Acta et Commentationes, Sciences of Education, nr. 4 (18), 2019 ISSN 1857-0623
p. 7-16 E-ISSN 2587-3636

CZU: 372.8004 DOI: 10.36120/2587-3636.v18i4.7-16

TEACHING COMPUTER PROGRAMMING IN SCHOOLS

Krassimir MANEV[*], professor, doctor

New Bulgarian University, Sofia, Bulgaria

*Lecturer in Discrete Mathematics, Programming and Algorithms in New Bulgarian University, Sofia, Bulgaria; co-founder of International Olympiad in Informatics (IOI); Team Leader of Bulgarian team; 11 years member of International Committee of IOI (including President of IOI '2009) and President of IOI for the period 2014-2017; in 2017 he co-founded European Junior Olympiad in Informatics and is its first President.

Summary. Starting with a brief historical incursion of the introduction of computer science into the school from the author's perspective, the article outlines some theoretical aspects regarding the modern trends in the formation and development of programming skills of students and their involvement in computer competitions.

Key words: computer programming, competitions in Programming, teaching Informatics in secondary school, task.

STUDIEREA PROGRAMĂRII ÎN ȘCOLI

Rezumat. Începând cu o scurtă incursiune istorică a introducerii informaticii în școală din perspectiva autorului, în articol se elucidează unele aspecte teoretice cu privire la tendințele moderne în formarea și dezvoltarea competențelor de programare la elevi și implicarea acestora în concursurile de informatică.

Cuvinte cheie: programare, concursuri de programare, predarea informaticii în școli, sarcină.

Introduction

I have started to learn computer programming more than 50 years ago – during the far 1966, in the National mathematical school in Sofia. My first computer was "Minsk 2" – a machine with 2K 36 bits words memory, practically without input/output devices – only punched tape reader and writer, and primitive printer that was able to print a single float point number on a line. There was no OS, no IDE and even no compiler from some used in that time language. Something more – there was no even Assembly language and Assembler. We created our first programs in the machine language, punched them on tape and operators executed them manually from the control console of the computer and brought us a short piece of paper with one-two numbers that was the result of our programs. Usually that was solution of some equation.

In my first years at the Sofia University we started to learn FORTRAN and PL/1 but without the possibility to execute our programs at all. Meanwhile the "modern" machine "Minsk 32" was installed in the Institute of Mathematics of the Bulgarian Academy of Sciences and we had the possibility to write our diploma thesis in Assembler. This machine had some simple OS functions. The programs were punched on much more comfortable 80 columns card and results were printed on 128 columns printers. The machine had 8 magnetic tapes reading/writing devices and we could save our programs to tapes and execute them many times from tape without reading the cards and reassembling for each execution. One researcher of the Institute created in that time, not so complex, plotter and I graduated with a diploma thesis on a system that could, by given from the cloth constructor cuts of man shirts, to create ant draw on the plotter any asked by the user size of a shirt cut.

When I have started my academic career as an assistant professor at the university the things changed totally. First the Eastern Europe's IBM 360 compatible machines – so called "ES series" – were installed everywhere, with a true OS, disk devices, Assembler and compilers from FORTRAN and PL/1. Teaching of Assembly language of IBM 360 (the best I have ever seen) goes in parallel with teaching of the procedural languages. Anyway the cycle of compiling a program, finding the obtained bugs of compilation or errors of the logic, punching replacing cards and submitting the deck of cards for new compilation/execution lasted usually one day. When a bit later the terminals were attached to our ES 1040 with a corresponding system for distant creation (what revolution was creating and editing the code on the screen of the terminal) and execution of the programs started to looks, more or less, like the nowadays technology of programming.

At the end of 70's and the beginning the 80's years of the past century computer programming entered the secondary schools of Bulgaria – exceptionally in the profiled mathematical school. Methodology of teaching (if something as methodology existed at that moment) was directly moved from the universities to the school. And university professors participated in this process en mass because there were no teachers in programming at all. Even in this early moment it was clear that secondary school students with mathematical abilities very quickly perceive the conceptions of the computer programming and very quickly get significant skills in solving tasks with computer programs. It was clear also that, for these talented students, the very limited school program in Informatics is not a challenge at all and we had to support their ambitions for development in the domain in some other way.

