ISSN 1822-7732

INTERNATIONAL OLYMPIAD IN INFORMATICS

INSTITUTE OF MATHEMATICS AND INFORMATICS

INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING

OLYMPIADS IN INFORMATICS
Country Experiences and Developments

Volume 1 2007

Selected papers of
the International Conference joint with
the XIX International Olympiad in Informatics
Zagreb, Croatia, August 15–22, 2007

OLYMPIADS IN INFORMATICS

Country Experiences and Developments

ISSN 1822-7732

`http://www.vtex.lt/olympiads_in_informatics`

Olympiads in Informatics, Volume 1, 2007.

2007.07.25. 11 leidyb. apsk. l.
Tiražas 170 egz. Užsakymas Nr. 1667.

Printed by Printing house “Mokslo aidai”, Goštauto 12, 01108 Vilnius, Lithuania
Cover design: Multimedia Center, Arab Academy for Science and Technology, Alexandria, Egypt

Olympiads in Informatics, 2007, Vol. 1, 3–4
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Editorial

The International Olympiad in Informatics (IOI) is one of the most prominent computer science competitions in the world. It was initiated by the United Nations Educational, Scientific and Cultural Organization (UNESCO) and endorsement by the International Federation on Information Processing (IFIP). The IOI is a truly international event, having been held on five continents and drawing delegations from six. It has been held every year since its foundation in 1989: Bulgaria, Belarus, Greece (twice), Germany, Argentina, Sweden, The Netherlands, Hungary, South Africa, Portugal, Turkey, China, Finland, Korea, USA, Poland, Mexico and this year in Croatia. It is thanks to the hard work of these host countries in organizing and funding the olympiad that the IOI continues to flourish.

The competition sets tasks that are of an algorithmic nature, however the contestants have to show basic skills including problem analysis, design (and knowledge) of algorithms and data structures, in addition to the programming and testing their solutions. The winners of the IOI belong, no doubt, to the best young computer scientists of the world.

The IOI regulations define its General Assembly, the body that is made up from members of each of the participating delegations, as “a temporary, short-term committee during IOI”. This has often been characteristic of the informatics olympiads community as a whole. We come together during the year for a variety of regional and world-wide events, before returning to our respective countries to run our national contests in our own individual vacuums. Yes, we communicate during these national and international events, but too frequently these are conversations between small groups, if not individuals, and often such conversations are piecemeal and quickly forgotten.

Many of the issues at the national level differ from country to country. We have different educational systems and the availability and take-up of information technology varies, but even here there are as many similarities as differences. We also face many of the same problems: How do we pick our students? How do we train them? What is suitable material? etc ... How about those outside of our community? There are currently around 200 countries in the world and about 80 participating at the IOI. How many of those other countries have national contests, or want to have national contests, and how can we as a community help them?

As a community, we have a great deal of accumulated experience gained by running our respective contests. The IOI presents an ideal forum for discussing these experiences and associated issues. It brings together this experience and knowledge from across the globe and offers a regular, annual forum. There is also the opportunity for sharing the experiences of our community with the local educators in host countries. During some previous IOIs, attempts have been made to bring delegation leaders and other educators together, e.g. workshops in South Africa (1997) and Finland (2001).

In Mexico (2006), the Chilean delegation leader Alexander Tobanov made a challenge to start organizing half-day conferences during IOIs. During this year's pre-IOI meeting in March, the International Committee agreed that it was time to start holding conferences in order to study our experiences and to develop future plans. The *Olympiads in Informatics* conference, to be held jointly with and supported by the IOI, was approved and the editorial board chosen.

The first *Olympiads in Informatics* conference puts attention on organizing olympiads at the national level. The 17 papers selected for this inaugural conference discuss the running of and issues facing several national olympiads. Some explore recent ideas and changes, and how experiments with them have worked at the national level. Ideas which, in several cases, have been tried simultaneously in other countries.

It is intended that this conference will have main topics each year. We have a lot of questions to be dealt with: tasks developments, automatic testing systems, teaching programming methods, software for training, curriculum, relations between other contests, etc... There are many such issues and we will set out, if not to lay down answers then to lay down the questions and record how the community is approaching them. We hope that that this will be a benefit, not just to the IOI community, but to the wider community of educators in our field.

Thanks are due to everyone who has contributed to this conference. In particular, we would like to thank Ivo Separovic and the Croatian organisation of this year's IOI for giving us the opportunity to host the conference. Without their generous assistance it would not have been possible to hold this event.

Valentina Dagienė
Arturo Cepeda
Richard Forster
Krassimir Manev

Olympiads in Informatics, 2007, Vol. 1, 5–14
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Brazilian Olympiad in Informatics

Ricardo de Oliveira ANIDO, Raphael Marcos MENDERICO
Instutute of Computing, State University of Campinas
Av. Albert Einstein, 1251, Cidade Universitária Zeferino Vaz, Campinas, São Paulo, Brazil
e-mail: {ranido, rmm}@ic.unicamp.br

Abstract. The Brazilian Olympiad in Informatics (OBI, in Portuguese) is a contest promoted by the Brazilian Computing Society (SBC) and its main purpose is to raise the interest of students in such an important science for a student's education as Computing Science, through an activity that involves challenge, ingenuity and a healthy dose of competition. The contest is composed of two different categories, for two levels of contestants: a programming contest for high school students (Programming Category) and a logic contest for elementary students (Logic Category). In this paper we will discuss the main aspects and challenges concerning the OBI organization.

Key words: programming contest, logic contest, grading systems, programming training camp.

1. Introduction

The Brazilian Olympiad in Informatics (OBI, in Portuguese) is a contest promoted by the Brazilian Computing Society (SBC), and is one of the Scientific Olympiads in Brazil – among Mathematics, Physics, Astronomy and others. The Institute of Computing (IC) at the State University of Campinas (UNICAMP) is in charge of OBI's organization and coordination since its first edition in 1999. The contest has as the sole sponsor *Fundação Carlos Chagas,* a non-profit organization that promotes education in Brazil.

The main purpose of OBI is to raise the interest of students in such an important science for a student's education as Computing Science, through an activity that involves challenge, ingenuity and a healthy dose of competition. The OBI is also a way to promote Computing activities at schools, which could help the students in their future career.

The Olympiad is composed of two different categories, for two levels of contestants: a programming contest for high school students (Programming Category) and a logic contest for elementary students (Logic Category). Both categories are further divided into two levels, depending on the age of the contestants. The contest is organized in two rounds. In the Local Round, contestants take the test at their own schools. The top ranked contestants in this first round are invited to take part in the National Round. For both categories, the best classified contestants are invited to a week of activities that takes place at UNICAMP during the winter break. For contestants in the Logic Category, activities are composed of introductory courses in programming; for contestants in the Programming Category, the activity is a classical Training Camp, with exams for selecting the members of the Brazilian team for the International Olympiad in Informatics.

The main challenge faced by the organization is to motivate schools to take part in the competition. In OBI's 9th edition (2007), about 8,000 students registered for the competition while 300,000 students registered for the Brazilian Olympiad in Mathematics. The main reason for this, we think, is that Brazil does not include Informatics, let alone programming, in its official elementary or high school curricula. Most schools do not have an informatics teacher, nor even computers available to the students, which makes it especially difficult to raise the students and schools' interest.

Many other challenges have to be addressed: the number of girls decreases enormously between the two contests (Logic and Programming), some schools in remote regions do not even have the money or resources to make copies of the exams, the Training Camp costs are enormous, mainly due to the price of air tickets (cities may be 3000 km apart, too far to travel by bus), and others. These challenges and the solutions we found are further discussed in other points of the paper.

This paper is divided as follows: Section 2 introduces the categories, levels and rounds adopted at OBI. Section 3 presents our grading strategies for Logic and Programming categories. Section 4 discusses the training camp and the training camp exams which select the contestant members of Brazilian Team for IOI. Section 5 shows the conclusions and future work.

2. Categories, Levels and Rounds

At OBI the contestants are divided by their knowledge level in categories; these are also divided into levels, according to which academic year each student is in at that moment and his age. These are the categories at OBI:

- Logic Category:
  - level 1, for students up to 7th year of elementary school,
  - level 2, for students up to 9th year of elementary school;
- Programming Category:
  - level 1, for students up to 1st year of high school,
  - level 2, for students up to 3rd year of high school, or who have been enrolled in high school until December of the previous year and are no older than 20 years on July the 1st of the contest's year.

The Logic category, with multiple choice questions, evaluates the logical reasoning of the students, trying to identify earlier some contestants which could develop good skills in computer programming. The questions are usually on same difficulty level as used on SATs (School Admission Test) in the USA. There is an example of a task of the Logic contest in Appendix 5.

The Programming category is composed of programming tasks, like the International Olympiad in Informatics. The purpose of Programming Category Level 1 is to identify students with basic programming knowledge and to develop advanced programming concepts like graphs and dynamic programming with them. The Level 2 contest is the competition which selects the four contestant members of Brazilian Team at IOI. The languages used in programming category are the same accepted at IOI: Pascal, C and C++. In Appendix 5 there is an example of a programming task used at OBI.

Each category and level of OBI is divided into two rounds:

Local Round: The objective of this round is to allow the highest number of contestants. To achieve that, the students do the exams in their own schools or, if it is not possible, in another school which accepts invigilating the exam for that student. Each school fills in a registration form and choose a school member (usually a teacher) to be the "Local Representative" of OBI at his school. This person is in charge of printing the tests, setting up the computer environment according to OBI regulations (if necessary) and sending the students' answers to the OBI coordination (at UNICAMP) to be graded. In both contests grading is done automatically, centralized at the coordination.

National Round: The top ranked 10 percent of contestants at each level are invited to participate in the National round, in which exams are done in universities located in the states' capital cities, important regional centres and cities with a large number of invited contestants. The "National Representative" of OBI is chosen by the National Committee of OBI and is also responsible for the tests and the computer environment, like the Local Representatives. A financial aid is provided to universities that host the National Round.

The contestant solutions, in both Categories, in both rounds, are all graded at UNICAMP, where the OBI central office is located. The office has the administrative support of one secretary hired by the Brazilian Computing Society. The secretary is in charge of keeping the website up-to-date and doing the administrative tasks concerning the OBI administration. The OBI office is also responsible for organizing the Local and National Rounds, and for grading the students solutions.

2.1. *Task Creation and Distribution*

There are two committees for task creation, one for each category. Usually the task creation committee members live in different cities, sometimes outside Brazil, separated by thousands of kilometers in some cases, and they have their own jobs and appointments; it is almost impossible to set a meeting with all the members. The management of the tasks and the committees is done electronically, by a customized bulletin board system that is password protected. All communication among committee members is cyphered with a public-key schema, typically DSA and Elgamal provided by GPG. The committee members are former contestants of OBI, ICPC contestants and former contestants and faculty members.

In 2007, about 20 Logic tasks were made, each one composed of 5 or 6 multiple choice questions for the four Logic exams. Each exam contains between 20 and 30 multiple choice questions. There were also about 25 programming tasks created for the four programming exams of OBI and the Qualification Exam (which will be described in Section 4).

Since the cost of distributing paper versions of task sets to schools would be too high, considering the distances in the country and the weight of the packages, tasks sets for both the Logic and the Programming Category are made available for download by the OBI's representatives in PDF format, using a secure OBI sub-system. The representative must print the task set locally and make copies for each contestant.

As the Logic Category exams are tests, a student solution is a paper form with check-boxes filled (answer forms). The answer form for each student is made available in the Internet for download, together with the tasks, in PDF format (the set of forms for one school is one single multiple page file). Separate answer forms, without contestant information, are also available in case of necessity (for example, when a student smears or blotches her/his answer sheet).

3. Grading Systems

Grading of solutions in both categories, in both rounds, is done automatically, centralized, at the OBI central office.

3.1. *Programming Category*

For the programming category, the contestants' solutions are sent to the OBI office through an electronic system, where the Local and National Representatives submit the tasks, after archiving the contestants solutions into one file. The system verifies the consistency of the archive (names and extensions of the individual submissions), providing a report; if inconsistencies are found, the system directs the Representatives to fix the problem and try again. After receiving all tasks, they are electronically graded, using a set of test cases for each task. A test case is composed of one or more instances and a contestant's solution is only accepted on a test case if every instance is considered correct.

The programming grading system is capable of grading programs implemented in a wide range of distinct compilers and configurations, like several versions of `gcc`. This is necessary because the schools have some freedom to set up their computer environment and, as a result, use many operating systems and compilers at the Local Round. During the National Round the computational environment is controlled and there is no need to use several compilers, but the grading system used in that case is the same as used on Local Round.

The main problems concerning the programming grading system are caused by wrong submissions from Representatives, who sometimes send the solutions by email or in an unrecognizable format. This requires a manual intervention to correct these problems and sometimes inserts the tasks directly into the grading system without a submission. The tasks are never corrected outside the system and no alteration is made in any contestants' solution.

3.2. *Logic Category*

The Logic Category exams are sent by mail to the OBI office and their answer sheets are transformed into digital images by a fast scanner and then digitally analysed to obtain the contestants' answer to each multiple choice question. Each set of answers is compared to the key answers of corresponding exam and graded according the number of correct answers.

The representatives print the task sets and answer sheets in any printer available. For the Programming Category this has no implication, but for the Logic Category this makes things more difficult for the automatic grader, because some printers may scale the pdf when printing. The scanner may also slightly rotate the sheet when scanning. For these reasons, the grader first "normalizes" the image, eliminating rotation and scaling, based on four registration marks in each corner of the answer sheet, before analysing the digital image for grading. The scanner used can scan about one page per second, and the grade program, written in python (using PIL and Numarray), also takes about one second to grade each sheet.

Unfortunately, some students do not follow the instructions and sometimes fill the answer sheet in a wrong way, for example not completely filling the checkbox, making it impossible to grade the sheet. In these cases manual intervention is required to correct these problems, if at all possible.

4. Training Camp

This section presents the training camp courses. The top ranked contestant in each category and level are invited to take part in the courses. Between 15 and 20 students are invited in each level, the exact number depends on the level ranking, which could contains some students with the same score. In 2007, 54 students were invited to participate in the Basic Programming Course (Section 4.1), 10 were invited for the Advanced Programming Course (Section 4.2) and 12 took part in the IOI Training Camp (Section 4.3).

In this section is also discussed the structure needed to the training camp courses (Section 4.4) and the costs involved in the camp (Section 4.5).

4.1. *Basic Programming Course*

The Basic Programming Course is given to the top ranked students on Logic Category and its purpose is to show basic programming techniques and problem solving using the computer. Obviously, a computer language is shown to the students, usually C. However, the aim of the course is the problem solving, with notions of algorithms complexity. The course lasts five days, with classes during the morning and laboratory classes in the afternoon. The topics discussed are:

- Day 1: Introduction to computer programming, computer-aided problem solving, programming languages.

- Day 2: C programming language, initial commands, input and output, decision structures.
- Day 3: Loops.
- Day 4: Functions.
- Day 5: Vectors and Matrices.

4.2. *Advanced Programming Course*

The Advanced Programming Course is given to the top ranked students on Programming Category Level 1 and invited students from Level 2 who could participate of OBI for at least one more year. Its aim is to improve the contestants' knowledge about programming, discussing topics like algorithm complexity and dynamic programming. The course focus on practical aspects of the selected topics, not on the more mathematical approach which is taught at colleges. The topics discussed are:

- Day 1: Introduction to algorithms and complexity.
- Day 2: Data Structures and greedy algorithms.
- Day 3: Graph Algorithms.
- Day 4: Dynamic Programming.
- Day 5: Geometric Algorithms.

4.3. *IOI Training Camp*

The IOI Training camps are given to the top ranked students on Programming Category Level 2. Their purpose is to prepare these students for the IOI. In this camp the four best students are chosen to form the Brazilian Team for IOI. There are two training camps:

- The first training camp takes place in June every year and last 6 days. About 10 contestants are invited to attend the course and take the qualification exams. The exam is divided into three short tests on third, forth and fifth days and a final exam on the sixth day. The local and national round exams' score is added to the qualification exam's score to determine the four contestant members of Brazilian Team for IOI. During the course the students learn advanced programming techniques and solve several programming problems from University of Valladolid problem set archive or other programming problem repositories.
- The second training camp takes place in July, together with the Brazilian Computing Society Annual Congress. The four members chosen to form the IOI Team are invited to attend a week of problem-solving classes.

The classes are given by former contestants of OBI and ICPC contests and faculty members. The qualification exam's tasks are prepared by the programming scientific committee.

4.4. *Structure*

The courses take place at Institute of Computing of UNICAMP. Two classrooms and two computer laboratories are used during the course. The computers have two operating systems (Linux and Windows 2000) and several development tools, like Pascal, C and C++ compilers, text editors and manuals on both operating systems. The Institute does not charge the OBI organization for the use of its structure.

Ten assistants are necessary to take care of the contestants during the training camp. The contestants are usually teenagers, from 11 to 19 years old, and it is necessary to look after them constantly during their classes, meals and on their way between the hotel and UNICAMP, especially the younger ones. The assistants are undergraduate and graduate students of Instutute of Computing.

4.5. *Costs*

The Brazilian Olympiad in Informatics is sponsored by *Fundação Carlos Chagas*, a non-profit organization that promotes education in Brazil. Our annual budget is about US$ 75,000.00 and covers all the costs of OBI, including the training camps, OBI office and the participation at IOI.

Our highest costs are the programming course and training camp costs. About half of our budget is spent to bring the contestants to Campinas, where the courses take place. The main reason for that are the high costs of transportation, especially the air tickets. For example, among the 40 students attending the Basic Programming Course in 2006, 20 needed to travel more than 2,000 km to attend the course. Each airplane ticket costs about US$ 250.00, resulting in US$ 500.00 per participant and a total of approximately US$ 12,000.00 in 2006 for all courses and training camps. Another US$ 500,00 was spent with participants who needed ground transportation to attend the course.

Moreover, although there is a student dormitory at UNICAMP, it is not possible to use it to accommodate the contestants during the camp, when forces us to allocate the students in a hotel. This cost is comparable with the transportation cost and they compose the largest part of the training camp budget. The rest is spent on teachers' and assistants' payments, meals and prizes for the students (certificates, medals and t-shirts). The 2006 training camp cost is shown on Table 1.

On 2006 our budget also included about US$ 10,000.00 to maintain the OBI office and about US$ 11,000.00 to take the contestants and leaders to IOI countries, which costs

Table 1

Training Camps' costs

Activity	**Cost (US$)**
Transportation (ground and air tickets)	12,500.00
Hotel	13,000.00
Meals	2,700.00
Medals, certificates and t-shirts	900.00
Payment to teachers and assistants	1,400.00
Total	30,500.00

Table 2

Brazilian IOI Team' costs

Activity	**Cost (US$)**
Transportation (air tickets)	8,000.00
Insurance and registration fee	1,300.00
Total	11,300.00

are detailed on Table 2. All values shown are approximate values and are given only for information.

5. Conclusion and Future Work

Here are some planned future work to be done for promotion and improvement of OBI:

- A booklet written by the members of Logic scientific committee and other professors, with Logic categories questions and key answers. Two thousand books will be printed and distributed by mail to every participant school of OBI 2007 or future participant schools or to every one who asks for a copy.
- A CD developed by the members of Programming scientific committee which contains a small version of the submission and grading system and past programming tasks with solutions and test cases, to be distributed as described above.
- On-line contests during the second semester of each year, to promote the main contests and raise interest in students and schools in logic and programming.

Appendix A. Logic Category Task Example

MP3 Player

Francisco must select three CDs to record on his mp3 player. He owns 6 CDs labeled K, O, S, T, V and W. Francisco must follow the following conditions:

- K must be selected, S must be selected or both must be selected.
- O or V must be selected, but neither V nor S could be selected together with O.

1. Which one is a valid CD set to be recorded on MP3 player?
   (A) K, O and S
   (B) K, S and T
   (C) K, S and V
   (D) O, S and V
   (E) O, T and V
2. If K and O were chosen, which item shows a valid set of Cd's that Francisco could choose without breaking any condition?

(A) S and V
(B) T and W
(C) V and W
(D) S, W and T
(E) V, W and T

3. If S were chosen, which CD must be chosen?
   (A) K
   (B) O
   (C) T
   (D) V
   (E) W
4. If V was not chosen, which CD pair must be chosen?
   (A) K and O
   (B) K and T
   (C) K and W
   (D) O and T
   (E) O and W
5. Which CD pair must not be chosen at the same time ?
   (A) K and O
   (B) K and T
   (C) O and W
   (D) T and W
   (E) V and W

Appendix B. Programming Category Task Example

Maze

A friend of yours is very excited about a new game he downloaded to his mobile phone. The game is a kind of maze which could be represented by a N by M matrix. Each cell of the maze contains a platform which is at a certain height from the floor, which could be represented by an integer from 0 (the lowest) from 9 (the highest). Initially, you are on cell $(1, 1)$ (superior left corner) and your purpose is to reach at the end of the maze at cell (M, N) (inferior right corner).

To leave the maze you should move from a cell to an adjacent one. The problem is: your avatar cannot jump, so, if the destination cell is more than one unit higher than your current height, you can't move.

In each round you can move to one of four adjacent cells (up, down, left, right) *if the height of the destination cell is less or equal your current cell height plus one*. That is, if the height of the cell is a, you cannot move to an adjacent cell if, and only if their height is less or equal than $a + 1$.

To make the things worse, in each round, *after the player's action*, each cell increase its height by one. If the height of a cell is exactly 9, its height becomes 0.

Observe that, in a round, the player does not need to move, he could just wait for the platforms movement. Moreover, notice that not every cell has 4 neighbors, because movements outside the maze are not allowed.

You, as a good programmer, decided to write a program which calculates the lowest number of rounds necessary to reach the exit of a given maze.

Task. Write a program which, given a maze, returns the lowest number of rounds necessary to reach the exit with the restrictions described above.

Input. The input must be read from the standard input device, usually the keyboard. The first line contains two integers N and M $(2 \leqslant N, M \leqslant 50)$ separated by a space, which represents, respectively, the amount of lines and columns on the maze. The N following lines contain, each one, M integers which represents the initial height (at round 0) of the platform. The platform is always between 0 and 9.

Output. Your program must print, on the standard output device a single line, containing the lowest number of rounds to reach the exit.

Source file: `maze.c`, `maze.cpp`, *or* `maze.pas`

Input
```
4 3
0 0 0
0 0 0
0 0 0
0 0 0
```
Output
```
5
```

Input
```
3 3
1 2 3
4 5 6
7 8 9
```
Output
```
12
```

Input
```
3 5
1 3 1 1 1
1 3 1 3 1
1 1 1 3 1
```
Output
```
10
```

R. O. Anido is the coordinator of the Brazilian olympiads in informatics. He has been attending the IOI since 1998, and has served as head judge for the South American ACM ICPC Regional since 2003. He holds a BEng in mechanical engineering from the Aeronautical Technological Institute (ITA), and a MSc in computer science from University of Campinas, both in Brazil. He took his PhD in computing at Imperial College, London, and spent a post-doctoral period at Institut National dés Télecommunications (INT) at Evry, France. He is currently an associate professor at the Institute of Computing, University of Campinas, Brazil, where he was vice-director and director from 1996 to 2004. His main research interests are distributed and mobile computing.

R. M. Menderico participated in OBI as a contestant in 2001 and since then has helped in OBI organization, initially as an assistant and then he has served on the scientific committee of OBI. He also attend the IOI as Brazilian Delegation's deputy leader in 2005 and team leader in 2006. Raphael received his BSc degree in computer engineering in 2005 at UNICAMP and is currently enrolled in PhD course at Institute of Computing at UNICAMP. He is also a part-time lecturer at UNICAMP.

Olympiads in Informatics, 2007, Vol. 1, 15–23
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Regular Competitions in Croatia

Predrag BROĐANAC
V high school
Klaiceva 1, Zagreb
e-mail: predrag.brodjanac@zg.t-com.hr

Abstract. Getting the Olympiad team which will consist of the best informaticians is a really big problem. In the Republic of Croatia, over the years, different competition models have been experimented with, often using the previous, abandoned models or introducing the new ones. Each of these models had its own advantages and disadvantages, but none of them have been the perfect one. The current model also has its own disadvantages, but the biggest drawback is that the competitors are overburdened at the time when they are supposed to do their best. The students are under a lot of pressure, so there have been some thoughts about the model, according to which additional time would be put between the final competitions.

Key words: programming, competitions, competitors, the Croatian Olympiad in Informatics, International Olympiads in Informatics, the Olympiad team.

1. Introduction

In order to become the member of a team which will represent the Republic of Croatia at the International or Central European Olympiad in Informatics, the competitors have to show their own quality in a whole series of competitions in informatics. If they don't achieve the excellent results in only one of these competitions, it is almost certainly that they will not be the part of the informatics team which will represent Croatia at the International Olympiads.

The competitions, where the selection of the informatics team starts, begins by the end of January at the lowest level of competition – the school competition. The next level is a county competition and afterwards the state competition, where the students compete for 2 days. Immediately after the state competition, the Croatian Olympiad in Informatics follows, at which up to 15 students, who achieved the best results in the state competition, are invited to participate. The 8 students who were top-placed at the Croatian Olympiad in Informatics are invited to participate in the Elective preparations. During these preparations students are faced with another two exhausting competitions, and the 4 top-placed students in these competitions will be the members of the team which will represent the Republic of Croatia at the International Olympiads in Informatics.

Such a large number of competitions is extremely stressful for the students, considering the fact that most of these students participate in some other competitions (mathematics, physics, etc.), but the years of experience have shown that such a large number of different competitions is, in fact, a prior condition which will show who the best competitors really are.

Apart from regular competitions, which are a prior condition if you want to participate in the International Olympiads in Informatics, there are some other competitions in informatics, which help the competitors to stay on good form, such as: COCI (Croatian Open Competition in Informatics), TopCoder and things like that.

Great attention has been given to the students' preparations for the competitions. Apart from the regular informatics classes and preparations which students have throughout the school year in extracurricular activities within the school, the Croatian Informatics Clubs Association organizes informatics summer camps and winter schools of informatics every year.

2. The Overview of the Regular Informatics Competitions in the Republic of Croatia

In the Republic of Croatia school education lasts for 12 years. In the first 8 years of their education, students attend elementary school which is compulsory for everybody. When they finish their elementary school, students can choose between 3 basic types of high schools (grammar schools, vocational and technical schools). According to the chosen program, high school lasts for 3 or 4 years.

The informatics competitions follow the structure of the education system. Accordingly, there are 2 basic categories of competitions: elementary and high school competition in informatics. At high school level there are two subgroups of competition: one subgroup is for the first and the second graders (at the ages of 15 and 16) and the second subgroup is for the third and the fourth graders (at the ages of 17 and 18). All the students solve the programming tasks using one of the following programming languages: Pascal, C or C++.

At the elementary-school level there are also 2 subgroups. The first subgroup is for the students up to the 6th grade (up to 12 years old) and the second subgroup is for the students in the 7th and 8th grade (they are 13 and 14). Unlike high school students, elementary school students can compete in two categories: BASIC/Pascal and LOGO.

All these competitions are held at several levels:

- school competition,
- county competition,
- state competition.

For high school students there are two levels of competition:

- Croatian Olympiad in Informatics,
- Elective preparations.

Exceptionally, elementary school students can be invited to the Croatian Olympiad in Informatics if they have achieved exceptional results in the state competition.

Up to now, only two elementary school students have been invited to the Croatian Olympiad in Informatics.

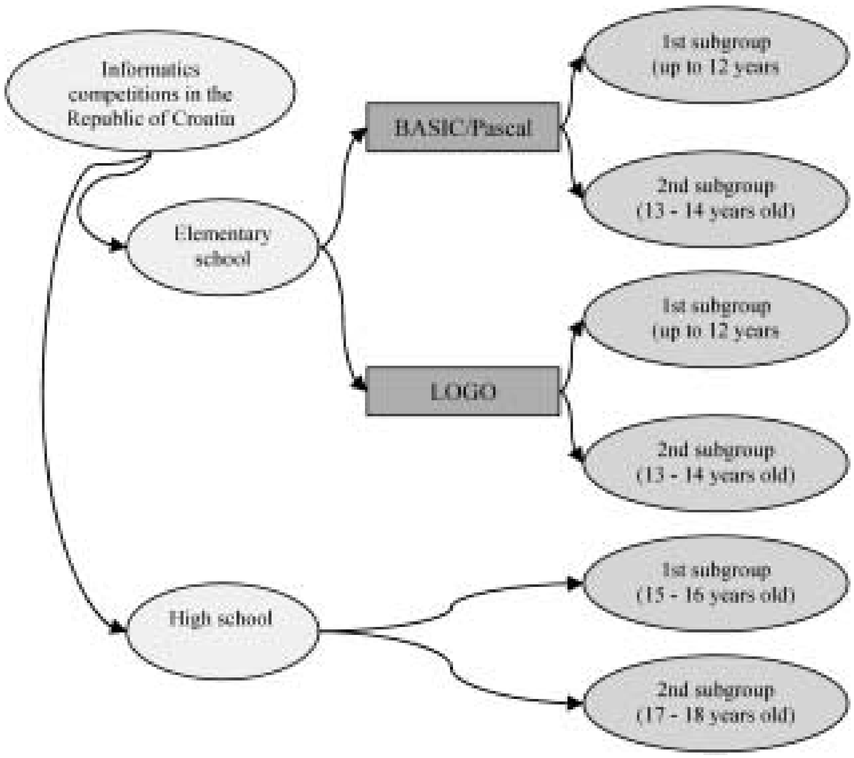

Informatics competitions in the Republic of Croatia
Elementary school
High school
BASIC/Pascal
LOGO
1st subgroup (up to 12 years
2nd subgroup (13 - 14 years old)
1st subgroup (up to 12 years
2nd subgroup (13 - 14 years old)
1st subgroup (15 - 16 years old)
2nd subgroup (17 - 18 years old)

Fig. 1. The overview of the regular informatics competitions in the Republic of Croatia.

