Olympiads in Informatics, 2019, Vol. 13, 21–30

DOI: 10.15388/ioi.2019.02

Training in Writing the Simplest Programs from Early Ages

Michael DOLINSKY, Mariya DOLINSKAYA
Faculty of Mathematics and Technologies of Programming, F. Skorina Gomel State University
Sovetskaya str., 104, Gomel. 246019. Republic of Belarus
e-mail: dolinsky@gsu.by, mkugejko@gsu.by

Abstract. This article describes the author approach for start programming teaching at the primary school, which is sequentially teach to create simplest programs that read some numbers, do necessary calculations and write the answer.

Keywords: primary school, programming teaching, distance learning tools.

Introduction

For many years, the authors have been training schoolchildren of the city of Gomel for olympiads in computer science and programming (Dolinsky, 2013). All work is carried out on the basis of the instrumental distance learning system (`http://dl.gsu.by`, hereinafter referred to as DL), created at the mathematics department of F. Skorina Gomel State University under the leadership of one of the authors (Dolinsky, 2017). At the same time, schoolchildren of increasingly earlier ages are drawn into programming training, and since 2007, learning begins in the first grade. According to the authors (based on the results of the performance of Gomel schoolchildren at republican and international computer science competitions), a rather effective system of education has been created (Dolinsky, 2016).

The two first stages of learning were described in detail earlier: mental skills development (Dolinsky, 2014) and learning the starting keywords for programming in Pascal (Dolinsky and Dolinskaya, 2018). This article describes in detail the authors' approach to the subsequent transition to programming training.

Starting from year 2019 Pascal can't be used at IOI, nevertheless the authors consider the article as very useful for our community for the following reasons. First of all, the article describes approach that can be used for any programming language. In addition, authors think that one can continue to teach programming to 7-9 years old children with existing effective system to provide fast and easy learning for early age children. Finally, our distance learning system provides transfer to C++ teaching for any one (Dolinsky, 2017).

№ 1	№ 2	№ 3
Example of data output -2-0-1-7-<2016>	**Example of data entry:** 77 70 69 **Example of data output** t:8(0232)777069	**Example of data entry:** 12 5 **Example of data output** 12=5+7 7+10=5+12
	Example of data entry: 71 23 67 **Example of data output** t:8(0232)712367	**Example of data entry:** 3 1 **Example of data output** 3=1+2 2+2=1+3

Fig. 1. The purpose of training.

In Fig. 1, our learning goal is concisely presented – to teach how to write programs, decisive tasks for regional programming competitions for pupils of grades 1–4, besides it presents the tasks of the Olympiad, which took place in April 2017.

In the first task you are required to display two specified sequences of characters. In the second task it is required to enter three numbers and put them in the correct order, adding other symbols to the output. To perform these tasks, it is required to use the read and write operators in the standard way (readln, writeln). Fig. 2 shows the author's solutions to these problems.

But the third task is already a real Olympiad task. It is not so simple to understand or guess what the program should do. In order to facilitate the understanding of the task and ensure the unambiguity of its formulation, TWO examples of input and output are

№ 1	№ 2

Begin
writeln('2-0-1-7-');
writeln('<2016>');
end.
	var
	a,b,c : longint;
	begin
	readln(a);
	readln(b);
	readln(c);
	
	writeln('t:8(0232)',a,b,c);
	end.

Fig. 2. Author's solutions to problems 1 and 2.

given, a comparative analysis of which allows you to reliably determine what needs to be done in the task. The conditions of the first 10 tasks of competitions for grades 1–4 historically look like two examples of input-output. Initially, we started from the fact that children could not read confidently and, with this in mind, we formulated tasks. On the other hand, the analysis of the conditions of the tasks presented in such a form, by definition, develops mental skills, brings the task closer to the Olympiad task style, and causes additional interest that increases the motivation to practice.

So, let us reproduce the possible analysis of the condition of the third task given. The analysis is made by a pupil. It is required to type two numbers on the keyboard (12 and 5). Next you need to display the line 12 = 5 + 7. 12 and 5 are the numbers that were input. The number 7 was not on the input, hence this is either a constant number, or a number that our program needs to calculate. In the second example, the number 5 is not at this position. It means that the program must calculate it. From the example it is clear how to calculate it. This should be such a number, which in total with 5 will give 12. Hence, to calculate it, subtract 5 from 12. Then output, in the correct sequence, the numbers entered, the calculated number, as well as the symbols "+" and "=".