At the end of 70's, almost the same time when in USA started the Collegiate Programming Contest for university students (famous nowadays all over the world as ICPC) we started the first competitions in Programming for secondary school students. Organizing programming contests on ES (IBM 360) machines was very hard. I could write a separate paper about difficulties we have to overcome. But here I will limit myself to a simple example – at that time we had to evaluate programs of the contestants reading the source code. The time of 4 hours was too short at that times, the contestants did not succeed to finish their programs and checking them by test cases was impossible.

The things changed revolutionary with appearing of the personal computer. Bulgaria started producing own Apple II compatible personal computer Pravetz 8 and any secondary school was able to install as many of these machines as the school needs. Any school student with corresponding interest could start to learn programming. In the middle of 80's we created the National Olympiad in Programming for school students. Each contestant used its own machine and could create and debug in depth her/his program. In 1987, by an idea of prof. Blagovest Sendov, we organized the first international contest in programming with participation of 7 countries, and with two age groups. Enthusiasm from the results was enormous and prof. Sendov asked from UNESCO approval to organize the first International Olympiad in Informatics (IOI) in Bulgaria. This year IOI has its 31st edition in Baku, Azerbaijan with participation of 327 students from 87 countries.

Almost 40 years of my life are closely related with the competition on programming. I started as a task creator and coach, was President of the National team for Olympiads in Informatics, Leader and Deputy Leader of the National team for many IOI and Balkan Olympiads in Informatics. I have served 11 years as a member of IC of IOI and from 2014 to 2017 was a President of IOI. After the end of my presidency I have created with a team of colleagues the first international programming contest for school students aged under 15,5 years – the European Junior Olympiad in Informatics (EJOI) [5]. The 3rd edition of EJOI took place this year in Maribor, Slovenia, with 90 contestants from 24 European countries.

During all this years I did not stop my work as University professor, mainly in Sofia University, but also in American University in Bulgaria, New Bulgarian University and other Bulgarian universities too, teaching Discrete Mathematics, Programming and Algorithms. As University professor I was coaching long years the team of Sofia University for Regional ICPC contest and 4 times participated in ICPC World finals. My participation in the International programming contests give me the opportunity to meet colleagues from many countries of the world and to share opinions with them on teaching programming and related topics.

During my academic career I have published more than 75 scientific and methodological papers; I have led a team that created more than 25 textbooks for Bulgarian school in Informatics and Information Technologies and published 2 textbooks for university students – in Discrete mathematics and in Algorithms in graphs.

I decided to start this paper with such extensive description of my academic career because, I suppose, I know almost all history of teaching programming in my country and the big part of the history of teaching programming in the world. This is crucial because in the next lines I would like to stress some negative trends in the process of preparing programmers – an activity that I consider extremely important for the future of the Humanity!

Teaching programming in the secondary schools and the universities

Teaching Informatics came in Secondary schools relatively later – in 70's – 80's years of past century. It came in the moment when Mathematics and the other natural sciences have their strong place in the school curriculum. And it happens that there is no room for teaching Informatics. That is why teaching the elementary things in Informatics – what is computer, what is OS and program, and how to create programs – is taught in out of class or in some elective forms. More serious teaching of Informatics we have only in some profiled schools (so called mathematical schools) but not in all. With wide spread of computers and computerized devices another discipline – Information technologies – came in the curricula and there is a tendency to almost totally displace teaching of Informatics.

For example, let see nowadays the curriculum of the Bulgarian school. We have elective classes (one hour per week) in Information technologies for 1st-4th degrees of schools that have the possibilities to organize such education. Then we have mandatory classes (one hour in week) in Information technologies for (5th-10th degrees). And mandatory classes (two hours in week) in Informatics for 8th degrees only and not in all schools! More Informatics is taught in mathematical schools when there are well prepared teachers. On this misbalanced background Bulgarian ministry in education introduced in 3rd-4th degrees new (third) discipline with the pretentious name Computer modeling! The idea was in this discipline classes to be introduced some programming environment for children (Skratch-like) but we strongly doubt that it happens.

This colorful and misbalanced process is implemented under the conditions of permanent lack of qualified teacher – not only in Informatics but in Information technologies and new discipline Computer modeling, too. The lack of teachers in Informatics and related disciplines is easy explainable when we have in mind that the software industry "sucks" almost each qualified young professional proposing 5-10 time higher salaries. That is why some of the teachers are not prepared to teach the material in assigned classes, especially the teachers in primary schools who have to taught many disciplines. So, "education" pass under the slogan "computer is yours, make what you wish".

