Olympiads in Informatics, 2017, Vol. 11, 29–39

DOI: 10.15388/ioi.2017.03

A New Generation Distance Learning System for Programming and Olympiads in Informatics

Michael DOLINSKY

Department of Mathematics, Gomel State University "Fr. Skaryna"
Sovetskaya str., 104, Gomel, 246019, Republic of Belarus
e-mail: dolinsky@gsu.by

Abstract. The article describes concept of new generation system to teach for programming and olympiads in informatics. The new system is oriented to support teaching and learning of arbitrary programming language, to provide personal approach, to guarantee quality of knowledge and skills of graduates. Special attention is devoted for supporting new programming language course authors as well as teachers and tutors. Many base functions of the new system already are implemented at DL.GSU.BY site. From July 2016 we are teaching C++ programming on the base of the new possibilities.

Keywords: programming teaching, olympiad in informatics, distance learning tools.

1. Introduction

At June 16, 2016 the author got an e-mail from IOI (International Olympiad in Informatics):

> *From: IOI-announce [mailto:ioi-announce-bounces@lists.ioinformatics.org] On Behalf Of Bernard Blackham*
> *Sent: Thursday, June 16, 2016 9:24 AM*
> *To: ioi-announce@lists.ioinformatics.org*
> *Subject: [IOI-announce] Proposed changes to allowed languages at IOI 2017*
>
> *Dear IOI community,*
>
> *The ISC and ITC will propose to the GA at IOI 2016 to remove Pascal and C as accepted languages from IOI 2017 onwards. We will also experiment with allowing Python as an accepted language at IOI 2017, but cannot guarantee solutions in Python will score 100%. The accepted languages at IOI 2017 would therefore be: C++, Java and Python. We will ask the GA to vote on the proposals at IOI 2016.*

Why drop Pascal? The number of competitors using Pascal has been very small for several years. In IOI 2015, four contestants submitted solutions in Pascal. The effort for the host country to support Pascal each year is quite significant, as the host writes solutions and graders for every task in every language. Dropping support for Pascal will make it feasible to experiment with newer languages at the IOI by moderating the workload on the scientific committee.

Why drop C? Even fewer C submissions were received (3) than Pascal at IOI 2015. Although supporting C requires less effort than Pascal due to its syntactical and run-time similarities with C++, it still requires model solutions in C to be written and validated. Solutions in C can generally be adapted with few changes to compile as C++ and perform similarly, so we do not expect students to be realistically affected by this.

Why add Python? Python is widely used as an introductory programming language in many countries. The intention of allowing Python is to make IOI tasks more accessible to newcomers to programming.

How will you handle the difference in execution speed of Python? Code written in Python is notably slower than the equivalent compiled C, C++ or even Java. The scientific committee will guarantee that it is possible to solve at least one sub-task of every task using Python, but will not guarantee that Python can score full marks.

Why not drop Java? Java has also had similarly low submission rates as C and Pascal last year. However, as Java was introduced quite recently, it is too early to assess whether we should continue supporting it at the IOI.

Kind regards,
Bernard
(on behalf of the ITC)

To be short, it's proposed to remove Pascal and C as well as to add Python as programming language at International Olympiad in Informatics starting from IOI 2017.

Meanwhile solely Pascal was key language for successful programming teaching and learning (Dolinsky, 2016) in the author's courses at DL.GSU.BY site since 1996.

So we are forced to think about elaboration of a new system oriented on some programming language allowed at IOI. On the other hand many university students and sometimes schoolchildren skip olympiad programming and move to professional programming before they finish studying in university and school accordingly. In addition, it often happens that people with another professional education ask author to help them to study programming to get a chance to find a more profitable employment.

Reflections on these themes led to the concept of a new generation distance learning system that is described below. Moreover we have successfully started to realize it. C++ was chosen by the author as new programming language to prepare for olympiads

in informatics.. C++ programming teaching piloted since July, 2016 and is in stable use since September, 2016.

Chapter 2 presents the base paradigms of the new generation system.

Chapter 3 describes content of the base education course.

Chapter 4 contains author's analysis of advantages and disadvantages of learning C++ vs Pascal in common sense as well as at DL.GSU.BY site.