2.1. *School Competitions*

Up to this year, schools alone have been in charge of the school competitions. Schools could organize the competitions within some period of time (most often, a week), setting their own tasks.

This is the first year that the informatics competition was organized in a way that the competition took place at all schools at the same time. Tasks for the school competition were not prepared by schools alone, but by the State Examination Board that prepares tasks, defined at the state level. That State Examination Board prepares exams for all levels of competitions.

The school competitions are organized at the end of January, or at the beginning of February. The exact size, according to categories, is unknown, but it is approximately about 800 elementary school competitors and 200 high school competitors (the overall number of students per age group in the Republic of Croatia is about 40000).

At this level, students solve 3 tasks which are relatively simple in order to attract as many competitors as possible. It is extremely important that, at least, one task (usually the first one) is very simple so that everyone can get some points.

2.2. *County Competitions*

The Republic of Croatia is divided into 21 counties. According to that, the next level of competitions, after the school competitions, is the county competitions.

An Examination Board, which organizes the competition and invites students to the competition, is formed for each county. Students who were top-placed in the school competitions are invited to participate in the county competition. Up to this year, county Examination Boards had a big problem with inviting students to the county competition.

As each school had different tasks, it was extremely difficult to establish the relationship between the weights of the competition between two schools. This year those problems were solved as the tasks were the same for all schools.

Table 1 provides the number of students who took part in some subgroups of the competition. It is important to emphasize that some elementary school students competed in the category BASIC/ Pascal and the category LOGO, so they are counted twice.

As for the school competition, the tasks are prepared by special Examination Boards which are created by the State Examination Board for the competition in informatics. It is mostly the same Examination Board which prepares tasks for the school competitions.

2.3. *State Competition*

The state competition in informatics is held during May. According to the rules of the Croatian Informatics Clubs Association, the general number of students who are invited to participate in each category is shown in Table 2.

High school students compete for 2 days. Every day 3 problems are solved and the total score is the sum of points from both days of competition.

On the first day high school students solve 3 problems within 3 hours while on the other day 3 problems are solved within 4 hours.

Elementary school students compete for only 1 day. In the LOGO category 4 problems are solved within 2 hours and in the BASIC/Pascal category 3 problems are solved. In order to enable elementary school students to participate in both categories of the competition, the LOGO category and BASIC/Pascal category competitions are held in different days.

Table 1
The number of the participants in each category in the county competition in 2006/07

School level	Category	Subgroup	Number of competitors
Elementary school	LOGO	1st subgroup	199
Elementary school	LOGO	2nd subgroup	179
Elementary school	BASIC/Pascal	1st subgroup	122
Elementary school	BASIC/Pascal	2nd subgroup	183
High school	Pascal/C/C++	1st subgroup	106
High school	Pascal/C/C++	2nd subgroup	137

Table 2

The general number of students who are invited and the number of students who paricipated in this year state competition

School level	Category	Subgroup	General number of students who are invited	Number of the invited students 2006/07
Elementary school	LOGO	1st subgroup	15	16
Elementary school	LOGO	2nd subgroup	15	17
Elementary school	BASIC/Pascal	1st subgroup	10	12
Elementary school	BASIC/Pascal	2nd subgroup	20	22
High school	Pascal/C/C++	1st subgroup	20	20
High school	Pascal/C/C++	2nd subgroup	25	27

2.4. *Croatian Olympiad in Informatics*

Generally, high school students are invited to the Croatian Olympiad in Informatics. Exceptionally, an elementary school student who achieved exceptional results at all levels of regular and special competitions can be invited.

When the second day of the competition is finished, the State Examination Board, which is in charge of the competition in informatics, decides on the students who will be invited to the Croatian Olympiad in Informatics. Who will be invited is determined by their success in the state competition over those 2 days. According to the rules of the Croatian Informatics Clubs Association, 5 to 7 competitors from the first high school subgroup and 10 to 12 competitors from the second subgroup are invited to participate in the Croatian Olympiad in Informatics.

This year 21 students, 8 from the first high school subgroup, 12 from the second high school subgroup and 1 elementary school student, are invited to the Croatian Olympiad in Informatics.

2.5. *Elective Preparations*

Basically, 8 top-placed students from the Croatian Olympiad in Informatics are invited to take part in the Elective preparations which, essentially, last for a week. Over a week, students have intensive preparations and they have competitions for 2 days. In each of these 2 competitions students can gain up to 200 points and the points gained in these 2 days are summed up. So, the maximum number of points gained in the Elective preparations is 400. The final rank-list is done after the Elective preparations, according to the overall number of points from the Elective preparations and the number of points at the Croatian Olympiad in Informatics (Table 3). 4 top-placed students from that rank-list are invited to participate in the international olympiads.

Table 3

Final rank-list

Overall rank-list for the Olympiad team selection which will represent the Republic of Croatia at international Olympiads		
Croatian Olympiad in Informatics (300 points maximum)	First competition within the elective preparations (200 points max.)	Second competition within the elective preparations (200 points max.)

3. Informatics in the Framework of the Croatian Education System

In spite of numerous efforts, informatics (computer science) has not become an obligatory subject in elementary schools. Rather, it is taught as an extracurricular activity and the number of students eligible to register for the classes is limited. Elementary school informatics is mainly based on acquiring fundamental informatics skills, where programming skills account for only 11% of the entire informatics curriculum. Since Croatian schools are rather poorly equipped and the profile of elementary school teachers is fairly low, and since there is an evident lack of adequate programs, the situation looks pretty chaotic at this point. The teachers often teach only what they like, and that, in most cases, does not involve programming. It happens quite often that elementary students who have studied informatics for four years do not know anything about programming. The situation is not much different in high schools either. Informatics is mostly taught for one year, except in certain technical schools and schools of science and mathematics where informatics is taught for more than one year. In the schools where informatics is taught for only one year, it is mainly reduced to learning how to use office tools such as MS Office, whereas in technical schools where informatics is taught for more than one year, it is mostly taught in connection with certain professional subjects and based on the use of hardware with just a touch of low-level programming. Only the students of the 2nd, 3rd and 4th grades of the schools of science and mathematics learn the skills needed for more comprehensive programming. In the course of three years, they are taught the basics of structural programming, using the Pascal or C language. The skills that the student acquires through high school education are often not enough for achieving success at even the lowest level of competition.

3.1. *Additional Work with Students within the School*

The more ambitious elementary and high school teachers give additional classes, outside the regular informatics classes, to students showing additional interest in programming. During such additional classes, the teachers work with students primarily on solving programming tasks and prepare them for competitions. The main problem of this kind of additional education is that it is primarily voluntary in nature. In most cases the teachers do not get paid for the extra hours they put in. After the students have reached a certain level of knowledge, the teachers cannot keep track any more. The additional classes serve to maintain the continuity of committed work, the teachers try to motivate the students and find problems that the students solve on their own. This type of education, in most cases, is organized in the evening when school time is over or on Saturday. In some schools this type of education is provided by ex-students, who achieved good results in competitions, along with teachers. This type of education is extremely important, which enables students to rise to the top and come to the higher level of competitions in Croatia. My personal experience says that only 5% of pupils who did not take part in extracurricular activities come to the state competition.

3.2. *Extracurricular Work with Students*

The students acquire the largest amount of knowledge and skills needed for competitions participating in extracurricular activities. The Croatian Informatics Clubs Association has been organizing informatics summer camps and winter schools of informatics for 13 and 11 consecutive years respectively. The summer camps and winter schools are mainly organized for students who achieve the best results at the state competition. The students in community where the summer camp or the winter school is organized can also register to attend the classes. In summer camps and winter schools of this kind, education activities are conducted in the form of well-designed workshops led by excellent teachers, mostly former competitors. The workshops differ in character and mainly teach programming skills. The students can take part in the following workshops:

- Algorithm Workshop – the students deal with various algorithms depending on their age. Elementary school students learn about sorting algorithms, recursions etc. High school students learn about dynamic programming and algorithms for graphs. Each algorithm workshop is complemented by a series of problems that the students work on with the help of their instructors. The teachers prepare the problems from the previous competitions.
- Programming Language Workshop (Pascal, BASIC, C++, Logo etc.) – the students acquire additional knowledge or are introduced to a new programming language. This workshop is intended primarily for elementary school students. They are introduced to the more advanced applications of programming languages and data structures (designing subprograms, working with arrays, matrixes, databases, strings, etc.). C++ STL is extremely popular among high school students. As in the case of algorithm workshops, programming language workshops are also designed to ensure the students an opportunity to use the acquired knowledge providing a number of problems related to the taught subject for them to solve.
- Undefined-topic Workshops (web design, C#, Java) – the purpose of these workshops is to give the students a kind of insight into what is currently going on in the world of programming. Such workshops are a kind of relaxing escape from algorithms and psychically significantly less demanding.

The fact that such workshops are mostly conducted by former competitors is very important because the students benefit from their rich, first-hand experience in competitions. Another very important fact is that the competitors and instructors/former competitors are relatively close in age which helps break the classic barrier between students and teachers and develop a friendly relationship that allows the students to ask questions with significantly less hesitation, talk about the problems even in their free time, etc. To fill their free time the students have the opportunity to participate in computer game contests, student excursions during which they hang out and talk, mainly about programming problems, algorithms etc., which is very important in terms of developing informal learning habits. The students who achieve the best results during the workshop are awarded at the closing ceremony.

After that type of school is over, students and teachers stay in contact by e-mail or in some other way. Teachers help students to solve programming problems, giving suggestions etc. That type of schools in particular, and relationship between teachers and students are one of the most important facts that has led to Croatia being awarded 64 medals in international competitions.

3.2.1. *Informatics Summer Camps*

Informatics summer camps are held during the summer months, usually in July or August, near the coast. They are extremely attractive to students because they include studying but also a fair amount of entertainment and free time to hang out with fellow students. The main attraction is the time that students spend at the coast. In informatics summer camps, special attention is paid to those students who intend to participate at the international informatics olympiads. They attend special all-day workshops conducted by the best former competitors during which they work on the problems similar to those that might appear at the olympiad and attend a series of lectures in various fields, very often associated with mathematics.

3.2.2. *Winter Schools of Informatics*

Winter schools of informatics are organized in winter, during the winter break, mainly in the continental part of the country. They do not differ much from the informatics summer camps, excepts that winter schools are easier than summer camps. Competitions, especially international, are far away so the workshops for students who intend to participate at the olympiad do not exist. For the potentially candidates for the international informatics olympiads workshops are organized with very complicated algorithms and data structures. This workshop is of an open type and any participant of winter school can join it.

References

Primary School Croatian National Educational Standard. Zagreb, Ministry of Science, Education and Sports, 2005.

Gymnasium Curriculums. Zagreb, Croatia Institute for Education, 2005.

Technical School Curriculums. Zagreb, Croatia Institute for Education, 2005.

Messenger HSIN_@ vol. 9, Croatian Informatics Society, Zagreb, 2005.

Program of rules for competitions and reviews of software work primary and secondary school Croatian students in informatics for 2006./07. school year. Croatian Informatics Society, Zagreb, 2005.

Messenger HSIN_@ – special edition. Croatian Informatics Society, Zagreb, 2005.

New International Program of HSIN in 2006. – Croatian Open Competition in Informatics (project request), Croatian Informatics Society, Zagreb, 2006.

`http://www.hsin.hr`.

P. Brođanac in 2000 graduated from the Faculty of Science, Department of Mathematics and therefore obtained a bachelor’s degree of engineering in mathematics. He has been working as a teacher of informatics at V high school in Zagreb since 2001 where he has achieved outstanding results in the national and international competitions. Since 2001 he has been a member of the Examination Board which is in charge of competitions in informatics.

He is the author of a series of books and reference books in informatics and technical articles in mathematical and informatics magazines. He is the author of several educational software for informatics and a lecturer on many occasions for teachers and professors of informatics in the Republic of Croatia.

He is a member of a professional working group which is in charge of introducing a state exam (school-leaving examination) in the Republic of Croatia.

Since 2002 he has been working as a subcontractor in the pharmaceutical company Barr (ex Pliva) where he deals with the projects connected to biochemical software.

Olympiads in Informatics, 2007, Vol. 1, 24–30
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Italian Olympiads in Informatics

Giorgio CASADEI
Computer Science Department, University of Bologna "Alma Mater"
e-mail: casadei@cs.unibo.it

Bruno FADINI
Computer Science and System Department, University of Naples "Federico II"
e-mail: fadini@cds.unina.it

Marta Genoviè De VITA
Italian Association for Informatics (AICA)
e-mail: mpi.genovie@flashnet.it

Abstract. We describe our 6-years long experience in training and selection of the Italian team for the IOI. Based on this experience, we outline our proposals and how we intend to proceed to improve the effectiveness of these processes.

Key words: training, problem solving, programming competition.

1. The Beginning

During the year 2000, the Italian Ministry of Education and the Italian Association for Informatics (AICA) came to an agreement to organize the Italian participation to the IOI. To this end, they formalized a National Committee for the Olympiad of Informatics (COI) whose primary objective is select and train the Italian Team for the IOI. AICA was already engaged with the Ministry of Education in promoting learning of Informatics in Italian Schools and this new cooperation was well inserted in the old one.

From the year 2002 COI organizes the Italian Olympiad in Informatics, as the final event for the selection of the National Team for the IOI.

In the 7 years of participation in the IOI, the Italian Team score is: 1 gold, 4 silver and 10 bronze medals.

2. The Organization Scheme

The Italian Organization is based on the National Committee for policy-making and decision process and on a Technical Group for treatment, development and administration of all specific activities relating selection and training processes.

2.1. *The National Committee for the Olympiad in Informatics (COI)*

The COI promote, coordinate and manage all decisions and actions relative to selection and training of students and make the choices to form the national team. The COI is composed by nine members:

– one delegate from the Ministry of Education,
– one delegate from AICA (Italian Computer Science Association),
– three experts from higher schools,
– four experts from University and Research.

These members elect a chairperson who lasts in the job for 3 yeas.

To carry out his program, COI has organized his work in two groups: a scientific group and an administrative group.

The scientific group is composed by the chairperson of the COI, two members from University and one expert not member of the COI. Purposes of this group are:

– to approve and validate texts for selection steps;
– to organize training activities;
– to suggest the composition of Italian Team for IOI;
– to prepare the budget for these activities.

The administrative group is composed by three members of the COI (one is headmaster of high school and manager of the full budget). Purposes of this group are:

– to manage contacts with schools;
– to organize logistics for selections steps and for the National Olympiad;
– to prepare the budget for these activities.

On the basis of sponsorship and budget proposals, the COI approve the overall budget for the next year. Members of the COI cannot receive any fee (only refunds for traveling expenses are allowed). The budget for the year 2006 is shown in the last section.

2.2. *Technical Group*

To organize and manage technical activities related to the selection process and to administer and perform training courses, the COI employs a technical staff (computer scientists from research institution, University and high schools) coordinated by the expert of the scientific group not member of the COI.

Main tasks of this group are:

– to provide problems (Tasks) for selection steps;
– to administer and evaluate selection papers and programs (with the support of computer tools);
– to teach and train the winners (gold and silver medalists) of the National Olympiad;
– to support the COI in the final selection procedure to build up the national team.

3. The Annual Process

Starting from the second participation to the IOI, Italian activities have been organized in the following phases (Table 1).

3.1. *School Enrolment*

By the middle of September, Minister of Education invites high schools to participate to a national selection (enrolment fee is 50 euro per school).

3.2. *First selection (involving about 500 schools)*

About 12000 students are involved in this selection. This tests take place by the middle of November; it consists of 15 logical problems and 15 programs: for each item, students must select the right answer out of 4 given choices. The National Committee select the text and a local teacher administers it. The best students of each school (from 1 to a maximum of 5) are invited to a Region Selection.

3.3. *Second Selection (in 20 Regions)*

About 1200 students are involved in this selection. The test takes place by the end of January in 20 different sites and is somewhat similar to that of the IOI (3 problems, not very difficult, to be solved in 5 hours). The National Committee prepare and manage the test. An ad hoc system has been designed and implemented in order to send the problems and to collect the answers of each student, via Internet. The best 75 students of this selection (at least 1 for each region, for promotion purpose) are invited to the National Selection.

3.4. *Third Selection (Italian Olympiads of Informatics)*

By the middle of March, in one site (different every year) we organize the Italian Olympiads of Informatics inviting the 75 best students of the Region Selections and the young winners of the Italian Olympiad of the previous year. This selection is quite similar to that of the IOI; at the end the best students receive medals (5 gold, 10 silver and 20 bronze).

Table 1

The annual process

October	school enrolment;
November	first selection in the enrolled schools (about 500 schools and 12–15000 students);
January	second selection in 20 region points (1200–1500 students);
March	third selection (National Olympiads: 80 students);
April–May	training stage and team formation.

3.5. *Training and Team Formation*

The gold and silver medal winners are invited to follow two (sometime three) stages (5-6 days each, between the end of April and the beginning of June) at the University of Pisa. At the end of the second stage, a final selection is organized to select the best 4 students to form the Italian team for the next IOI.

During the current year, two on-line training periods were successfully experienced before and between formal training. The objective of these on-line activities was to give basic theoretical knowledge and to stimulate student capabilities.

4. Comments and Development Strategies

In Italy, Informatics is quite absent in high school regular curricula. It is necessary to promote initiative to stimulate interest in this discipline (among students and teachers alike).

The actual organization of selection and training activities is not satisfactory; it does not assign enough time to training activities (remember that most students have no training in regular courses). We are studying to arrange all the activities of selection and training as lasting two years, so that we can involve a greater number of students and lengthen the time of specific training between Regional Selection and team formation.

Taking into account that Italian high school last for 5 years (with students 15–19 years old), the COI is inclined to act in the following way:

- to accept enrolments of all students in high school (this is active from 2006, before only students of the last three years were accepted);
- to teach formal courses (at least a basic course and an advanced one) and then to develop selection and training activities during two years;
- to make earlier Regional and National Selections to allow a longer time for training;
- and finally, to promote a closer work between teachers of school and university staff involved in this project.

The project is an ambitious one and must be checked from various points of view:

- Is the cooperation between different entities (Ministry of Education, Universities and School teachers) possible?
- Are the students motivated to make a two-year training for a non-standard curriculum?
- Is there a budget to cover it?

However, we prefer to stimulate local experience and to continue the presentation of the project in order to promote the discussion on these ideas in the IOI environment. In particular, we present two courses, the basic and the advanced one, that belong to the project.

4.1. *Basic Course*

This course is intended for students of the first two (three) classes (15–17 years old) with the aim of introducing general concept of computer programming and simple recursive scheme. The attendance to this course is free; any school can be involved with the payment of a symbolic fee (e.g. 50 euro).

The objective is to involve (many) hundreds of schools; that is possible only with:

– the cooperation of at least a teacher for each school involved in the project;
– the availability of e-learning system to support distance learning.

Every year, at the and of the course, the best 30–50 students could be invited (and granted) to attend a free summer course to facilitate the attendance of the advanced course of the next year.

4.2. *Advanced Course*

This course is intended for students of the last two (three) classes (17–19 years old) with the aim of introducing "*algorithmics* (*the spirit of computing*)" and practising IOI-like problems.

Each year, at the beginning of October, a selection could be organized; the best students passing the admission test can follow the advanced course.

During the year two other selection can be organized and the best 40 could be invited to the National Olympiad of Informatics.

The gold and silver medallist will follow training and team formation activities as described in the previous point 2.5.

5. Conclusions

Following the suggestions described in the previous section we hope to obtain these results.

– To increase the involvements of schools in programming curricula.
– To stimulate a greater number of students toward the study of algorithms and computer science.
– To improve competence of students in problem solving.

A great and consistent help could be given by activities promoted by the IOI International Committee to facilitate cooperation and exchange of experiences among countries as for examples:

– **To approve the syllabus**,
– **To modify the structure of the competition** (not only make the problems harder and harder),
– **To offer free tools** to support distance teaching and learning and to administer and evaluate tests.

6. Budget for Year 2006 (Euro)

Income

AICA support	89.000
Fee paid by schools to join Olympic selection	24.900
Ministry support	60.000
TOTAL	**173.900**

Expenses

		AICA	MPI	TOTAL
COI and Groups Meeting		9.000,00	4.000,00	13.000,00
Contribution to schools for regional selection		30.000,00	0	30.000,00
Starting annual activities	Advertising, press, posters, bill	2.500,00	2.500,00	5.000,00
National Olympiad	Refunds for traveling expenses: COI members, teachers and students	2.500,00	11.000,00	13.500,00
	Stationery, gadget, medals and plates	3.000,00	0	3.000,00
	Contribution to Institute managing the budget	0	3.400,00	3.400,00
International Olympiad	Enrollment, traveling expenses,	5.000,00	6.000,00	11.000,00
Secretarial staff		0	2.500,00	2.500,00
Training in Pisa	Refunds for overnight stay and traveling expenses for teachers and students studenti e docenti	2.600,00	9.300,00	11.900,00
Manager, teachers and tutors for activities in Pisa		22.000,00	0	22.000,00
Technical assistance for networking and evaluations		14.000,00	0	14.000,00
Rewards		11.000,00	0	11.000,00
Assurance for students and teachers		0	500,00	500,00
Reserve fund		7.000,00	500,00	7.500,00
TOTAL		**108.600,00**	**39.700,00**	**148.300,00**

References

Andronico, A., A. Carbonaro, G. Casadei, L. Colazzo, A. Molinari and M. Ronchetti (2003). Models and services for mobile learning systems. In *International Conference on Technology Enhanced Learning TEL'03*, Milano, Italy.

Carbonaro, A., G. Casadei, and S. Riccucci (2004). An adaptive assessment system to evaluate student ability level. In *IFIP World Computer Congress* (*WCC2004*), *International Conference on Artificial Intelligence Application and Innovations*, France, August. Kluwer Academic Publishers.

Mignani, S., S. Cagnone, G. Casadei, A. Carbonaro (2005). An item response theory model for students ability *Evaluation using Computer-Automated Test Results*". In *New Developments in Classification and Data Analysis*. Springer-Verlag, Germany, pp. 325–332.

Riccucci, S., A. Carbonaro, and G. Casadei (2005). An architecture for knowledge management in intelligent tutoring system. In *Proc. of Cognition and Exploratory Learning in Digital Age* (*CELDA 2005*), IEEE Technical Committee on Learning Technology and Japanese Society of Information and Systems in Education, Porto, Portugal, December, pp. 473–476.

Riccucci, S., A. Carbonaro, and G. Casadei (2006). A framework for knowledge acquisition in intelligent tutoring systems. In *Proc. of Informatics in Secondary Schools Evolution and Perspective* (*ISSEP 2006*), Vilnius, Lithuania, November.

G. Casadei (1936), university degree in physics in 1959, researcher in numerical analysis and computer programming from 1959 to 1976 and full professor in computer science since 1976. Teacher of a basic course in artificial intelligence for a curriculum in computer science and of elements of informatics for a curriculum in science of education. Main research interest is the role of computer programming in educational processes and history of computing. Member of the Italian Committee for National Olympiad in Informatics since 2001.

B. Fadini (1937), university degree in engineering in 1961, is full professor of computer architectures in the Department of Computer Engineering and Systems of the University of Napoli "Federico II" since 1973. Past-president of CINI (Italian Universitary Consortium for Informatics) and AICA (Italian Association for Informatics), is president of the Italian Committee for National Olympiad in Informatics since 2000. He has been principal investigator for several national and international research projects. Main research interests are computer architecture, software engineering and e-learning. He has co-authored more than 100 publications and 10 books.

M. G. Vita (1934), university degree in economy and commerce, is inspector of the Ministry of Education and adviser AICA. She is member of the Italian Committee for National Olympiad in Informatics since 2000.

Olympiads in Informatics, 2007, Vol. 1, 31–36
© 2007 *Institute of Mathematics and Informatics, Vilnius*

The Informatics Olympiad in Mongolia

Lhaichin CHOIJOOVANCHIG
School of Computer and Information Technology, Mongolia State University of Education
e-mail: choijoovanchig@msue.edu.mn

Sambuu UYANGA
School of Mathematics and Computer Science, National University of Mongolia
e-mail: uyanga@magicnet.mn

Mendee DASHNYAM
Institute of Finance and Economics
e-mail: dashnyam@ife.edu.mn

Abstract. The Informatics Olympiad plays key role in introducing Information and Communication Technology (ICT) to Mongolian secondary schools. It is one of the biggest ICT related competition among Mongolian secondary school teachers and students. The goals of the Informatics Olympiad are to stimulate interest in informatics and information technology, and to bring together exceptionally talented teachers and students from all over Mongolia. Mongolian Informatics Association organizes annual national informatics Olympiads in cooperation with the Ministry of Education, Culture and Science (MOECS) and other universities for the 21st year. For past years, a number of activities were implemented to enhance the informatics Olympiads, such as training of informatics subject teachers, development of training manuals and handbooks with tasks and problems for the informatics Olympiads, various activities to support participation of Mongolian teams in the International Olympiad in Informatics. In this paper, we describe national Informatics Olympiads in Mongolia, informatics education, Mongolian participation in the International Olympiad in Informatics (IOI), and related key issues and problems.

Key words: information and communication technology, informatics education, informatics competitions, programming contests, informatics olympiad, Mongolia.

1. Informatics Education

Meanwhile, the Ministry of Education, Culture and Science has used Vision-2010 as a model to implement ICT in the education sector, developing an action plan which was officially approved in 2001. The Vision for ICT in education has four major components, covering following areas:

- training: full utilization of ICT in each educational level's curriculum and contents in order to introduce opportunities provided by ICTs and gain knowledge and skills to use it;

- hardware: supply of hardware allows the conduct of training according to different level of modern ICT development and provides possibilities of free access to information;
- teaching staff: supply of teaching staffs which have the capabilities to develop themselves in terms of their own knowledge and skills in line with rapid development of ICT;
- information ware: creation of possibilities of available and accessible information service by establishing educational information database and network (MOECS ICT Vision, 2001).

The informatics education in secondary schools plays key role to reach to the implementation of the Vision for ICT in education. The informatics as a subject has been included in the secondary school curriculum in Mongolia since 1988. The old curriculum covered basic concepts of informatics, basics of algorithms and programming and it was not fully covered due to lack of hardware facilities such as computers, trainings were mostly concentrated on providing programming and algorithm development skills.

For past years, a number of activities were implemented to enhance the informatics curriculum, such as development of standards, training of informatics teachers, development of training manuals and materials for the informatics subjects in secondary schools. One of the most important steps taken by government to improve informatics training was development of first standard for informatics education during year 2000–2004. Within this standard Informatics subject taught starting from 5th grade from the academic year 2005–2006. This standard has the following advantages (Uyanga, 2005):

- development of the educational standard of informatics by using the content standard of informatics in complete secondary schools;
- focused more on competence based goal than the subjective goal;
- the content standard is based on domains of systematic knowledge of the informatics science;
- assess not only to knowledge and capability, but also the competences accumulated;
- abundance of individuals needs, more than the social needs;
- the standard is tailored to primary, secondary and complete secondary education respectively;
- the content standard has clear focus, that the trainees gain knowledge and skills to use the informatics, computer and information technology effectively and efficiently, and to resolve the issues met in real life situation and the other trainings by using them;
- needs and demands of informatics education and standards are determined based on the needs of individuals and society;
- the standard is supervised that teachers of informatics not only teach the informatics, computer and information technology, but also develop the skills of students to use them effectively and individually;
- the standard instructs that the teachers of informatics should create the environment to implement the standard successfully by supporting other teachers to widely use informatics, computer and information technology in their teaching;

- comprised the correlation between other educational fields;
- the content is well suited to the international standards according to the contents of documents and standards for ICT education by specialized international organizations;
- independent of certain tools and types of information technology.

The new standard does not cover programming, so the students have to study programming by themselves or they can choose to study it as an additional subject after consulting with their teachers.

Due to the implementation of new Informatics Education Standard in academic year 2005–2006, Informatics textbooks for 5–11 grades are being written.

Also institutions which train professional informatics teacher are planning to update their curriculum to reflect the changes. Nationwide re-trainings for informatics teachers are constantly organized since academic year 2003 as to follow the new standards. The "ICT Vision 2010 in Education Sector of Mongolia" has objectives to conduct training and re-training of teaching staff in secondary schools, expansion of professional teachers' training activities considering the increase of professional teaching staff in informatics up to 90% by year 2007. However, secondary schools still lack of professional informatics teachers. In order to increase supply of informatics teachers, government is taking various measures to foster applicants from rural areas such as by means of tuition fee discount; scholarship under local government contract; retraining of teachers etc. These measures still do not solve needs for informatics teacher. Graduates with informatics teacher certification in most cases move to work in non-educational sectors, in government, non government organizations, private enterprises and companies.

2. National Informatics Olympiad

The first National Informatics Olympiad was organized in 1987. Mongolian Informatics Association is responsible for organizing all annual national informatics Olympiads in cooperation with the Ministry of Education, Culture and Science, other universities and ICT companies. There are some public and private organizations who support national Informatics Olympiad. The Olympiad is organized annually in three levels: districts, city/province and national. Students acquiring high points are admitted to the next level. The winners of the national Olympiad participate in the International Olympiad in Informatics. The Olympiad is organized in two categories: students and teachers. The students contest consists of two days computer programming and one day for teachers. The winner students receive invitation to study in IT related local universities. The Government pays tuition fee for first two year for these students. The software application competition among students is organized during Informatics Olympiad. Students usually develop a computer application using Delphi, Visual Basic, Flash and other technologies.

3. Mongolian Team in IOI

The Mongolian Informatics Association sends a team with four members to represent Mongolia at the International Olympiad in Informatics (IOI) each year. The Ministry of Education, Culture and Science allocates annual budget for the team to participate at the IOI. Mostly it only covers half of total expenses. For past years, Mongolian students and teachers have participated in IOI 1989–1991 (in Bulgaria, Belarus and Greece), IOI 1999–2000 (in Turkey and China), in IOI 2002 (Korea) and IOI in 2004–2006 (in Greece, Poland and Mexico). The Mongolian team had participated in nine IOI and received two bronze medals from IOI 2005.

