first quarter of the century it had become generally accepted. Nevertheless, question about how to incorporate programming into the preschool education system remains open. A systematic attempt to answer this question, based on real robotics, is outlined in the boor [5]. The idea of replacing real robotics with virtual led to the creation of the wonderful Scratch Junior programming learning environment [6]. For Russia, with its traditionally centralized and even federalized education system, where new courses, after some experiments, should be introduced simultaneously and everywhere, the question is very important whether it is possible to introduce new literacy - coding - in the existing system of preschool education by the forces of the current corps of teachers with reasonable costs for retraining. This article answers this question in the affirmative. A specific methodology for teaching programming to preschoolers is described and six years of experience in the development and large-scale application of the one-year course "Algorithmics for Preschoolers", accumulated jointly by SRISA/NIISI RAS and the Department of Education of Surgut, Western Siberia, are summarized. Since the fall of 2018, all children of all preparatory groups of all preschool educational institutions of Surgut - about 6 thousand children aged 6 to 7 years annually - undergo a one-year course of study [7]. One of the conclusions of this study is that the methodology developed by us made it possible to introduce an initial programming course in all preschool institutions of the city with a population of about 400 thousand people with the help of the existing corps of teachers [8]. This result was achieved with minimal retraining costs, from 24 to 72 academic hours. The main difficulty in retraining personnel who received their basic education many years ago and were familiar with modern information technologies at the household level was the choice of a set of basic concepts that teachers would fully understand themselves and were ready to demonstrate in the game and discuss with preschoolers. The selection of this set of concepts and the development of a methodology for its presentation is the focus of this article.

Another question that we were able to answer is the maximum size of the group with which one teacher works. Before starting the experiment, we assumed that a teacher would be able to teach in a group of no more than 6-7 children, but the practice has shown that by adjusting the methodology and software, the size of a group working with one teacher can be increased without compromising the quality of education up to 10-12 children. Below will be described the features of our software and the methodology to achieve this result. These features have been tested and refined in 6 consecutive organized annual courses. In the zero year of the experiment, we started working with one kindergarten; at the final stage, several dozen kindergartens participated in the experiment - all kindergartens in the city of Surgut. The course is based on the textless educational programming environment PiktoMir [9; 10].

II. A FEW WORDS ABOUT THE PIKTOMIR SYSTEM

History and current state. The development of the PiktoMir system was launched by SRISA/NIISI RAS about 10 years ago [9]. Originally conceived as a purely electronic game capable of acquainting selected motivated preschoolers with programming, such as the Box Island game [11] and the Lightbot game [12], the system turned out to be in demand by the preschool education system in Russia and began to develop rapidly as software to support the course of systematic programming for preschoolers. Work on the development of the system continues today under the state order "Development, implementation and application of a family of integrated multilingual programming environments for kindergartens, schools and pedagogical universities ..." [13] issued for the Russian Academy of Sciences by the Ministry of Science and Higher Education of the Russian Federation.

Emphasis on real, not virtual world. The PiktoMir system and the programming course for preschoolers are built on the metaphor of computer, programmed control of real-world objects. PiktoMir allows you to control real robots and virtual robots and virtual scenes appear on the screen as models of real robots and real scenes. Therefore, unlike the Blockly Games [14] and Scratch Junior [6] systems, PiktoMir pays little attention to graphic and sound effects and does not challenge the child to develop their own drawings, music, dialogue games and cartoons. Today the world has accumulated a lot of experience in creating textless programming systems and robotic complexes for teaching preschoolers to program. The use of these software systems in the educational process of different countries may be constrained by national characteristics. For example, Hopster Coding Safari for Kids [15] remains an unsurpassed tool for introducing programming to children aged 3-4 years, unfortunately, in Russia this system cannot be used by kindergartens, since working individually or collectively with electronic screens in the educational process for children under 5 years old, in today's Russia is prohibited by federal medical authorities. This is one of the reasons why PiktoMir focuses on working in the real world with the real toy-robots and screenless programming.

Of course, concerns of teachers and parents about the negative impact on the health of children under 5 years of age from working with a keyboard and mouse or prolonged screen work exist in other countries and are codified in one way or another, but usually in the form of recommendations, and not in the form of prohibitions. For example, the American Academy of Pediatrics recommends limiting non-educational screen time to about 1 hour on a weekday and 3 hours on weekends for children 2-5 years old.

Pikto language. The Pikto language is supported in PiktoMir, in which a dozen basic program constructions are implemented. With one exception, all of these constructs are illustrated in the program in Fig. 1 consisting of the main section, function A and function B

- calling the robot's order-command (main)
- calling the robot's question-command that returns a boolean value (function A, loop WHILE)
- calling the robot's question-command that returns a non-negative integer value (function B, loop REPEAT N TIMES)
- one-letter header of a function without parameters (functions A and B)
- calling a function without parameters (main)
- header of the REPEAT N TIMES loop (main, function B)
- integer literals from 1 to 6, used as loop header (main)
- header of the WHILE loop (function A)
- header of the IF statement (not used in Fig. 1 program).

