

# An Approach to Teaching Introductory-Level Computer Programming

Michael DOLINSKY

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

**Abstract.** This article describes a methodology, proposed by the author, for teaching beginners to computer programming from scratch. The methodology is dedicated to teaching groups of students with various levels of knowledge and motivation. The technical base of the methodology is the distance learning system “Distance Learning Belarus”, briefly DL, created by the author.

**Key words:** programming, programming languages, improving classroom teaching; authoring tools and methods, interactive learning environments, intelligent tutoring systems.

## 1. Introduction

Teaching to program today is very hard and interesting task. This introduction is a brief sketch of the recent effort of researchers to make the teaching process more effective. The author thoroughly studied the publications of the journal “Computers & Education” because it is in accordance of the theme as well all the papers of the journal being open access. The main directions presented in the research are: interactivity of the classes; enhancing the learning performance using personalized diagnosis; designing an adaptive web-based learning system; analyzing effects of different types of feedback in a computer-based learning; different methods of examination; dynamic assessment; web-based learning; discussion forums; combining cooperative learning methods; computerized tool support for software engineering education; automatic evaluation; team work; computer simulations; developing computer skills of young children.

Interesting ideas are proposed in the articles devoted to introductory-level computer programming courses by Hawi (2010), Kordaki (2010), Mouraa and van Hattum-Janssenb (2011). These works also pointed the main problem of introductory-level classes: the various levels of motivation and skills of the students in one class.

The author of this article has, for many years, taught introductory-level computer programming course at the Gomel State University “Fr. Skaryna”(Belarus). The paper summarizes his experience and methodology developed by him for teaching programming which could be interesting and useful for other teachers and researchers.

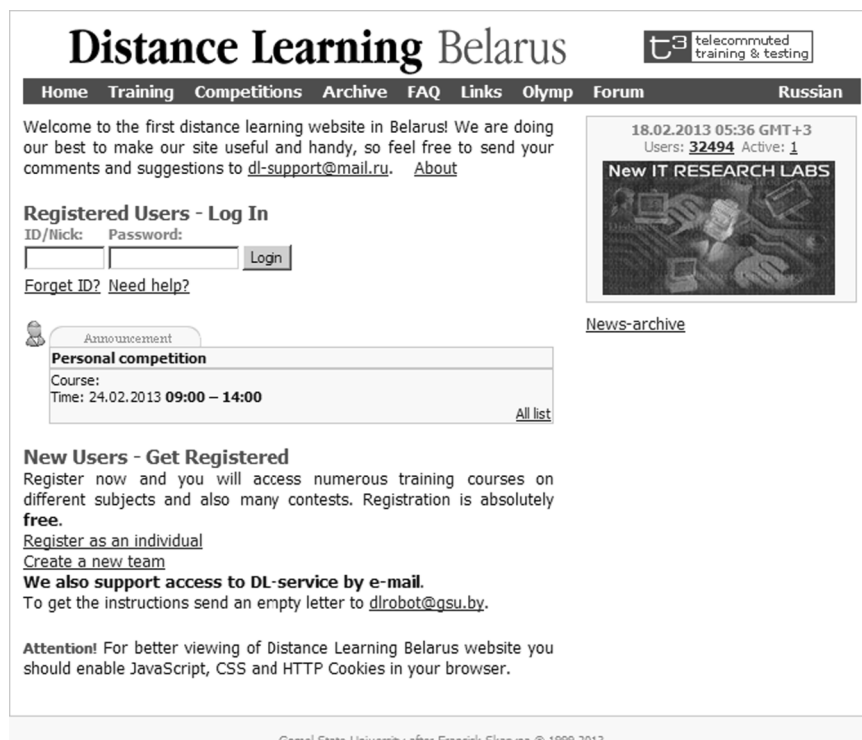
The technical base of the approach is the distance learning system DL available at the address [dl.gsu.by](http://dl.gsu.by); Section 2 presents this software tool. The site has Russian and English

version, but essential part of the training materials is only in Russian. Section 3 gives a general overview of the training content. Sections 4 and 5 stress on the specificity of the theoretical and practical classes, respectively. Section 6 describes the marking scheme for students. Section 7 explains self-managed students work. Finally Section 8 contains conclusions.