The key problems of Mongolian team to participate in the IOI are followings:

- Weak English language. Due to language barriers students can not fully understand tasks, use online internet sources. It creates problems for students to attend in online Olympiads.
- Financial problems. In most cases the budget that allocated by the Ministry of Education, Culture and Science could not cover all expenses.
- Difficulty with visa. As a developing country, for Mongolians to get a visa to developed countries is a big problem. Due to visa problem Mongolian team did not participate in IOI 2001 (Finland), 2002 (Germany) and 2003 (USA).
- Lack of students skills and knowledge related to modern technologies.
- Lack of programming and algorithm development skills. Current standard for informatics education does not cover programming.

4. Key Issues and Problems

The quality, skills, interests, knowledge, and competence of the participants are increasing from year to the year. Also the teachers who prepare and train the participants are working with their students very hard. The Ministry of Education, Culture and Science and other universities and organizations in Mongolia, support the National Informatics Olympiad. Even if these organizations contribute for the National Informatics Olympiad, there are still some obstacles. ICT development is still not good enough in Mongolia. However Informatics training at rural schools are limited by computer hardware and skilled teachers supply.

1. The teachers' development and skills in rural area is poorer. The students participating in the Olympiad from the bigger and central cities show higher success rate than those participating from provinces or rural areas.
2. The National Olympiad is held once a year because of poor finance and it is one of the factors that give bad influence to the Olympiad. So the participants in the Olympiad do not have enough experiences and they make some technical and other mistakes.
3. The contents of the national Olympiad also can not meet the secondary curriculum and standard.

4. Experience of the participants is weak because they could not participate in regional and other international Olympiads.
5. The number of the participants is decreasing.
6. The number of teachers who prepare their students for the Olympiad is less and their skills is also not good enough.

Considering conditions mentioned above and current situation, it is appropriate to improve quality of Mongolian Informatics Olympiad and students' knowledge and skills.

The following steps are need to be taken in order to improve the quality of Mongolian Olympiad:

1. To organize domestic Informatics Olympiad in different levels regularly /regional, province, district, and honored contests/.
2. To re-train and prepare teachers who have enough skills for preparing their students for the Informatics Olympiad.
3. To increase the opportunity to teach programming via advanced and elective modules or training for talented students.
4. There are still some problems to be solved according to the evaluation of the Informatics Olympiad. Especially it is needed to develop and use the automatic and online evaluation system which meets the international requirements.
5. The content of Mongolian curriculum must be improved and increased.
6. The English knowledge of the students must be improved and they have to learn technical terms found in Informatics.
7. To increase and extend the scope of Informatics Olympiad among students.
8. Due to inclusion of informatics subject in primary school, it is better to organize mini Olympiads for those students.
9. Foster students and teachers to share their experiences and best practices.
10. To develop an electronic database for Informatics Olympiad tasks and problems.

5. Summary

Mongolian Informatics Association has been very active and contributed in Mongolian Information and Technology development since it was established. In the future it will do lots of things to develop ICT education, to increase the Olympiad development and to meet the Olympiad quality to the international standard. Mongolian Informatics Association works hard to contribute Mongolian Information and Technology Development and make its strategic and operational plan closer to Electronic Mongolia National Program, Concept of ICT Development of Mongolia by Year 2010, and ICT Vision 2010 in Education Sector of Mongolia.

References

MOECS ICT Vision (2001). *ICT Vision 2010 in Education Sector of Mongolia*.
Government of Mongolia (2000). *Concept of ICT Development of Mongolia by Year 2010*.

Government of Mongolia (2005). *E-Mongolia National Program*.

Uyanga, S. (2005). The usage of ICT for secondary education in Mongolia. *International Journal of Education and Development using Information and Communication Technology*, **1**(4).

L. Choijoovanchig is a professor of the School of Computer and Information Technology, Mongolian State University of Education. He is one of the founders of the Mongolian Informatics Association and is working as a president of this association since 2000. His research interests include informatics education, school development and teacher training.

S. Uyanga is an associate professor at Department of Information System, National University of Mongolia. She is PhD in ICT and educational studies, and MSc in computer science. She is a member of working group for Informatics Curriculum Standard for Primary and Secondary Education at Ministry of Education, Culture, and Science, Mongolia. Her research interests include computer and ICT curriculum, ICT in education, web based distance learning.

M. Dashnyam is a lecturer at the Institute of Finance and Economic. He is a secretary of Mongolian Informatics Association since 2004.

Olympiads in Informatics, 2007, Vol. 1, 37–49
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Contests in Programming: Quarter Century of Lithuanian Experience

Valentina DAGIENĖ, Jūratė SKŪPIENĖ
Institute of Mathematics and Informatics
Akademijos str. 4, Vilnius, 08663 Lithuania
e-mail: dagiene@ktl.mii.lt, jurate@ktl.mii.lt

Abstract. The paper deals with development of informatics competitions in Lithuanian and in particular of the Lithuanian Olympiads in Informatics over the past 25 years. The role of the Young Programmer's School in both introducing programming to secondary school students, organizing programming competitions and later encouraging and training students to participate in informatics olympiads is emphasized. The evolution of the national olympiad in informatics from the first pen and paper competitions to the current four round contest, held using contest and grading systems, is described. The paper also gives a short overview of other related contests like the Baltic Olympiads in Informatics, and the Beaver contest.

Key words: teaching programming, informatics, algorithms, contests, informatics olympiads.

1. The Early Years

The first talks about teaching programming at schools started in the beginning of the seventies. In those times some large electronic computers were available in Lithuania, however not for teaching. Only a few educators thought that it would be possible to involve kids in the world of computers and programming. Some researchers from the Institute of Mathematics and Informatics supported this idea and worked on it to become a reality. The first few after-school classes in programming were established in several Lithuanian secondary schools in 1970–1975. There also appeared the first ideas to establish a school to teach programming secondary school students (Dagienė, 2006).

A significant role in designing a methodology for teaching programming was played by the scientist of the Institute of Mathematics and Informatics. In 1979 comprehensive materials for teaching programming were prepared, including tasks, texts, and answers. The Ministry of Education of Republic of Lithuania agreed to try this method in ten schools. Scientists from the Institute created a Pascal translator for the electronic computing machines of the unified system (Baliūnaitė, 1979). The Pascal translator was designed specifically for the learning process, putting emphasis on debugging program source so the user (the student) could receive comprehensive output after trying to compile and execute their first program. The translator even corrected some mistakes and afterwards reported what was changed and how.

The Young Programmers' School by Correspondence (JPM – *Jaunųjų programuotojų mokykla*) was founded in 1981 by the Institute of Mathematics and Informatics (Dagys, 1994; Dagys, 2006). The curricula consisted of teaching main concepts of procedural programming and basic algorithms using Pascal. The learning material and assignments used to be published in the national youth daily newspaper *Komjaunimo tiesa* (cur. *Lietuvos Rytas*) twice a month. The students had to sent their algorithms which were evaluated and the solutions were published afterwards. The best students were invited to summer camps where they not only had a chance to see and touch a computer, but also to compete. The students had to solve algorithmic tasks while designed algorithms had to be written on a piece of paper in Pascal.

In 1985 it was decided to organize the first contest of young programmers. Anyone aged below 30 could participate in it. The contest took part in two rounds. Tasks of the first round were published in the daily newspaper *Komjaunimo tiesa*, in the journals *Informatika* and *Mokslas ir technika* (*Science and Technology*). The tasks even used to be announced through the national television channel. In the first round there were five tasks to solve using one of the three allowed programming languages: Pascal, Fortran, PL/1. Later Fortran and PL/1 languages were removed from the list and C added. The participants had one month to solve the tasks and send the algorithms together with justification to the scientific committee. Sources of the programs could have been written by hand, printed with typewriter or printer.

The Project of Distance Teaching of Informatics using Electronic Mail got support from UNESCO in 1993 (Dagys, 1993). Teaching via electronic mail was mainly performed indirectly through programming contests. The project also took advantage of transmission of source code using media which was quite new and educative in those times. The effect of program execution were observed and evaluated through distance.

In 1996–1997 *The School Of Teaching Algorithms Via El. Mail In Baltic Countries* was established (Dagiene, 1997). Over 100 students from Estonia, Latvia and Lithuania took part in the school. There were two learning and one contest sessions.

2. Young Programmers' School – Search for Talent

The core curricula of the *Young Programmers' School* is teaching algorithms. Programming languages as well as the computer are considered to be learning tools and only the basic information regarding those tools (as much as it is needed to write down and execute the algorithm) is provided. Using Pascal as a working language minimizes the time cost to learn basic constructions of programming languages.

All the teaching materials of the Young Programmers' School consisted of several chapters: 1) identifiers, variables, constants, assignment statement and sequence of statements, 2) conditional statements, 3) repetitions of actions, 4) programs and their execution by a computer, 5) logical values, 6) functions and procedures, 7) recursion, 8) discrete data types, 9) real numbers and records, 10) arrays, 11) programming style, 12) program design. We would like to emphasize again that all the theory is taught only through program comprehension (e.g., given fragments of a program which has to be completed, corrected, etc.) and program design assignments.

There exists a steady attitude in the Young Programmers' School that a student has not only to become acquainted with the basic constructions of programming but also has to learn how to justify and describe an algorithm, and design clear and simple solution.

For most children, theory is less attractive than practical activities. Thus the basic principles of the theory were delivered in an indirect way through problem solving. The set of programming problems was chosen in accordance with the requirements dictated by theory and good programming style (Grigas, 1990).

From the start of the Young Programmers' School until now there have been many changes in the teaching of informatics in general and in programming in particular, due to the increase in the number of computers in educational institutions and the introduction of informatics as a compulsory discipline in secondary schools. The changes in the structure of the Young Programmers' School may be characterized by five periods: 1) Universal (general) programming teaching (1981–1986), 2) Learning effectively: differentiation by students' abilities (1986–1993), 3) Intensive teaching of gifted students (1993–1999), 4) Preparing students for the olympiads (since 1999), and 5) Using new media while learning algorithms (since 2005).

The first Informatics Olympiad for the enrolled students of the Young Programmers' School took place in summer 1982. The tasks were designed with extreme accuracy and forethought. They were attractive and challenging (Dagienė, 1991). Programming olympiads used to take place every year but only for the students studying in Young Programmers' School. Therefore the Young Programmer's School was an impulse to start the *Lithuanian Olympiad in Informatics*.

3. Evolution of the Lithuanian Olympiad in Informatics

The first Lithuanian nation-wide Olympiad in Informatics was organized in 1990, i.e., the year after the first International Olympiad in Informatics (IOI).

In the beginning the olympiad consisted of the three rounds: 1) school round 2) regional round (about 60 regions), and 3) national round. Since 1993 the national round has been split into two parts. The first part was organized using e-mail, the second was on-site competition. The structure of four rounds is convenient and has been kept until now.

The final stage of the national olympiad is organized in a different region each year (Fig. 1). About 50 participants from all over Lithuania are invited. Organizing the event in different regions not only allows the contestants to get to know the region but also gives a possibility to the teachers of local schools to look at the olympiad from inside – to observe how the final versions of tasks are being prepared, and to look closer at the competition system and grading.

3.1. *Using E-mail for On-Line Contests*

The organization part of the national round of the first national olympiad was quite complicated. Each of the sixty counties in Lithuania designation winners of their regional

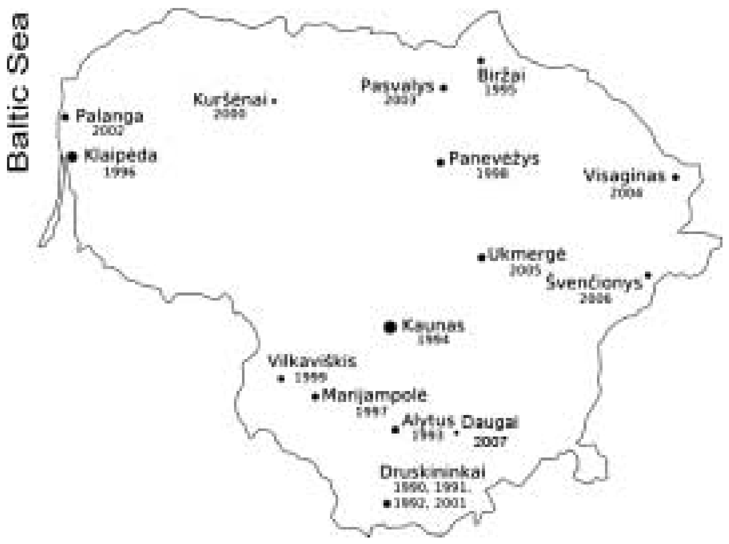


Fig. 1. Regions that hosted finals of the national olympiad in Informatics.

competition for the national round. As it was not possible to arrange an on-site competition for about two hundred students, the first part of the national round used to be arranged in several selected municipalities. The organizers of the olympiad would send their representative with tasks to each municipality, the representative would observe and coordinate the competition and bring back solutions to the scientific committee for evaluation.

A significant breakthrough became possible in 1993, when the Fidonet computer network became available for some schools in each region. It was decided to organize the first part of the national round in each region using e-mail. Then it was both a brave and innovative decision. On the one hand even though e-mail was available at schools many teachers still did not know how to use it nor that there was a real need for it. Organizing a competition in such a way stimulated teachers to learn how to use e-mail.

However in the first years there was not an easy job for organizers of the olympiad to manage the contest. Some e-mails would not reach the contest organizers and the organizers had to call the region and find out what happened, there were lots of problems with attaching solutions. Teachers either did not know how to attach a file or the attachment received was un-decodable and the organizers spent a lot of time consulting teachers how to do it. A lot of problems were caused due to ignorance of file naming instructions. Many solution files would be given random names and contained no contestant name in the comments.

It took several years until the teachers got accustomed to using e-mail and the transfer of solutions to the scientific committee became fluent. Olympiads had positive educational effect also on promoting the use of e-mail in Lithuanian schools.

Solutions were delivered through e-mail and afterwards graded using black-box testing for the ten years from 1993 till 2002.

3.2. *Contest Management and Grading Systems*

In order to test solutions automatically programs have to comply with certain input/output formatting requirements. In several IOIs there were severe problems regarding automated grading when programs with typographic errors in the data file name or those leaving extra space at the end of line were not given any points. Similar problems were also encountered in Lithuanian olympiads.

However in order to motivate students the scientific committee tried to be more objective and to make distinctions between formatting errors and more serious mistakes. This was a huge load of work for the scientific committee. Nearly every program which did not get full score had to be checked manually searching for formatting errors.

This was especially important for younger students as for many of them this was their first competition and getting zero points for a program with a good algorithm and a typographic error might have resulted in a decision to quit the olympiad. Later the grading became more strict, especially in the senior division. On the one hand it was decided that part of this workload could be done by the contestants themselves (i.e., they should analyze their solution and write an appeal), on the other hand in some cases it was hard to distinguish between formatting and non-formatting errors and it was decided not to change the code of the contestant at all.

The first Contest and Grading System that allowed the submission of programs via a web-interface during contest time, and to check whether they compile and comply with format requirements, was designed and used in Finland in 2001. Such a system has been used in Lithuanian contests since 2003.

In the IOI the use of a Contest and Grading system during the competition was accepted with support from the contestants. However this was not so in Lithuania. The top contestants who had participated in IOIs supported the use of a Contest system in Lithuanian olympiads. However many inexperienced (especially younger) contestants were shocked when they tried the system for the first time. Typically they tried to submit a program which produced correct output on their computer and when the contest system rejected the program they were lost and did not even know how to try to debug it as their debugging skills were quite low. Some of them could not even understand what happened, as they could not believe that a program which works on one computer might have failed on another. Even if the problem was a typographic error in the file name for some students it was impossible to find it out. Again there started coming a huge numbers of e-mails to the technical committee asking for help and complaining that the system did not work correctly. The Contest and Grading system has been used in Lithuania for five years and many teachers have got accustomed to using it. However still in some schools some people find it difficult to use and can not understand that error message during the contest is actually a hint that they should use (Skūpienė, 2004).

3.3. *Participants*

All students in secondary education under the age of 20 are invited to compete in Lithuanian Olympiads in Informatics. Approximately 3000 students take part in national competition each year.

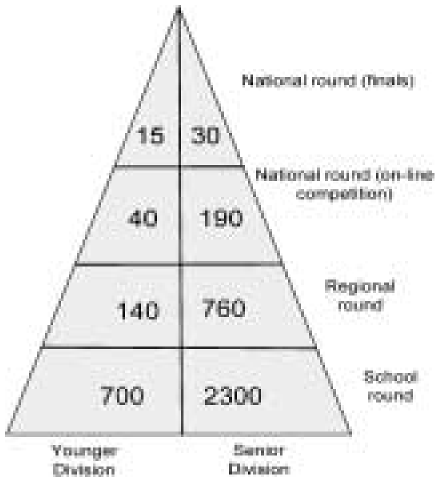


Fig. 2. Fig. 2. Average number of participants in Lithuanian Olympiads in Informatics.

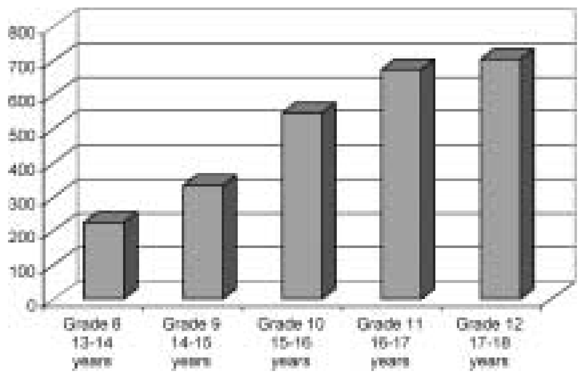


Fig. 3. Distribution of participants in the first round of LOI'2006 according to grades.

The number of younger (grades 8–9) participants has significantly increased since 1997 when a separate division for younger students was established and 30% of the places in the finals of the national competition were reserved for students from younger division. This motivated both younger students and their teachers.

4. Tasks – Keystone of Contests and Resources for Problem Solving

Interest and engagement is very important in competitions as well as in teaching problem solving (e.g., in assignments in the Young Programmers' School) and it essentially depends on problems. Therefore the problems have to be designed taking into account different aspects of each problem, i.e., its educational power and how to interpret its attractiveness to the students (i.e., whether it stimulates learning or not). Attraction, invention, discoveries are the desired features of a good problem set. Here are some other desired characteristics of problems:

- interesting and attractive formulation;
- algorithmic solution lies behind;
- variety in difficulty;
- do not require specific, especially technical knowledge;
- short, elegant formulation.

Therefore, one should try to present problems from various spheres of science and life, with a lot of real data. Processing large amounts of data becomes challenging and important aspect when learning programming.

However, many the textbooks and teaching materials do not contain actual problems but just small exercises. They are mainly oriented towards checking syntax of a particular programming language. The selection of tasks at the distance education school is very important: they must cover as many theoretical problems as possible, teach students algorithms and programming methods, and what is most important, to acquire the skills of using them (Dagienė, 1993; Dagienė, 1999; Mayer, 1990).

While developing the methodology of teaching algorithms for the School of Young Programmers, we have raised the principle that it is highly important to classify the problems into the sets of problems that would actualize the purposes of teaching algorithms. Two large groups of problems were distinguished: 1) reading problems (for analysis); and 2) writing or design problems (Grigas, 1989).

When someone starts learning programming, active and passive learning methods can be distinguished. This deals with fixing priorities: whether they are taught just programming language constructions or problem solving using programming languages. An active teaching is when problems are solved while the languages constructions are mastered gradually when they are needed in the solution. Therefore it is highly important that the tasks of the first rounds in the national olympiads take this into account, i.e., we would seriously consider which language constructions are needed to solve the problem, so that the beginners would be able to solve the problems.

A variety of task books with problems and their solutions have been prepared and published.

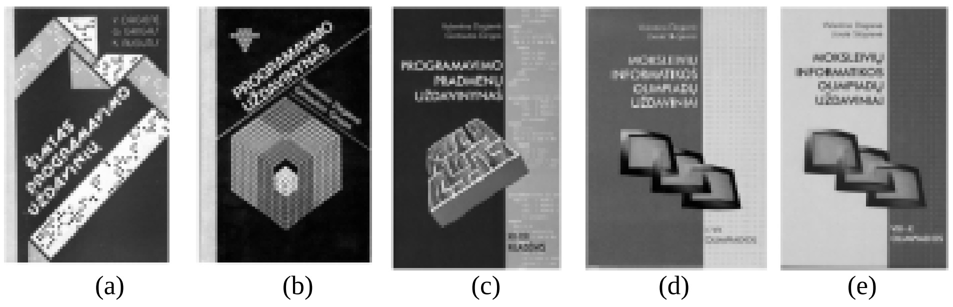


Fig. 4. Books in problem solving for olympiads in informatics: (a) *A Hundred Tasks in Programming*; (b, c) Programming tasks; (d, e) *Olympiads in Informatics* (volume 1 and 2).

5. Achievements of Lithuanian Students in IOI

The first Lithuanian contestant Andrius Čepaitis participated in IOI in 1989 where he was awarded a gold medal. At that time Lithuania was still part of Soviet Union and Andrius was included into the team of Soviet Union. Lithuania joined the IOI in 1992 soon after restoring independence. Since then each year Lithuania selects sends a team of four contestants to IOI and the Lithuanian contestants were awarded 44 medals in total.

Team selection for the IOI is conducted in the following way. The best six contestants from the national competition senior division are invited to take part in Baltic Olympiad in Informatics which usually takes place in April or May. In cases where it is not possible to do a fair selection of the BOI team due to a small difference in points, extra competitors take part in the BOI on-line, under surveillance of members of NOI. The team of four contestants to represent Lithuania at the IOI is selected taking into account points gained in finals of national competition, points gained in BOI, medals in International Olympiads in previous years and age (younger contestants have priority).

6. Other Contests in Informatics

The national olympiad in informatics is the main however not the only contest for Lithuanian students interested in algorithms and problem solving.

Table 1
Achievements of Lithuanian students of in IOI's

	Gold	Silver	Bronze
IOI 1989	1		
IOI 1992			3
IOI 1993			3
IOI 1994		2	2
IOI 1995	1		3
IOI 1996	1	2	
IOI 1997			1
IOI 1998		1	2
IOI 1999		1	2
IOI 2000		2	1
IOI 2001		1	2
IOI 2002		1	2
IOI 2003		1	1
IOI 2004		1	2
IOI 2005		1	1
IOI 2006		2	1
Total	3	15	26

In 1997 scientists from Kaunas Technology University together with an American Lithuanian Dr. J.P. Kazickas, whose fund sponsors the events, established the *Dr. J.P. Kazickas Regional Competition of Students Computerists*. Ethnically Lithuania is divided into four regions (Aukštaitija, Žemaitija, Dzūkija ir Suvalkija) and the contest takes place in each region once a year. The winners of regional contests are invited to the final round which take place in Kaunas Technology University. The winners of the competitions are given the priority against other candidates if they want to study in the University (Kazicko forumas, 2007).

In order to bring programming and concepts of computer science to younger students, Logo was chosen to be introduced into Lithuanian schools many years ago. As it is known that competition makes learning more attractive, in 2000 it was decided to establish *Logo Competition-Olympiad*. Students in both primary and secondary education are invited to compete. The competition consists of several separate parts and the contestants decide in which particular part to participate. There are contests of pictures, uncontrolled animation, controlled animation, as well as algorithms. Thus younger students become involved in studying algorithms at quite a young age, even before they learn procedural programming language.

A Saturday school of participants of informatics olympiads was established in 2003. The school has three levels. Young students who have no experience in programming, but proved themselves to be good in math competitions, can study at the introductory level. The curricula of the first level covers introductory to programming course as well as some very basic algorithms (e.g., sorting). At the end of the first level (which lasts for one year) the students take a test and those who pass can study at the intermediate level. Also the students who successfully participated in the national informatics olympiad in younger division can study in this level.

During the intermediate level the course covers the main algorithms and methods used to solve tasks for informatics olympiads (e.g., Dijkstra algorithm, dynamic programming, etc.).

At the advanced level the students who passed the final test of the intermediate level as well as participants of finals of national informatics olympiad can study. At the advanced level the students solve complicated and advanced algorithmic tasks and it is possible to study at this level for several years (i.e., until they graduate school). Currently about 50 students from all over Lithuania study in the school.

In 2004 Kaunas Technology University Gymnasium together with charity and support fund of M. Rostropovich initiated the project *National Student Academy* for gifted students. Students having high achievements in various areas (mathematics, chemistry, biochemistry, physics, informatics, economics, writing and music) can enrol at the Academy. During the year they are working on assignments from their subject. There are two or three sessions throughout the year which last from one to two weeks. Scientists are invited to give lectures to the Academy students as well. The exceptional feature of this project is that students from different areas of interest join together to work, create and learn.

In order to achieve high results in informatics olympiads, one has to study and practice for several years, and not so many students are able to do it. We felt that there was a

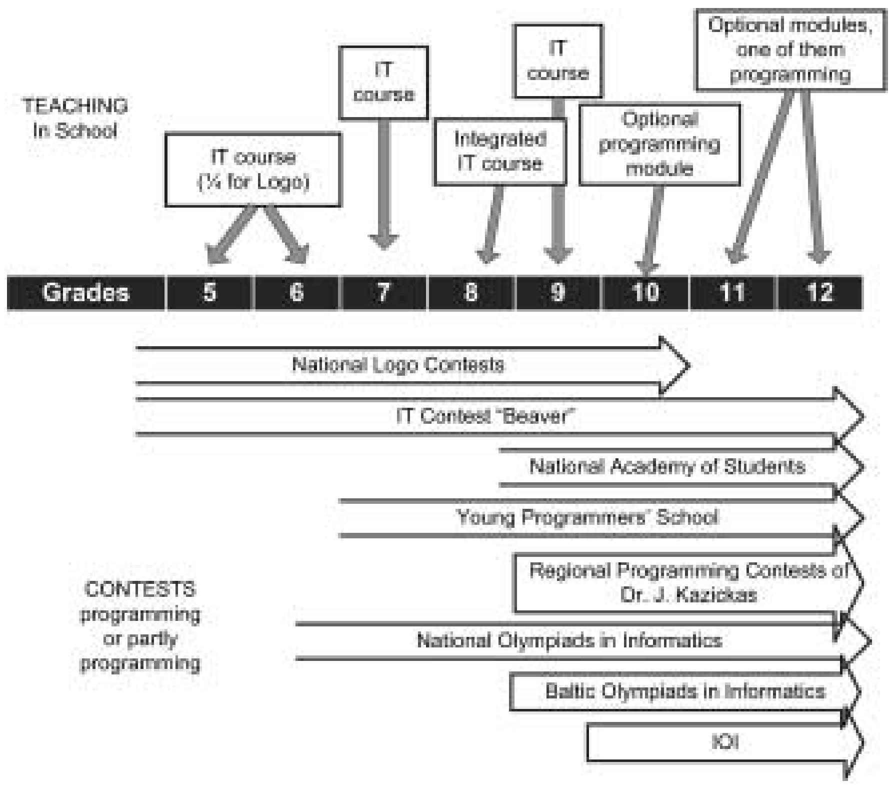


Fig. 5. Teaching Information Technology (IT) and programming in schoolls and various of contests.

lack of a competition in informatics where everyone could participate and have fun. Such a competition existed in mathematics – "Kangaroo". In 2004 it was decided to arrange a similar competition in informatics. The history of "Beaver" began on September 25, 2004, when an experimental trial, in which 779 school students participated, was held. The aim was to check selected technologies of the contest and to evaluate the level of complexity of the presented problems. After a month, on October 21, the first Lithuanian "Beaver" contest took place. As many as 3470 pupils from 146 schools participated. In 2006, the International "Beaver" organizing committee was established which included representatives of seven countries. Tasks are designed for three different age groups (11–14 years, 15–16 years, 17–18 years) and they are multiple-choice tasks. Grading is automated. Average time for solving one task is 2–3 minutes, so task descriptions are short in form. There is still no defined syllabus for the contest, but the discussed topics are: a) general logic; b) ICT in everyday life; c) practical and technical issues; d) information comprehension; e) algorithms and programming; f) mathematics and underlying CS; and g) history and trivia (Opmanis, 2006).

In order to ensure better preparation for the IOI and to strengthen regional relations, various regional olympiads are being organized. Baltic Olympiads in Informatics (BOI) were established by the initiative of the three Baltic countries (Estonia, Latvia, and

Lithuania) in 1995. Year by year six other Baltic countries (Denmark, Finland, Germany, Poland, Sweden and Norway) joined the BOI. Compared to the IOI, BOI is a short-term (the duration is 5 days) and inexpensive event. It can be distinguished by a cosy and good neighbourly atmosphere.

Even though the BOI is a mini-model of IOI it differs significantly. The organization of the scientific part of the BOIs is based on mutual trust of participating countries. The leaders of all the participating countries take part in proposing and selecting problems for the coming BOI. After draft problem formulations are presented, the problems are discussed via e-mail and the each country votes for the problem set for the competition. Most of the problems are translated into native languages by the leaders before leaving to BOI. During the competition leaders are involved in solving various problems which might occur, for example, some misrepresentation in the formulations of contest problems. This a unique possibility for country representatives to gain experience in organizing scientific part of a small international olympiad (Dagienė, 2004).

The BOI is also a form of learning for its participants. The organizers of BOIs try to follow as close as possible the newest IOI trends in problem types, compilers, platforms, contest systems. It is not always possible to do that in national contests. Besides, the competition tasks are always proposed by different countries. Even though all the tasks are of an algorithmic nature they represent cultural and methodical differences. Many students come to the BOI to gain international experience after participating in domestic contest. The BOI can be considered as a pre-arranged international form of learning.

7. Conclusions

Lithuania has long traditions of both teaching programming and algorithms in secondary schools and organizing informatics contests. We have noticed three basic challenges:

1) initiating students to start to learn programming and to do it in attractive and proper way;
2) when students learn the basics of programming they start trying to find an event where they could demonstrate their skills – contests and olympiads are the right place to do this;
3) there are many students who are interested in problem solving and would be interested in participating in the contests but they need instructors or some kind of schools to help them to grow their skills.