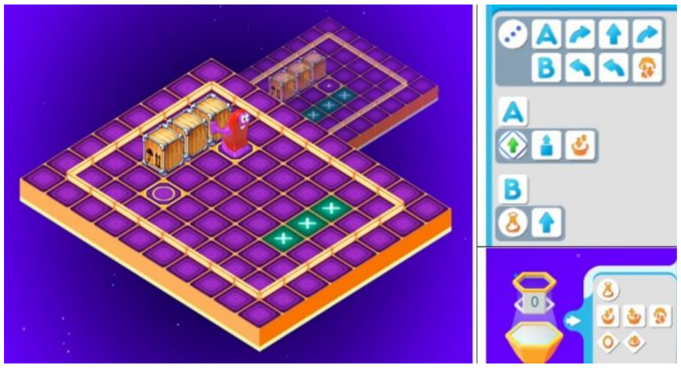


Fig. 1 Universal program using Magic Jug. Two similar working environments are given. The towing robot Tractor has to set the boxes to the designated positions.

This set of constructs includes a set of structured programming constructs, the understanding of which, according to federal law, is mandatory for graduates of the 9th grade of schools in the Russian Federation. There are no variables in the language, but there is one built-in counter - Magic Jug With Stones. The magic lies in the fact that there is an inexhaustible pile of stones next to the jug, and the jug itself is never overfilled.

With the help of the Magic Jug, programs using counting can be created. For example, in the program in Fig. 1 after each step of the robot in function A, a stone is added to the jug. In function B the current number of stones in the jug plays the role of a repeater, in the main section jug is cleared at the end of each loop body execution.

The language has no means of input-output of text information or audio messages. To a limited extent, such I / O can be organized using command-orders and command-questions of robots, which are controlled by the program.

PiktoMir supports the mode of composing a program from tangible objects, similar to the modes in the Primo Cubette [17], MatataLab [18] or KIBO-robot [19] systems. Unlike the listed systems, this mode does not require special devices for placing program blocks or encoding information in the form of bar codes. The tangible objects themselves in PiktoMir are just pictures, recognized by the software modules of artificial intelligence, executed on the tablet in about one or two seconds. Our teaching aid kits use two types of tangible objects, flat magnetic cards like refrigerator stickers (good for a teacher demonstrating a program on a magnetic board) and painted wooden cubes (Fig. 5).

Like Scratch Junior, PiktoMir allows you to compose primitive parallel programs. In this case, for each movable real or virtual robot, its own control flow is created. Each such control flow controls only one instance of a given type of robot. The commands of the stationary robot-counter Magic Jug With Stones and other stationary objects are available in all control flows. Sequential threads prepared as part of a parallel program are launched simultaneously and executed synchronously, clock by clock. Synchronization between threads controlling different robots is done only by analyzing changes in the environment produced by other robots using busy-looping by each robot. Any collisions lead to failure. There is no event mechanism in the language.

Parallel cooperative programming mode. PiktoMir supports the cooperative programming mode, when sequential Programs controlling different robots operating in the same environment are compiled by different children sitting side by side at the same table and each working on their own computer on the same local network [20]. This mode is popular with children and is remarkable in that it requires intensive preliminary communication between team members before coding begins and develops teamwork skills.

The procedure for using the PiktoMir system by kindergartens. PiktoMir system and the courseware for 'Algorithmics for Preschoolers' are distributed on the free-of-charge basis. They can be downloaded from the SRISA / NIISI RAS website to tablets with iOS or Android or accessed using a browser via a web-interface [10]. The kindergarten administrator at the beginning of each school year usually downloads PiktoMir and the complete courseware set on all kindergarten tablets. The tablets are used without any personalizations. At each lesson, the student has access to the formulations of all the tasks of the course, but not the results of this student on their implementation during the year. When working with preschoolers, authorization at the beginning of the lesson is not made, the results of the students' work at the end of the lesson are not saved. This allows you to distribute tablets to children in random order at the next lesson in the group.

III. Innovations in the PiktoMir System

In the process of using the PiktoMir system in several consecutive courses "Algorithmics for Preschool Children", the system has undergone significant changes aimed at increasing its efficiency with the support of systematic motivation of children in the process of plugged work in the system and reducing the load on the teacher leading the lesson in order to ensure operational the teacher's reaction to a request for help from any child. A number of innovative, in our opinion, solutions were introduced into the system. Two of them, the systematic use of the Magic Jug and the Cooperative Programming Mode have been described above. Virtual reality innovations will be described in a separate publication. Below are a few more innovations that we found worth mentioning.

Controlled solution variability. In accordance with the practice of modern programming, when composing a program in Piktomir, a student almost never starts developing it from scratch. The child is usually given a program template, which simultaneously serves as both a limitation and a hint and an example of recommended practice for organizing the program. Similar hints-restrictions are found in the Lightbot environment in the form of a rigid program template that prescribes the nomenclature and sizes (number of commands) of functions. Methodically, this approach is very effective and we use it widely. Constraint hints are also used in the Box Island environment in the form of an upper a priori constraint on the number of different control structures and robot command calls that are allowed to be used in the program.