## 2. Distant Learning with DL

Our DL is based on modern Internet technologies (Fig. 1). Hence it can be used for distance learning, decreasing the importance of geographical close positions between students and teachers or study materials, as well as excluding the necessity of participation of students and teachers in the study process at the same time. Moreover, the DL is used successfully to increase the quality of the education process in our own university.

The system distinguishes the following kinds of users: *viewer*, *student*, *tutor*, *teacher*, *author*, and *administrator*. The permissions of the users are increasing from viewer to administrator. Viewer is the only kind of DL user that could be unregistered. A viewer can only see the published results of any active or archived course. A student has access to the theoretical lessons and related tasks and can submit problem solutions to the system



The screenshot shows the start page of the 'Distance Learning Belarus' website. The page has a header with the title 'Distance Learning Belarus' and a logo for 'telecommuted training & testing'. Below the header is a navigation menu with links: Home, Training, Competitions, Archive, FAQ, Links, Olymp, Forum, and Russian. The main content area includes a welcome message, a 'Registered Users - Log In' section with a login form, a 'Personal competition' announcement for a course on 24.02.2013 from 09:00 to 14:00, and a 'New Users - Get Registered' section. A sidebar on the right shows the date and time (18.02.2013 05:36 GMT+3), the number of users (32494), and active users (1), along with a 'News-archive' link. The footer contains the text 'Gomel State University after Francisk Skaryna © 1999-2013'.

Fig. 1. Start page of DL.

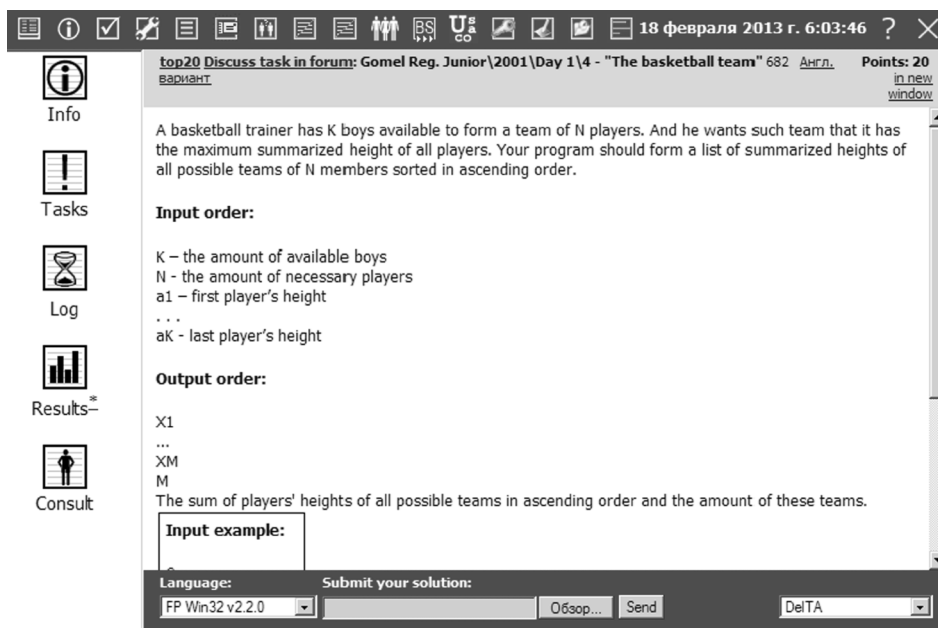


Fig. 2. Typical page for problem solving.