We try to investigate and to work in those three directions. In primary education, we introduce Logo: kids aged 11–12 years have chance to develop their own procedures using very simple programming statements (mainly primitive commands, loops and procedures). In lower and upper secondary school students have opportunity to choose optional modules of programming (each are for 70 hours). While learning programming students have possibilities to participate in various contests: Logo Olympiad, IT contest “Beaver”, Lithuanian Olympiad in Informatics, etc.

Consequently both teaching of programming (it can be optional, but it should be available) and contests complement and stimulate each other. At the same time variety in contests give more possibilities for interested and talented students and allows them to choose the contests which are most interesting and challenging for them.

References

Baliūnaitė, A., V. Dagienė, G. Grigas (1979). Transliator jazyka PASCAL v operacionoj sisteme DOS/ES i jevo ispolzovanyje dlia učebnych celej. *Operativno-informacionyj material*. Novosibirsk: SO AN SSSR (in Russian).

Bulotaitė, J., K. Diks, M. Opmanis, R. Prank (1997). *Baltic Olympiads in Informatics*, Inst. of Math. and Inf., Vilnius, Lithuania.

Dagienė, V. (1991). *Lietuvos jaunųjų programuotojų olimpiados*. Kaunas, Šviesa.

Dagienė, V. (1997). Learning via electronic mail: what and how? *Education for the 21st Century*, 2–4 December, Cape Town, 1–10.

Dagiene, V. (1999). Programming-based solution of problems in informatics curricula. *Communications and Networking in Education: Learning in a Networked Society*, IFIP WG 3.1 and 3.5 (with 3.6). Aulanko, Hämeenlinna, Finland, June 13–18, 88–94.

Dagienė, V. (2006). *The Road of Informatics*. Vilnius, TEV.

Dagienė V., G. Grigas (1993). Development of problem solving skills and creativity through distance teaching of programming. In G. Davies and B. Samway (Eds.), *Teleteaching: IFIP Transactions (A–29)*. Elsevier, Sc. Pub., pp. 179–182.

Dagiene, V., J. Skūpiene (2004). Learning by competitions: olympiads in informatics as a tool for training high grade skills in programming. In T. Boyle, P. Oriogun, A. Pakstas (Eds.), *2nd International Conference Information Technology: Research and Education*. London, pp. 79–83.

Dagys, V. (1994). The work principles of Lithuanian Young Programmers School by correspondence. Human Resources, Human Potential, Human Development: the Role of Distance Education. In *Proceedings of the European Distance Education Network (EDEN) Conference*. Tallinn, 182–184.

Dagys, V., V. Dagienė, G. Grigas (2006). Teaching Algorithms and programming by Distance: Quarter Century's Activity in Lithuania In V. Dagienė, R. Mittermeir (Eds.), *Proc. of the 2nd Int. Conference "Informatics in Secondary Schools: Evolution and Perspectives"*, Vilnius, 402–412.

Dagys, V., A. Klupšaitė (1993). Distance teaching of programming and possibilities of e-mail. *Informatica*, **4**(3–4), 303–311.

Grigas, G. (1990). Some aspects of teaching the art of programming by correspondence. *Informatica*, **1**(1), 156–166.

Grigas, G. (1989). Informatics and creative Ttinking. *Third International Conf. "Children in the Information Age"*. Sofia, pp. 229–240.

Dr. J. Kazicko moksleivių kompiuterininkų forumas (accessed 2007.06.20). http://pilis.if.ktu.lt/˜forumas/

Mayer, R.E. (1990). Teaching for transfer of problem-solving skills to computer programming. Computer-based learning environments and problem solving. In E. de Corte etc. (Eds), *ATO ASI Seties*, Springer-Verlag, 193–206.

Opmanis, M., V. Dagienė, A. Truu (2006). Task types at "Beaver" Contests. *Infomatics Education*, the Bridge between Using and Understanding Computers. *Proceedings of International Conference in Informatics in Secondary Schools – Informatikon Technologies at School*, November 7–11, 2006, Vilnius, TEV, 509–519.

Skūpienė, J. (2004). Automatinis testavimas informatikos olimpiadose. *Informacinės technologijos 2004*, Konferencijos pranešimų medžiaga, Kaunas, Technologija, p. 37–41.

V. Dagienė is a head of Informatics Methodology Department at the Institute of Mathematics and Informatics, also Professor at Vilnius University. She published over 100 scientific papers, wrote more than 50 textbooks in informatics for secondary education. She coordinates the Young Programmer's School, has position of the Chair of The National Olympiads in Informatics, established IT contests "Beaver". V. Dagienė is the national representative of the IFIP for Education (TC3), member of the European Logo Scientific Committee, an elected member of the IOI Scientific Committee. She is an Executive Editor of international journal "Informatics in Education".

J. Skūpienė is younger research fellow in Informatics Methodology Department in the Institute of Mathematics & Informatics. She has published about 10 scientific papers. She is a member of the Scientific Committee of National Olympiads in Informatics since 1994 and a team leader in IOI since 1996. For a few years she was director of studies of Young Programmers' School, since 2004 she has been a coordinator of Informatics section in the National Academy of Students.

She is author/co-author of four books on algorithms and algorithmic problems.

Olympiads in Informatics, 2007, Vol. 1, 50–56
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Polish Olympiad in Informatics – 14 Years of Experience

Krzysztof DIKS, Marcin KUBICA, Krzysztof STENCEL
Institute of Informatics, Warsaw University
Banacha 2, 02-097 Warszawa, Poland
e-mail: {diks,kubica,stencel}@mimuw.edu.pl

Abstract. This paper presents the organization of the Polish Olympiad in Informatics, together with tasks preparation process and evaluation. It is a result of over 14 years of experience in organization of programming contests for high-school pupils. We believe, that although described procedures are rather widely know, their rigorous implementation is the key to organization of a successful programming contest.

Key words: algorithmic problem solving, programming contest, informatics olympiads.

1. Introduction

Polish Olympiad in Informatics (POI) originates from a smaller contest, called *Contest in Informatics*, and after evolving for a couple of years, it gained status of the national olympiad in 1993. It is addressed to high-school students, however middle-school pupils can also take part in it. The detailed organization rules can be found in (XIII Olimpiada Informatyczna, 2006) or on the web page of POI: `http://www.oi.edu.pl/`.

POI consists of three stages. In each stage, a number of tasks of algorithmic nature is presented to contestants. Solution of each task is either a computer program, or computed data. The supported programming languages are: C, C++ and Pascal.

The first stage is usually organized in October and November. It gathers over a thousand of contestants. Among them, about 40 contestants are from middle-schools. The contestants are presented five or six tasks, that should be solved at home within a month and sent backfor evaluation. About 360 contestants are qualified to the second stage of competition.

The second stage is organized in six regional centers, located at universities cooperating with POI and takes three days. The first day is a preparation day – the contestants have to solve one or two tasks during a three hour session, however the results do not count in the competition. The objective of the preparation day is twofold: first, the contestants can get familiar with the environment, second, the organizers can verify that everything is ready. During the second and third day, the contestants have to solve two or three tasks during five hour sessions. The solutions are collected from all the centers and then evaluated. Preliminary results are usually known a couple of hours after the competition is finished. The results are approved in two weeks, which gives time to process possible contestants' complaints. About 70 contestants are qualified to the third stage of competition.

The final and third stage is organized in one place and takes five days. Traditionally, it is organized in Sopot, a small city at the Baltic shore. Similarly to the second stage, there are three competition days. The first day is also a preparation day, when the contestants have to solve one or two tasks in three hours. on each of the other two competition days the contestants have to solve three tasks in five hours. The preliminary results are known just after the competition. The fourth day is a leisure day for contestants, while for organizers it is a day reserved for processing possible complaints. On the last day, there is an awarding and closing ceremony.

POI loosely follows international tradition in medal allocation. There are three categories of medals: golden, silver and bronze. The number of medallists rather does not exceed half of the number of finalists, and the ratio between the number of golden, bronze and silver medals is close to 1:2:3, but it is not a rule.

Each year, POI requires preparation of approximately 17 tasks. Tasks preparation is a continuous process. Before each stage of the competition, 5 to 7 tasks for this stage are selected from the pool of about twelve tasks ready for the competition.

The next section of this paper presents the tasks preparation process, from a task idea to a moment when the task is ready to be used. In the following section the evaluation process is described.

2. Tasks Preparation

The main objective of the tasks preparation is to assure good quality of tasks. But what does it mean? What makes a good task? We should take the following aspects into account:

- Task formulation – it must be clear, comprehensive and not too long.
- Differentiation of contestants' skills – there should exist many ways of solving the task, of different difficulty; moreover, it should be possible to distinguish these solutions by testing.
- Thoroughness of analysis – task analysis should take into account wide spectrum of solutions, covering all ways of solving the task accessible to the contestants, and different programming languages and usage of STL (where necessary).
- Thoroughness of testing – tests should distinguish correct and incorrect solutions; they also should distinguish different classes of solutions, regardless of the programming language used to implement them (and possible usage of STL).
- Correctness – all example programs should obey input/output specifications and should produce correct outputs; if necessary, an output checker is also needed.

Describing the tasks preparation process, we will emphasize requirements needed to achieve the above objectives.

The tasks preparation process consists of the following phases: review of task ideas, formulation, analysis, verification, and calibration.

2.1. *Reviewing Task Ideas*

Initially we need just a task idea. It only has to define the algorithmic problem to be solved. When reviewing the idea, we should answer the following questions:

– Can the task be formulated in a short and comprehensive way? If it is too complicated or requires explanation of many terms, then it is not suitable for a competition.
– Is the task a ‘handbook’ one? If so, then it would test knowledge of a particular algorithm/technique rather than creativity. Hence it is not appropriate.
– Is the task unique, to the best knowledge of the reviewer?
– Can it be solved in a polynomial time? If not, then it is rather not possible to evaluate it in a reasonable time. However, exceptions are possible.
– Are there many ways of solving the task, with different difficulties and different (time) complexities? If not, the task is probably not appropriate for a contest, or it may not be possible to distinguish different classes of solutions.
– Can it be solved by a high-school student? There is no universal answer to this question. Our requirements are a little bit higher than those defined in (Verhoeff *et al.*, 2006). The expected knowledge is covered by most general handbooks on algorithms, e.g., (Cormen *et al.*, 1989) (skipping more advanced chapters) covers it all.

2.2. *Task Formulation*

In the task formulation, all elements missing in the task idea should be added. In particular: a short story can be added to make the task more attracting. The language should be simple. One should avoid complex sentences. All new notions should be introduced before they are used. Greek symbols should be avoided. If a coordinate system is needed, then the standard one should be used. Other detailed guidelines can be found in (Verhoeff *et al.*, 2006).

Input and output specifications must be precise – limits on the data sizes can be left undefined. Preferably, the output should be short, hard to guess (e.g., not a yes/no answer but rather some integer) and unequivocally determined by the input. However, this last requirement is not crucial. The task formulation should contain an example, preferably with a picture. The task should fit on one or two pages. Three pages are an absolute limit. The task formulation should be also accompanied by a short description (one or two paragraphs) of author’s solutions – it will be taken into account during the analysis.

2.3. *Task Analysis*

Task analysis is the most time-consuming part of preparation. The outcome of the analysis should consist of: a document summarizing the analysis, a number of programs and tests. Also, all missing elements in the task formulation (e.g., limits on the data sizes) should be defined.

The analysis document is an internal document, so it can be written in a professional language. The analysis document should discuss different solutions: the optimal solution (within contestants' scope), other possible solutions and a couple of incorrect solutions that could be expected. It should discuss all the solutions proposed by the author of the task, but by no mean should it be limited to these solutions. Of course not all incorrect solutions can be foreseen, but a good sample is valuable.

The solution descriptions for pupils are prepared post factum. However, if they are to be distributed during the competition, they should be prepared together with the analysis document.

All correct solutions should be implemented both in C/C++ and Pascal. Moreover, if application of STL is relevant, such solutions should be implemented in C++ using STL and in C (not using STL). The rationale is that we need to know the actual running time of these solutions. For other solutions, e.g., incorrect ones, just one implementation is enough, since they should produce incorrect outputs.

In case of batch (i.e., typical) tasks, the set of 10 to 20 tests should be prepared. The primary objective of tests is to distinguish correct and incorrect solutions. However they should also distinguish all the classes of correct solutions, that are of different difficulty. Tests should put stress on the asymptotic time-cost rather than absolute running time. As a rule of thumb, solutions up to twice slower then model solutions should score the full points. Moreover, the result of testing should not depend on the choice of the programming language, or usage of STL. Some version of IOI-50% rule can be applied – 30%–60% of points should be allocated to correctness tests. In other words, correct but not efficient solutions (however running in a reasonable time) should score 30%–60% of points. The rest of points should be granted for efficiency. Such a thoroughness of testing could be hard to achieve. The usual solution is to increase the data sizes. However, the amount of available RAM and expected testing time can limit it.

If necessary, tests can be grouped – a solution is granted points for a group of test only if it passes all the tests from the group. Grouping should be used when the correct result could be 'guessed' with high probability, or when more than one test is needed to distinguish correct from incorrect solutions.

Tests can be prepared in form of files or a generating program can be provided. The latter option is especially useful for generating huge efficiency tests. If such a program uses random number generator, then it should also set the random seed, so that it always generates exactly the same set of tests. Tests should be accompanied with a program verifying their correctness. Such a program should verify all conditions stated in the task, what is sometimes quite not trivial.

If the task is an output-only one, then the set of tests should be prepared in a similar way, however we cannot control the running time. Contestants can even use different programs to solve different tests. Usually we cannot measure efficiency of contestants' solutions. However in this type of tasks, the running time is not so crucial, or the competition time is a sufficient limit. So, we can skip the requirement to implement all correct solutions in all supported programming languages.

If the task is an interactive one, then we have to provide modules that should be compiled with contestants programs, in all supported programming languages. There should be two versions of such modules: one provided during the competition and one for the evaluation. The first version should just allow testing contestants' solutions. The second one should implement a couple of 'strategies'. Different interacting strategies and different initial configurations correspond to tests. The module used for evaluation should also be secured against reverse-engineering attacks or misuse. It can be divided into a separate process and simple communication module that is compiled with contestants solution. Then, the separate process is protected by the operating system. Moreover it can be implemented in just one programming language. The communication can be done via standard input output. However it should contain verification/check-sum codes, to prevent contestant's code form interacting with such a process.

2.4. *Verification*

The main goal of the verification phase is to assure correctness. The two main ways to achieve it is thorough inspection of the analysis document and model solutions, and cross-checking the model solutions and tests. The inspection should cover: task formulation, analysis document, model solutions and programs for test generation and verification. An independent model (but not necessarily optimal) solution should be implemented. The result given by such a solution should agree with those produced by the model solutions. Also, a program verifying correctness of tests has to be implemented. All model and incorrect solutions should be evaluated on all the tests, and it should be possible to distinguish classes of solutions of interest.

2.5. *Calibration*

All other phases of tasks preparation can be done in advance. However it is not possible to define the actual time-limits without knowledge of the hardware used to evaluate solutions. And this is usually known just before the competition. Hence, the calibration of time-limits must be done as soon, as the hardware used for evaluation is installed.

3. Evaluation

In POI solutions to all tasks are graded automatically. A special software system has been created for this purpose. It has been evolving from 1992. First it was written for MS-DOS, then ported to Windows NT, thrown away and totally rewritten, and then finally ported to Linux when it got the current shape of a full-fledged web system known as SIO (Information System of Olympiad[1]). Contestants submit their solutions to the SIO system which grades them and provides results. In most cases the full publication of the results is delayed until the end of the competition. Only the result of the test case from the task description is publicized immediately.

Most tasks require writing a program which is to read the input data and produce a result. Such a task will be called a *batch task*. A small set of test cases is prepared for each

[1]The suspected permutation of letters is phantom. The abbreviation comes from the Polish name of the system.

batch task. Each submitted solution is run for each test case. The SIO system measures time and kills the program if the real time of the run exceeds the time limit twice. The time limit does not concern the real running time but the system time plus user time of the process. We allow doubled time limit for safety, but only the process running time counts. If the process time exceeds the time limit, the solution will always get no points for the given test case. The SIO also controls the security. It will stop a solution, if it executes a forbidden system call (e.g., forks and network routines) or opens any file (solutions to batch tasks are reading from the standard input and are requested to send the result to the standard output).

If for a test case the solution does not exceed the time limit, terminates with the exit code 0 and provides a correct answer, the solution will get points for this test case. In all other cases, the solution will get no points for this test case.

There are two problems with such a grading procedure. POI is proud of finding good yet simple solutions to them. The first problem is the discontinuity of the function which maps running times to points. The second problem consists in outputs which are relatively frequent in the result space (e.g., the answer 'NO' in a task which requires a specific sequence of numbers as output).

The time-to-points function is flat from zero to the time limit (its value is the maximum number of points for the test case). Then, it instantly drops to zero and stays flat until infinity. Apparently, this function is not continuous in the point of the time limit. If a solution fluctuates over the time limit, its result will vary much in subsequent evaluations. In order to avoid it, we decided to make the time-to-point function continuous. In POI this function is flat from zero to the halved time limit. Then it linearly descends to zero in the point of time limit. It is now continuous and if the running time of a solution fluctuates, the number of point will never change rapidly. The final required remark here is that time limits are set in such a way that the model solutions always terminate long before the time-to-point function starts its descend.

The problem of answers frequent in the result space has been eventually adopted by the IOI, but it has been invented for POI. The solution to this problem is as simple as the continuous time-to-point function. Each test case consists of a number of test runs. Each test run is connected with specific input data. Thus, running each test case means running a solution program for each test run separately. A solution will get points for a test case only if it solves all test runs correctly. Let us consider an example task where the answer is a sequence of integers with some properties, but if such a sequence does not exist, the program must output one word 'NO'. Of course we don't want to award any points to a solution program which always prints 'NO'. Thus, all test runs for which the correct answer is 'NO' are grouped in test cases together with test runs which do produce sequences of numbers. This way programs which solve nothing will get nothing.

Last but not least we should mention *errare humanum est*. Indeed, the grading procedure is configured and maintained by humans. Particularly, it concerns preparation of test data. Thus, before the decisions on the qualification to the next round or the medal allocation are made, contestants have access to the preliminary grading of their solution on all test cases. These days it is easy, since we have the Internet. All this information is available on contestants' accounts in the SIO system. The contestants can appeal (of course through SIO). Yet after all the appeals are concluded, the results are verified by those who are the most interested. The decisions on medals and qualifications are made using these strongly verified results.

4. Conclusions

Running an annual programming contest is a never-ending job. We have described the organization of the Polish Olympiad in Informatics, with focus on tasks preparation and evaluation. We hope, that it can be fruitful to organizers of other contests.

References

Cormen, T.H., C.E. Leiserson and R.L. Rivest (1989). *Introduction to Algorithms*. The MIT Press and McGraw-Hill Book Company.

Kanarek, P., and A. Iwanicki (Eds.) (2006). *XIII Olimpiada Informatyczna*. Komitet Główny Olimpiady Informatycznej.

Verhoeff, T., G. Horváth, K. Diks and G. Cormack (2006). A proposal for an IOI syllabus. *Teaching Mathematics and Computer Science*, **IV**(I).

K. Diks (1956), PhD hab. in computer science, associate professor at Warsaw University, director of Institute of Informatics of Warsaw University, chairman of Polish Olympiad in Informatics, member of IOI-IC since 2001, former chairmen of IOI'2005 in Nowy Sacz, CEOI'2004 in Rzeszów and BOI'2001 in Sopot, Poland. His research interests are: algorithms and data structures, parallel and distributed computing, and graph theory.

M. Kubica (1971), PhD in computer science, assistant professor at Institute of Informatics, Faculty of Mathematics, Informatics and Mechanics, Warsaw University, scientific secretary of Polish Olympiad in Informatics, former IOI-ISC member and former chairman of Scientific Committees of IOI'2005 in Nowy Sacz, CEOI'2004 in Rzeszów and BOI'2001 in Sopot, Poland. His research interests focus on combinatorial algorithms and computational biology.

K. Stencel (1971), PhD hab. in computer science, at the moment works at the Faculty of Mathematics, Informatics and Mechanics of Warsaw University. His research interests are connected with non-relational databases. From 1995 he has been the chairman of the jury of Polish Olympiad in Informatics. He was also the chair of jury at CEOI'97, BOI'2001, CEOI'2004 and IOI'2005.

Olympiads in Informatics, 2007, Vol. 1, 57–65
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Slovak IOI 2007 Team Selection and Preparation

Michal FORIŠEK
Department of Computer Science, Faculty of Mathematics, Physics and Informatics
Comenius University, 842 48 Bratislava, Slovakia
e-mail: forisek@dcs.fmph.uniba.sk

Abstract. We describe the process of selecting and preparing the Slovak IOI team. The presented information is related to the situation in the year 2007. We present the structure of the Slovak national Olympiad in informatics and of the Slovak selection camp, including the rationale behind their current form. We list several other national activities that help our students acquiring extra-curricular knowledge in computer science. We discuss our cooperation with other countries in the region. Finally, we give several examples of tasks used in our Olympiad.

Key words: olympiad in informatics, Slovakia, IOI, training, preparation, programming competition, programming contest.

1. National Olympiad

1.1. *Structure*

The Slovak national Olympiad in informatics (OI) is divided into two categories. The easier category B is intended for ages 15 to 16, the harder category A for ages 17 to 18. (The rules specify exact limits based on the attended class, here we provide approximate ages instead.)

In category B the contest has two rounds – a home round and a regional round. In category A, there is an additional national round. The corresponding rounds in both categories have the same structure, we will describe them below.

1.2. *Home Round*

This is a long-term round. The contestants have at least a month of time to work on their solutions. Contestants are given four theoretical tasks of algorithmic nature. Refer to Appendix A for an example task.

One of the four tasks is accompanied by a study text. This study text (usually) describes a non-traditional computational model that will be used in tasks throughout the year, i.e., in all three rounds. In the accompanied tasks the students are supposed to solve a problem within the given computation model. The actual tasks used form a graduated difficulty series – many times a task from an earlier round serves also as preparation for harder tasks in later rounds. For more details and an example refer to Appendix B.

For each of the other three tasks the contestants are supposed to find a correct algorithm that is as efficient as possible, implement it, and provide a discussion of its correctness and time complexity. All submitted solutions are evaluated by human judges and awarded points on a scale 0 to 10.

Usually about 150 students participate in this round, and all of them are invited to the regional round.

1.3. *Regional Round*

The regional round takes place in all (eight) regions of Slovakia at the same time. In each region, the students are invited to some place where they have to solve four tasks in four hour. The type of tasks is identical to the home round (i.e., three algorithmic tasks and one task related to the computational model introduced in the home round).

All four tasks are solved on paper only. After the round is over, each region evaluates the submitted solutions. All solutions are then sent to the Olympiad's national committee for coordination. During the coordination process all solutions are re-evaluated (checking for potential mistakes in the preliminary evaluation), and the same point scale is applied to all solutions.

After the coordination the regional ranking lists are merged to create a national ranking list, and based on this ranking list approximately 30 best students are invited to the national round.

1.4. *National Round*

The national round consists of two competition days. The first day is similar to the previous rounds (four and a half hours, three theoretical tasks, one of them using the computational model).

On the second day the students have four hours to implement their solutions of the two algorithmic tasks. The solutions are evaluated using automated testing and points awarded to a solution are proportional to the number of test data batches it is able to solve correctly.

Note that the second day is similar to the International Olympiad in Informatics (IOI). For readers not acquainted with the IOI contest format we recommend referring to (Pohl, 2006) and (Dagiene, 2006). Same as at the IOI, compilers from C/C++ and Pascal are available on the practice day.

For each of the days the maximum is 30 points. Based on the total results of both days, the national champions are announced, and approximately ten best participants are invited to the national selection camp where the representation for the IOI is determined.

1.5. *Contest Format Rationale*

We strongly believe that the thinking process (in other words, problem solving process) is the most important skill we want to see in our contestants. This is what they will need in their future lives, should they pick a career in computer science.

The IOI, “practice only” competition style fails to achieve this goal properly. Practical competitions are, and always will be, at least partially focused on the actual implementation and marginal issues such as debugging techniques, library knowledge, etc.

In our point of view, the IOI competition format is in some sense just a necessary evil – the number of contestants and the amount of available time make it practically impossible to evaluate the IOI in any other way.

Slovakia’s 5 million inhabitants make it a relatively small country and this is one of the factors that enable us to pick a different competition type for our national Olympiad. Therefore we still keep the competition mostly theoretical, and award points based on the thought process of the contestants. We are convinced that from a long term point of view, this competition type will actually be more beneficial for the contestants.

The contest format where the evaluation is based on ideas presented in the submitted work is also assumed to make the competition more accessible to girls. See (Fisher and Cox, 2006; Boersen and Phillipps, 2006) and (Boersen, 2006) for a discussion of related topics. We may support their theories. In 2007, out of 29 participants in the Olympiad’s national round, four were girls, the average in past years is about two. Many more girls take part in the correspondence seminar (described below).

One more note on the IOI competition itself: With a strong theoretical background it is much easier for the contestants to subsequently adapt to the IOI competition style – they are halfway there, all they need is to practice implementing their ideas in an efficient and correct way (see also appendix B for rationale on including tasks in non-traditional computation models).

2. National Training

2.1. *Selection Camp*

The national selection camp is a camp organized for approximately ten best participants of the national round. The program of the camp is almost entirely focused on practical programming. During a typical day, there is a half-hour warm-up session and a three hour coding session in the morning, a four hour coding session in the afternoon, and a presentation of sample solutions thereafter.

In the warm-up sessions the contestants have half an hour to solve two simple theoretical problems on paper. The coding sessions try to mimic the IOI environment as much as possible. The set of tasks used involves a cross-section of past competition tasks used in national and international competitions. The task topics are selected in such a way that they cover most topics that tend to appear at the IOI.

Based on the total results of this selection camp the four best contestants are selected as the IOI team, four contestants are selected as the Central European Olympiad in Informatics (CEOI) team and six contestants (including the IOI team) are selected for the Czech–Polish–Slovak preparation camp (CPSPC, see below).

For CEOI, it is our usual practice to treat it as a practice competition before the IOI. Thus, last-year students that have already taken part in an international competition are usually not eligible for the CEOI.

2.2. *Correspondence Seminar*

For more than 25 years, the students at Comenius University in Bratislava organize a correspondence competition for secondary school students. The original motivation for starting this competition was the lack of qualified informatics teachers in places other than the largest three or four cities. In this way, all talented students are able to acquire extra-curricular knowledge, references to study materials, etc. And last but not least, the correspondence seminar plays a vital role in making these students interested in computer science and enabling them to choose a career in this field.

After the 25 years have passed, we have to sadly conclude that the main motivation behind having the correspondence seminar is still here. With the level of salaries at secondary schools, the number of good informatics teachers decreases at a steady pace. Thus the correspondence competition still plays a vital role in preparing our talented students for the IOI.

The competition consists of four series spread throughout the year. There are three different categories; we can roughly describe them as "beginner", "ordinary", and "extreme". The last category had roughly 15 participants in the last year, and almost all contestants that qualified for this year's selection camp were previously solving tasks of this category.

In each series, contestants in each category are given several tasks to solve. The contest mimics the home round of the national Olympiad. The difference is that the contestants receive their evaluated solutions back along with personal comments from the evaluators.

2.3. *Correspondence Seminar Camps*

Twice a year a camp is organized for about 30 best participants of the correspondence seminar. This camp takes about a week of time. An usual day consists of two series of lectures in the morning, and of sports, games, competitions (both scientific and other), and team-building activities during the rest of the day. The lectures are mostly focused on efficient algorithm design and implementations. Most of them are prepared by undergraduate university students who organize the camp, but usually there are several guest lectures held by our university lecturers. In each series of lectures two lectures of various difficulties run in parallel, to accommodate different skill levels of the participants. Much of the camp's program is focused on presenting computer science as a fun and interesting topic, and on building a community of young people interested in it.

You can find more on the importance of this community building in (Forišek and Winczer, 2006). Refer to (Bell *et al.*, 2002) for an idea of how many of the camp's activities look like, and to (Verhoeff, 1997) for a general overview of related topics.

2.4. *Programming Hatchery*

In the recent years I developed the following train metaphor: Computer science as whole and programming competitions in particular, can be regarded as a train. The train has already left the station and it is still speeding up. For new passengers, catching it and getting on is becoming harder and harder. But is there any way to help them?

Actually, there is. Use a second train. Make it slow enough for them to get on and then speed up and allow them to jump to the first train.

This is exactly what we tried to achieve by starting an e-learning portal called Programming Hatchery. The first "chapter" of this portal is a gentle introduction to the area of programming contests. Subsequent chapters that are opened later offer a variety of graduated difficulty task series accompanied by study texts. The entire portal is available in Slovak, thereby compensating for the lack of good Slovak textbooks. Most of the study texts were adapted from the knowledge accumulated during the years by the organizers of the correspondence seminar. The students are given lots of valuable resources at their disposal, while being able to choose their own pace of progress. To date, this portal has got 655 registered users.

3. International Cooperation

3.1. *Preparing Problems for the National Olympiad*

Since 1993, when Czechoslovakia was split into Slovakia and the Czech Republic, we maintain an intensive cooperation with the organizers of the Czech national Olympiad. Both contests share the same problem set. For half of the years this problem set is prepared by one nation, for the other half by the other nation.

Thanks to this cooperation we have more time to gather and prepare more interesting competition problems.

3.2. *Czech–Polish–Slovak Preparation Camp*

Since 1999 the three nations mentioned in the title organize a joint week-long camp for their best students. Usually six contestants from each country take part in the camp. The organizers prepare a set of tasks for four IOI-like competition days. The camp usually involves some sports in the afternoons, and one day-long trip.

As an interesting bit of trivia, the main reason behind the order of country names in the camp's title is that the current abbreviation CPSPC is a palindrome.