Finally Chapter 5 contains conclusions.

2. Base Paradigms of the New Generation System

Supporting of arbitrary programming language. Currently there are three programming languages at IOI: C++, Java, Pascal (that will be replaced by Python soon). In the national olympiads Pascal will remain for some time. Many other programming languages are used in professional programming. At the same time there is tremendous layer of common base skills obligatory for everybody, who starts to learn any programming language. What are those skills? First of all the following:

- To understand what need to be done (from the problem's description).
- To elaborate appropriate algorithm.
- To accumulate own library of known algorithms.
- To properly code the chosen algorithm.
- To write readable and structured programs.
- To check program correctness on test data including corner cases.
- To debug code and fix possible errors.
- To train practical skills to code in fast and reliable manner.

Such knowledge and skills are marginally connected with programming language. At the same time it's desirable to give student and teacher the possibility to choose arbitrary programming language they like to study.

Our existing courses based on Pascal contain thousands of exercises, some of that have be prepared manually or generated automatically (Dolinsky, 2013). This was a huge amount of work (that lasted for 20 years) that just can't be timely repeated for another programming language (or several languages). This led to the idea to generate "on the fly" all needed preliminary exercises, using a dictionary of chosen programming language's keywords, reference solution in that language, and first three tests (input/output data) of the problem.

First three tests are used in the first exercise where student need to manually calculate output data for the proposed input data. This exercise ensures that students understand what exactly the program should do.

The dictionary is a list of keywords of programming language and their translation into native language (Russian in our case). These words can be used both to better remember their translation and to train novice pupils' typing skill. The dictionary is also used to translate reference solutions' algorithms to native language which are used in auto-generated preliminary exercises.

Currently we use the following kinds of auto-generated on the fly exercises that pupils can go through in case they are stuck at difficult problem:

- Manually calculating and typing answers for first three tests of the problem (output data for first test is shown as example, Fig. 1).
- Reconstructing the program from its lines permutation given the correct screenshot (Fig. 2).
- Reconstructing the program from its line permutation without a hint.
- Typing the program line by line given correct code example (Fig. 3).
- Typing the program line by line given native algorithm description.
- Typing the program without hints but with error highlighting (Fig. 4).

Figures Fig. 1–Fig. 4 show examples of these exercises:

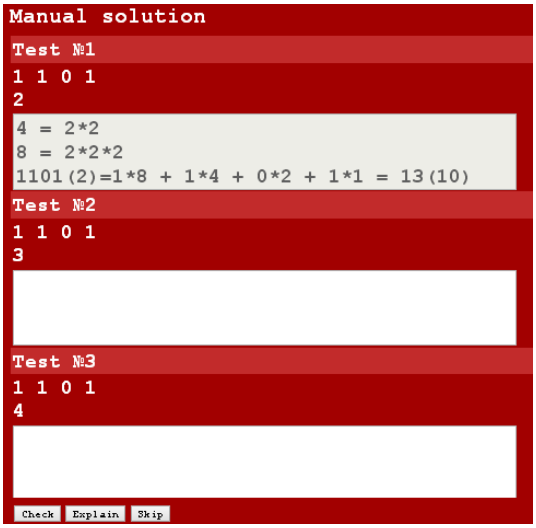

Fig 1. Calculating and typing the answers without programming.

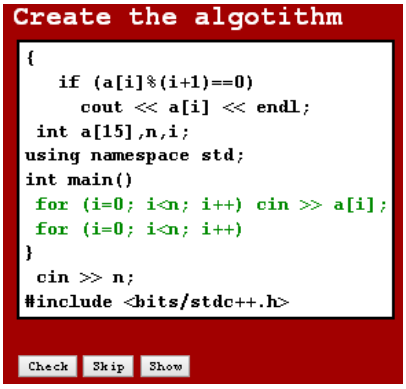

Fig 2. Reconstructing the program from permutation of its lines.