This approach is primarily intended to force the trainee to use short looping programs instead of long linear programs. We considered this approach to be far from programming practice and do not use it. Instead, we use a different trick. In the task for programming in the PiktoMir system, not one program template can be specified, but several. For example, to solve the same problem, it may be proposed to use a linear program, a program with a loop, and a program with a main program and one function. The task may require solving the problem in all three ways or in any one of them. Choosing the conditions of the problem so that the solutions in all three ways are feasible, but with a wide spread of labor intensity, we put the student to the condition when he voluntarily chooses the most effective solution method for him.

Tasks for the development of universal programs. If a robot working according to a program is able to execute command-questions, that is, it can transmit to the computer - the executor of the program - information about the situation in which it finds itself, it becomes possible to develop the so-called universal programs. The same universal program can control a robot for similar tasks in similar environments. Problems of this kind are encountered in various competitions and exams, in particular, in the final exams in computer science in 9 grades of schools in the Russian Federation. In tasks of this kind, an informal description of the environments in which the program should work successfully is usually given in natural language. For preschoolers who do not know how or are not very fond of reading, this approach is unacceptable. Therefore, PiktoMir provides tasks for the development of a universal program that works in several specific environments provided by the author of the task. For example, in the program in Fig. 1, two environments are specified. In one of them, program needs to drag each box by 5 cells, in the other, drag each box by 2 cells.

Observability of signals given by the computer to the robot. We show children how a simple real robot is controlled by three different sound commands: one, two or three beeps. Therefore, children are able to observe the process of controlling the robot by commands given from the sound remote control by a person or a computer while executing a program; children can imitate the robot and execute commands given by the remote control or computer. To avoid interruptions caused by noise in the classroom, the audio commands are technically duplicated by radio signals, but the children are not taught this duplication.

Piggy bank of commands - logging mode with rollback. In this mode, when the command icon is activated, it is immediately executed, and the command icon flies into the piggy bank (added to the stack of executed commands). You can undo the last command you ran by tapping the command icon in the piggy bank. The sequence of commands accumulated in the piggy bank can be inserted into the program, starting from any place. In this mode, you can execute not only atomic commands of the robot, but also functions that have single-letter names in PiktoMir. This opens up the possibility of developing programs with subroutines using bottom-up technology.

The authors would like to draw the reader's attention to this mode, which, in our experience, provides a significant reduction in the time a child spends on composing a program.

IV. Selecting a System of Basic Concepts in the Course of Programming for Preschoolers

Several factors influenced this choice. Firstly, in the process of preparing Russian kindergarten teachers for the course, it turned out that teachers first of all want to fully understand what changes and with what objects occur during the preparation and implementation of the program, in what terms and using what analogies they should tell children, what happens when compiling and executing programs, what are the functions of a computer. When working with virtual worlds on the screen in systems like Scratch Junior, the computer performs too many interconnected functions: it helps to create an environment on the screen in which characters will act, allows you to choose and change their appearance, helps a person create a program that will control these characters, and finally executes this program, demonstrating the movements of the characters on the screen and the changes in the surrounding virtual world that the characters produce.

In this situation, it is almost impossible to fulfill the teacher's request to single out from the above actions of the computer those actions that children could imitate in the game without a computer. If the controlled objects are real objects in the real world, then they can be easily imitated by children. Therefore, we tried to transfer everything that is possible from the virtual world to the real one.

Secondly, since we plan to start mastering programming with children aged 4 years old, whose work with the screen is not allowed due to their age, we are forced to begin acquaintance with programming using objects of the real, not the virtual world, and naturally this emphasis on working with the real world save throughout the course.

Digitalization of preschool education is a very important issue for modern civilization [21]. In this area, there are still no generally accepted, mass-tested solutions. This article is mainly focused on one specific task, namely the organization of the first lessons of the programming course for preschoolers and teachers preparing to teach programming to preschoolers.

This question is especially important, because it is in the first lessons that the foundation of the course is laid and a set of basic concepts is mastered. In our research, we tried to answer two closely related questions: what set of scientific concepts should be used as the basis for a programming course for preschoolers and what methodological and technical means are needed to organize the effective mastering of these concepts by preschoolers and educators in preparation. Following Papert's approach, we decided to base the system of concepts on the so-called *principle of programmed control*.

The principle of programmed control. We share the point of view expressed in [22] that one of the goals of studying computer science by children should be 'mastering the informational picture of the world.' This picture today includes robots and other technical devices controlled by a person or a computer system. This part of the picture is well described by the so-called Principle of Programmed Control:

Any work that a person can do by commanding a mechanical assistant (robot) can also be assigned to a computer, if the person can develop a program for the robot doing this work.

This principle can be described in terms and images that a six-year-old can consciously grasp if provided with an appropriate set of teaching aids and involved in a series of specially designed play practices. Therefore, following the classical approach of Seymour Papert, we choose for the first steps in programming a set of concepts that describe *the principle of programmed control*.

Programmed control can be achieved both with feedback (closed-loop control systems) and without feedback (open-loop control systems). Methodologically correct would be a two-stage presentation of the principle of programmed control, in which at the first stage the simplest version of the principle is presented, that is, without feedback. Practical work with preschoolers has shown, that it is the first stage that turns out to be fundamental and more difficult, than the second stage, where feedback is introduced.