Classes in Information technology are too much, and in these classes, in a "spiral", are repeated exercises with the applications from MS Office. Some of the students know well these programs and the other have no interest and teaching loses sense. Classes in Informatics are not enough even for introducing the main conceptions of the domain. If some programming is included in the Curriculum it is dedicated to creating very small applications with graphical interface, which is rather technology (creating screen forms and linking some standard function events of the form) but not real programming. And especially when the teachers are not prepared enough they leave students to play games and to surf in the net. What happens in the classes of Computer modeling is full mystery.

We have information that some teachers are reading fairy tales in these classes (this is 3rd-4th grade any way and there is not too much to expect).

In other papers, discussing the situation, we proposed a reasonable solution of this not normal state of the art. We would like to repeat it here. The school needs only one discipline – Informatics and Information technology – but from the first to last grade and with the existing number of class hours of the three disciplines. In primary school the student have to be taught the main skills for working with the computer – for example, correct work with the keyboard (10-fingers typing) and the mouse, principles of the OS (executing programs, works with the file systems) and some introductory abilities to use application (typing and editing of text, for example).

In middle grades (5th-7th) students have to be taught, as in the moment, to more deep usage of the applications (formatting big documents, structuring data in tables, presenting data, and drawing with a graphic editor). But in parallel some lessons in Informatics have to be introduced in these years, increasing each year the number of class hours. In 8th-12th degrees of the regular schools the main part of the time has to be dedicated to topics from Informatics – carefully chosen parts of Discrete Mathematics (not such a thing in the secondary school at all), Operating Systems (role and functions of OS are hidden under the graphic interface and students do not understand well them), Programming and Algorithms, Networking. Lessons in programming have to repeat the ontogenesis of the discipline. The students have to be able to write procedural code, before to pass to Object-Oriented programming and making applications with graphical interface and network applications. Some technology classes in this period have to be included too – on Data Bases and Information systems. In profiled mathematical school the number of classes could be 2-3 times bigger and teaching could be more intensive and comprehensive [4].

Only with such ultimate discipline in the secondary school it is possible to expect that the students accepted in university programs will be able to study successfully and to graduate in time. And most of the university programs in Informatics and IT did not consider this particularity of the day – now the students following programs in Informatics and IT are much more than in time when the corresponding Curricula are created. And the average preparation level of the students that come from the Secondary schools is below the level in that time.

Students have not mathematical culture (they just know to solve some typical tasks of the school mathematics but not to reproduce proof of a theorem or to construct their own proof), so they do not understand the lectures in Discrete Mathematics [1]. As a result they are not able to understand and use complex abstract data type and to structure data. As well as they are not able to understand more complex algorithms (because each complex algorithm is based on some more or less deep theorem). So the universities produced mainly Web programmers and programmers of the routine programs with graphical interface. The same are doing many “academies” created by big software companies of societies of professionals.

The university could not fill the “whole” in the education in Informatics of secondary schools created during of years neglecting this education in the school. To tear down significantly the level of the university programs means to turn the universities in professional schools. No one university will do such suicide. Because the constant increasing of necessity of programmers, the solution is to mobilize the efforts of the community for reforming the education in programming and related topics in the secondary school. And the universities could make some compromise with the level of education till the moment when the gap between the school and university is minimized enough. Another way to fill this gap is creating an intermediate level of education, which to prepare students for the university, or to work in a software business as developers of low qualification (the business need such developers).

Programming contests for secondary school students

At the end, I would like to share some things that happen in the programming contests for secondary school students – mainly in IOI but in all related competitions where contestants are prepared for IOI – National Olympiads and Regional Contests too [2]. Because these contests are the place where the young programmers fund the base of their future career in programming, they could help the contestants to elaborate skills and habits that could be very helpful in their future work – to plan carefully the implementation, to analyze in details the necessary algorithm and the most efficient structuring of the data, to structure and type the code in such way to be easy readable, to debug efficiently the code and so on. But these contests could also create habits that are not so helpful and even harmful. Some of these negative habits that the nowadays contest in IOI style nor simply encourage but make them inevitable I would like to comment here.

The programming contest task usually states some algorithmic problem. The statement of the task describes in details what kind of data will the program receive on the input, how this data will be formatted, what kind of results it has to produce and how the results have to be formatted on the output. In order to solve the task the contestant has to write corresponding piece of programming code and to submit it to the grading system of the contest. The grading system performs the task – append some piece of code from the author, compile the obtained program and execute it against a set of test cases. Efficiency of the used algorithm is estimated by corresponding time limit for each execution. The contestant receives grading marks depending of the quantity and quality of the test cases that the program solved.