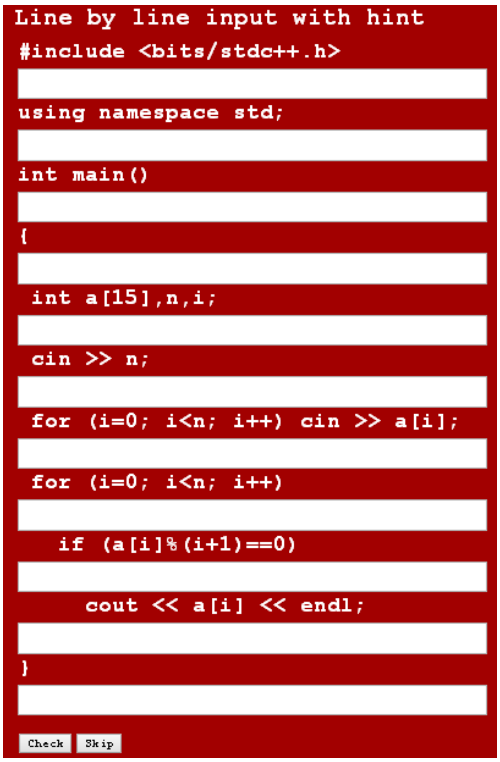

Fig 3. Typing the program line by line with hints.

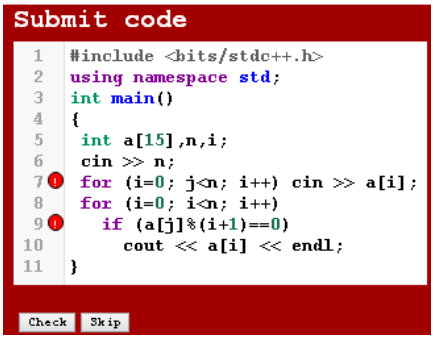

Fig 4. Typing the program with errors highlighting.

Note that colour hints are widely in all exercises. Correct lines are marked green in typing and permutation exercises and wrong lines are marked red in typing exercises.

A pupil can navigate preliminary exercise hierarchy using “I don’t know” and “Skip” buttons. When “I don’t know” button is pressed pupil passes one level deeper in the tree. When “Skip” button is pressed, pupil passes to the next exercise at the current level.

Error highlighting in typing exercises is accomplished with help of standard C++ spell checker. Typed program is at last automatically sent to grading system and then the following problem is shown to pupil.

There are plans to generalize such typing exercises to a powerful WDE (web development environment) with incremental syntax analysis, displaying server-executed program's output and even step-by-step debugger. Web debugger implementation is easier for JavaScript but it should also be achievable for the chosen minimal subset of C++.

Eventually, it's sufficient to add language dictionary and base problems' reference solutions in that language to implement support for a new language in existing courses and training system.

Considerations. New learners can come with wide variety of previous experience and cognitive skills. One may need to complete just one exercise but another would need to go through ten of them to gain same educational effect. One may be familiar with some topics of the course but need to study deeper the other topics. And a good learning system should strive to provide everybody his own versatile effective educational path. For example, a newcomer could get a series of qualification tests on each topic. If test successfully passed then corresponding topic is skipped, otherwise in-depth learning starts to that topic.

Differential study. Base learning course is composed of topics. Each topic consists from four blocks: Technical Minimum, Control, Control Advanced, Practice.

Technical Minimum block contains obligatory exercises with new theory with optional in-depth fine-grained preliminary exercises. .

Control block contains minimal number of exercises to thoroughly check the student's understanding of the theory and ability to make use of it.

Control Advanced block contains advanced exercises that although don't depending on another theory, yet demand smart approach to be solved. Usually we move here exercises from Control Advanced block that are difficult for majority of students.

Practice block contains demonstrative problems from various programming contests where current theory is involved.

Every of Technical Minimum, Control, Control Advanced, Practice blocks can have structured subtopics.

If a student knows how to solve the problem then he types the solution, sends it, scores and gets the new task. Otherwise he can press button "I don't know" button and get a series of preliminary simpler tasks according to one of the following differential study strategies (some already implemented and others planned):

a. **Linear.**

   When "I don't know" button is pressed, student gets a series of described above exercises based on reference solution of the problem (manual answer calculation, code lines permutation, etc.). It is doing so in any from blocks Technical Minimum, Control, Control Advanced, Practice.