Now let's turn to the second line of the sample output. 7 is the number we have just calculated, by the second example we check our guess: there is a number 2 on the corresponding position, also calculated as the difference between the entered numbers 3 and 1. On the right side of the output, after the "equal" sign, there are 5 and 12 that were on input. But in the left part there is also a new number 10. Again, it is clear from the example that this is a number that, in total with the number 7 we calculated earlier, should give the sum of the numbers 5 and 12. Therefore, it should be calculated as the difference (5 + 12) – 7. Similarly, for the second example, the right number 2 is calculated as the difference (1 + 3) – 2.

The author's solution of task 3 is shown in Fig. 3.

As for the Olympics, the first such contest took place on October 27, 2008. Training of hundreds of children since that time has led us to a deeper understanding of the problems arising in teaching and to the updating and development of the automatic learning system.

```
var
   s1,s2,s3,s4 : longint;
begin
   readln(s1);
   readln(s2);
   s3:=s1-s2;
   s4:=s1+s2-s3;
   writeln(s1,'=',s2,'+',s3);
   writeln(s3,'+',s4,'=',s2,'+',s1);
end.
```

Fig. 3. Author's solution to problem 3.

Teaching Technology

Thus, the complete list of tasks for learning to write the first program in Pascal (manipulating numbers) seems to us like this:

- Move from words to the text of the first program (input-output numbers, Fig. 4).
- Learn how to work in the Turbo Pascal environment.
- Launch Pascal from the desktop icon.
- Type and edit the program.
- The minimum set of keys for work:
  F2 – save program.
  F9 – check errors.
  Ctrl + F9 – execute the program.
  Alt + F5 – see results.
- Send solutions for testing to the system DL.GSU.BY.
- Use the test assignment (watch the input and output, on which the program gives the wrong answer).
- Search and correct errors in the program by comparing the correct output to the output of the program.

Note that by this time, after completing the first two parts of the training ("Learning to think", "Learning words"), pupils already know how to turn on / off the computer, log into the network, launch the desired program by clicking on the desktop shortcut, to log in with a personal account in the DL system and understand the numerical task. In connection with the automatic differentiated learning, the system itself issues to each pupil a task on which he stopped.

At the same time, it is very important to preserve the motivation of children to practice, so the tasks should be diverse in form, interesting in content and provide differentiated learning, so that each pupil could find a comfortable mode of assignment.

The tasks themselves are lined up in a tree-like form, where the pupil can either proceed to the next task if he has coped with the current one, or to the task system that teaches how to complete the current task.

The words	**Program**
Program	program p1;
var	var
longint;	s : longint;
begin	begin
readln	readln(s);
writeln	writeln(s);
end.	end.

Fig. 4. Words and the program «Input-output numbers».

The standard view of the tree of learning the source code for solving a specific problem is presented in Fig.5.

Fig. 7–14 present the corresponding tasks for teaching the solution of the problem presented in Fig. 6.

The first task that is offered to a pupil who does not get to write the required program is the task (Fig. 7), offering the pupil to do the work instead of the program. That is, to indicate what the program will display if the specified numbers are entered at its input. This ensures that the pupil understands what the program should do. In case the child cannot cope with this task, there is a task in which the numbers necessary for input are displayed on the keyboard. This is especially convenient for self-study. In the classroom, an understanding of the condition may also arise from discussion of a problem with a teacher or another pupil designated by the teacher.

After the pupil has completed the task "Manual solution", presented in Fig. 7, he is invited to type Pascal-program (Fig. 9). In this task, the algorithm for solving the problem is presented on the left, and on the right, you need to enter a program corresponding to this algorithm. And from this point on, differentiated learning begins to work most intensively.

There are children who just have to look at this picture to exclaim "I understand", to press this button and return to the original task (Fig. 6), write the necessary program correctly, check, send it for testing and passing, go to the next task.