V. Objectives of the course "Algorithmic for Preschoolers" and requirements for choosing a system of basic programming concepts that will allow achieving these goals

In semi-professional terms, the Principle of Programmed Control can be described as follows. A program is a plan for future actions that allows one object (computer) to control another object (robot) in accordance with a program previously developed by a human (programmer) in appliance with previously known rules for developing programs (programming language). The execution of a program by a computer is a process by which a computer, following a program, in some predetermined manner gives commands to a robot, which the latter executes and informs the computer about the execution of each such command and its readiness to receive the next one. However, the computer can execute the program only if it was previously informed about this program by loading the program into the computer's memory.

The above description explicitly introduces 12 concepts: 6 objects, 1 subject, and 5 interactions between the objects and the subject.

A. The objects

program; computer; computer memory; robot;
rules for developing programs (programming language); robot's command.

B. The subject

programmer;

C. The interactions

- the programmer *develops* a program;
- the computer *executes* the program by *giving commands* to the robot;
- having received a command, the robot *executes* it and waits for the next one;
- the computer *loads* the program into its memory.

Of course, the set of concepts and the accents made above in the explanation of the principle might have been different. The authors chose this set of 12 concepts based on purely practical considerations.

The fact is that our course is designed to simultaneously solve two problems:

- to achieve intuitive understanding and practical assimilation by children of the 'rules of the game', ie, intuitive awareness of the proposed system of concepts and their practical use;

- to enrich the vocabulary of children, teach them to use terms that express the concepts they have studied, to express understood concepts in verbal form.

The solution of the second problem is impossible without the solution of the first, but not vice versa. The second task is posed in our course as an independent and self-valuable one.

So the system of concepts should not only be easy to understand intuitively, but also accessible for expression in verbal form by children of six and seven years of age.

In our experience, intuitive understanding and confident practical use of the concepts listed above is achieved after 12-15 half-hour group sessions of an annual course, and the ability to express understood concepts in a verbal form is achieved only by the end of the course.

It is important that all these concepts form kind of a closed system. Most likely, a six-year-old toddler taking the course will master the first interconnected system of scientific concepts in his life. And the lessons of the course should be organized so that the system of concepts is fully assimilated by each child.

According to L.S. Vygotsky [23], the awareness of any general principle requires comprehensive mastering by the child of a certain system of scientific concepts: '*Scientific concepts are the gates through which awareness enters the kingdom of children's concepts*'.

The principle of programmed control (without feedback) can be understood and realized by a child only after he has mastered the rather complex system of 12 scientific concepts as described above. Of course, verbal explanation by an adult of these concepts to a 6 or 7-year-old child is not possible. Conscious assimilation of these concepts will only become possible if the child is offered some activities that will help him to 'get used to' all these 12 concepts in a game, to 'pass them through' his mentality.

Let us emphasize here that the principle of programming control can be implemented in different ways by changing the set of concepts that are presented as the basic ones and of those that are introduced implicitly. The set of 12 basic concepts as proposed above was selected for the purpose of making it as easy as possible for a child to master each such concept and the entire system of concepts in the activity-game form.

VI. What methods ensure the assimilation of the concepts of the course 'Algorithmics for Preschoolers' by children

The authors managed to structure the first lessons of the course so that each child in the group could simulate the execution of all five actions/interactions as specified above by alternately taking on the roles of the robot, computer, and programmer and could work with the 'materialized' program, i.e., develop the program, load it into the computer memory, and tackle the difficulties, should he/she violate the program development rules.

This was achieved due to the following methodological and technical solutions used during the very first lessons:

1) As suggested by S. Papert as long as half-century ago [24], children get to work not only with virtual robots on the screen, but also with real toy robots that move around across the playroom floor imitating the movements of virtual robots on the tablet's screen (see Fig. 2 and Fig. 3).

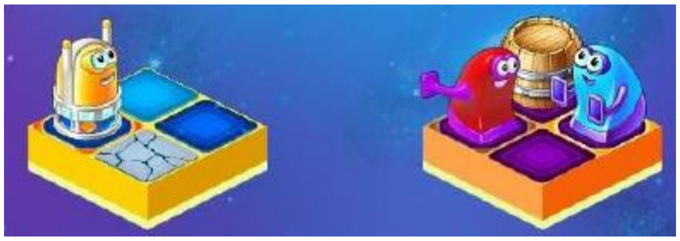


Fig. 2 Virtual robots. Vertun paints the slabs. Tractor pulls boxes and barrels behind it, and Dvigun pushes them in front of it.

Fig. 3 Actual toy robot.

2) A real robot is controlled by a remote control (see Fig. 4) by sound commands: one, two or three beep sounds. These commands can be 'observed' (heard) by the children. After receiving a sound command, the robot announces command's name by voice, and after executing the command, it says 'done', demonstrating its readiness to execute the next command. Children learn to imitate a real robot, and they like to command each other and imitate the speech reactions of the robot outside of the course.