3.3. *Cooperation with the Swiss Olympiad*

The Swiss Olympiad is a much younger contest than most other national Olympiads, currently it had 12 years. To be able to keep up with the more established contests, the Swiss Olympiad started several new activities to prepare their contestants. In some cases the Slovak Olympiad organizers volunteered to help. In the last year, there were two jointly organized events.

The first of these events was a Swiss Olympiad winter camp in Davos. Roughly 20 participants of the Swiss Olympiad were invited to this event, along with 3 participants from Slovakia. The camp organizers were both from Switzerland (two) and from Slovakia (three, one of them currently at ETH Zurich).

The second event was that four Swiss contestants were invited to take part in the Slovak selection camp – which served as a week-long training for them.

4. Conclusion

We presented the most important activities related to the Slovak Olympiad in informatics. The main point in preparing our contestants for the IOI is in giving them a strong theoretical background, the ability to reason about algorithms and to express their thoughts, and only then we focus on the implementation part of the competition. We try not to forget that the competition is not a goal, but just a means to make computer science more popular, and to motivate talented students to study it.

5. Appendix A: Example of a Theoretical Task

5.1. *Problem Statement*

"The fictional mountain"

Michael and Joseph were planning a trip. Each of them took his own map and started to examine the track they agreed upon. On each of the maps, some points of the track had their altitudes displayed. After a while, Michael claimed: "That's interesting. If I only consider the altitudes on the track, it seems that we will only be going over one mountain – first up, then down." "That's nonsense!" answered Joseph. After a while they found out the reason: Michael's map was less precise, and it was missing some of the altitudes displayed on Joseph's map.

Your task is to write a program that reads the list of $N < 100000$ altitudes written on Joseph's map, and computes how many of them could have been shown on Michael's map as well.

EXAMPLE.

Input: $N = 12$, altitudes: `112 247 211 209 244 350 470 510 312 215 117 217`

Output: 9

Explanation: E.g., Michael's map could have contained these nine altitudes:

$$112, 211, 244, 350, 470, 510, 312, 215, 117.$$

Worse solutions:

* for 7 points out of 10, solve the task with the constraint $N < 1000$,

* for 4 points out of 10, solve the task with the constraint $N < 20$.

5.2. *Solutions*

This task admits multiple correct solutions with various degrees of efficiency. The simplest way is to check all 2^N subsets of the given numbers, and for each of them check whether it represents a mountain. This leads to a solution in $\mathrm{O}(N \cdot 2^N)$ time.

The efficient solutions are based on the concept of the longest increasing subsequence (LIS). Note that the mountain is a subsequence that contains first an increasing, then a decreasing part. The correct solution is to compute the length of the LIS ending at each element of the sequence, and the length of the longest decreasing subsequence beginning at each element of the sequence. (The latter is just the LIS in the reversed sequence.) Having these lengths, the answer can be computed in $\mathrm{O}(N)$ time. The lengths can be computed using one of the known LIS algorithms in either $\mathrm{O}(N^2)$, or $\mathrm{O}(N \cdot \log N)$ time. For discussion of these algorithms see (Cormen *et al.*, 1989).

There are other, less efficient solutions – e.g., one can use brute force to pick the top of the mountain and for each top compute LIS in each of the parts from scratch. This leads to a solution running in $\mathrm{O}(N^2 \cdot \log N)$ or $\mathrm{O}(N^3)$ respectively, depending on the LIS algorithm used.

6. Appendix B: Example of a Computation Model Task

6.1. *Rationale*

For the selected computation models we do not expect the contestants to have any prior knowledge. The study text accompanying the problem statements is supposed (and designed) to be their first introduction to the area in question. The text is written in such a way to be accessible to (talented) secondary school students – the model is explained as simply as it gets, parts that won't be used in tasks are omitted, etc.

The contestants' main task is to understand the study text. They have to be capable of understanding the given model, and forming a suitable mental abstraction. The tasks used in the home round usually serve as a kind of a verifier – the contestant who understood the model correctly should be able to solve them without any problems.

Occasionally, the home round task statement also helps to understand the model by pointing out some interesting concept that will be referenced in subsequent rounds.

In the following sections we will show an example justifying the claims above.

6.2. *Past Models*

In the past years, these topics from theoretical computer science were used as background for our competition tasks: deterministic Turing machines, Wang tiles (Wang, 1961), alternating Turing machines (Chandra *et al.*, 1981), Markov's registry machines, D0L-systems, a-transducers, reversible computers, and many others.

6.3. *Example Study Text (partial)*

In this section we present a part of the study text for alternating Turing machines – a computational model with non-deterministic decisions of two types, corresponding to a "parallel and" and a "parallel or" on a massively parallel machine. Note that our formulation makes the topic slightly less formal, but much more accessible.

"*The Parallelizer*"

Beyond seven mountains and six valleys, the inventor Kleofas managed to construct a wondrous machine. He decided to call it the Parallelizer. At the first glance, it was just an ordinary computer. However, there was a small twist. Under some special conditions, the Parallelizer was able to run several copies of its program in parallel, without slowing down.

Kleofas designed a simple programming language for his new machine. The language looks just as Pascal for ordinary computers, there are only a few differences. First, we don't have the function `Random`, thus for each program its computation and its result are uniquely determined.

Upon termination, each program will return an exit code: either zero (meaning that its run was successful) or one (meaning that it was not). For clarity, we will have special commands `Accept` (terminate with exit code zero) and `Reject` (terminate with exit code one).

The final change will be the commands allowing us to make parallel computations: `Both(x)` and `Some(x)`. The command `Both(x)` stops the current program and creates its two identical copies. In the first one the variable x is set to 0, in the second one it is set to 1. Both copies are executed in parallel. If both of them terminate successfully, the original program is resumed. The command `Some(x)` works in the same way, with the difference that the original program is resumed as soon as at least one copy terminated successfully.

(The rest of the study text contained a more precise definition of these two commands, and two simple example tasks with solutions and commentary on the program's execution.)

6.4. *Example Task*

Given are two strings *haystack* and *needle*. Write a program for the Parallelizer that will successfully terminate if and only if the string *needle* is a substring of the string *haystack*.

6.5. *Solution*

While really simple, the optimal solution of this task contains two important concepts: using `Some` to make a parallel search (i.e., to emulate non-determinism, but this concept is never referenced explicitly), and using `Both` for parallel verification. In both cases, the lesson learned is that $\mathrm{O}(N)$ things can be done in $\mathrm{O}(\log N)$ time by repeated forking.

Many contestants were able to define functions `Exists(x,N)` and `Forall(x,N)` that do the same as `Same` and `Both` for x in the set $0, ..., N-1$. Using these functions a solution to the given task fits on a single line. Both functions were introduced in the sample solutions of the home round, and used in the next rounds as a building brick.

References

Bell, T., I.H. Witten and M. Fellows (2002). *Computer Science Unplugged*. A special version of the book available (retrieved Jun 2007) at
`http://www.google.com/educators/activities/unpluggedTeachersDec2006.pdf`

Boersen, R., and M. Phillipps (2006). Programming contests: two innovative models from New Zealand. Presented at *Perspectives on Computer Science Competitions for (High School) Students*. Retrieved Jun 2007 from
`http://bwinf.de/competition-workshop/RevisedPapers/1_BoersenPhillipps_rev.pdf`

Boersen, R. (2006). Inclusive programming contests = Inclusive problem sets. In V. Dagiene and R. Mittermeir (Eds.), *Information Technologies at School*, pp. 535–544.

Chandra, A.K., D.C. Kozen and L.J. Stockmeyer (1981). Alternation. *Journal of the ACM*, **28**(1), 114–133.

Cormen, T., C. Leiserson and R. Rivest (1989). *Introduction to Algorithms*. MIT Press.

Dagiene, V. (2006). Information technology contests – introduction to computer science in an attractive way. *Informatics in Education*, **5**, 37–46.

Fisher, M., and A. Cox (2006). Gender and programming contests: mitigating exclusionary practices. *Informatics in Education*, **5**, 47–62.

Forišek, M., and M. Winczer (2006). Non-formal activities as scaffolding to informatics achievement. In V. Dagiene and R. Mittermeir (Eds.), *Information Technologies at School*, pp. 529–534.

Pohl, W. (2006). Computer science contests for secondary school students: approaches to classification. *Informatics in Education*, **5**, 125–132.

Verhoeff, T. (1997). The role of competitions in education. Presented at *Future World*: *Educating for the 21st Century*. Retrieved Jun 2007 from
`http://olympiads.win.tue.nl/ioi/ioi97/ffutwrld/competit.pdf`

Wang, H. (1961). Proving theorems by pattern recognition. II. *Bell System Tech. Journal*, **40**(1), 1–41.

M. Forišek is currently a graduate student at the Comenius University in Slovakia. He received a master's degree in computer science from this university in 2004. His research interests range from theoretical computer science to education of mathematics, informatics, and algorithms. He is an elected member of the International Scientific Committee of the International Olympiad in Informatics. For many years he has been an active organizer of national and international programming contests, including several years of the Internet Problem Solving Contest (IPSC) and Central European Olympiad in Informatics 2002. As a student he was a very successful participant in programming contests, he was awarded gold medals both at the IOI and at the ACM ICPC World Finals.

Olympiads in Informatics, 2007, Vol. 1, 66–78
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Olympiads in Informatics: Macedonian Experience, Needs, Challenges

Metodija JANCESKI
Faculty of Natural Sciences and Mathematics, Institute of Infomatics
Skopje, Macedonia
e-mail: meto@ii.edu.mk

Veno PACOVSKI
Civil Engineering Faculty
Partizanski Odredi bb, Skopje, Republic of Macedonia
e-mail: pacovski@gf.ukim.edu.mk

Abstract. The introductory part of the paper includes the brief description of the current Macedonian educational system, including information about national ICT policy, primary and secondary education curricula, technological equipment, the level of using ICT in education, and projects related with ICT.

Presently, there is enforced computerization of our country (an enormous increase in numbers of computers in primary and secondary schools, free education for computer illiterate and efforts are made for full computerization of every aspect of the society).

This paper is about national informatics competitions in Macedonia for secondary school students, their organization, and scope. Computer Society of Macedonia (CSM) is the organizer of National competitions. The regular activities of CSM also include: organization and execution of seminars for permanent education of IT teachers in secondary and elementary schools, and there are efforts made to influence national education of IT.

CSM organizes four levels of state annual competitions with two groups of original problems. The top ranked four contestants at the Olympiad represent Macedonia at the international competitions: BOI and IOI.

Some of the conclusions or recommendations follow:

– **state education:** IT education, especially programming is not continuous (it should be); mentors fall behind with developments in the field of programming, they need guidance, seminars, materials, all kinds of support;
– **finances:** permanent funds have to be established; companies still do not see their interest to support the competitions, it has to change;
– **materials:** there are plenty of materials, tasks with their solutions, discussions and tests; many of them are available at the IOI official site; but, there is need of further guidance in using them, even including some additional education material;
– **coordination:** the coordination at the higher level is needed, (when and how the young students should start with programming, what should be their first programming language, Pascal, C, C++, C# or Java).

Key words: education, competition, programming, computerization, olympiad in informatics, Macedonia.

1. Brief Description of the Macedonian Educational System

1.1. *General Data*

The Republic of Macedonia has a territory of 25.713km^2 and population of 2,022,547 citizens, according to the last census.

The school education system in Macedonia has two levels – primary (grades 1 to 8, until this year and Grades 1 to 9 from the next school year) and secondary (grades 1 to 4) each culminating with a state certificate, namely, the certificate for completion of primary education and the secondary school diploma. Primary education consists of two stages – Elementary (grades 1 to 4) and lower secondary (grades 5 to 8). Children start their initial education at ages six or seven. Under Macedonia's constitution, primary education is compulsory, and there are efforts for secondary school to be compulsory by next school year.

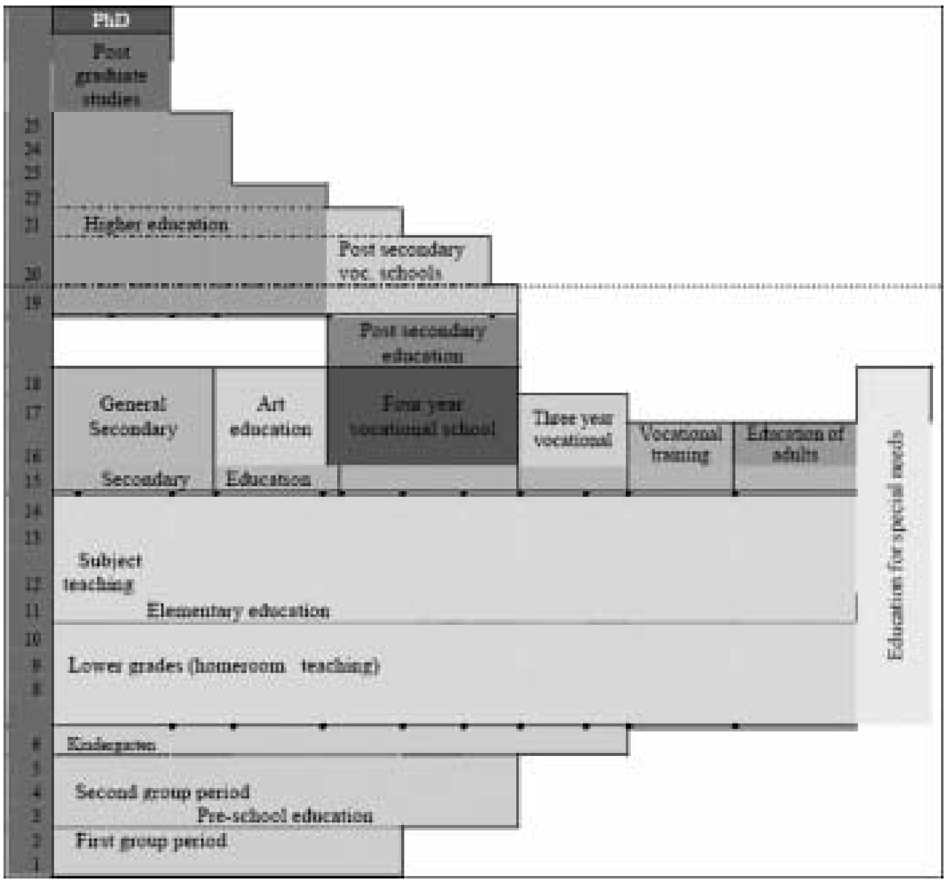


Fig. 1. Graphic display of the system of education in the Republic of Macedonia.

The basic educational level is uniform countrywide, both in terms of curriculum and school-leaving certificates. The secondary level of education consists of two educational branches – general education and vocational education. Duration of study in general education is 4 years, and in vocational education – 3 or 4 years.

There are 341 primary schools and 92 secondary schools in total in Macedonia, with approximately 330,000 primary and secondary school students.

Three strategic national documents in the field of education and information society are the "National program for the development of education 2005–2015", the "National Strategy for development of Information Society" and the "National Strategy for Development of Electronic Communications and Information Technologies". These strategies are expected to be a base for drafting policies that will define all aspects of the information society in Macedonia.

1.2. *"National Program for the Development of Education 2005–2015"*

In April 2006 The Parliament of the Republic of Macedonia has accepted the "National Program for the development of education 2005–2015". It sets out a roadmap to ensure coherent and systematic reform towards decentralization in vocational education and training. Also, at the same time, the following accompanying program documents were accepted:

- program for development of pre-school education;
- program for development of primary education;
- program for development of secondary and post-secondary education;
- program for development of higher education;
- program for development of ICT in education;
- program for professional development of the education staff;
- program for development of institutional reforms;
- program for provision and quality control of education;
- European Language Portfolio.

1.3. *"National Strategy for Information Society Development" and "National Strategy for Development of Electronic Communications and Information Technologies"*

In 2005, as a result of the partnership between Government, civil and business sector, the "National Strategy for Information Society Development" with accompanied "Action plan" were drafted and adopted with wide consensus. They define the measures related to the basic infrastructural foundation for information society which are necessary for implementation and development of the basic postulates of information society (e-governance, e-education, e-business, etc.), and the ultimate goal of the strategy is the goal we are all striving at – improving the quality of living in Macedonia.

The next step in the development of information society is drafting the "National Strategy for Development of Electronic Communications with Information Technologies" that will be fundamental document that will define the infrastructure of the information and communication technologies.

1.4. *Trends in Macedonian Education*

As for education, the general approach in Macedonian schools is the traditional one. We have to reflect attention to, and understanding of, some of the particular challenges facing educators in the educational community – lack of student motivation and preparation for the rigor of learning different subjects, accompanied by a lack of appreciation of the value and utility of the kind of skills acquired in introductory courses. There is a need for digital content and free access to digital libraries. Primary and secondary educators are not satisfies with the level of communication and coordination with the universities. They need organized and traditional professional training and other forms of permanent education, including educational electronic or printed magazines, as good examples of additional learning material.

In accordance with the best European and world educational models, practices and standards, and accepting the basic concepts of the modern national educational policies in the developed countries: long life learning, student centered education, education as the key for economical success, learning society etc., we have to emphasize:

1. We have to get students motivated by connecting what they do in the classroom to the world in which they live and work, to help students gain a deeper understanding of the world of sciences through a series of interactive lectures, guided discussions, and to help students realize that what they are learning will help them tackle problems in other courses and give them a “can do” attitude that will serve them well in all types of careers.
2. The next challenge is the real need for increasing the usage of the ICT in different subject education. It includes various multimedia tools, Internet, digital materials, etc. The educators and students prefer labs instead class-rooms. It is also worth including some forms of distance education in our schools.
3. There is also a need for rapid and widespread dissemination of “best practices” in Math and Science education and a need for dissemination mechanisms that include thoughtful reflection on the intellectual and pedagogical basis for such innovations and mechanisms to aid others in adapting those new approaches in the classroom and lab.
4. We need to force active-engagement methods that improve learning and situations where Math and Science students work with peers to make predictions, help construct knowledge and solve regular, and more complex problems.

2. Accelerated Computerization of the Macedonian Schools

Thanks to the former Macedonian President Boris Trajkovski and the donation of the Government of the People’s Republic of China (PRC), 5300 computers were obtained and installed in Macedonian primary and secondary schools in the period 2003–2005.

In that context, during the last couple of years USAID became one of the main actors in the education reform and transformation process in Macedonia. The result of USAID involvement in this process was establishing the following projects: "Macedonia Connects", "E-school" and "Primary education project".

"Macedonia Connects project" enabled broadband Internet access to be readily available and affordable throughout the country and facilitated its use by all sectors of society, besides schools. "Macedonia Connects" has resulted in Macedonia becoming the first all-wireless internet country in the world.

"E-Schools Project" resulted in creation of 460 computer labs with PRC computers in all primary and secondary schools. Also, a series of training programs were conducted for most of the secondary and primary school teachers, focusing on use of ICT through project-based learning strategies and networking. One of the main "e-School project" activities was the translation and adaptation of the software package titled "ToolKID", based on Comenius Logo, for use by K-4 children. This educational software, together with four manuals was donated to 100 primary schools throughout Macedonia. Besides, a large number of teachers passed the training for using this software. This process continues.

As was described before, the PRC donation and USAID projects were the main factors for initialization of the process of computerization in our schools.

Since the last elections, the new Government has made outstanding efforts to accelerate the computerization of our schools. They tried to establish a new Ministry for IT affairs, but unfortunately, it was not accepted by the opposition parties, who prefer IT Agency, instead Ministry. However, a new Minister for Informatics Society was included in the Government.

For the first time in Macedonia, the subject "Informatics" will be obligatory in primary schools by the 2007/2008 year. Besides, as the result of the achievements of the former and current projects in this field, as well as the newest government efforts for full computerization of Macedonian society (an enormous increase in numbers of computers in primary and secondary schools, free education for computer illiterate, etc.), we are witnessed the increased level of implementation of ICT in the Macedonian education.

One of the last initiatives of the Government in this field was establishing the Council for implementation of ICT in primary and secondary school in Macedonia. The members of this Council are people from: universities, schools, actual of former IT projects, IT companies, and other governmental and nongovernmental organizations. The main areas of work of this Council are: computer installation and maintenance, creating digital materials, as well as teacher ICT training.

This year, the Government is promoting two very important IT projects: "Computer for each child Project", and "Free Internet for all citizens Project".

In the framework of the first project, the Government has a very ambitious computer procurement plan: 50.000 computers (2007), 100.000 computers (2008 and 2009). In this year (2007) budget, 10 million euros were predicted for this procurement.

Outside of education, the Government is making efforts for full computerization of the Macedonian society. It is worth mentioning the project "Electronic health personal file for each citizen", as one of the EU requirements, and the project in collaboration with Ministry of justice, for creating a system for court's document management.

There is serious lack of IT staff in Macedonian administration. All unemployed IT experts that apply, will be accepted. They could be a part of the Macedonian history in the process of creating the informatics society. After opening the new University in Shtip, this year, it is expected more than 2.000 IT students to enroll in our universities annually. The additional impact will be 125 grants for IT students that will be approved by the Government, annually.

One of the observed options to increase the number of IT experts is organization of IT trainings for the prequalification of the unemployed graduated electrical engineers and other technician experts, by the Government and the private sector.

Many dilemmas, such as "laptops or desktop computers", or "free and open source software or proprietary software", need to be overcome. Taking into account the lack of proper school electricity infrastructure and the lack of the proper protection from thefts and damages in the past, security, insurance, and maintenance of the computers still remain the issues to be considered. The other questions that remain in the new, large-scale situation are the following: motivation of the teachers and students to use the increased ICT capacity, awareness for the computer as student's need, digital content according Macedonian curricula, fast and high quality internet access, etc.

The two important initiatives to take into consideration as support in the computerization of the schools are: the USAID's "Primary Education Project" (PEP) and **"**Portal for primary and secondary schools".

The "Primary Education Project" is the latest USAID's initiative targeting all public primary schools in Macedonia. PEP seeks to improve the quality of instruction and increase employment skills in youth. This project is a continuation of a cluster of USAID-funded projects to strengthen education in Macedonia and support decentralization efforts, as Macedonia seeks entrance to the European Union. This project has four components: renovate select schools and improve energy efficiency; increase access to and improve use of information technology; improve math and science education; and improve student assessment.

The educational portal was funded by the USAID, and prepared and maintained by both the "Macedonia Connects Project" and the "E-School Project". From the beginning of the 2007/2008 year, this portal will be responsibility of the Ministry of education and science.

This portal will link the teachers and the students from all primary and secondary educational institutions in a virtual working environment. The education portal would provide on-line resources for the school directors, teachers and the students and would enable them to easily share their experiences and practices. The education portal will mark the beginning of a new quality of the education system in Macedonia, thus bringing it closer to the modern education trends and practices in the world.

The goal of this portal is to enable public access to all information regarding the secondary and primary schools and a wide range of services for the students and the teachers.

Through this portal, the visitors have access to the list of all schools and their contact information. All secondary and primary schools have the opportunity to create their own web page which will be hosted on the portal, and all students and teachers will be able to get an e-mail address.

The on-line collaboration and exchange of information and experiences of all teachers and students from primary and secondary schools and the educational system institutions are enabled through the discussion forums which are already functioning on the portal. Other functionalities, such as the calendar and the library of uploaded documents, are under construction.

Pedagogical, psychological, methodological, social and other aspects of this process of computerization are also important. It is not just a question of developing basic computer literacy, but also of improving teaching and learning, improving curricula and teaching methods, improving assessment, and promoting: critical thinking, active inquiry-based learning, problem-based learning, student-centered learning, etc., that will support building of the 21 century student skills.

3. COMPUTER SOCIETY OF MACEDONIA (CSM)

On the initiative of a group of professors at the Institute of Informatics, Faculty of Natural Sciences and Mathematics, and Ss. Cyril and Methodius University – Skopje, in 2000 in Ohrid the Computer Society of Macedonia – CSM (or Association of IT teachers in Republic of Macedonia) was formed. CSM continued the activities of the computer scientists which, until 2000, were part of the Association of mathematicians and computer scientists of Macedonia.

One of CSM's primary efforts is to promote the informatics society in Macedonia. Members of CSM are people from different profiles: enthusiast computer scientists, teachers in elementary and secondary schools, professors and teaching assistants at universities in Macedonia, and others.

The main goals of CSM are:

1) establishment, promotion and advancement of informatics, as well as its application;
2) implementation of informatics in all parts of society, especially in education;
3) organization of competitions in informatics for secondary school students and college and university students.

The first steps towards implementation of informatics in Macedonia were made in the early 1980s. In 1990 in Prilep, the first state competition of informatics for secondary school students was held. Three years later, the local competitions started to take place regularly. This year both the 18th National Competition and the 15th Local Competition took place.

In 1997, The Macedonian Olympiad of Informatics (MOI) was established as an annual event. This year the 11th MOI took place.

As Macedonian representatives, teams of young computer scientists each year participate in the Balkan Olympiad of Informatics (BOI) and in the International Olympiad of Informatics (IOI).

Today CSM successfully works on the aimed goals, and its membership continues to grow.

Further on, some of the regular activities of CSM will be mentioned. They include: organization and execution of seminars for permanent education of IT teachers in secondary and elementary schools and national and international competition of informatics.

3.1. *Seminars for Informatics Teachers*

Since 2000, the seminars for primary and secondary school informatics teachers take place annually. Generally, the seminars are held during the CSM annual meetings or the national competitions.

CSM has a long term experience in organizing trainings for IT teachers in the framework of their permanent and life-long education. All these years after being established in 2000, CSM is the main organizer of such trainings and workshops that include the following topics: didactics and methodology in teaching computer science, interactions in education, preparing IT lessons, implementation of ICT in the primary and secondary schools in Macedonia and different software packages (FrontPage, Dream Weaver, Photoshop, PageMaker, Microsoft Word, Microsoft Excel, PowerPoint, etc.). Most of these trainings were held in the lecture rooms and labs of the Institute of informatics under the Faculty of natural science. CSM is the link between the Institute of informatics, an institution with 20 years experience in the field of IT teachers' education and IT teachers after their graduation.

The following lists some of the lectures that were organized by CSM.

- "Report on the survey in the Primary and Secondary schools in Macedonia about the state of computer equipment and informatics courses in the process of education".
- "Current issues about the state of informatics courses in the Macedonian schools",
- "Current states of the informatics instruction in the primary and secondary schools in Macedonia",
- "National program for the development of education in Republic of Macedonia 2005–2015",
- "Indicators of applicability and the impact of ICT in the education",
- "Some aspects about the informatics courses and IT teacher preparing for instruction".

Also, CSM organized several workshops for different software packages (Power Point, Front Page, Corel Draw, PhotoShop, Front Page, and Page Maker).

The most significant activity in 2005 and 2006 was the procurement of the free annual distance learning program "Administering computer networks" for 219 informatics teachers from primary and secondary schools in the Republic of Macedonia, in the period between 30th September 2005 and 30th September 2006. This training, which consisted of 35 Soft Skill courses, was organized by Clear View and financed by USAID, at the initiative of Computer Science of Macedonia.

The training "Creating and Maintaining Web Pages with MAMBO Tool" for all IT teachers from the secondary schools was organized and realized in two phases: training of the core group of 21 trainers, and training for all IT teachers from the secondary schools in Republic of Macedonia. The partner organizations for these trainings were "World Learning", "Macedonia Connects" and CSM. In May 2007, the same training was organized for over 700 primary school teachers, most of them IT teachers.

3.2. *Competitions of Informatics*

CSM is the sole organizer of competitions of informatics for secondary school students in Macedonia. There are four levels of domestic competitions held annually from February to May:

- electronic competitions (over 200 contestants),
- local competitions (150 contestants, in average),
- national competition (50 to 75), and
- Macedonian Olympiad of Informatics (15–20).

According to the total of the results of all the competitions, the best four contestants are selected and they represent our country at the international competitions.

By the year of 2005, there already have been held:

- 6 electronic competitions of informatics,
- 15 local competitions of informatics,
- 18 national competitions of informatics,
- 11 Macedonian Olympiads of Informatics – MOI.

Each year the Macedonian team, consisting of Leader (usually a university professor), Deputy Leader (a professor or teaching assistant) and four contestants that were the best at the domestic competitions (electronic, local, national and MOI) takes part in BOI and IOI.

It is worth mentioning that Macedonia participated for the first time at the 3rd BOI, 1995 in Bulgaria, and at the 8th IOI, 1995 in Hungary. Macedonia organized BOI 2000 in Ohrid.

3.3. *Other Activities*

Some of the activities of CSM address the curricula and syllabi in education, with the purpose of improvement and more successful implementation of informatics in the education and society as a whole:

- accepting the basic concepts which rest on the contemporary national educational policies in the developed countries:
  - "lifelong learning",
  - "a society that learns",
  - "education as a precondition for economic success";
- improvement of the information system;
- enhancement of the level of informatics knowledge of teachers and students;
- increased application of ICT in education;

- management and coordination of projects.

At this year's annual meeting a decision was made to update the membership of CSM. According to the approved plan for 2007, the main activities of CSM in the following period of time will be:

- participation in the 15th BOI in Moldavia,
- participation in the 19th IOI in Croatia,
- organization of seminars and training for permanent education of the IT teachers,
- dealing with the issues teachers meet in everyday working environment,
- active participation in future activities concerning the formation of the Chamber of educational staff in RM, which will unite all teachers' associations, societies and federations in Macedonia,
- taking an initiative for organizing informatics competitions for the primary school students,
- enabling free digital materials for IT teachers for their further (long life, permanent, and continual) education
- introduction of informatics and increasing of the number of classes in informatics an all levels in education, from preschool to university,
- reactivation of BIT^{+} – magazine for affirmation of informatics.

Nowadays, there is a process of enforced computerization of our country (an enormous increase in numbers of computers in primary and secondary schools, free education for computer illiterates, and efforts are made for full computerization of every aspect of our society). The question remains how to use all these resources for producing better contestants, thus promoting IOI concepts and ideas.

4. IOI, Plenty of Materials, Reorganization, Responsibility

IOI has accumulated a wealth of knowledge and experience during almost 20 years of existence. The decision has to be made what to do with it, and what is the best way for using it.