The first trend that I would like to comment is so called *feedback*. When the contestant submit a solution then the grading system could evaluate the solution without telling to the contestant the result of the evaluation – such grading is called *without feedback.* But it is possible that grading system reports to the contestant the result from evaluation on a part of the test cases – *partial feedback* or on all test cases – *full feedback.* Giving partial or full feedback was introduced in IOI and related contests less than 10 years ago because the tasks became very difficult and the assumption is that with some feedback contestants will have more chances to solve such difficult tasks.

We could accept such an assumption and consider it reasonable. But, unfortunately, some colleagues decided that it is necessary to grade each task with full feedback. I would not like to discuss the fact that, having full feedback, some contestants started to make attempts to guess answers of the test cases instead to write a program that solve them. And responsible committees had to search how to prevent such possibilities. Much more dangerous is the behavior of almost all contestants to stop debugging their programs. Having full feedback the contestant just submits the code without debugging it. If the submit is successful the contestant moves to next task. But if the full feedback says that the solution is wrong then the contestant makes some changes and without debugging the code submits it again. And so on till the limited number of submissions.

As it was mentioned above debugging of the code is one of the main activities of the developers – debugging needs much more time than writing the code. And when one is developing commercial software there is no feedback at all. Giving full feedback in the competition of programming denude contestant of the possibility to train one of the most important professional activity in their future career and could be very big damage. Reasonable compromise in contest round with 3 tasks is simple – one task with full feedback, one task with partial feedback, and one task without feedback at all.

The second trend that I would like to discuss is connected with the form of supplying the input data of the competition tasks. Because of the constant increasing of the speed of modern processors authors of the tasks started to increase the size of the input (and/or output) data in order to be able to estimate more precise the speed of the used algorithm. As a result the time for execution of the solutions and giving the promised full feedback starts to increase to and this overloaded the grading system. The proposed solution of the problem was that instead of submitting complete program solution contestant had to submit, **for all tasks**, some functions to be called by the main function of the author. The necessary data respectively structured, are provided through a pointer argument and the results are passed back to the main function through some pointer too.

I would not comment the mass of technical difficulties that such approach creates. A negative trend in the preparation of young programmers in this case is that the contestant stopped to parse input data. We could not deny that interface of nowadays commercial software is organized through the screen forms where parsing the input data is almost not necessary. But ability of parsing text input is important for the qualified developers because of existence of applications in which parsing text input is inevitable and will be such forever. The compilers are very good examples for such applications.

The normal solution is obvious in this case – for tasks with extremely large input/output will be natural the statement of the task to demand creation of sub-function. But for task in which input is not so large it is normal to demand that the program is able to parse the input.

The last negative trend that we would like to stress is splitting of competitive tasks in *subtasks*. The idea is that not so experienced contestant to be able to solve some of the easy subtasks that correspond to their level of preparation. We still not deny this idea in principle. Before the era of subtasks the authors gave such possibilities to contestants making the test case in such way that that solution of complexity $O(n^3)$ to earn 10% of the points, a solution of complexity $O(n^2)$ to earn 20% of the points, a solution of complexity $O(n\log n)$ to earn 30% of the points and a solution of complexity $O(n)$ to earn 100% of the points. In that time the contestant was writing a single program trying to implement the best algorithm they could invent and if there is time then to search other, more efficient.

What happen now? For some tasks 5, 6 or more subtasks are formulated, some of them so different that it is impossible to solve one of subtasks ameliorating the algorithm that solved another subtask. Something more, to receive a grade for some subtasks it is necessary to solve all test cases for this task. It is quite possible that the contestant has a single unsolved test case in each of the tasks and instead of usual 80% of the points to obtain zero. And more, sometime authors create complex relations among the tasks – even if the contestant solved all test cases for subtask *D* she/he could not obtain a mark if was not solved subtask *C*, but grades for correct solving subtask *C* are assigned if and only if the subtask *B* or subtasks *A* is solved. In such a way the normal solving a competitive task by searching better universal algorithm is replaced by considering a set of not coherent algorithms and difficult navigating through the relations of the subtasks.

We consider such trend very negative because it is not teaching young programmers to some abilities that will be necessary in their professional career. In this format of the tasks sometime tasks appear which have no natural splitting of subtask. Even interesting such task is rejected or is split formally in some subtasks just for the principle that task has to be in subtasks. In this format IOI tasks are, the same are the tasks of ICPC and related contests. But for comparison, one ICPC-style contest tasks set is usually composed of 12 tasks that has to be solved by a team of 3 university students for 5 hours. In IOI-style contests a single secondary school student is put in front of 15 or more subtasks (sometime with inner dependencies among them) that she/he has to solve for the same 5 hours.