(Fig. 2). Each submitted solution is automatically tested by DL. The results of testing are collected in a *course results table*. A student can see her/his own *test protocol*. A tutor is nominated by a course teacher and can do part of the teacher's work when it is assigned to them by the teacher. A teacher has access to all test protocols and solutions of their students as well as to create a group of their own students. The author's role is to create new courses, as well as to upload/change/replace theoretical materials and task sets of their courses. An administrator controls permissions and processes at DL.

DL allows usage of multimedia presentations not only of the theoretical materials but also of the task tests. A good example for these possibilities is the course "English DEMO". The tests for students in this course contain graphics, pictures, sound and video.

DL liberates the teacher from a considerable amount of work in organization and control of study process. In addition, DL automatically displays the current results of the study process, stimulating the students to work hard in order to ameliorate their published announced results.

Currently, DL is used for teaching students of mathematical department of Gomel State University in the following courses: "Computers and programming" (for first-year students), "Computer foundations" (for third-year students), and "Foundation of computers" (for fourth-year students).

In addition, DL is actively used for studying informatics by secondary school students (from 1st to 11th degree) as well as for preparing them for competitions in informatics and programming. For this purpose the following training and competitive courses are recommended: "ACM tasks", "Preparing for programming contests – Profy", "Preparing

for programming contests – Beginner”, and ”Introduction in informatics”.

Annually, the Gomel town round and the regional round of the national olympiad in informatics are using DL. In addition, from 1997 to 2006 Gomel Computer Science Week (<http://www.gsu.by/gcsw/>) was using DL to provide the set of contests: in Pascal/C programming for IBM PC; in solving chess problems; in solving mathematical problems; in digital system design; in assembler programming for microcontrollers Intel 8051/8086, Motorola 68HC05/08, Atmel AVR, Texas Instruments TMS370, Microchip PIC.

Now we have more than 32000 registered users from 80 countries.

### 3. Content of Training

Here and in the following sections we will concentrate on the introductory-level computer programming course that is studied by the first-year students in their first semester. The main difficulty in such course is the enormous difference in the students' level of knowledge and skills. Studying informatics in a Belarus high schools is oriented mainly to usage of computers, e.g., preparing documents with text and graphics editors, writing e-mails and surfing in Internet. Only a few students achieve deep knowledge in computer programming due to special lessons or individual work. The main studied programming languages are Pascal and C++.

The learning technology in the author's classes is based on weekly cycles. Each week begins with a theoretical lesson, then some practical lessons follow and an examination closes the week. The topics of the classes, arranged by months are given below:

September: Introduction to programming. Debugging. One-dimensional array (standard and non-standard algorithms).

October: Two-dimensional array. Geometry. Strings – elementary algorithms. Strings – standard functions.

November: Strings – user defined functions and procedures. Sorting. Queue.

December: Recursion. Recurrence relations.

Most of the students are using Pascal because it is easier for novices, but the experienced students that know C++ can use it to solve the tasks. The main objective of the course is not to teach the programming language syntax, but the fundamental knowledge about basic algorithms and program development, testing, and debugging. Special attention is paid to good structuring of programs that helps to understand easier what the program is doing and how is doing it.

In the mentioned above topics the following concepts are considered:

*Introduction to programming*: data types (char, string, longint, real), arithmetic operators (+, -, \*, /, DIV, MOD).

*Debugging*: execution – line by line, to the cursor, to the end; open of watch window, add variable to watch windows, windows control.

*One-dimensional array*: sum, count, maximal, minimal, operators for and while.

*Two-dimensional array*: rows, columns, diagonals. Algorithms for walking a two-dimension array.

*Geometry*: coordinates of a point, distance between two points, one-dimensional array of the distances from point to set of points, two-dimensional array of distances between all pairs of points of a set.

*Strings*: standard procedures and functions (*copy*, *delete*, *insert*, *pos*, etc.), user defined procedures and functions.

*Sorting*: bubble sort, exchanging sort, counting sort.

*Queue*: problem of knights on a chessboard, filling fields.