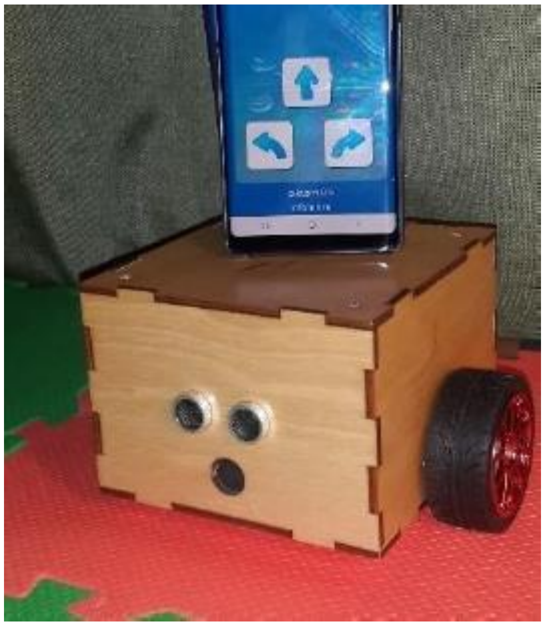


Fig. 4 Remote control for toy robot.

3) Programs are composed of material objects (cubes with pictograms drawn on their faces, representing robot's commands, repeaters, and other constructions of the programming language) (see Fig. 5 and Fig. 6).

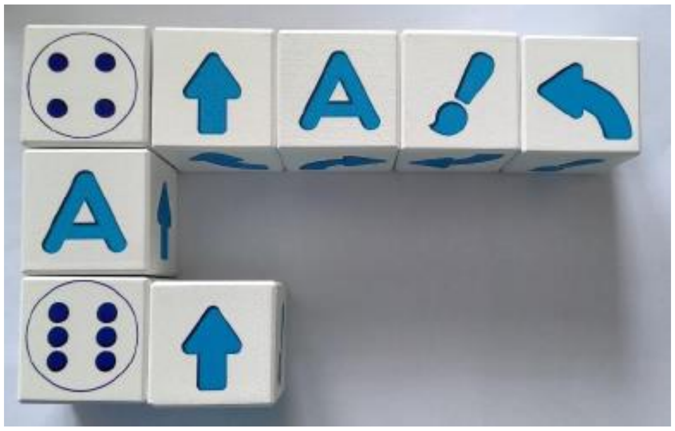
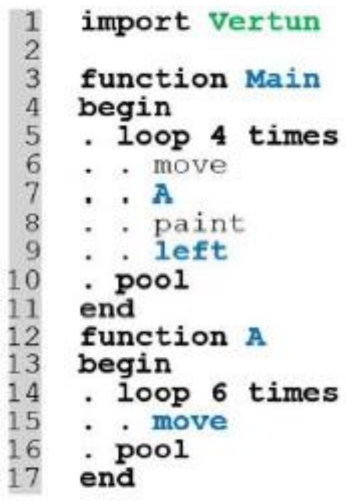


Fig. 5 Program made of cubes by a preschooler.

4) In the first lessons, the tablet acts as a control computer and is not used for other purposes - neither for compiling a program - nor for displaying virtual robots on the screen.
5) The action of loading the program into the memory of the computer (tablet) is also materialized. A child can load a program into the memory of a computer (tablet) by explicitly taking a picture – with the help of the tablet camera – of the program laid out on the table as a certain configuration of cubes. This explicit action is followed by an implicit (hidden) 'understanding' of the program by the computer, i.e., recognition of the photo with cubes by the tablet processor using neural networks. The child can see the result of this understanding on the tablet screen; however, how exactly this understanding happens, is not discussed with the child.



```
import Vertun

function Main
begin
. loop 4 times
. . move
. . A
. . paint
. . left
. pool
end
function A
begin
. loop 6 times
. . move
. pool
end
```

Fig. 6 Shown in Fig. 5 program recoded by a seventh grader into the Pascal-like programming language Kumir

6) Since the elements of the program are presented to the child in material, tangible form, and the creation of the program occurs by mechanical movements of these material objects, the rules for compiling programs are also expressed in visual, understandable to the preschooler, terms. The first syntactic rule for program development requires the cubes to be laid out in fairly even rows, and the rows to be located one under another. The first semantic rule for program development is as follows: a program is executed by reading pictograms in rows from left to right and by reading rows from top downwards. At the same time, the child has to face the fact that the computer fails to ‘understand’ programs compiled in violation of the rules, i.e., the computer ‘does not understand’ the location of cubes on the table, if it is difficult or impossible to mentally detect any rows (see Fig. 6).
7) Two more complex programming rules — the numeric repeater and the one-letter subroutines (functions) — are introduced as a means of solving problems that the child faces directly. The repeater is introduced as a way to facilitate the process of composing a program with repeated chunks. The construction of a subroutine is introduced as a way to ‘encrypt’ program fragments. At a later stage, this construct is also seen as a way to reduce the size of the program. The rules for compiling programs are described in conjunction with the rules for executing programs and therefore can be mastered in practice, in situations where a child imitates a computer and tries to ‘understand’, that is, to execute a program made of wooden cubes by another child. These rules are easily mastered by children, as they are geometrically clear, based on the spatial structure of the program, namely, on indents.

Fig. 7 Illustration by artist Mikhail Gladkov for the report by A.P. Ershov titled ‘Programming is the second literacy’ [25].