   The system remembers for each student the list of tasks where "I don't know" button was used. When student passed the last problems of a topic he is automatically presented with a random series of tasks from his past "Don't

know" list. If he presses "I don't know" button again, then the problem remains in the list. Transition to the next topic happens only after "Don't know" list is cleared. Teacher cans advice student to start from Practice block before Control block.

b. **Accelerated.**

As was written above each topic has 4 blocks: Technical Minimum, Control, Control Advanced, Practice.

After Technical Minimum a student is directed into Control block. If he can't solve a task there and press "I don't know" button, he is automatically redirected to Practice block. He proceeds in "linear" mode there. But in any moment he can return in Control block and continue there. But if student can't solve next Control task, he is again redirected to Practice where he stopped last time. If a student solves all tasks in Practice (and also cleared his "Don't know" list if it wasn't empty) then he is also redirected to last unsolved Control task and further he continues like "linear" strategy is turned on. When student finishes "linear" strategy of the Control block he is then directed to Control* block, with "linear" strategy which also allows skipping too difficult tasks.

c. **Adaptive.**

Each task in the Control block is connected by topic name with a group of tasks in the Practice block. If a student can't solve Control task he is automatically redirected to corresponding Practice tasks and returns back after hi has them finished.

d. **Dynamic difficulty levels.**

We define task's difficulty metric as number of days that it's open for submission divided by number of students that solved it. For example, task difficulty of 1 means that it is solved by somebody averagely every day, value of 30 means average monthly success submission rate and difficulty of 365 implies that the task surrenders only once a year.

Taking into account that the course has been open for many years so far, this automatically calculated metric nicely reflects real difficulty of solving the task for students.

The author of the course can observe difficulty metrics of all problems and pointed difficulty levels. For example, three difficulty levels: 1–10, 11–100, 100+. These levels are used to automatically subdivide tasks of Technical Minimum, Control and Practice blocks into subsets of different difficulty. Each subset can be alone passed along as in "adaptive" strategy.

e. **Diagnostics (for input/intermediate/output control).**

Pupils are presented with a sequence of several tasks in order of ascending difficulty peeked from each topic of Control block. Only passed topics are used in case of intermediate screening. Tasks can be skipped, time may be restricted.

Entry diagnostics can be used for exact determination of entry point to study. Intermediate diagnostics can be used for regular control of the education quality. Exit diagnostics can be used as a base for graduate certification.

Guaranteed knowledge and skills of graduates. Use of intermediate and exit control guarantees the quality of education.

Support of teachers and tutors. We name a teacher the person who will teach group of students "off-line" (in the classroom, for example). Teacher are provided with convenient tools to view the studying results of their students alone and among all other students. We name a tutor the person who will help to teach one or more students not being with them physically in the same classroom or even in the same city. Convenient communication tools for tutors and their students need to be provided.

Gamification. Adding game elements into educational process stimulate increase in motivation to learning. For example:

- Hall of Fame.
- Public rating.
- Prizes for best students from interested companies – T-shirts, icons, notebooks, and pens.
- Time spent and efficiency metrics.
- Dynamic forecast of education results (interpolation based on statistics of all other students).
- Seasonal competitions: Autumn, Winter, Spring, Summer Cups and Person of the Year.

Support of the course authors. Currently we support automated testing of programs in the following languages: Pascal, C++, Java, Kotlin, Python, Perl, Ruby, JavaScript. For a new programming language to become supported it is first of all needed to negotiate installation of its compiler on testing servers.

Then an author interested in particular new language would have to provide the dictionary of its keywords and their translations in native language. It is necessary to translate existing task descriptions into desired native language too in case it's not Russian. In case native language is English, dictionary can be absent and all exercises with translation involved would be automatically hidden from course tree. An author then needs to prepare reference solutions for course tasks in the desired programming languages and validate them in test system. There is a tool to automatically apply chosen correct submissions as reference solutions for a range of tasks.