*Recursion, Recurrences*: general knowledge and simple examples.

#### 4. Theoretical Lessons

The quantity, as well as the quality, of the theoretical material for each topic is carefully selected. All necessary fundamental knowledge is presented in such way that it can be acquired by the good students (about 1/3 of the group) within 15–30 minutes. The remaining time of the lesson is used for working in small teams (2–3 students) on comprehending the new knowledge and using it for solving specially chosen tasks with ascending difficulty.

Theoretical classes are held in an auditorium, equipped with beamer, screen, mobile notebook for the lecturer, wireless access to the university network, as well as power supply for students laptops. Each student of the class has a computer in front of him. These computers can be used by students as an alternative to the big screen, where the lecturing material is projected. After the lecturing part of the theoretical lesson is finished, students can browse the theoretical materials uploaded in the system and post questions on the topic.

To stimulate students' cognitive activity, from the beginning of the lecture (to be more exact, from the break before the lecture) a team contest starts. The contest consists of set of task on the topic with ascending difficulty. The students can discuss tasks in the teams. They have to write programs that solve the problems and submit them for automatic evaluation. During the contest the teacher's screen displays the dynamically changing table of the contest's results. At the same time a special personal web-based learning page is open for weak teams that help them understand the theory and to succeed in solving some of the easier tasks of the contest. The tasks are chosen in such a way that no team may solve all tasks (otherwise new tasks are added) and at the same time each team can solve at least one problem (thanks to the help provided during the contest for solving easiest tasks).

#### 5. Practical Lessons

There are three types of practical lessons: *learning lessons*, *common exams*, and *individual exams*. For learning lessons students can choose one of the three possible levels of decreasing difficulty: *individual tasks*, *learning tasks* or *preparing for learning tasks*. During the lessons of the last two types the system automatically provides to students problems

Table 1  
Numbers of main/all tasks by topics

Theme	Number of main tasks	Number of all tasks
Introduction to programming	28	4126
One dimension array	44	704
Two dimension array	19	430
Geometry	26	160
Strings	139	1552
Sorting	12	124
Queue	18	147
Recurrences	8	23

from a *problems tree* prepared for personalized instruction. “*Preparing for learning*” tree contains the easiest problems in order to provide to beginners possibility to start up. The tree is chosen instead of a sequence of tasks to provide learning more adaptive to a student’s preparation level. Some students need more detailed explanations but for some a less detailed explanation. If a student makes a mistake solving exercises, the system automatically present them with the first one from the corresponding learning tree. Additionally there are the buttons “Don’t know” and “I understood” given to students for their own navigation on the problems tree. All touched (by students) exercises get the color (green if it solved and red otherwise), so students as well as teacher (for any student) can analyze the learning path.

Because of the automatic delivery of problems to the students we succeed in achieving adaptive and individual teaching of students, independent of the teaching of the others. For each topic we have uploaded in the system small number of *main tasks* which are compulsory for solving by each student and large number of *additional tasks* (see Table 1) organized in tree structures. Trees of tasks are visualized when student cannot solve the main task or some of the additional tasks.

Additional tasks could be very different by form and content. The main task always demands student send the text of a program (in Pascal or C/C++) that solves the given task (Fig. 2). But the additional problems are dedicated to teach the student how to write the corresponding program if they cannot do it themselves. The first type of such tasks demands the student submit an output for given task input. The goal of such an exercise is to provide the student with understanding what the program must do. Then the student has to do exercises for choosing right algorithm, constructing a program with given program lines (Fig. 3), typing the program with help from the system (Fig. 4), filling the gaps in a program (Fig. 5), etc.

Note, that exercise in Fig. 4 gives to student essential help, highlighting by red color mistaken letters, and by green color right letters.