The basic principle of conducting the first lessons of the course ‘Algorithmics for preschoolers’. We have structured the course in such a way that in the first lessons, children do not work on computers, but play role-playing games with their peers and teaching aids (a robot and a set of cubes with pictograms). Experiments over many years have convinced us that in the first lessons, a computer should only be used for one and only purpose - that is, to execute programs loaded into its memory, but not for other purposes, such as compiling programs or creating images. virtual robots and virtual environments on the screen. Under this condition, the behavior of the computer is reduced to sequential issuance of sound commands to the robot, which can be easily understood and imitated by a child. This approach accelerates the development of the principle of programmed control. In the first lessons, the program, the robot and the environment in which the robot works are not virtual, but real, tangible objects, and all interactions are real processes in which material objects are involved. Thus, in the first lessons, the studied objects - a robot, a program, a computer - are spatially separated and actions with objects are observable, i.e. visible and audible. Thus, children can gradually master the roles of objects in the course of the session. Here is a short list of the steps that a child goes through in the first lessons of the course.

1) A child becomes familiar with the concepts of the robot, the commands of the robot, watching the teacher command a real robot using the remote control.
2) A child learns all the capabilities of the robot, independently controlling the robot using the remote control. The child remembers the pictograms representing the commands of the robot, seeing these pictograms on

the screen of the remote control.

3) A child 'gets used to' the role of a robot, imitating a real robot in the game, obeying sound commands from the remote control or from the computer.
4) A child can act like a robot by following sound commands received from a computer or from another child who acts like a computer; a child can act like a computer, running a program designed by another child, and command a third child, who acts like a robot.
5) A child can work as a programmer and independently develop a program by moving wooden blocks with icons on the table, and then move on to a new method of developing a program by manually moving the icons on the touch screen of the tablet.

VII. The Structure of the Annual Course 'Algorithmics for Preschoolers'

Most of the lessons of our annual course are devoted to the study of basic concepts and the practical development of control programs without feedback. Each such program solves only one specific problem: it provides the desired behavior of the robot in a specific environment, shown to children on the floor of the playroom or, graphically, on a tablet screen. No-feedback programs are developed using only two explicit control structures (a numeric repeater and a function with one-letter name) and one implicit construct (sequential execution of a linear program block). The 'per minute' manual of the one-year course 'Algorithmics for Preschoolers' includes descriptions of 30 lessons, and 4 more lessons are left as backup. The teacher can use them for repetition, competition, etc.

Sequential execution of a linear section of the program is included in the very first lesson. The repeater construct first appears in Lesson 10 and is introduced as a way to reduce the size of the program. The subprogram (function) construct first appears in Lesson 15 and is introduced as a way to 'encrypt' program that leads to the treasure. At a later stage, this construct is also viewed as a way to reduce the size of the program.

Of course, the expressive power of the two selected control structures is small. All programs that can be developed using these constructs are equivalent to linear programs. However, these linear programs are described by complex nested control structures.

Practice has shown, that a set of meaningful tasks that can be solved using these two control structures is sufficient to keep the children's attention throughout the year (the first 25 lessons out of 30), on condition that a sufficient variety of robots and their graphic representations are created. Currently, the 'Algorithmics for Preschoolers' course uses 5 virtual robots and 1 real robot (Polzun, which means 'Crawler'), which imitates one of the virtual robots. Despite the very short duration of the computer part of each lesson in the course (from 15 minutes in the first half of the year to 20 minutes in the second half) it is possible to have each child perform 4 or 5 tasks in each lesson, in other words, over the one-year course, each preschooler develops 120-150 simplest programs on his own. Independent accomplishment of more than a hundred exercises seems to be a necessary condition for the reliable mastery of the theoretical and practical material of the course.

During the last lessons of the first year of studies, a transition from control without feedback to control with feedback begins, yet is not accomplished (see Conclusion):

1) question-commands are added to the robot command systems, and branching and repetition are added to the programming language constructions provided by the PiktoMir system.
2) a new type of command (a question-command) is introduced into the system of basic concepts and a new type of interaction is added (the robot *answers yes* or *no* to the question-command given by the computer).

Along with the concept of 'feedback', the concept of number (non-negative whole number) is introduced to the system of basic concepts. To 'materialize' the concept of number, a virtual device 'Magic Jug with Stones' is introduced to play the role of a counter. By counting the number of steps, the counter allows it to address the robot control problems, such as 'reach the nearest wall and return to the starting point'.

Of course, introduction of new concepts is accompanied by games. While playing the role of the robot a child answers question-commands 'yes' or 'no'; while acting as the jug-counter he executes commands by adding or removing stones from a real jug and answers the question-commands 'Is the jug empty?', 'Is the jug not empty?', or 'How many stones are there in the jug?'. The introduction of feedback in the 'Algorithmics for Preschoolers' course takes at least 5 lessons. To reliably master these concepts during the next year of study, 12-15 more lessons are needed, which do not fit into the annual course (see Conclusion)

After the feedback was introduced, the children can be given tasks related to the development of 'universal' programs that can work not just in one, but in several different environments. These tasks are also given in the graphical form, without any verbal description of the class of environments in which the program should work; simply at the next level of the game you've got to develop a program that works not in one (like previously), but in two or three given environments, see Fig. 1.

VIII. Why Learning Programming Is a Difficult Issue and How to Reduce Difficulty

Unbreakable internal non-triviality of programming.