The full feedback leads to very negative behavior of some contestants. Without attempting to make a perfect solution of some of the tasks, which has to be the main goal of programming contest in educational prospective, they submit 6 routine solutions of the contest tasks and won bronze medals. In IOI’2019 a bronze medal was earned with 41,7% of the points, in IOI’2018 – with 31,17% and in IOI’2017 – with 22,86%. It is clear that the smartest children in our domain will learn how to survive in the complex competition conditions. But the question is “Why they need to do it?”

One of the most important features of the nowadays programming contests is the using of grading systems. Contemporary grading systems are high quality instrument for automatic testing of programs. They are not created to be used in education in programming and have not some functions that are important for teaching – to show some of the tests cases, to show answer of some of the test case, to provide teaching material concerned solved task etc. Extending a classic grading system with such functionality will make it priceless assistant of the teachers, making the educational process much more intensive and efficient [3].

In addition to programming contest in “Olympic” style, we have to mention another kind of contest – in them the contestants present software projects made “in home”. We consider these contests also very important, because not each student has the psychological quality of a contestant – high mathematical culture, deep knowledge of Algorithmics and libraries of predefined abstract data types, ability to work under time pressure and so on. There are students who are very good in working on a large project that needs months for implementing. These students are very big reserve of future programmers and their efforts have to be encouraged, too.

But let us see what happen in this very large domain of education of young programmers. Instead of directing their efforts to the important topics and instruments of Informatics and creating applications, many tutors push their talented pupils to create presentations, visualizations, games interfaces, static web sites or web sites with trivial access to simple data base and so on. Something more, the tutors and members of the Juries of such competitions of projects pretend that the results of these students are of high quality and importance and deviate in such way some of the students from the development of the serious professional skills in programming.

Conclusions

Education of young programmers is one of the most important tasks of the global education process of the planet, because the necessity of programmers will increase in the future. In contrast with this the conservative secondary education system still does not find an adequate place for teaching Informatics in secondary school almost all over the world. Instead the authorities try to replace teaching of Informatics with teaching of Information technology where many things are included in the curriculum that student know from their everyday use of mobile devices.

We deeply believe that secondary schools have to find enough room in the curriculum for an integrated discipline *Informatics and Information technologies* (in a similar way as integrated disciplines *Philosophy and Society, Biology and Human health, Chemistry and Human environment*, and so on). I&IT has to be mandatory and to be included in the curricula of all grades – from the first to last. In the beginning the stress will be, of course, on usage of computers and computerized devices, and on the technology. But from one moment (5th grade for example) computer programming has to be introduced in the program in such way that, in the last 3-4 years in the secondary school, classes in I&IT to be dedicate to Informatics almost. Only in such way the students that continue their education in the universities will be prepared for the ambitions of the Departments of Informatics to produce well educated programmers.

When this idea becomes realistic, then the Olympiads in Informatics, as well as the Project contests, will obtain a new vision. The number of participants, especially in middle grades (4th-7th) will increase, the results of the countries that now participate on the principle "participation is more important than the victory" will ameliorate. This will be in accordance with needs of the society of more and more qualified programmers.

References

1. Manev Kr. Mathematics and Discrete Mathematics. Proceedings of Seventh International Conference "Discrete Mathematics and Applications". South-West University, Blagoevgrad, June 2004. p. 131-138. ISBN 954-680-363-4.
2. Manev Kr., Kelevedjiev E., Kapralov St. Programming Contests for School Students in Bulgaria. In: International Journal "Olympiads in Informatics". v.1, Vilnius, 2007. p. 112-123. ISSN 1822-7732.
3. Manev Kr. Grading Systems in Teaching Programming. Proceedings. of International Scientific Conference "Informatics in the Scientific Knowledge". Varna, June, 2014. p 230-239. ISSN 1313-4345.
4. Manev Kr., Maneva N. On a Metodology for Creating School Curricula in Computing. Olympiads in Informatics, 11, 2017. p. 93-107. ISSN:1822-7732.
5. Manev Kr., Yovcheva B. First European Junior Olympiad in Informatics, Olympiads in Informatics, 11, 2017. p. 171-173. ISSN:1822-7732.