Every week one of the practical lessons is dedicated to a group exam. It includes 20–30 tasks (common for all students) with various difficulties. Because the problems are common, students have good opportunity to discuss their solutions after the class.

```

for i:=2 to 10 do
max:=a[1];
program max10;
  if a[i]>max then max:=a[i];
end.
writeln(max);
var
  a : array [1..10] of longint;
  max,i : longint;
begin
  for i:=1 to 10 do readln(a[i]);

```

Fig. 3. Constructing the program from lines.

```

program max10;
var
  a : array [1..10] of longint;
  max,i : longint;
begin
  for i:=1 to 10 do readln(a[i]);
  max:=a[1];
  for i:=2 to 10 do
    if a[i]>max then max:=a[i];
  writeln(max);
end.

```

Fig. 4. Typing the program with help.

Something more, for most difficult tasks special educational materials are prepared that become available after the exam.

To combat cheating during the exams by students (where they send solutions to colleagues or receive solutions from them) there are individual exams. Such exams includes 10 tasks on topics taught during the semester. Each student gets their own personal set of tasks, chosen from the problem bank randomly. Another important feature of individual exam – it cannot be done from student’s notebook but only from a stationary computer of the university class and in special account that has read/write permissions only on a special empty folder in the local computer. Each student can repeat the individual exam until they gets the needed mark.

```
program max10;  
var  
  a      : array [1..10] of longint;  
  max,i  : longint;  
begin  
  for i:=1 to   
  max:=  
  for i:=2 to 10 do  
    if   
  writeln();  
end.
```

Fig. 5. Filling the gaps.

## 6. Evaluation

Evaluation of the course is built to achieve the following objectives:

- to encourage students to attend each theoretical and practical class;
- to encourage students to work hard each minute of each class;
- to make each student's mark as objective as possible.

The final score is the minimum from the score of individual exam and the average score earned during the semester. The average score is accumulated from marks from theoretic examinations, practical examinations, solving learning problems, solving individual problems, bonuses and attending classes. Bonuses are assigned by teachers to encourage students' conscientious learning and cognitive activity. For example, bonuses earned by a team during theoretical classes are proportional to the number of solved tasks. Skipping classes reduces the average score.

## 7. Self-Managed Students Work

All theoretical and practical classes are performed on the base of the web-based e-learning system dl.gsu.by. Beside the other functionality it gives students good opportunities for independent work. In particular, students can work off the skipped classes by solving individual problems, problems from learning or preparing for learning sets. Essential help in the independent students work is provided by the *forum*, where everybody can post their own questions and get answer from other students or the teacher. In addition, the solutions of all individual problems are described in the special topic in the forum. Links to these solution descriptions are systemized by the teacher, so it becomes an additional bridge for passing from solving relatively easy learning problems to solving more difficult individual problems.

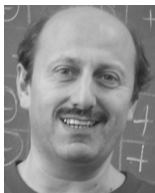


## 8. Conclusion

This paper describes the author's methodology of teaching introductory level programming course. The methodology is oriented to teaching groups of students with various levels of motivation and preparation. An excellent technical base of the course is the web-based teaching/learning system DL ([dl.gsu.by](http://dl.gsu.by)), created with the supervision of the author. Applying the described teaching methodology provided essential shift up in the quality of teaching and especially in the teaching of the less prepared and motivated students. At the same time, all other students, including the most prepared and motivated, were also satisfied with such an approach for studying. Their opinion can be found in the forum of the site, where all students are encouraging to answer the questions "What you like (or dislike, or propose to change) in theoretical lessons, practical lessons and evaluation system?"

## References

- Hawi, N. (2010). Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course. *Computers & Education*, 54(4), 1127–1136.
- Kordaki, M. (2010). A drawing and multi-representational computer environment for beginners' learning of programming using C: design and pilot formative evaluation. *Computers & Education*, 54(1), 69–87.
- Mouraa, I.C., van Hattum-Janssenb, N. (2011). Teaching a CS introductory course: an active approach. *Computers & Education*, 56(2), 475–483.



**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](http://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.