The programming metaphor, which introduces the concept of a program, the rules for its compilation and execution, and the concept of an agent performing this execution, is relatively new for our civilization and has its own intrinsic complexity and nontriviality.

A theoretical indicator of the nontriviality of this metaphor is the fact that the behavior of a program, in the general case, cannot be predicted by its formal analysis carried out while executing another program. This fact is often illustrated by the *halting problem incompleteness theorem*. The theoretical impossibility of automating the process of understanding the program in practice translates into the fact that the result of executing even the simplest program is often difficult to predict.

Therefore, mastering the basics of programming is a difficult task that is not reduced to pure logic and is provided only by the practical study of constructions and practices invented by mankind that structures the program and makes it easier to understand the process of its implementation. While mastering programming, one also cannot do without acquaintance with the practice of compiling and debugging programs. In a beginner's programming course for learners of all ages, instructive examples are more important than correct verbal explanations. This is why we systematically use program templates.

The basic constructions and methods of procedural programming form some compact core. This core must be studied in its entirety, whichever procedural programming language and programming environment are chosen. With any approach to building a course, mastering a dozen concepts and practices of this core is possible only if the students independently compile at least a couple of hundred programs. Thus, in order to learn the basics of programming, you can restrict yourself to the simplest programming language and environment that supports basic programming constructs and methods and is rich enough to prepare several hundred tasks that require the use of basic programming constructs and their combinations.

For example, the Pikto language and a programming environment for real and virtual robots such as the PiktoMir system or other textless programming environments such as Scratch Junior.

Coding is the literacy of the 21st century. The use of a textless graphical programming environment opens the way to teaching programming to preschoolers who do not yet know how to read and write. We share the point of view of many experts that our civilization should use this opportunity. Renowned specialist Marina Umaschi Bers in [3] made the following remark in this direction: "*If coding is the literacy of the 21st century, we need to start teaching it early, at the same time that we teach students how to read and write . . . We know that, as a literacy, coding will open doors, many of them that we cannot anticipate now. But we also know that these young coders are children. It is not enough to copy models of computer science education that were developed for older students.*"

The last statement of this quote is appropriate to comment on and develop. We have extensive experience in teaching the basics of programming to beginners of all ages: preschoolers from 6 years old, schoolchildren of all ages, undergraduates, kindergarten teachers and school teachers. It turned out that at any age, traditional approaches to teaching beginners are not effective and lose to the approaches developed for preschoolers. The first steps in learning programming are best done in comfortable, text-less programming environments originally designed for preschoolers. Just compare the effort required to compose or comprehend the miniature program in Fig. 5, consisting of 8 icons, and the text form of the same program in Fig. 6, occupying a respectable one and a half dozen lines.

IX. Conclusion

Based on a 6-year experiment with thousands of preschoolers in the city of Surgut, western Siberia, we draw the following conclusions.

- The choice of a set of basic programming concepts and methods of teaching them to preschoolers, made by us and described in this article, as close as possible to the material world, allows retraining the existing corps of kindergarten teachers for conducting an initial programming course for preschoolers with minimal costs.
- The proposed set of "scientific" in the sense of L. Vygodsky programming concepts is easily mastered by children and can be the subject of meaningful speech practice. Enrichment of the preschooler's vocabulary and the development of monologue and dialogue skills using the accumulated ’professional’ vocabulary is just as important goal of the course as the acquisition of the skills of independent development of the simplest programs in the educational gaming environment.
- The pictogram programming language Pikto and the PiktoMir programming environment make it possible to provide conditions under which each preschooler, by the end of the year's course, gains the experience of independently compiling more than a hundred of the simplest control programs for real and virtual robots.
- Without compromising the quality of teaching, the size of the group with which one teacher works can be brought up to 10-12 children.
- The first lessons of the course can be conducted in a non-screen form.
- An annual programming course for preschoolers, provided that one lesson is held per week, does not allow preschoolers to master the compilation of algorithms with feedback and gain experience in compiling universal programs that work in several similar installations. It is advisable to develop a three-year programming course associated with an elementary school programming course.
- Preschoolers show great interest in the construction of the real toy robot used in the course. The one-year programming course developed today and the multi-year course under development do not satisfy this interest. Therefore, the developed course should be included in a set of STEM courses for preschoolers, covering, in particular, the basics of electrical engineering and other engineering issues related to the device of robots.

The novelty of the work is that the proposed system of concepts and the methodology for teaching programming based on it have proven their effectiveness when introduced into the preschool education system in a city with a population of 400 thousand people.

Let us comment on the conclusions drawn.

Our experience in Surgut, based on working with thousands of children and hundreds of teachers, described in detail in [3], shows that modern techniques and teaching aids allow children to master the basics of programming many years earlier than provided for by the current curriculum. As our experience shows, almost all basic programming concepts and skills provided by the federal school curriculum in Russia can be learned and mastered by preschoolers in a textless programming environment.

With an increase in the duration of the course 'Algorithmics for Preschoolers', all the basic concepts provided by the compulsory part of the Russian school curriculum can be mastered at the preschool level of education.