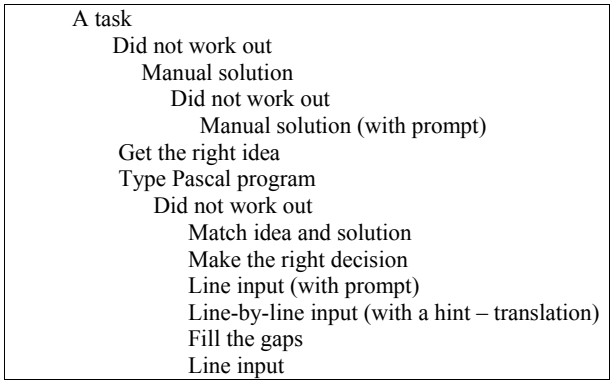
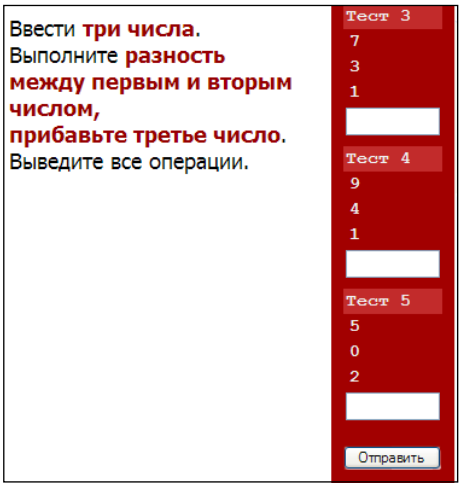


Fig. 5. Standard learning task tree.

Example of data entry:
3 5
Example of data output
3+1=4 4+5=9

Fig. 6. Example task to learn.

There are children who, exclaiming «I understand», still cannot write the program correctly and therefore have to return to the task presented in Fig. 9 again and write the program line by line on the model. If they can do this without a single error, they will return to the task in Fig. 6. If they make a mistake in at least one symbol or immediately press the “I do not know” button, they will get on the task “Create the right idea”, shown in Fig. 8. Here it is required using the permutation of the lines (by clicking on the two lines that need to be swapped) to make the algorithm presented in the previous task. For



Enter three numbers
Determine the difference between the first and second number, add the third number.
Print all operations.

Fig. 7. Manual solution.

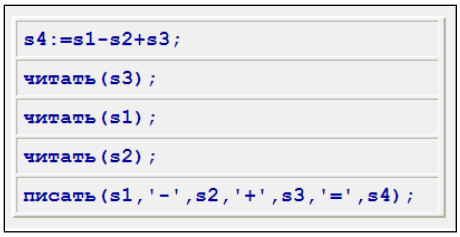


Fig. 8. Get the right idea.

Left side: explanatory lines describing what needs to be done (originally in Russian)	Right side: the lines from the Turbo Pascal solution that perform the necessary actions
Program olymp	
variable	
s1, s2, s3, s4 : number	
begin	
read s1	
read s2	
read s3	
s4=s1-s2+s3	
write s1, ‘-‘, s2, ‘+’ , s3 , ‘=’ , s4	
end	

Fig. 9. Type Pascal program.

those who cannot cope with this task correctly, there is a button “Show correct answers”, by clicking on which the lines are arranged in the correct order, however, the possibility of moving to the next task is blocked, and for the pupil it is required to click on the “Disable hint «and perform the task yourself.

Such a process can be repeated several times, until the pupil memorizes correctly the sequence of actions in the algorithm. After that, he is offered the task “To Match the Idea and the Solution”, presented in Fig.10. On the left there is the algorithm already mastered by the pupil in the past assignment, and on the right – the lines of the corresponding Pascal program, located in random order. The pupil must permute the lines of the program to turn it into the correct one. After that, the pupil can click “I understand” again and take the decision of the main task (Fig. 6.). Some pupils do this, they succeed, and they move on.

Some pupils fail, and they come back to this point. Some are not distracted and continue to perform learning tasks consistently. In order not to repeat – at any moment any pupil can press the “I understand” button, to try to take the completed task and, if it is decided wrong, to go back.

So, then the pupil is invited to the task “Make a decision”, presented in Fig. 11. In this task, permutation of the lines is required to make a program. In case of failure, you can use the buttons “Show correct answers” / “Disable hint”. In the case of this task, the transition to the “Line input” task, shown in Fig. 12, takes place.

An important feature of this task is the color support. That is, if the child is typing correctly, then everything that he is typing glows green, and when the first erroneous character is typed, the entire line of the set becomes red to signal the pupil about making a mistake at the exact moment when he made it. If the pupil succeeds to dial a few characters, “by inertia”, without noticing the mistake, it is enough to delete them one by one from the end until the line turns green again.

I would like to emphasize that we work with all the pupils who come – with completely different skills. And such a task helps the teacher to cope with “problem” pupils.