A logical thing to do would be further dissemination and coordination with all the participant countries in order to achieve better results and further improve programming skills of contestants.

The latest project of the Computer Society of Macedonia is the mass process of collecting and indexing various valuable electronic materials (electronic books, essays, papers, multimedia tools, programs, etc.) accessible on the Web, by IT teachers in three different areas: programming, informatics education, and computer science. There is a team, established by CSM, responsible for accepting these materials from the teachers, that selects the most appropriate ones, classifies them, in order to make them accessible for all IT teachers. The common rules we respect during this process is meticulous care for copyrights and intellectual property, and prioritizing the free materials. The fact that each material contains information for the teacher(s) who "discover" it on the Web, displays the effort of each volunteer and their contribution to the process. Motivating teachers to do the best, team work is enhanced. All teachers are given the chance to express themselves thus building common values.

The positive experiences of this micro-level project (i.e., Macedonian experiment) encourage its recommendation to be practiced on the macro (i.e., IOI) level, as well as proposing the following improvements in further IOI society functions:

- IOI recommending continuing IT education to all IOI country Governmental educational and IT departments and to contribute with recommendation in the following fields: IT curricula, implementation of ICT in all subjects, teachers preparation and professional development;
- each country (responsible person to be announced, for example, each team leader in IOI 2007) should be responsible for the following things:
  - ◦ to send profile of the main organizer/organization of the national competitions, including: structure, funding, main activities, objectives etc.,
  - ◦ brief explanation of the national competitions system (team, coverage, selection process, trainings etc.),
  - ◦ to send a brief reports for all domestic competitions annually, before the IOI event, including students involved, results, tasks (task texts, different solutions in a certain programming language, additional explanations, and tests; all of these in English),
  - ◦ to send links to learning materials or information used during the competition preparation;
- eIOI establishing a special organizational team (body) which will analyze all this data and make efforts to:
  - ◦ organize this data,
  - ◦ make a selection of offered materials,
  - ◦ produce practical methodological manuals divided in chapters for all programming areas (graphs, dynamics programming, theory of numbers, etc.) for both, teachers and students, including theoretical parts, exercises, tasks, different solutions, and comparing solutions; going step by step from the easiest tasks to the most complicated. These manuals are supposed to help the teachers to overcome the gap between them and contestants thus enabling them to constantly upgrade their knowledge. It will also enable them to follow modern trends and use freely the most competent books for competitions preparations.
  - ◦ establish digital library of all previous IOI competitions;
- edeveloping an IOI distance learning education system for both teachers and contestants.

5. Conclusions

IOI has to think about extending the IOI competition platform with other forms of competition; junior Olympiads in programming, competitions such as IT projects, IT quizzes, creating Web sites and blogs, competitions in other languages (e.g., Logo), etc.

The afore mentioned proposals will ensure the gap between the developed and developing countries can be overcome. We have to create such situation with equal starting positions for each country, to create an excellent working environment for students and contestants, to enable IOI members to be proud of their involvement and to motivate them to increase the level of contribution in the future, thus spreading the IOI culture.

Here are some of the conclusions or recommendations:

1. state education: IT education, especially programming is not continuous (it should be); mentors can not follow on their own, they need guidance, seminars, materials, all kinds of support;
2. finances: permanent funds have to be established; companies still do not see their interest to support the competitions, it has to change;
3. materials: there are plenty of materials, tasks, with their solutions, discussions and tests; many of them are available at the IOI official site; but, there is need of further guidance in using them, even including some additional education material (see the comments above);
4. coordination: the coordination on the higher level is needed, (when and how the young students should start with programming, what should be their first programming language, Pascal, C, C++, C# or Java).

Appendix. Macedonian achievements at Balkan and International Olympiads of Informatics (BOI and IOI)

Table 1
Macedonian achievements at BOI, so far

3rd BOI 1995, Varna, Bulgaria	Zarko Aleksovski – bronze medal
4th BOI 1996, Nikozia, Cyprus	Zarko Aleksovski – silver medal
	Igor Trajkovski – bronze medal
5th BOI 1997, Drama, Greece	Zarko Aleksovski – silver medal
	Vladica Sark – bronze medal
6th BOI 1998, Ankara, Turkey	Vladica Sark – bronze medal
12th BOI 2004, Plovdiv, Bulgaria	Nikola Postolov – bronze medal
13th BOI 2005, Rodos, Greece	Nikola Postolov – bronze medal
14th BOI 2006, Nikosia, Cyprus	Dimitar Mishev – bronze medal

Table 2
Macedonian achievements at IOI, so far

8th IOI 1996, Vezsprem, Hungary	Andrej Bogdanov, bronze medal
	Zharko Aleksovski, bronze medal
9th IOI 1997, Capetown, South African Republic	Zharko Aleksovski, bronze medal
15th IOI 2003, Kenosha, USA	Nikola Postolov, bronze medal

References

National report for development of education in the Republic of Macedonia 2001–2004, Ministry of education and science; Bureau for development of education.
`http://www.ibe.unesco.org/international/ICE47/English/Natreps/reports/macedonia.pdf`

Council for implementation of ICT in education Web site.
`http://www.mon.gov.mk/ikt`

National Program for the Development of Education in the Republic of Macedonia 2005–2015.
`http://www.npro.edu.mk/english/index-en.htm`

National Strategy for Development of Electronic Communications and Information Technologies.
`http://www.metamorphosis.org.mk/content/view/848/57/lang,en/`

National Strategy for Information Society Development and Action Plan of the Republic of Macedonia.
`http://www.kit.gov.mk/WBStorage/Files/Strategy%20and%20Action%20Plan%20ENG.pdf`

USAID Macedonia press releases.
`http://macedonia.usaid.gov/english/EDU/Macedonia_Connects.htm`
`http://macedonia.usaid.gov/english/EDU/E-schools_eng.htm`
`http://macedonia.usaid.gov/English/EDU/pep.htm`
`http://english.schools.edu.mk/`

M. Janceski (1965), master of IT sciences, teaching/research assistant at the Institute of informatics under the Faculty of natural sciences and mathematics in Skopje, president of the Computer Society of Macedonia since 2004, member of the Governmental Council for implementation of ICT in Macedonian schools since 2007. The leader of the Macedonian team on 6 BOI and 3 IOI, since 1998. Consultant and trainer in the crucial national projects in the field of IT and informatics.

V. Pacovski (1965) holds a Ph.D. in Computer Science, at the moment works at the Faculty of Civil Engineering; main field of work is Information Retrieval; has an extensive programming experience in database design and various programming languages, including Delphi as a RAD tool; Deputy leader at BOI '96 and IOI '05.

Olympiads in Informatics, 2007, Vol. 1, 79–89
© 2007 *Institute of Mathematics and Informatics, Vilnius*

New Task Types at the Canadian Computing Competition

Graeme KEMKES
Department of Combinatorics and Optimization, University of Waterloo
200 University Ave West, Waterloo, Ontario, N2L 3G1 CANADA
e-mail: gdkemkes@waterloo.ca

Gordon CORMACK, Ian MUNRO, Troy VASIGA
David R. Cheriton School of Computer Science, University of Waterloo
200 University Ave West, Waterloo, Ontario, N2L 3G1 CANADA
e-mail: {gvcormack, imunro, tmjvasiga}@cs.uwaterloo.ca

Abstract. In the 2006 competition workshop held at Dagstuhl, Germany, there were many fruitful discussions about the difficulties facing computer science competitions today. Our competitions have several purposes: to foster interest in the discipline, to create a community, and to promote achievement, for example. Balancing these various purposes may require many tradeoffs. Several participants identified areas where we need to improve our competitions. Tom Verhoeff (2006) discussed the problem of giving a meaningful ranking to incorrect solutions. Maryanne Fisher and Tony Cox (2006) pointed out that some groups of students are disadvantaged by the present format. Many participants made suggestions for improving the competitions. One of the suggestions in our paper (Cormack *et al.*, 2006) was open-ended tasks. A task is *open-ended* if there is no known optimal solution to the problem. Points are awarded for correct submissions in proportion to how well they do. A vast number of real-world applications, such as pattern recognition, information retrieval, and compiler optimization appear suitable for this purpose. At Canada's national informatics olympiad, the Canadian Computing Competition, we have been exploring several of these suggestions. In this paper we describe the experiments we have performed and we analyze whether the objectives have been achieved.

Key words: computing competitions, open-ended tasks, informatics in Canada.

1. Introduction

Computing competitions have generally been focused on determining the top students who can solve algorithmic problems. For various reasons (efficiency, quantitative bias, competitiveness) the focus has been on tasks that are known to have an optimal solution: the student must find this optimal solution to get full credit on the given task.

This narrow goal, however, should not be the only goal of computing competitions. In particular, computing competitions should also focus on other goals, if they are to serve the informatics community as a whole and cultivate continued growth in the field. These other goals should include fostering interest in the discipline in order to ensure a continued source of future informatics students; to promote achievement of *all* sorts, in order for students to have a sense that informatics is an accessible field of study; and to create a community amongst students that have an interest in the field of informatics, since most students (in the Canadian context, at least) are geographically isolated from other students who have an interest in informatics. Some of these goals have been highlighted by Verhoeff (2006) (to award students who do not have the "best" solution) and Fischer and Cox (2006) (to promote achievement amongst students who are generally disadvantaged).

One idea that the Canadian Computing Competition (CCC) has adopted to address some of these other goals of computing competitions is the use of open-ended tasks. We call a task *open-ended* if there is no known optimal solution to the task. It follows from this definition that all NP-complete problems are open-ended, but there are also problems that do not fit into the NP-complete model that also have open-ended features about them (such as text compression, spam recognition, character recognition, etc.). These latter open-ended problems tend to be focused on current research areas, and thus provide students with "real-world" problems, that can, in turn, be a motivating factor to consider the field of informatics as worthwhile to pursue in higher education.

This paper outlines the Canadian experience of computing competitions to place the "typical" computing experience of secondary school students in context. We then outline the benefits of open-ended problems as a contrast to the typical experience students have in computing competitions or computing education. Following this, we outline two open-ended tasks which have different features. We conclude this paper by analyzing the effectiveness of these particular tasks and outline issues to consider when creating other open-ended tasks.

2. The Canadian Experience

2.1. *Informatics in Canada*

In Canada, the educational system is controlled by each provincial or territorial unit (there are 10 provinces and 3 territories), and the amount of informatics education (typically called "computer studies" in Canadian secondary schools) varies greatly between regions. If there is a computer studies course in secondary schools, the curricula focuses on learning programming languages (especially at the Grade 11 level, where Java is the prevalent language of instruction). Most secondary schools have at least one computer lab which is connected to the internet. The challenge in creating a national competition is to manage the diversity of the competing students: tasks should be approachable to (almost) all levels of ability and yet challenging enough for students with a very high level of experience and ability.

2.2. *Organization of the Canadian Computing Competition*

The Canadian Computing Competition (CCC) is divided into two stages: Stage 1 and Stage 2.

The Stage 1 competition is further subdivided into two separate contests: the Junior level (intended for students with limited programming experience, typically Grade 9 or 10 students, who are typically 14 or 15 years old) and the Senior level (intended for students with more programming experience, or students who have competed in the CCC before, typically Grade 11 or 12 students, who are typically aged 16 or 17 years old). This competition is written in secondary schools across Canada, and is composed of 5 algorithmic problems to be solved in 3 hours, with the score based exclusively on the student code providing the correct output on given input. Any programming language is allowed (students have used Visual Basic, Pascal, Turing, PHP, C, C++, Java, Perl) and the contest environments are heterogeneous. The evaluation of the student programs is done by the individual teacher, and results are sent into the CCC office for final processing. The top students in five geographical regions of the country are awarded prizes.

The Stage 2 competition gathers the top 20 students who competed in the Senior competition and provides a week-long program of activities (social activities, lectures by faculty members) and two competition days (similar to the IOI: three problems per day in a three-hour time period) where the allowed programming languages are C, C++ and Pascal (the current "official" IOI programming languages). The top four competitors are selected for the Canadian IOI team, again, based on the evaluation of their programs in terms of providing the correct output on given input.

3. Open-Ended Problems

As outlined in Section 1, we define an open-ended task as one where there is no known optimal solution. In this section, we describe the benefits of this type of task in terms of addressing the computing competition goals of fostering interest in the discipline, creating a community and promoting achievement.

3.1. *Fostering Interest in the Discipline of Informatics*

One method to foster interest in the discipline is to provide students with problems that are manageable (i.e., they are able to develop at least a partial solution to them) and yet are interesting or appealing. To appeal to students, problems that they encounter should be realistic and meaningful. By asking open-ended problems, such as spam recognition or text-compression, which are both research-level problems, students can be motivated to consider computer science as an active field, where not all of the interesting problems have been solved. In other words, the problems are "real". Moreover, as with other discoveries and insights, it may be that the idea of a student that causes forward progress to be made on these problems. Finally, real data can be used to evaluate this type of task (such as real email messages, or real text that needs to be compressed), and the use of real data further underscores the applicability of computer science to real problems.

3.2. *Creating a Community*

Though the problems stated here were not posed as such, open-ended tasks can be done collaboratively. This more community-based version of the task utilizes the exploratory nature of the open-ended task, and is more likely to lead to future discussions which are more lengthy and exploratory in nature. This contrasts to the more typical closed-task, where once the optimal solution is known, there is nothing further to discuss, since all avenues of exploration have been exhausted. Furthermore, as noted by Fischer and Cox (2006), typical programming tasks may appeal to a certain subsection of the population, but with open-ended tasks, there is the potential to appeal to a broader group of students, since students with various levels of ability may be able to collaborate and create a meaningful and insightful solution.

3.3. *Promoting Achievement*

When assigning marks for open-ended tasks, evaluators can reward students on other dimensions of achievement, including the typical measurement of technical skill. For example, it is possible to measure originality of a solution (to some extent) to reward interesting use of data structures, or thought processes or consideration of a special subcase.

As well, since there is no optimal solution, there is no perfectly correct solution, and thus, there is no incorrect solution (other than solutions which do not compile, or do not meet the specifications). Therefore, “incorrect” solutions would not receive zero points, and thus, students who can follow the minimal requirements of the task can have a non-zero score and a sense of accomplishment. In particular, it is possible to give students a basic strategy which, if implemented correctly, gives students a minimum level of achievement. This ensures all students have some chance of providing a partial solution to the problem: frustration or hopelessness is lessened.

Since open-ended tasks are typically evaluated on a continuous scale, a wide diversity of achievement can be rewarded, and, as an added benefit, the probability of tie scores is significantly reduced. (The concept of continuous scoring is discussed more thoroughly in (Cormack *et al.*, 2006).)

4. Open-ended Tasks at the Canadian Computing Competition

In this section, we outline two open-ended tasks: Paint-by-Number (an NP-complete open-ended task) and Codec (a non-NP-complete open-ended task) that were used at the Canadian Computing Competition.

Codec is a problem to design and implement an algorithm to compress and decompress English text.

Paint-by-numbers is a logic puzzle (similar to Sudoku or other number-puzzle problems where a solution is a satisfying assignment to a collection of constraints) which is computationally difficult.

We first present the task descriptions as given to the students in the CCC Stage 2, then briefly discuss the student attempts and evaluate the success of these tasks. We conclude by offering some suggestions for future open-ended tasks.

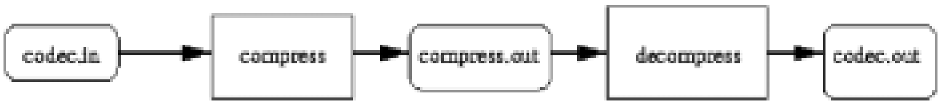
4.1. *Gordon Cormack, Codec (CCC 2006 Stage 2)*

The problem of lossless data compression is to transform some data into a compressed form such that:

(a) the original can be reproduced exactly from the compressed form.
(b) the compressed form is as small as can reasonably be achieved.

You are to write two programs – `compress` that performs lossless compression and `decompress` that reproduces the original data from the compressed form. The data to be compressed will be plain English text represented using printable ASCII characters (i.e., all characters with ASCII values between 32 and 126 inclusive). The compressed form is a string of binary bits. For convenience, we will represent this string of bits as a character string containing only 0s and 1s.

`compress` reads the original data from the file `codec.in` and writes the compressed form `compress.out.decompress` reads the compressed data from a file called `compress.out` and writes the corresponding original data to `codec.out`. Of course, `codec.in` and `codec.out` must be the same file. Pictorially, we have the following flow of information:



`compress` must output only 0s and 1s and `decompress` must exactly reverse the effects of `compress`. That is, condition (a) above must hold for any English text. If `compress` and `decompress` meet these criteria, your score will be determined by the relative size of the input and output by

$$\text{score(as\%)} = 50 \cdot \sqrt{\frac{8c - b}{c}},$$

where c is the total number of characters in the original text and b is the number of bits in the compressed form. Note that scores may exceed 100%, but scores that are less than 0 will be given 0% (i.e., no negative marks will be given, but bonus marks may be awarded).

Discussion and Hints

It is well known that any ASCII character can be represented using 8 bits. Such a representation would achieve a score of 0 using the formula above. Since there are fewer than 128 possible symbols in the input, it is possible to represent each one with 7 bits. Such a representation would receive a score of at least 50%.

A smaller representation can be achieved, with high probability, by observing that some letters are more common than others. Suppose we estimate that a character α occurs with probability p_α in a given context. The best possible code will use $-\log_2(p_\alpha)$ bits to represent that character. If one estimates p_α one can construct a *prefix* code with about $-\log_2(p_\alpha)$ bits for each character in the following manner:

- build a binary tree with one leaf for each character α;
- organize the tree so that the depth of α is approximately $-\log_2(p_\alpha)$;
- use a binary representation of the path ($0 =$ left, $1 =$ right) to represent α in the compressed data.

One way to estimate p_α is simply to compute the fraction of characters equal to α in a sample of data similar to that to be compressed. Another is to use an adaptive method, in which the data is compressed one character at a time, and the sample consists of the text already compressed. A sample of English text is available in the file `sampleText.txt`.

It is also possible to estimate p_α using the context in which it occurs; for example, in English a "q" is very likely to be followed by a "u" (e.g., quick, quack, quit, quiz, but not qiviut, which happens to be the wool of a musk-ox).

Use this information, or any other information at your disposal, to build the best Compressor and Decompressor you are able.

Input

The input to `compress` will consist of n characters ($1 \leqslant n \leqslant 1000000$), as described above.

The input to `decompress` will consist of m 0s and 1s ($1 \leqslant m \leqslant 8n$).

Output

The output of `compress` will be a sequence of 0s and 1s, with no other characters (i.e., no newline characters should be outputted).

The output of `decompress` is a sequence of at most 1000000 uppercase letters, lowercase letters, spaces, newlines and punctuation symbols.

Sample Input (to compress)

```
To be or not to be?
```

Possible Output for Sample Input (`compress`)

```
0111001000010110000100110100001100010011110000011110010000
1011011111
```

Sample Input (to `decompress`)

```
0111001000010110000100110100001100010011110000011110010000
1011011111
```

Output for Sample Input (`decompress`)

```
To be or not to be?
```

Explanation

The sample compressor systematically uses the following codes for each of the input characters:

<space>	000
o	010
n	01100
r	01101
T	01110
t	011110
?	011111
b	10
e	11

The compressed output uses these codes, as shown below:

```
0111001000010110000100110100001100010011110000011110010000101101111
TTTTTooo    bbee    ooorrrrr    nnnnnoootttttt    tttttttooo    bbee??????
```

4.2. *Paint by Numbers (Sandra Graham, CCC 2006 Stage 2)*[1]

Way back before you were born, there was a really bad craft/hobby called *paint-by-numbers*: you were given a line drawing, with numbers in each enclosed region, and the number corresponded to a particular colour. An example is shown below:

The problem you have to solve is much more linear, in a way.

You will be given an n-by-m grid ($1 \leqslant n, m \leqslant 32$) which you will “colour” in with either a dot (‘.’) or a star (‘*’).

Of course, the grid will not be specified in the usual paint-by-numbers way, since this would be too easy.

Instead, you will have to infer which cells are blank and which contain a star. The only information you will be given is a collection of $n + m$ sequences of numbers, one sequence for each row and column. The sequence will indicate the size of each continuous block of stars. There must be at least one dot between two consecutive blocks of stars.

An example is shown below (which is supposed to look fish-like):

		1	1					
		1	1	2	2	4	4	2
2	2							
	5							
	5							
2	2							

(Unsolved Puzzle)

		1	1					
		1	1	2	2	4	4	2
2	2	*	*	.	.	*	*	.
	5	.	.	*	*	*	*	*
	5	.	.	*	*	*	*	*
2	2	*	*	.	.	*	*	.

(Solved Puzzle)

[1]It has been pointed out by one of our referees that this task was used at IOI 1992 as the problem “Islands in the Sea”. The problem description can be found at `http://olympiads.win.tue.nl/ioi/ioi92/tasks92.txt`

You may notice that some paint-by-number patterns are not uniquely solvable. For example,

	1	1
1		
1		

has two solutions

	1	1
1	*	.
1	.	*

and

	1	1
1	.	*
1	*	.

For this problem, you may assume that *any* solution which satisfies the specification is correct.

You should note that 50% of the marks for this question will come from test cases where $1 <= n, m <= 6$.

Input

Input consists of a total of $n+m+2$ lines. The first line of input consists of an integer n $(1 \leqslant n \leqslant 32)$, the number of rows. The second line of input consists of an integer m $(1 \leqslant m \leqslant 32)$, the number of columns. On the next n lines, there will be sequences which describe each of the n rows (from top to bottom). Each line will contain some positive integers, with a space between adjacent integers, and the sequence will terminate with the integer 0. The next m lines describe the m columns (from left to right), the same format as the rows are specified.

Output

Output consists of n lines, each line composed of m characters, where each character is either a dot ('.') or a star ('*').

Sample Input 1

```
4
7
2 2 0
5 0
5 0
2 2 0
1 1 0
1 1 0
2 0
2 0
4 0
4 0
2 0
```

Sample Output 1 for Sample Input 1

```
**..**.
..*****
..*****
**..**.
```

Sample Input 2

```
4
4
2 1 0
3 0
3 0
1 1 0
4 0
3 0
3 0
1 0
```

Sample Output for Sample Input 2

```
**.*
***.
***.
*.*.
```

5. Conclusion

5.1. *Effectiveness of the Particular Open-Ended Tasks*

Based on the solutions provided by the competitors, despite the fact that Codec required students to create two pieces of software (the encoder and decoder) to work together correctly, it was the case that 18 of the 22 students made a serious attempt at this problem. On the other task, however, only 7 out of 22 made a serious attempt at Paint-by-Numbers.

There could be a number of factors that may explain the difference in the attempted solutions. In a typical contest, many students leave a question blank, since there are only 3 hours to complete the competition and there are 3 difficult tasks. Students must make difficult time-management decisions in order to optimize the number of points attained. Thus, rarely all students complete all problems. In the problem statement of Codec, we described a simple base-line solution, whereas for Paint-by-Number there was no base-line solution given, and even the statement that significant marks may be attained by dealing with “small” cases was not enough to encourage more students to attempt this task. Students were given an outline for the baseline solution in Codec, which they are to implement, which models more closely their typical classroom experience, unlike Paint-by-Numbers which required students to devise their own solution from scratch.

However, we feel that both problems were successful, in the sense that students were creative and implemented different approaches. For instance, in the Paint-by-Numbers, we saw submissions that used

- recursive search,
- recursive search with heuristics (such as dealing with either completely full or complete empty rows or columns),
- simulated annealing, and
- random search.

For Codec, the majority of students implemented the baseline suggested solution, but others attempted a dictionary encoding using a variety of methods, including using a prefix tree and frequency counting.

5.2. *Future Considerations and Recommendations*

We have found that open-ended tasks are a useful tool to provide interesting problems for a wide diversity of students: students at both the highest level of ability and at the lower level of ability can both have positive achievement on these tasks.

We suggest that it is important to describe a baseline solution and promise that a certain number of marks will be given for implementing this baseline solution. Providing this assurance gives students from a wide diversity an opportunity to succeed.

We found that open-ended tasks encouraged discussion amongst the students, which both helped foster interest in the discipline of informatics and provided a sense of community to the competitors.

We will be exploring the use of collaborative versions of open-ended tasks in future Stage 2 competitions (perhaps as a task outside the typical individual tasks) and we hope that our experience can be transferred to other problems, and other competition organizers can use open-ended tasks to enhance their competitions.

References

Cormack, G., G. Kemkes, I. Munro and T. Vasiga (2006). Structure, scoring and purpose of computing competitions. *Informatics in Education*, **5**, 1–22.

Fisher, M., and A. Cox (2006). Gender and programming contests: mitigating exclusionary practices. *Informatics in Education*, **5**, 47–62.

Verhoeff, T. (2006). The IOI is (not) a science olympiad. *Informatics in Education*, **5**, 147–159.

G.V. Cormack is a professor in the David R. Cheriton School of Computer Science, University of Waterloo. Cormack has coached Waterloo's International Collegiate Programming Contest team, qualifying ten consecutive years for the ICPC World Championship, placing eight times in the top five, and winning once. He is a member of the Canadian Computing Competition problem selection committee. He is currently an elected member of the IOI Scientific Committee. Cormack's research interests include information storage and retrieval, and programming language design and implementation.

G. Kemkes has participated in computing contests as a contestant, coach, and organizer. After winning a bronze medal at the IOI and two gold medals at the ACM ICPC, he later led and coached the Canadian IOI team. He has served on the program committee for Canada's national informatics olympiad, the Canadian Computing Competition. Kemkes is currently writing his PhD thesis on random graphs in the Department of Combinatorics & Optimization, University of Waterloo.

I. Munro is professor of computer science and Canada Research Chair in algorithm design, at the University of Waterloo. His research has concentrated on the efficiency of algorithms and data structures. He has served the International Scientific Committee of the IOI as well as on the editorial boards of CACM, Inf & Comp, and B.I.T., and the program committees of most of the major conferences in his area. He was elected fellow of the Royal Society of Canada in 2003.

T. Vasiga is a lecturer in the David R. Cheriton School of Computer Science at the University of Waterloo. He is also the director of the Canadian Computing Competition, which is a competition for secondary students across Canada, and has been the delegation leader for the Canadian Team at the International Olympiad in Informatics.

Olympiads in Informatics, 2007, Vol. 1, 90–104
© 2007 *Institute of Mathematics and Informatics, Vilnius*

The Modern Contents of the Russian National Olympiads in Informatics

Vladimir M. KIRYUKHIN
Moscow Physical Engineering Institute (State University)
31 Kashirskoe Shosse, Moscow 115409, Russian Federation
e-mail: vkiryukhin@nmg.ru

Abstract. The Russian Olympiads in Informatics (RusOI) had begun in 1988 a year before The International Olympiads in Informatics (IOI) started. Since the first Olympiad the most important issue was to build up understanding of the RusOI contents in particular for students of secondary school. This paper describes main stages of development of the RusOI contents, connecting it with requirements to studying informatics at secondary schools and the modern RusOI contents, allowing to select talented young people in the country and to create for them the necessary conditions for their further development in the field of computer science.

Key words: computer science, secondary school education, olympiads in informatics, contents of the olympiads, competition tasks.

1. Introduction

The contents of any Olympiads in informatics are entirely determined by those competition problems which are offered in Olympiads. When such Olympiads had started in the Soviet Union in 1988 (in that time Russia was one of the Republics of the Soviet Union) there was no experience of composing competition problems. The closer from the point of view of competition problems was model of Olympiads on mathematics. Namely it has been put in a basis of the contents of competition tasks on informatics.

Since then more than 19 years have passed, but at this time much has changed in definition of the contents of the Russian Olympiads in Informatics (RusOI). The form of carrying out the Olympiads has changed, the computer equipment was improved, there were new information technologies, and together with this, the RusOI contents was gradually formed and the technique and technology of development competition tasks were improved. Now for participants of modern RusOI the first competition tasks represent not such the big interest from the point of view of the contents, but they are those tasks which have determined an initial point and a vector of their further development.

2. Stages of Development of the Contents of Olympiads in Informatics

The development of the RusOI contents had evolutionary character and nevertheless it is possible to point to three basic stages of it [1]. The first stage has been connected with the pioneer Olympiads when there was the Soviet Union. Then Olympiads passed in two competition days. The first competition day was theoretical (without use of computers) and the second one was practical (with use of computers). The first competition tasks were very close under the formulations to mathematical tasks and more reminded tasks of the raised complexity at school subject of informatics. The basic problems by their preparation were the problems connected with definition of basic distinctions between the competition tasks of theoretical and practical competition days, the representation form of participant solutions and criteria of their evaluation.

It was obvious, that on computer competition day the contestant should be required to produce only a single source file. That should be the competition task content of theoretical competition day and what should be produced by contestants as a result of solving of theoretical tasks, was rather vaguely. It was not clear, what kind of competition tasks and algorithms will be accessible to participants and that they should represent as a solution of this kind of tasks. To demand the proof of any properties of algorithm from participants would be not absolutely correct as not many secondary school students possessed corresponding mathematical skills, and in school informatics it practically was not considered.

For that case it was accepted the following decision. On theoretical competition day it was authorized to describe algorithms of the tasks solution both on the native language and on any algorithmic language known to the students. But there was a big problem how to evaluate theoretical tasks solution. Opinions were so much, how many was judges, but any more or less comprehensible solution for this problem was not. All of them had subjective character as without the participant it was very difficult to understand that has been presented as the solution. Moreover, if it was the description of algorithm in any programming language in the written form, it was not clear how to concern to syntactic errors available there.

The RusOI contents has essentially changed in 1990. From this year theoretical competition day has been replaced by computer competition day, that is, both competition days became with using computer. From this year the second stage in development of the RusOI contents had begun. Its basic feature – competition tasks became multilevel and participants were given only one task in the competition day. In particular, each such task contained some subtasks of a various degree of the complexity, incorporated by the common idea and located in ascending order of complexities. The main advantages of multilevel tasks were the opportunity to differentiate participants on a level of preparation and minimization of volume of the text of task statements. Thus weak participants could something solve and award by some points, and strong participants had an opportunity to prove to the full.