3. Content of the Base Course

A new "Accelerated course-2016" was created for described new style education model. This course currently contains 8 topics:

1. Introduction to programming.
2. One dimensional array.
3. Two dimensional array.
4. Geometry.

5. Sorting.
6. String.
7. Text problems.
8. Research.

Each of the topics includes Technical Minimum, Control, Control Advanced, Practice blocks.

The themes have the following subthemes:

Introduction to programming: formatted input/output, built-in function, calculus systems.

One-dimensional array: input and output, sum of elements, counting of elements with given conditions, maximum value and its index, minimum value and its index, search for elements with given conditions, combined and modified algorithms.

Two-dimensional array: input and output a whole array or just one row, one column or diagonals; application of base algorithms for one-dimensional array described above for two-dimensional array and it components (row, column, diagonals).

Geometry: distance between two points, distances from one point to an array of points; neighbouring distances; distances between all pairs of points; base and modified algorithms on one- and two-dimensional arrays in application to geometry problems.

Sorting: ascending and descending, coordinated sorting of two arrays.

String: reverse, counting, maximum, search, strings of brackets, all different symbols of the string, shift of strings, string generation, array of strings, splitting in tokens.

Text problems: one-dimensional array, two-dimensional array, geometry, strings.

Research: for (brute force of variable values, brute force on a numeric range); nested for loops (brute force of two variables; brute force of three variables; brute force of digit combinations); while loops; elements of number theory.

4. About Transition From Pascal to C++

Switching students who already study Pascal to another language should be considered wisely. The following variants are possible in author's opinion, but every student take the decision himself:

- Don't switch to C++ at all.
- Switch to C++ after getting national diploma.
- Switch to C++ after passed the "Accelerated Course-2013".
- Switch to C++ from the "Accelerated Course-2016".
- Start learning C++ from scratch.

Advantage of switching to C++:

- Capability to participate in IOI since 2017.
- C++ STL provides ready-to-use algorithms and data structures.
- Reference solutions at USACO, COCI, Codeforces, etc. are mostly in C++.
- Saving time on don't learning Pascal.

- At the beginning is easier to learn keywords:
  C++: main int { cin>> cout<< }.
  Pascal: program var longint begin readln writeln end.
- [Some years later] capability to participate in national Olympiads in Informatics.
- Brighter employment prospects (perhaps?).

Disadvantage of switching to C++ (at least for now):

- More sophisticated IDE for coding and debugging.
- More difficult to ensure code correctness.
- Less advanced learning path for C++ for kids since primary school in our system as Pascal courses are more mature and elaborated.

At that moment author is going to act as follows:

- All beginners since grade 5 and above should start with C++ in the "Accelerated Course-2016".
- The beginners of 1–4 grades from other school start with "Learning to think" course. Further learning path would depend on the progress.
- The beginner of 1–4 grades from our base school start with the language and course preferred by their direct teacher.

5. Conclusion

This paper represents a concept of a new generation distance learning system for programming and olympiads in informatics. The new system is oriented to support teaching and learning of arbitrary programming language, to provide personal approach, to guarantee quality of knowledge and skills of graduates. Special attention is devoted for supporting the new programming language course authors as well as teachers and tutors. Many base functions of the new system are already implemented at DL.GSU.BY site. From July 2016 we are teaching C++ programming on the base of new possibilities.

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 childrens. *Olympiads in Informatics*, 8, 63–68.

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

Dolinsky M. (2005). *Algorithmization and Programming with TURBO PASCAL: From Simple to Olympiad Problems: Tutorial*. Sankt-Petersburg: "Piter" (In Russian: *Алгоритмизация и программирование на Turbo Pascal: от простых до олимпиадных задач: Учебное пособие*. СПб.: Питер).

Dolinsky M. (2006). *Solving of Sophisticated Olympiad Programming Problems: Tutorial*. Sankt-Petersburg: "Piter" (In Russian: *Решение сложных* и *олимпиадных задач* по *программированию: Учебное пособие*. СПб.: Питер).

Performance Statistics of Gomel pupils in international and national olympiads in informatics from 1997 to 2016. (In Russian).
`http://dl.gsu.by/olymp/result.asp`

M. Dolinsky is a lecturer in Gomel State University “Fr. Skaryna” 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.