Given the great developmental potential of studying programming in the education of computational thinking, it is natural to set the task of developing a complete ABC programming course (in Russia the term Algorithmics is often used) for preschoolers. For Russia, where today 99% of children aged 3 to 7 attend kindergartens, this task is especially important. That is why, together with our colleagues from the Samara Institute of Educational Technologies, we started a new project - the development and testing of a complete text-less programming course for preschoolers, designed for three years of study [26]. In this course, a child aged 4 to 7 years old has been studying programming for three years with an intensity of one lesson per week. Individual lessons on tablets begin from the second year of study. The development of such a course can be considered as concretization of the proposals of the legislators of the Russian Federation on the introduction of elements of informatics into the standard of preschool education [1]. In parallel, we are working on the continuation of this course in primary school and the inclusion of PiktoMir in the family of mixed, pictogram-text programming environments for elementary school students [13].

X. Acknowledgements

The work was carried out on the topic 0580-2021-0010 of the 2021 state assignment in the department of educational informatics of the Federal State Institution Scientific Center of the Russian Academy of Sciences (Registration No 121031300134-3).

References

[1] Head of State Duma Education and Science Committee sees it fit to include computer science in the preschool curriculum: Proceedings of the 18th Congress of the United Russia party // TASS, December 8, 2018. URL: https://tass.ru/obschestvo/5888487.

[2] Richtel, M. Reading, writing, arithmetic, and lately, coding // New York Times. May 10, 2014. URL: https://www.nytimes.com/2014/05/11/us/reading-writing-arithmetic-and-lately-coding.html (accessed August 30, 2021).

[3] Matthew Lynch By Marina Umaschi Bers,
Coding as a Literacy for the 21st Century
Education Week, 2018/01
https://www.edweek.org/education/opinion-coding-as-a-literacy-for-the-21st-century/2018/01 (accessed August 30, 2021).

[4] Leonhardt T., Bergner N., Schroeder U., Paving the Way for Computer Science in German Schools, August 2018, Lecture Notes in Computer Science 11011:590-609. DOI: 10.1007/978-3-319-98355-4_34

[5] Bers, Marina Umaschi. (2008). Blocks to robots; learning with technology in the early childhood classroom. New York: Teachers College Press. pp. 154. ISBN 978-0-8077-4848-0.

[6] 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

[7] 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): pp. 28-40. (In Russ.) URL: https://doi.org/10.32517/0234-0453-2020-35-10-28-40

[8] 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,
https://www.niisi.ru/tr/2020_T10_N5.pdf . (In Russ.). (accessed August 30, 2021).

[9] Rogozhkina, I., Kushnirenko, A, "PiktoMir: teaching programming concepts to preschoolers with a new tutorial environment." Procedia – Social and Behavioral Sciences, 2011. Vol. 28. pp. 601–605. doi: 10.1016/j.sbspro.2011.11.114.

[10] PiktoMir: The project starting page at the website of the Federal Scientific Center NIISI RAS URL: https://www.niisi.ru/piktomir

[11] Box Island: http://boxisland.io/

[12] Lightbot: https://en.wikipedia.org/wiki/Lightbot

[13] Beschaposhnikov, N. O., Kushnirenko, A. G., Leonov A. G., Maly, A. A. PIKTOMIR-K - PROJECT OF A BILINGUAL PICTOGRAM AND TEXT LEARNING ENVIRONMENT FOR PROGRAMMING // Fourteenth Conference 'Free Software in Higher Education': Conference Proceedings.]. Pereslavl, 2019, pp. 64-66. (In Russ.). http://0x1.tv/20190126K (accessed August 30, 2021).

[14] Blockly Games: https://blockly.games

[15] Hopster Coding Safary for Kids: https://www.commonsensemedia.org/app-reviews/hopster-coding-safari-for-kids (accessed August 30, 2021).

[16] https://www.python-course.eu/python3_blocks.php (accessed August 30, 2021).

[17] Primo Cubette: https://en.wikipedia.org/wiki/Primo_Toys

[18] MatataLab: https://matatalab.com/en

[19] 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.

[20] Besshaposhnikov N.O. PERFORMANCE OF PARALLEL-COOPERATIVE TASKS IN EDUCATIONAL SYSTEM OF PROGRAMMING FOR PRESCHOOLERS AND PRIMARY SCHOOL PUPILS. Proceedings in Cybernetics. 2017;(4):154-163. (In Russ.).

[21] Kalas I. Recognizing the potential of ICT in early childhood education. Analytical survey. UNESCO Institute for Information Technologies in Education. pp.1-149. (2010) URL: https://iite.unesco.org/pics/publications/en/files/3214673.pdf (accessed August 30, 2021).

[22] 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). DOI https://dx.doi.org/10.14357/19922264200414

[23] Vygotsky. L. S. Myshlenie i rech' [Thinking and Speech]. Moscow, Labirint, 1999, pp. 352 (In Russ.).

[24] Papert, S. (1980) Mindstorms: Children, Computers and Powerful Ideas, New York, Basic Books.

[25] Ershov A. (1981) Programming, the Second Literacy, North-Holland Publishing Company, Microprocessing and Microprogramming 8 (1981) pp. 1-9. URL: https://doi.org/10.1016/0165-6074(81)90002-8

[26] Institute of Educational Technology, Samara, Russia, PiktoMir project participants. https://inott.ru/projects/piktomir/uchastniki-doshkolnoe-obrazovanie/ (accessed August 30, 2021).