Use of only one multilevel task on each competition day has led to understanding that small number of competition tasks narrows a scope of computer science topics considered on Olympiads. Really, practically not probably to compose one or two tasks for which solution could capture many topics of computer science. In this connection, the number of tasks on each competition round had been increased gradually (see Table 1), and from 1995 up to now in Russia there are three competition tasks on each competition day. The same tendency was on the IOI.

Table 1

Number of the RusOI tasks from 1990 to 1995

Year of RusOI	Number of the first round tasks	Number of the second round tasks
1990	2	1
1991	1	1
1992	1	1
1993	1	2
1994	2	1
1995	3	3

It is necessary to note also, that at the second stage in development of the RusOI contents there were shown up restrictions on dimension in task statements. If before a competition task were evaluated only by means of small dimension tests then occurrence of more perfect computer equipment and information technology has led to more perfective methods of evaluation of participant solutions. It had became to evaluate efficiency of more stronger algorithms and gifted participants could prove themselves on Olympiad from the best side.

The third stage in development of the RusOI contents has come in 1996 and proceeds to this day. This stage is connected with the advent of the automated evaluating systems. Use of such systems has essentially expanded the contents of Olympiads in informatics as there was an opportunity quicker and objective to evaluate solutions of participants. In particular, it had became to use three types of competition tasks: Batch tasks, Reactive tasks and Output-tasks.

The important feature of these tasks is the following. They are multilevel tasks too, but they do not consist of subtasks in task statement. They assume to use solution algorithms of various complexity which depend on dimension of the task. The smaller test case requires more simple solution while the bigger test case requires rather complex solution. Thus it is important, that are evaluated not only full solution, but also partial. It enables to evaluate a level of creative development of participants, instead of a level of computer science acquisition and training that is characteristic for competitions for higher school students.

3. Connection of Russian State Educational Standard in Informatics and the RusOI Contents

In Russia the Olympiad in Informatics for secondary school students is the official action of the Ministry of Education and Science and enters into system of certification of students. It includes five levels (school, municipal, regional, federal district and final), and students of the final stage acquire the right to enter to the university or institute without passing an examination. Such situation has influenced on the RusOI contents. On the one hand, it should meet all requirements shown by the Russian state educational standard in informatics. On the other hand, it should promote search of talented secondary school students and creation of conditions for their further accelerated perfection in the field of computer science.

Now in Russia the basic document determining the contents of secondary school education in informatics is the state educational standard in informatics accepted according to the decision of the Government of Russia in 2001. According to this document teaching of informatics at the secondary school is directed on achievement of the following purposes [2]:

- familiarization and ordering of the knowledge concerning with mathematical objects of computer science; construction of descriptions of objects and the processes, allowing to carry out their computer modeling; means of modeling; information processes in biological, technological and social systems;
- acquisition of skills to describe mathematical objects of computer science, including logic formulas and programs in the formal language, to create programs in the programming language under their description; to use program tools and to adjust them for needs of the use;
- development of algorithmic thinking, abilities to formalization, elements of system thinking;
- education of feeling of the responsibility for results of the work; formation of intention on positive social activity in an information society, on inadmissibility of the actions breaking legal, ethical standards of work with the information;
- getting of experience of design activity, creation, editing, registration, preservation, transfer of information objects of various type by means of modern software; implementation of computer models, collective realization of information projects, an information work in the various spheres demanded on a labour market.

It is important to note, that the same purposes characterize RusOI. However many teachers till now consider, that the contents of Olympiads in informatics essentially fall outside the limits of the curriculum on informatics for ordinary secondary school. Actually it is not absolutely so. Comparing the RusOI contents with themes of this curriculum it is possible to draw a conclusion that this curriculum in a certain degree covers the RusOI contents. But, as experience has shown, for successful participation in RusOI, especially in the final level, familiarization of this curriculum is not enough. In addition to this it is necessary to use forms of individual teaching with help of top level teachers and also actively to develop at school profile training and various elective courses.

4. Approaches to Determining the RusOI Contents

At determining of the RusOI contents it is necessary to take into consideration that informatics is the same science as mathematics and physics. Secondary school students should not only perceive it as a tool for solving of numerous problems by using a computer, but also to concentrate in its conceptual bases. For example, to answer such questions, as: what principles is informatics based on?; what new concepts are introduced with computer science in our world?; what kind of questions are called attention by scientists engaged in computer systems?; what methods are used for solving problems with use of computer equipment and information technologies?.

Proceeding from this, competition tasks should be composed so that to allow secondary school students:

- to get acquainted with fundamental knowledge of computer science;
- to promote development cognitive models of training to this knowledge;
- to encourage development of the skills which is necessary for application of conceptual knowledge on practice;
- to be prepared to the full for studying computer science at a professional level in higher school.

As most of RusOI participants after leaving secondary school become students of higher school at determining of the RusOI contents it is necessary to take into account experience of development of computer science curricula for higher education. The analysis of such programs [3] has allowed to allocate the following three approaches to development of the RusOI contents:

- programming-oriented approach;
- algorithms-oriented approach;
- breadth-oriented approach.

Programming-oriented approach was used at the initial stage of development of the RusOI contents. Therefore very often Olympiads in informatics is named as programming Olympiads. Formation of such incorrect opinion was affected with the following practical and historical factors:

- knack of programming is necessary skills for the further training in informatics, and not only for those who is going to study computer science further;
- computer science became academic discipline late enough, and by this moment in the majority of educational institutions there are introduction courses on programming;
- in 1970–1980 many courses under the name ”Introduction to computer science” contained the most part of topics connected only with programming and not reflecting development of algorithmic thinking.

Use of programming-oriented approach at determining of the RusOI contents has the following weaknesses:

- the accent on programming due to exception of other topics of computer science gives the limited understanding of computer science as science;
- theoretical knowledge of computer science which should strengthen understanding of a practical skills fade into the background, that leads to biased opinion on supporting role of the theory in the further studying computer science;
- consideration of programming as bases of the RusOI contents leads to that secondary school students concentrate on coding more, instead of on development of algorithm, analysis and testing of solutions;
- orientation to programming can lead secondary school students to believe, that the writing of the program is the unique approach to the solution of problems with use of a computer.

Algorithms-oriented approach is focused on studying of the basic algorithmic concepts and logic structures irrespective of any programming language. From secondary school students here is required the substantiation and an explanation of algorithms which they create. It allows them to work with a wide range of types of data and logic structures needlessly to struggle with various specific features which inevitably are present at popular programming languages. Possession of students of the basic algorithms and data structures allows them to prosecute more productively subjects of effective programming, debugging and testing of programs. Moreover, this approach can include additional theoretical topics, such as estimations of efficiency and the rudiments of computability

At the same time, use on Olympiads in informatics only the approach with orientation to algorithms has some critical weaknesses. First of all, for the solution of such theoretical tasks the pseudo-code for designing algorithms is used, as a rule, and secondary school students always wish to realize the ideas and solution on a computer, instead of abstract language on a paper. As the algorithm which has been written down in the programming language - is written not for people.

Secondly, orientation to a pseudo-code or even any language of writing of algorithms relieves students of necessity to show, that their algorithms correctly work. While the process of getting a program to compile and execute correctly is sometimes frustrating, it is also a critical skill that students must master early in their education. Therefore students should have appropriate skills for this at the earliest stage of the computer science training.

Thirdly, using algorithms-oriented approach there are big problems at evaluation of competition task solution. Now in most cases evaluation of competition task solution of participants occurs by means of the automated evaluating systems. The evaluation of a pseudo-code on a correctness in this case is very difficult for realizing, and to do it manually is not a simple problem too since this demands to recruit a plenty evaluators and often it is subjective.

Breadth-oriented approach is the approach with the maximal scope of topics on computer science. It has arisen as alternative to the two approaches mentioned above. For many years experts in the field of Olympiad informatics worried that those both approaches gives student too limited sight at informatics This science is constantly developing science which determines many other kinds of activity, and it is impossible not to consider this. Otherwise, competition tasks will not allow secondary school students to prove themselves in many other fields of knowledge, being a part of modern computer science.

Use breadth-oriented approach in practice has appeared some problems too. Expansion of number of topics leads to necessity of development of a plenty various competition tasks. They should be such that students could show the creative abilities and thinking. It is very much a challenge for developers of competition tasks. However it becomes even more difficult for students as to capture a plenty of topics on informatics with demanded immersing on the necessary depth of their familiarization becomes for them practically not real.

Discussions about what approach should be put in a basis of determining of the RusOI contents have been debated on all extent of development of Olympiad movement in informatics in the country. Each of the considered approaches has pluses and minuses, and nobody can tell unequivocally which is the better. Experience of carrying out of the RusOI has shown that true as always lays on the middle. Therefore at determining of the RusOI contents the new approach based on a harmonious combination all three mentioned above approaches has been used.

5. Structure and Short Description of the Modern RusOI Contents

The conclusions formulated in chapter 3 have allowed to allocate basic topics of computer science which determine the modern RusOI contents. In particular, as such topics the following have been chosen:

1. Mathematical Basis of Informatics.
2. Developing and Analyzing of Algorithms.
3. Programming Fundamentals.
4. Computer Literacy.
5. Operating Systems.
6. Basis of Programming Technology.
7. Fundamental Methods of Calculations and Modeling.
8. Introduction to Network Technologies.

The topic "Mathematical Basis of Informatics" substantially is connected with discrete structures and is a fundamental basis of computer science. It is especially important for participation in Olympiads in informatics as it is difficult to achieve success on competitions without good preparation in the field of set theory, logic, graph theory, combinatory theory and so on.

It is also important for the students to continue education in higher school. Moreover, escalating complexity of computer science methods influences the solution of practical professional problems. To solve this problems in the future for today's students are extremely necessary to have stable knowledge and skills in the different fields of mathematics especially discrete structures.

At successful familiarization of topic "Mathematical Basis of Informatics" secondary school students should:

know/understand:

- fundamental notions of functions, relations and sets;
- permutations, arrangement and combinations;
- formal methods of propositional logic;
- basis of construction of recurrent expression;
- the basic proof techniques;
- basis of numbers theory;

be able:

- to carry out the operations connected with sets, functions and relations;
- to calculate permutations, arrangement and combinations of set and also to interpret their values in a context of a specific task;

- to solve typical recurrent expression;
- to carry out formal logic proofs and a logic reasoning for modelling algorithms;
- to determine what kind of the proof approaches for the solving of a specific target is better;
- to use the basic algorithms of theory of numbers.

use the above-mentioned knowledge and skills at solving of practical tasks.

The basic themes of the topic “Mathematical Basis of Informatics” are:

1. Relations, Functions and Sets.
2. Basic Geometry.
3. Basic Logic.
4. Basics of Counting.
5. Proof techniques.
6. Basis of Theory of Numbers.
7. Basics of Algebra.
8. Basics of Combinatorial Calculus.
9. Basis of Graph Theory.
10. Basis of Probability Theory.
11. Basics of Game Theory.

The topic “Developing and the Analyzing of Algorithms” is very important for Olympiads in informatics and determines a basis of productive activity of students, their creative self-expression. In this field of computer science participants of Olympiad in informatics have an opportunity to show the best creative qualities at the solution of competition tasks.

Importance of algorithms theory is difficult to overestimate. Actual value of any program or program system depends on two factors: applied algorithms and efficiency of their implementation. Therefore development of good algorithm has crucial importance for productivity of any program system. Besides studying of algorithms allows to penetrate more deeply into a current task and can prompt methods of the solution which are not dependent on the programming language, a paradigm of programming, hardware maintenance and other aspects of implementation.

Studying of algorithms theory helps to develop at student’s ability to choose the algorithm most suitable for the solution of the given task or to prove, that such algorithm does not exist. This ability should be based on knowledge of algorithms which are intended for the solution of the certain set of known problems, their understanding strong and weaknesses, applicability of various algorithms with an estimation of its efficiency.

At successful familiarization of topic “Developing and the Analyzing of Algorithms” secondary school students should:

know/understand:

- elements of theory of algorithms;
- basic data structures;
- basic notions of graph theory and graph properties;
- relate graphs and trees to data structures, algorithms and counting;
- properties inherent in ”good” algorithms;
- big O notation for the description of the amount of work done by an algorithm;

- determining the time and space complexity of simple algorithms;
- computing complexity of the basic algorithms of sorting, search and hashing;
- concept of recursion and the general recursively presented problem statement;
- hash function and its assignment;
- simple numerical algorithms;
- basic combinatory algorithms;
- basic algorithms of computing geometry;
- the most widespread algorithms of sorting;
- the most important algorithms of string processing;
- representations of graphs (adjacency list, adjacency matrix);
- fundamental algorithms for graphs: depth- and breadth-first traversals, shortest-path algorithms, transitive closure, minimum spanning tree;
- basic algorithmic strategy: brute-force algorithms, backtracking, ”greedy” algorithms, ”divide-and-conquer” algorithms and heuristic algorithms;
- basis of dynamic programming;
- elements of game theory;

be able:

- to choose suitable data structures for the solution of problems;
- to use the above-mentioned algorithms during solving of problems;
- to determine memory and run-time complexity of algorithms;
- to determine computing complexity of the basic algorithms of sorting, search and hashing;
- to use big O notation for the description of the amount of work done by an algorithm;
- to implement recursive functions and procedures;

use the above-mentioned knowledge and skills at solving of practical tasks.

The basic themes of the topic “Developing and the Analyzing of Algorithms” are:

1. Algorithms and their properties.
2. Data Structures.
3. Basic Algorithmic Analysis.
4. Algorithmic Strategy.
5. Recursion.
6. Fundamental Computational Algorithms.
7. Numeric Algorithms.
8. String Processing.
9. Graph Algorithms.
10. Dynamic Programming.
11. Algorithms of Game Theory.
12. Geometrical Algorithms.

The topic “Programming Fundamentals” and a high technological level of its possession are necessary conditions of successful performance of any students on Olympiads in informatics. To participate in Olympiad in informatics, each secondary school student should know and put into practice even one programming language. Moreover, it is also desirable to master by participants of Olympiad at least two paradigms of programming because in RusOI is permitted to use three programming languages: C/C ++, Pascal, Visual Basic.

It is important to notice, that knowledge and skills in the field of programming, which are important for practice of programming, irrespective of an applied paradigm of programming. Therefore the given topic includes sections under fundamental concepts of programming, the basic structures of data and algorithms and also actually programming languages. Programming languages are the basic means of dialogue of the student and a computer during solving of competition tasks. Students should not simply be able to write the program in any one language, they should understand the various styles of programming inherent in different languages. The understanding of a variety of programming languages and various paradigms considerably facilitates fast study of new languages by them.

As a result of successful study of topic "Programming Fundamentals" secondary school students should

know/understand:

- basic structures of programming;
- concept of data type as sets of values and operations above them;
- basic data types;
- basic data structures: arrays, records, strings, stack, queues and hash tables;
- data presentation in memory;
- bases of input/output;
- operators, functions and parameter transmission;
- static, automatic and dynamic memory allotment;
- memory management during execution of the program;
- methods of realization of stacks, queues and hash tables;
- methods of realization graphs and trees;
- mechanism of parameter passing;
- features of realization of recursive solutions;
- useful strategy at program debugging;

be able:

- to analyze and explain behaviour of the simple programs including fundamental structures;
- to modify and expand the short programs using standard conditional and iterative operators and functions;
- to develop, realize, test and debug the program which to use all the most important structures of programming;
- to apply methods of structural (functional) decomposition to divide of the program into parts;
- to realize the basic data structures in high level language;
- to realize, test and debug recursive functions and procedures;

use the above-mentioned knowledge and skills at solving of practical tasks and confidently to program even on one of programming language permitted to use in RusOI (C/C ++, Pascal, Visual Basic).

The basic themes of the topic “Programming Fundamentals” are:

1. Programming Languages.
2. Basic Programming Constructions.
3. Variables and Data Types.
4. Data Structure Types.
5. Mechanisms of Abstraction.
6. Fundamental Programming Species.

The topic “Computer Literacy” plays an important role in studying of computer science because a computer is the integral tool which students use during competition in informatics. Secondary school students should not perceive a computer as a black box executing the programs by means of unknown magic. All students during solving competition problems should know the main components of which the computer consists and understand how they operate, their main characteristics, productivity and interaction between them. The understanding of the computer and its organization allows also to write more effective programs.

At successful familiarization of topic “Computer Literacy” secondary school students should:

know/understand:

- logic variables, operations, expressions;
- scale of notations;
- formats of representation of numerical data;
- presentation of data with fixed and floating point and connection with accuracy;
- internal representation of non-numerical data;
- internal representation of symbols, strings, records and arrays;
- instruction representation at a machine level;
- basis of input/output;
- basic types of memory;
- bases of memory control
- access to data on hard disk;

be able:

- to convert numbers from one scale of notation in another;
- to use mathematical expressions for the description of functions of simple consecutive and combinational schemes;
- to transform numerical data from one format to another;
- to adjust the programmer’s workbench for solution of competition tasks;

use the above-mentioned knowledge and skills at solving of practical tasks so that the students feels themselves confidently at work with a computer at solving of competition tasks.

The basic themes of the topic “Computer literacy” are:

1. Digital Logic.
2. Data Representation.
3. Computer Engineering Principles.

4. Memory.
5. Communications.

The topic “Operating systems” gives to students convenient abstraction of hardware maintenance of a computer. For many years operating systems and their abstraction became more and more difficult in comparison with usual applied programs. Nevertheless, for successful performance on Olympiad in informatics secondary school students should know:

- functions of modern operating systems;
- difference of primitive batch systems from complex multi-user operating systems;
- understanding of a logic level;
- page and segment organization;
- various ways of economy of memory;
- distinctions between the mechanisms used for communications with devices of a computer;
- advantages and shortcomings of direct memory access;
- requirements to recovery from failures.

Acquired knowledge of this topic should provide to students for the confident work with operating system at implementation of solution of competition tasks by means of a computer.

The basic themes of this topic are.

1. Operating Systems Basics.
2. Basic Functions of Operating Systems.

The topic “Basis of Programming Technology” is a part of software engineering, considering the application of corresponding knowledge and skills to effective implementation and operation of programs and program systems. Software engineering studies all phases of life cycle of program system: analysis of requirements, development of specifications, designing, implementation, testing, operation and support. Though on Olympiads in informatics there is no speech about development of program system, nevertheless, this topic plays a big role in solving of competition tasks and demands from students of quite certain knowledge and skills.

At successful familiarization of topic “Bases of Programming Technology” secondary school students should:

know/understand:

- purpose and structure of programming facilities and environments;
- role of program tools during development of the software;
- properties of designing of the ”good” software;
- basic methods of program testing and debugging;

be able:

- to choose and prove a set of programming facilities for support of development of the software;
- to use programming facilities and environments in development of software product;

- to compose tests;
- to test and debug of programs;
- to develop the program in the form of ready software product;

use the above-mentioned knowledge and skills at solving of practical tasks.

The basic themes of the topic “Bases of programming technology” are:

1. Programming facilities and environment.
2. Program Testing and Debugging.

The topic “Fundamental Methods of Calculations and Modeling” represents the area of computer science closely connected with calculus mathematics and numerical methods. As computers began capable to solve more and more challenges, similarly to computer science as a whole this area got the increasing value and importance. Moreover, by the end of the twentieth century scientific calculations have affirmed as the independent discipline having close connections with computer science, but, nevertheless, not identical with it.

In pure form methods of calculus mathematics and numerical methods are practically not used on the RusOI, exception is made only with methods of modeling. Nevertheless, the certain topics in this area play the important role in training of student for Olympiads in informatics. Their knowledge allows to approach more substantially to solving of competition tasks by students.

At successful familiarization of topic ““Fundamental Methods of Calculations and Modeling”” secondary school students should:

know/understand:

- notion of precision loss, computational stability, machine accuracy and error of calculus of approximations;
- sources of errors in calculus of approximations;
- basic algorithms of solving calculus mathematics tasks (calculation of value and roots of function, calculation of perimeter, square and volume, calculation of a point of crossing of two segments etc.);
- notions of model, modeling and simulation, basic types of models;
- specifications of computer model (input, output and state variables, state change functions, output function, time advance function) and ways of their description;
- basic phases and features of implementation and use of computer models;

be able:

- to calculate error estimation of calculus of approximations;
- to use basic methods of calculus mathematics at solving tasks;
- to formalize objects of modeling;
- to develop computer models of the elementary objects;

use the above-mentioned knowledge and skills at solving of practical tasks.

The basic themes of this topic are:

1. Basics of Computational Mathematics.
2. Modeling Basics.

The topic “Introduction to Network Technologies” is came into the RusOI contents in connection with last achievements in the field of networks and the telecommunications.

On Olympiads in informatics it is necessary for students not only the nobility that such computer networks, but also directly to carry out competition tasks in local computer network environment and the corresponding software. Possession of network technologies includes both theoretical knowledge, and practical skills. Students have to get basis of this knowledge and skills to feel during competition more confidently. The same situation and with wireless and mobile computers which actively start to be used on the RusOI, and an example to that - the final stage of the RusOI in 2006.

As a result of successful study of topic "Introduction to Network Technologies" secondary school students should

know/understand:

- basic structure of network architecture;
- most important network standards;
- roles and the responsibility of clients and servers for various applications;
- problems of networks safety arising because of viruses and the attacks directed on initiation of refusals in service;
- basis of wireless computer networks;

be able:

- to customize programmer's workbench in client- server network;
- effectively to use a number of the widespread network applications, including web-browsers and automated evaluating systems;
- to work with applications using mobile and wireless communications;

use the above-mentioned knowledge and skills for solving competition tasks in the environment which is installed for conducting RuOI.

The basic themes of the topic "Introduction to Network Technologies" are:

1. Basis of Networks and Telecommunications.
2. Introduction to Wireless Networks.

6. Conclusion

Short description of the RusOI contents presented in this paper is a result of longstanding experience of carrying out the Olympiads in informatics in The Russian Federation. The basic idea of these contents is to give all that are interested in participation in Olympiads in informatics a guiding line for development of an individual trajectory of training in Olympiad field in computer science. Moreover it gives good possibility for secondary school students, teachers, and composers of competition tasks to achieve a better common understanding of the knowledge and skills assumed of contestants of the RusOI.

The RusOI contents do not restrict the actual content of the RusOI and a process of deciding about the appropriateness of candidate competition tasks. It constantly develops, as well as computer science. Therefore to speak about any restrictions on this process it is not necessary.

The author hopes that the experience of Olympiads in informatics in Russia will be interesting to representatives of other countries.

References

Кирюхин В.М. (2005). *Всероссийская олимпиада школьников по информатике.* М.: АПК и ППРО.

Кирюхин В.М., М.С. Цветкова (2006). *Всероссийская олимпиада школьников по информатике в 2006 году,* М.: АПК и ППРО.

CC2001 – *Computing Curricula 2001 Computer Science,* Рекомендации по преподаванию информатики в университетах, Пер. с англ.: Спб.: Издательство СПбГУ (2002).

V. M. Kiryukhin is chairman of the methodical commission on informatics which is responsible in Russia for carrying out the national olympiads in informatics. He writes many papers and books in Russia on development of olympiad movements in informatics and preparations for olympiads. From the first IOI he is the permanent Russian team leader.

Olympiads in Informatics, 2007, Vol. 1, 105–111
© 2007 *Institute of Mathematics and Informatics, Vilnius*

USA Computing Olympiad (USACO)

Rob KOLSTAD,
15235 Roller Coaster Road, Colorado Springs, CO, 80921
e-mail: : kolstad@ace.delos.com

Don PIELE
University of Wisconsin-Parkside, 900 Wood Road, Kenosha, WI, 53405
e-mail: piele@uwp.edu

Abstract. The USA Computing Olympiad (**USACO)** supports pre-college computing around the world through computer programming competitions and training materials. The **USACO** holds six Internet-based contests each year. Unique in the contest community, they

- are open to every pre-college programmer around the world at no charge;
- feature three divisions of escalating difficulty and one beginning level contest;
- are machine-graded, with instant feedback on simple errors found during submission;
- are analyzed with solutions and results provided;
- are translated into foreign languages for the contestants;
- are created by a handful of coaches with the help of assistant coaches who interact through a comprehensive website that collects problems, test data, solutions, data validators, and discussions.

The **USACO** has reached out to the international community by

- inviting international students to challenge our students at the USA International Computing Olympiad (USAICO) here at Colorado College during our summer program;
- providing a discussion form on our website for contests and training pages;
- grading and ranking international students along with our own;
- providing a grading system for IOI 2001, 2003, 2004, 2006 and 2008 (expected).

In April, the **USACO** conducts the US Open, a proctored exam for US students, and, one day later, an Internet exam for international students. Based on the results of these contests, 16 students are invited to an all-expense-paid training camp in the early summer, where 4 students are selected to be the US Team at the International Olympiad in Informatics (IOI).

The **USACO** is sponsored by USENIX, SANS, and IBM. All **USACO** contest administration is staffed by 100% volunteers.

Key words:
USACO, USAICO, internet programming competitions, pre-college programming competitions, programming training, IOI, USA team selection.

Goals

Since 1992, the USACO has served the youth of the global pre-college computer programming community with four primary goals:

1. Provide pre-college students with opportunities to sharpen their computer programming skills in order to enable them to compete successfully at the international level.
2. Enhance the quality of pre-college computer education by providing students and teachers with challenging problems, training materials, and competitions that emphasize algorithm development and problem-solving skills.
3. Recognize excellent students with outstanding skills in computer science and encourage them to pursue further opportunities in the profession.
4. Provide educational, motivational, and competitive materials in the form of programming competitions and web-based training via the Internet.

USACO Scope

The USACO sponsors six Internet-based contests each year. Unique in the contest community, they:

- are open to every pre-college programmer around the world at no charge;
- feature three divisions of escalating difficulty;
- are machine-scored, with instant feedback for simple errors found during submission.

Fig. 2 graph shows the growth in participation. Now anywhere from 900–1,000 enter each competition.

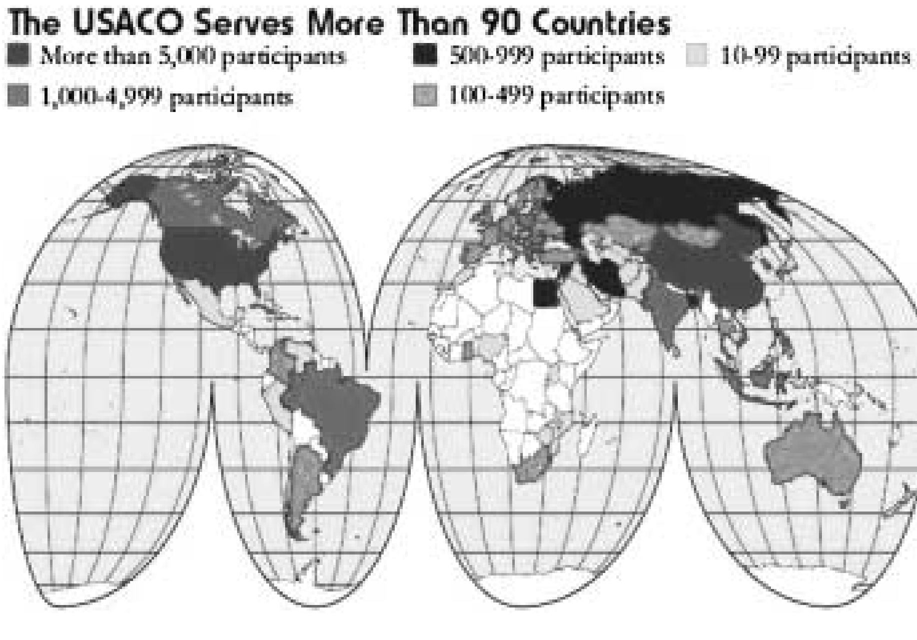


Fig. 1. USACO scope.

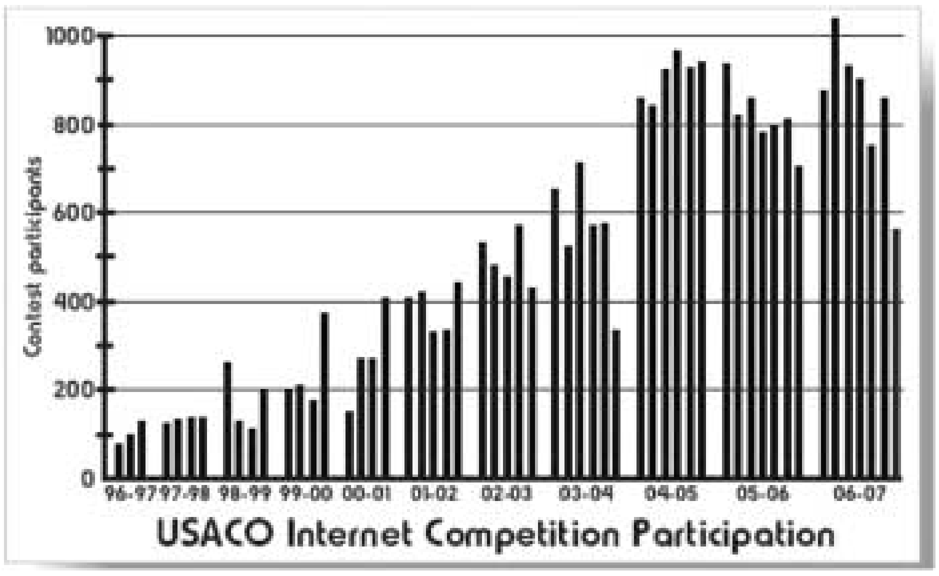


Fig. 2. Internet participation.

USACO Training

The USACO offers 200 hours of Internet-based training in the form of instructional texts and challenging programming tasks. Over 62,000 participants have registered for the training pages; over 321,000 tasks have been successfully solved. IOI world champions ship competitors from many countries extol USACO's training. Fig. 3 shows the growth of monthly training system logins.