`программа olymp;`	`s4:=s3+s2;`
`объявление переменных`	`readln(s2);`
`начало`	`program olymp;`
`читать(s1);`	`s3:=s1+1;`
`читать(s2);`	`end.`
`s3:=s1+1;`	`writeln(s1,'+1=',s3);`
`s4:=s3+s2;`	`var s1,s2,s3,s4 : longint;`
`писать(s1,'+1=',s3);`	`begin`
`писать(s3,'+',s2,'=',s4);`	`readln(s1);`
`конец.`	`writeln(s3,'+',s2,'=',s4);`

Fig. 10. Match idea and solution.

Fig. 11 Make a solution.

Fig. 12. Line input.

After all, the standard alternative is that the teacher should look for (himself or with the help of other pupils) every pupil who made a mistake in the case where the pupil cannot cope with it himself.

Moreover, there are other assignments tasks presented in Fig. 13–14, in which you also need to perform keyboard input of the program, which also provides color error prompts.

In task 13 there are tips in Russian, in task 14 – it is emphasized what exactly has changed in this program in relation to the previous ones. The last task in this series (not shown in the figures) is a line-by-line entry of a program with a color prompt of errors "from memory", that is, in the absence of prompting texts. In the case of this task, the pupil is assigned to the task shown in Fig. 9, which checks whether the pupil is able to type the program in the absence of color error prompts. This ensures that the program is memorized by a pupil with any level of prior training. At the same time, more prepared children undergo training faster; less prepared children spend exactly as much time as each of them needs personally to learn how to solve this problem.

Fig. 13. Line Input 2.

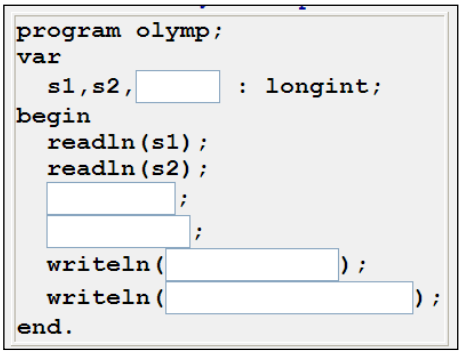


Fig. 14. Fill the gaps.

Differentiated Training

Under the differentiated training, the authors understand the creation of such a system of studies, when each pupil receives tasks that are feasible for him in complexity, and at the same time leading (albeit at different speeds) to the overall final goal.

Differentiation of training is provided by a rich choice of tasks for working at the table and a computer, automatic issuance of tasks on a computer depending on the results of the previous task, as well as a large set of task packages supporting the multiplicity of entry points to training.

Conclusion

In this article the author's approach to the beginning of learning programming in elementary school is considered. Many tasks of varying complexity, both on a computer and for working at a table, are of particular importance. Their proper use provides a differentiated approach to pupils with different training and motivation to practice. It is also important for the authors that, as practice shows, this training system is easily scaled and can be used even by teachers and parents, who were originally far from programming. At first, co-education may occur. But in the end, the children "gaining speed", have the ability to successfully engage in their own study.

References

Dolinsky, M. (2013). An approach to teach introductory-level computer programming. *Olympiads in Informatics*, 7, 14–22.

Dolinsky, M. (2014). Technology for the development of thinking of preschool children and primary school children. *Olympiads in Informatics*, 8, 63–68.

Dolinsky M. (2016). Gomel training school for Olympiads in Informatics. *Olympiads in Informatics*, 10, 237–247.

Dolinsky, M. (2017). A New Generation Distance Learning System for Programming and Olympiads in Informatics

Dolinsky, M., Dolinskaya, M. (2018). How to Start Teaching Programming at Primary School? *Olympiads in Informatics*, 12, 13–24.

Performance Statistics of Gomel pupils at international and national olympiads in informatics since 1997 up to 2018. (In Russian): `http://dl.gsu.by/olymp/result.asp`

M. Dolinsky is a lecturer in Gomel State University "Fr. Scoryna" from 1993. Since 1999 he is leading developer of the educational site of the University (dl.gsu.by). Since 1997 he is heading preparation of the scholars in Gomel to participate in programming contests and Olympiad in informatics. He was a deputy leader of the team of Belarus for IOI'2006, IOI'2007, IOI'2008 and IOI'2009. His PhD is devoted to the tools for digital system design. His current research is in teaching Computer Science and Mathematics from early age.

M. Dolinskaya is student in Gomel State University "Fr. Scoryna" from 2005 then graduate student from 2017. Since 2006 she is one of developer of the educational site dl.gsu.by as well as teacher of pupils from first grade. Her current research is in teaching programming from early age.