The hs-computing mailing list was created to distribute Internet competition problems and exchange information about advanced pre-college computing education and training. Its subscribers include high school teachers, coaches, and students. The list currently includes over 28,000 correspondents from more than 90 countries.

USACO Invitational

Each year, the USACO invites 16 USA students and (sometimes) the best of the international competitors for a week of competition at the USA Invitational Computing Olympiad.

Challenging contests (the equivalent of three complete IOI world championship competitions) are complemented by an academic, recreational, and cultural program to stimulate competitors both intellectually and athletically.

The intense week culminates in an awards ceremony that includes the week-long winner and names the four members of the USA international traveling team. Travel to competitions is the biggest reward and incentive for USACO contestants. The June USAICO contest challenges the top 16 USA programmers to compete for the traveling team.

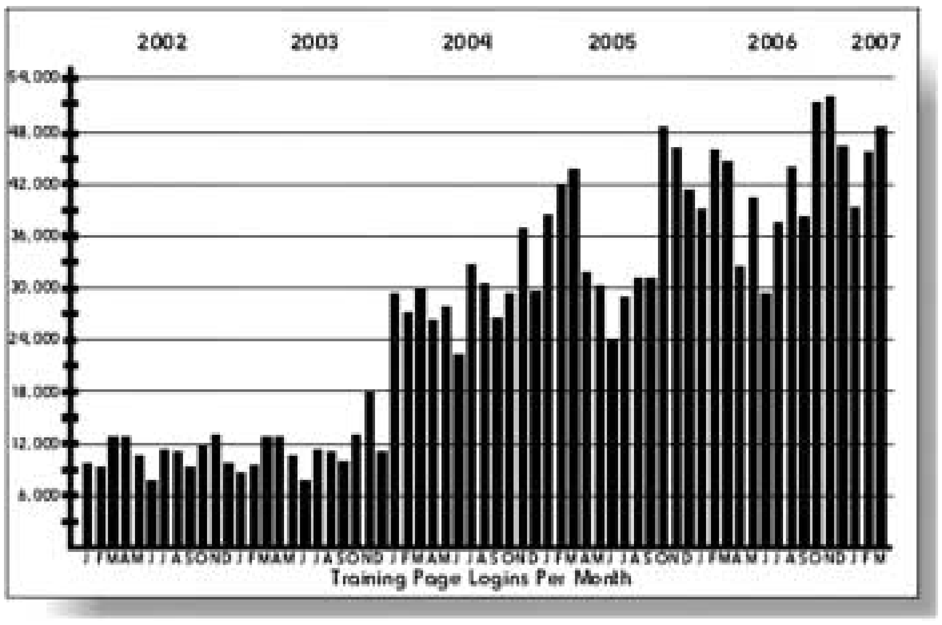


Fig. 3. Training page logins.

The top four USACO programmers compete against programmers from 75 other countries at the annual IOI (the world championships) in addition to occasional trips to the Central European Olympiad in Informatics and other international competitions.

USACO International Service

USACO staff contributes significantly to the international competition community. In 2003, USACO director Don Piele hosted the IOI world championships at his home campus at the University of Wisconsin-Parkside. USACO staff supervised all online grading for the IOI world championships in 2001, 2003, 2004, 2006, and is on deck for Egypt's 2008 International Olympiad. Coach Greg Galperin served on the International Scientific Committee; Don Piele retired in 2006 from the 15 member IOI governing committee after 12 years of service.

USACO Internet training and contests also serve the international community.

USACO Problems

USACO contest problems focus on the signature mascot of cows. The tasks require extensive knowledge of computer algorithms and are very challenging.

One recent contest's tasks were translated into 14 languages, including Chinese, German, English, French, German, Indonesian, Polish, Russian, Serbian, Spanish, Turkish,

USACO competitor Ben Joeris blackboards graphical algorithms with coach Brian Dean

Fig. 4. Teaching.

Turkmen, Ukrainian, and Vietnamese. The problem below was suggested by Canada's Maria Plachta.

Problem 2. The Wedding Juicer

Farmer John's cows have taken a side job designing interesting punch-bowl designs. The designs are created as follows:

A flat board of size W cm $\times$ H cm is procured.

On every 1 cm $\times$ 1 cm square of the board, a 1 cm $\times$ 1 cm block is placed. This block has some integer height B.

The blocks are all glued together carefully so that punch will not drain through them. They are glued so well, in fact, that the corner blocks really don't matter!

FJ's cows can never figure out, however, just how much punch their bowl designs will hold. Presuming the bowl is freestanding (i.e., no special walls around the bowl), calculate how much juice the bowl can hold. Some juice bowls, of course, leak out all the juice on the edges and will hold 0.

USACO Faculty

The coaching ranks also include several associate coaches. Many of these associates use the USA's training and contest resources for their own country's competitions. The list includes gold medal winners, other country's coaches, and both foreign and domestic graduates of the USACO invitational competitions.

Guest coaches at the 2006 competition included repeat IOI Gold Medal winner Bruce Merry from South Africa and Canada's Troy Vasiga from the University of Waterloo.

Canada's wunderkind Richard Peng has been invited to coach at the 2007 USAICO.

The regular faculty include:

- Director Dr. Don Piele is a Professor of Mathematics at the University of Wisconsin-Parkside. Founder of the International Computer Problem Solving Competition and USACO, Don has been organizing programming competitions since 1977 and took the first United States Team to IOI in 1992.
- Leader and Head Coach Dr. Rob Kolstad conducts the Internet competitions. A veteran of supercomputer startup companies and technical associations, he has organized programming competitions since 1973. Rob manages the Internet competitions, the online training, the automated grading system, and USAICO.
- Leader/Deputy Leader Dr. Brian Dean has coached since 1996 and recently completed his doctorate at MIT. He has interned at Akamai, Alta Vista, and Microsoft. Brian earned multiple awards for teaching at MIT. He is now an Assistant professor at Clemson University.
- Deputy Leader Liang joined the coaching staff in 2002 and is currently a Ph.D. student at Stanford, studying machine learning and natural language processing. His industrial experience includes IBM, Intel, ITA software, Microsoft Research, and Google.

The USACO is privileged to have a number of veteran organizers, contest champions, and performers in both the academic and industrial world on its staff. They have decades of experience and include both organizational veterans and enthusiastic former competitors.

- MIT student Alex Schwendner first attended USACO camp in 8th grade. He has won the US Open, the inaugural USAICO, and the USACO National Championship. Four trips to the IOI yielded two silver medals and two gold medals.
- MIT student Eric Price attended four USAICO Olympiads. A Silver and Gold Medalist at the IOI, he achieved the rare perfect score in 2005 (in addition to a Gold Medal at the International Mathematics Olympiad). Eric also organizes the Harvard/MIT Math Tournament.

USACO Sponsors

The USACO currently has three sponsors:

- USENIX: The Advanced Systems Computing Association;
- IBM;
- SANS: Security Training.

References

The USACO genera web site: http://www.usaco.org
The USACO Discussion Board: http://ace.delos.com/bb/
The USACO registration: http://ace.delos.com/usacoregister
The USACO training pages: http://train.usaco.org
The USACO Internet Contests: http://ace.delos.com/contestgate

R. Kolstad is the head coach of the USA Computing Olympiad and a consulting in the computer industry. Previous employment includes a startup supercomputer company, a large workstation manufacturer, a startup internet server corporation, and non-profit technical organizations. His current consulting revolves around the world of software support for patent litigation.

D. Piele recently retired from his long-time position as professor of mathematics at the University of Wisconsin-Parkside. Active not only in computer olympiads as the USACO director (and founder of other contests including the popular and long-running International Computer Problem Solving Competition), D. Piele also works extensively with mathematica.

Olympiads in Informatics, 2007, Vol. 1, 112–123
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Programming Contests for School Students in Bulgaria

Krassimir MANEV
Faculty of Mathematics and Informatics, Sofia University
1164, 5 J. Bourcheir blvd., Sofia, Bulgaria
e-mail: manev@fmi.uni-sofia.bg

Emil KELEVEDJIEV
Institute of Mathematics and Informatics, Bulgarian Academy of Sciences
1113, 8 G. Bonchev str, Sofia, Bulgaria
e-mail: keleved@math.bas.bg

Stoyan KAPRALOV
Department of Mathematics, Technical University of Gabrovo
5300,4 H. Dimitar str., Gabrovo, Bulgaria
e-mail: skapralov@tugab.bg

Abstract. Competitions in programming for secondary and high school students in Bulgaria have long traditions. The National Olympiad in Informatics started in 1985. Even before 1985, competitions on national and regional level were popular. Bulgaria is founder and the first host of the International Olympiad in Informatics in 1989. The paper presents the current situation and challenges in the area of Informatics competitions in Bulgaria. The structure of the competition system, including the Bulgarian National Olympiad in Informatics is outlined.

Key words: programming contests, olympiad in informatics, IOI, training, preparation.

1. Introduction

Education and competitions are closely related. We may present at least two reasons to assert this (Verhoeff, 1997):

– it is natural for children to compete, so they can easily understand any attempt of the teacher to put the competitions into use of education;
– competitions may be important in adult life, so we should especially teach children to compete – this is a part of the education.

Not all opinions on the role of competition in education look fairly straightforward. Educational theorists do not agree on whether competitions should be encouraged or constrained. Some of them assert that, because competitions are part of every culture and because education should transmit culture, it is necessary to incorporate competitions into education to help children get ready for it later. The others contradict that, because competitions are opposed to collaboration, and therefore are an evil element in culture, they should be avoided.

Nevertheless, competitions in education have their long history, which can be traced back to the centuries B.C. We present a brief time line:

– in first century B.C. Marcus Verrius Flaccus, a Roman teacher, introduced the principle of competition among his students as a pedagogical aid;
– in 1500 Battista Guarino, an Italian scholar, wrote that the students are stimulated best by competitions, instead of physical punishment;
– in 1894 the time when Pierre de Coubertin struggled to revive the Olympic Games, the Eötvös Loran University in Budapest organized the first national contest in mathematics;
– the idea of Science contests spread through Central Europe and Russia. In North America William Putnam started a mathematics competition for college students in 1938;
– in 1959 the first International Olympiad in Mathematics was hosted in Romania;
– in 1989 the first International Olympiad in Informatics was hosted in Bulgaria (below we will describe how this happen).

The competitions have much to offer in education. They are a good measure of how well a discipline is accepted. The competitions should be further developed; their organizing (especially organizing good competitions) is a major challenge.

2. At the Beginning

2.1. *First Programming Contests in Bulgaria*

The programming contests for school students in Bulgaria started in early 80’s of the past century. In the schedule of traditional Winter Mathematical Competitions, organized by the Union of the Bulgarian Mathematicians (UBM), a Programming tournament was included. The participants had to write a program in one of the languages FORTRAN or PL/1, that solve a given algorithmic task, to punch source on cards, compile it (computers was IBM/360 compatible machines from the ES-series) and to try to debug the program for 3–4 runs (no more runs were possible in limited to four hours contest).

In 1982 Bulgaria started to produce the Apple II-compatible machine Pravetz 8. Very soon each Bulgarian school had at least one computer lab. So, each participant in Winter tournament had the possibility to work on an individual computer. The languages BASIC and Pascal (UCSD) replaced FORTRAN and PL/1. Evaluation of solutions in those years was pure manual and some quantity of marks was assigned for the style of programming.

2.2. *National Olympiad*

Four years expertise from Winter tournament was enough for the Team of the UBM to dare to organize a National Olympiad in Informatics (NOI). In 1985 the First NOI took place. This was a two day contest. In the first day contestants had to solve some theoretical task – concerning the algorithmic knowledge and knowledge of the programming language. The second day was similar to the Winter tournament – the contestants had to write and debug a program. Since the Second NOI, in each of two days contestants had to solve one task by writing a corresponding program. There was no special qualification for participating in the Final round of the NOI. Each school organized its own contest to decide which students would be sent to the Final.

In parallel with the NOI some efforts were made to involve as much as possible students in the game. In the program of Winter tournament a new age group was included – for students in 5th–7th grade. The older students participated both in Winter tournament and Final round of the NOI.

2.3. *International Contest*

In 1987 Sofia – the capital of Bulgaria – hosted the International Conference of IFIP and UNESCO "Children in Information Age". Prof. Blagovest Sendov, member of the Academy of Sciences and President of the Organizing committee proposed to the Team from the UBM to organize during the Conference an international programming contest for school students. The contest was organized in two age groups (younger and older than 14 years). Students from 7 countries took part in this event (Check–Slovak Republic, Federal Republic of Germany, German Democratic Republic, Poland, Rumania, Soviet Union and Bulgaria).

The results of this experiment were fantastic. All participants were very enthusiastic about the future of programming contests. They shared their experience in preparation of teams and some ideas about organization of the future contest. As a result Prof. Sendov asked the authorities in UNESCO for permission to start a new Olympiad – International Olympiad in Informatics, using the model of the other scientific International Olympiads for school students and especially the model of International Olympiad in Mathematics.

The first International Olympiad in Informatics was held in June 1989 in Pravetz, small town, placed about 50 km North-East from Sofia. The town was not only the center of Bulgarian computer industry, but the birth place of the Head of State Council (equivalent to President) of Bulgaria in that time, too. The term *Informatics*, not so popular in English-speaking countries was chosen, after long discussions, to replace *Computer Science* or *Programming* because of very good looking abbreviation IOI (not simply palindrome, but a graphical palindrome, too). The President of First IOI was Prof. Peter Kenderov, a mathematician with huge experience from mathematical Olympiads.

Students from 13 countries took part. The teams were composed of a Leader and 3 students. Contest was organized in one day. The students had to solve the following task (here is a simplified version of the statement):

Task. A sequence of $2N$ boxes is given. In $N-1$ of the boxes white pieces are placed, in other $N-1$ of the boxes – black pieces, and 2 consecutive boxes are empty. The following movement is permitted – to take pieces from two consecutive boxes and to move them, conserving the order, into empty boxes. Write a program to arrange, with minimal number of movements, white pieces leftmost of black pieces (the final place of two empty boxes does no mater).

The absolute winner of the contest was 15 year old student from the Second Bulgarian Team – Teodor Tonchev. He had a very precisely planned BFS and solved the largest test case. For more details about the proposed tasks, the contests, and results of the First International Olympiad in Informatics see (Kenderov, Maneva, 1989). We are planning a new edition of this brochure for the celebration of 20-th anniversary of the First Olympiad during IOI'2009.

3. Current State of the National Programming Contests

3.1. *Organization*

Many institutions and people are involved in the organization of the national competitions in Informatics.

The two main organizers at the conceptual level are the Ministry of Education and Science (MES) and UBM. The link between them is the National Committee (NC). The chair and the members of the NC are proposed by the UBM and are approved by the Minister of Education and Science. The membership in the NC is a kind of a voluntary activity. The main responsibility of the Committee is carrying out the annual competition schedule that includes NOI and a number of national tournaments.

At more practical level the organizers of the national competitions are universities and schools. There are different kinds of high schools in Bulgaria. The most popular are math schools and language schools. The best students as a rule attend these two kinds of schools. Typically the education in high schools is from 8th to 12th grades but in many math schools there are also students from 5th to 7th grades. The math schools are the main source of competitors for the national contests in programming.

3.2. *Structure of NOI*

The NOI started with one age group in 1985. Now the contestants are divided in 5 age groups – E (4th–5th grade), D (6th–7th grade), C (8th–9th grade), B (10th–11th grade) and A (master group). Contests in different groups have different duration – 3 hours for E, 4 hours for D and C, and 5 hours for B and A. Contests are purely conforming to the format of IOI. Does not matter in which group participates, each contestant has an own work place and must solve three task of algorithmic type, writing the corresponding programs in one of the official languages of IOI – C/C++ or Pascal.

For 22 years the NOI of Bulgaria totally changed. Nowadays we have 3 rounds – Local/School (in February), Regional (in March) and Final/National (in April). The schools or villages where some students declared a will to participate in the round are free to prepare their own tasks. For helping schools (especially in small villages) that are not able to prepare tasks, the NC proposes a set of sample tasks for this round but does not participate in grading. The round is not a formal qualification. The teachers, that evaluate the contestants, decide who is ready to participate in the Regional round.

Tasks for Regional and Final round are prepared by the NC. Regional round is organized in schools in one day with common start. Solutions of the pupils are sent immediately to the work groups of NC that evaluate and grade them with common set of test data. The round is a qualification for the Finals of the NOI – 50–60 students in group A, and 10–20 from the other groups are invited to take part in the Finals round.

The Final round is organized in a different town each year. There is one contest for groups B, C, D and E, and two contests (in two consecutive days) for group A. The first 10 students from the Final round in group A form the long list of National Team for Balkan Olympiad in Informatics (BOI) and IOI. A sample of the tasks from the Final round in this group is given in Appendix 1. Since this year some Bulgarian students aged less than 15.5 year will participate in Balkan Olympiad in Informatics for Juniors (JBOI), which will be finished at the moment of presenting this paper. Following the above described procedure, this year, 12 best students (aged less than 15.5 years) were selected from the Final round in group C and formed the long list of Junior National team. A sample of the tasks from contest in group C is given in Appendix 2.

3.3. *National Tournaments*

In parallel with the NOI a system of National Tournaments (NT) is organized by the UBM and with the help of the MES. The season starts with the Fall tournament in November. The traditional Winter tournament is in January and the final for the season is the Spring tournament in the end of May or the beginning of June. The NT's are open. Each student could participate in each NT. Format of the contests in NT's is the same as those of the NOI. But, in principle, tasks are more difficult, and frequently, during these contests, some experiments are made. Some new topics or types of tasks usually first appear in one or two NT's before to be given in the NOI.

The NT's are not only a possibility for students to have regular contest during the season. By the results of Fall tournament, Winter tournament and the two days of the Final round of the NOI, the last 2 places in the long list of National Team are filled. It is not a rare case when students obtained one of this two "wild cards", later took place in the National team. Something more, the Spring tournament is a decisive contest for the election of the four students, which will represent Bulgaria in the BOI and IOI (since this year the same is valid for JBOI, too).

4. Preparing Contestants and Training the National Teams

Programming competitions are attractive, because the winners are considered among the classmates as programmers of a very high rank and the competitors with the best results have the right to enter university usually without other exams, which is not negligible. In many countries, including Bulgaria, the question how to help students to prepare themselves for competitions in Informatics arises, especially for those who do not attend specialized schools.

4.1. *Out of Class Forms*

The official curriculum in Bulgarian secondary schools include one year teaching of Informatics (9th grade) and one year teaching of IT (10th grade) for regular students. In some schools (especially math schools) students could take part in special (profiling) education of Informatics. But neither regular nor profiling education in Informatics is enough to prepare a student to participate in programming contests. It happens in out of class CS-schools. The famous CS-schools in Varna, Shoumen, Rosse, Stara Zagora, Veliko Tarnovo, Bourgas, Yambol, Gabrovo, Pleven, as well as these in City Math School of Sofia and National Math School (situated in Sofia also) recruit practically all members of the National teams recently.

Lecturers in the schools are qualified teachers and professors from the local universities with the active support from ex-contestants that study in Bulgarian and world universities. These are the people preparing the tasks for the national contests as well. Unfortunately, we are speaking for less then 25 persons that work with great enthusiasm, practically on voluntary principle, lead their CS-schools, prepare tasks for the National contests, give lectures during the preparation camps, write teaching materials and so on.

4.2. *Preparation Camps*

Traditionally the National team for IOI is trained during a summer camp organized a couple of weeks before the Olympiad. The camp is a mixture of lectures and competition days, simulating the International Olympiads. The camp is organized in the Black Sea area for 8–10 days and is accompanied with sport activities, visiting of the beach and cultural events (the famous Jazz Festival in Varna is scheduled in the same time as our camps).

Because of the complex situation with lack of qualified teachers, since 2006, the NC starts to organize some training camps for students of small age groups C, D and E. In 2006 three and in 2007 two such camps were organized with duration of one week each and in the format of training camps of the National teams. For this purpose, a National ranking list of contestants was elaborated, based on the results of students in the National contests, and 10–15 pupils from each age group are invited for a camp. This approach significantly ameliorated the quality of contestants in small age groups and stimulated the interest to competitive programming in towns, hosting the camp.

4.3. *The Balkan On-Line Training Program* ***campion***

One of the interesting forms of preparation, in which Bulgarian school students take part, is the Balkan on-line training program **campion**, organized by our Romanian colleagues. This is a form of training of Romanian students, similar to the American USACO, but since 2006 it is open for participants of all Balkan countries. The training consists of regular on-line contests for three age groups (one contest per month in average) and a final round for best ranked students in each age group. Participation of our students in this training program was very stimulating and helpful.

4.4. *Distant Competitions of Bulgarian Magazines*

On the pages of some related to IT Bulgarian magazines (*Computer Magazine* of The New Tech Publishing company and *PC Magazine* of IDG group, supported by software company Musala Soft), distant competitions in programming has being maintained during the past 20 years. We will point out the following main features of that kind of competitions:

– student has a plenty of time to solve the proposed problem – typically a month;
– student's solution has to contain not only a programming code, but also some explanations;
– the evaluation uses test examples;
– the evaluation also uses examination of student's explanations;
– after the end of the period, when the participants should submit their solutions, the author of the proposed problem publishes a detailed description of his own solution, accompanying it with explanations and discussions of students' works.

4.5. *Other Activities*

Among the other activities for preparation of contestants it is worth to mention the publishing activities and maintaining of a site dedicated to the programming contest.

An old idea of the NC is to publish series of books dedicated to the competitive programming. Despite the big difficulties three such books were published till now – an introductory book on programming in C/C++ for 11–12 years aged students (Yovcheva&Ivanova, 2006), one book on competitive programming for same age (Kelevedjiev&Dzhenkova, 2004) and one on dynamic programming for elder contestants (Kelevedjiev, 2001). Two other books – a second part of the introduction to programming in C/C++ and an introduction to algorithms in graphs have been prepared and will be published soon.

A Bulgarian Internet portal for competitive programming was created more than 10 years ago by the talented contestant (at that time) Svetlin Nakov. Latter the portal was generously hosted and maintained by the software company Musala Soft and could be found at address `http://infoman.musala.com`. All materials of the Bulgarian programming contests (statement of the tasks, tests, answers, checkers, as well as the solutions of all contestants) are regularly published on the pages of that site. In such way the NOI is the most transparent of the scientific Olympiads in Bulgaria.

Something more, the site is regularly publishing analysis (mathematical explanations and model solutions) of the competitive tasks, written by the authors, by students or editors of the site. The site also informs for incoming events, gives references to training materials and other resources and maintains a forum, where Bulgarian contestants could share experience, discuss the problems, etc.

5. Conclusion

As a general supporter of the activities concerning the International Olympiad in Informatics we should note the American foundation for Bulgaria, which is founded by some successful American businessmen with Bulgarian origin and leaded by a former participant in (and Golden medalist from) the International Olympiads in Mathematics. We would like to express, in behalf of the Bulgarian programming contests community, our deep acknowledgements to these people.

Bulgaria has long traditions in organizing programming contests for school students. After the First IOI in 1989 we hosted two Balkan Olympiads (1995 and 2004). Bulgaria was chosen to organize IOI'2009, just for celebration of the 20th anniversary of the beginning of the IOI. We hope that during the IOI'2009, in the picturesque city of Plovdiv, we will be able not only to demonstrate our hospitality but to share more of our experience, too.

Appendix 1: Example of Tasks for Group A

Below a typical set of tasks from the master group A of the Bulgarian NOI (Day 2 of the Final round of 2007) is given

Task A1. Area. A rectangle Q with sides parallel to the axes of orthogonal planar coordinate system and a point T, which is internal for the rectangle, are given. N lines are also given ($0 < N < 50$), not passing trough T. For each line let us consider the half of the plane defined by the line, that contains the point T and to form the area of the plane that is an intersection of all such half planes. Write a program **area** to find the face of the obtained area.

Input. The first line of the standard input contains the coordinates (x_B, y_B) of the bottom left and (x_E, y_E) of the upper right corner of the rectangle Q. The second line contains the coordinates of the point T. The third line contains the number N. Each of the following N lines contains the coordinates (x_1, y_1) and (x_2, y_2) of couple of points that define one of the given lines. All coordinates are non negative integers, less than 10000. All lines, including the lines defined by the corners of the rectangle Q are such that no three lines that pass through a common point.

Output. The program has to print on the standard output one integer – the found face truncated after the decimal point.

EXAMPLE.

Input	**Output**

`0 0 5000 5000`
`4000 2500`
`2`
`2800 4100 400 4300`
`800 2200 4600 80`
	`14348737`

Task A2. Numbers, numbers, ... Let N be a natural number and D is the product of its digits in decimal system. Let us define an operation over N giving as a result the numbers $N_1 = N - D$ and $N_2 = N + D$. Let us apply the operation to N_1 and N_2, to the numbers obtained from them, and so on. A question: is it possible, starting with a given number N and applying the operation, to obtain the number N again? For some numbers it is easy to answer, positive or negative, of the question, for other numbers it will be difficult to find the answer and there are numbers for which finding the answer seems impossible.

Input. Ten files are given, named `numb.01.in`, `numb.02.in`, ..., `numb.10.in`. In the single line of each file 10 different integers between 0 and 1000 will be given.

Output. For each file `numb.xx.in` you have to produce a file named `numb.xx.out` containing string of length 10, composed of characters `0`, `1` and `2`. Each character corresponds to one number from the input. The character has to be 1, if you established that the corresponding number in the input could be obtained by itself with applying the operation. The character has to be `0`, if you established that the corresponding number in the input could not be obtained by itself with applying the operation. If you where not able to establish the true then the character has to be `2`.

EXAMPLE.

numb.xx.in	**numb.xx.out**
10 11 12 13 14 15 16 17 18 19	1000100010

Evaluation. If your output file is the same as the file of the author then 10 points will be assigned for the test. For each `2` in your output, placed where `0` or `1` is expected, the result will be decreased by 1 point. If in your input there is `1` in place, where `0` is expected or the opposite – 0 points will be assigned for the test.

Task A3. Strings. The string S of length L is composed of the characters of given set T. Write a program `string` to find the number of different strings X of length P, composed of the characters of T, such that S is not a substring of X.

Input. On the first line of the standard input the string S of length L will be given ($1 \leqslant L \leqslant 2000$). Second line of the input contains also a string such that each character of T appears once in it. The characters of T are small and/or capital letters of Latin alphabet (so the size of T is no more than 52). On the third line of the standard input the number P will be given ($1 \leqslant P \leqslant 2000$). In 30% of tests the set T contains 2 letters and $P \leqslant 20$.

Output. On the single line of the standard output the program has to print required number of strings reduced by modulo 10^6.

EXAMPLE.

Input	**Output**
aa ab 3	5

Appendix 2: Example of Tasks for Group C

Tasks of groups C are especially interesting because of the approaching first international contest in programming for students aged less than 15.5 years – the Balkan Olympiad in Informatics for juniors. In the moment when this paper will be presented the Olympiad will be finished, we will know the tasks that were selected for the contests and if these tasks are appropriate for the students or not. Now we can only present the vision of the Bulgarian team about hardness of the tasks for an international contest for the mentioned age. Here are some tasks for group C from the Spring tournament, held in June 2007.

Task C1. Exchanges. Sporting activities are part of preparation of National team for Balkan Olympiad in Informatics for juniors. In the break between two lectures sporting coach of the team proposed following game. On the periphery of a large circle she has drown N small circles (as many as the students), labeled with the numbers from 1 to N. Then she distributed students in small circles – one student in a circle and placed in each circle one of N plates labeled also from 1 to N. By the sign of the teacher each student had to run from her/his circle to the circle pointed by the plate. The movement generated a big disorder! When each student reached the goal the teacher gave a sign again and the students had to make the exercise again. The game finished when all students reached their initial positions simultaneously. Funny, isn't it? Write a program `change`, to calculate how many movements will be necessary to finish the game.

Input. On the first line of the standard input the number N will be given $(5 \leqslant N \leqslant 100000)$. On the second – list of the plates in order they were placed in the circles $1, 2, \ldots, N$, respectively.

Output. On the single row of standard output the program has to print the number of exchanges. In 80% of test cases the result will be less than 2000000000.

EXAMPLE.

Input	**Output**
`5`	`6`
`3 5 4 1 2`	

Task C2. Game. A positive integer N and K positive integers, less then N, are given ($1 \leqslant N \leqslant 10^8$, $2 \leqslant K \leqslant 15$). Two players play the following game. First player choses one of the K numbers and divides N to it (integer division). Then the second player choses one of the K numbers and divides to it the result of the first player. Then the first player moves again and so on. The player that first obtains result 0 is the winner. Write a program `divgame` to check is it possible for the first player to win the game, dose not matter how the second player will move and if yes – how many among the K possible first divisors lead to a victory.

Input. For each run the program has to check two games. On the first row the numbers N and K for first game are given and on the second – the K divisors. On the third and forth line the data for second game are given in similar way.

Output. For each game the program has to print on standard output the number of divisors that lead to victory. If the number is not zero – a second line has to be printed for the game with a list of winning divisors.

EXAMPLE.

Input	Output
6 2	1
2 3	2
18 2	0
2 3	

Task C3. Net

An IT company has to connect computers in a local net. A number less than 15000 identifies each computer. A list of M couples of computers that have to be linked is given. The two computers in the couple have to be linked directly or trough one other computer. Write a program `net` to find the minimal number of direct links that are enough for creating the net.

Input. First line of the standard input will contain the number M ($M < 5000$) of couples that have to be linked. Each of the next M lines contains one of the couples – two identifiers separated by single space.

Output. On the single line of the standart otput the minimal number of direct links has to be printed.

EXAMPLES

Input	3 1 2 1 3 2 3	6 1 2 0 2 5 1 0 1 5 2 6 5	7 1 4 6 10 4 6 2 3 5 1 4 10 1 6
Output	2	4	5

References

Verhoeff, T. (1997). The role of competitions in education. Presented at *Future World*: *Educating for the 21st Century*. A conference and exhibition at IOI'97, December.

http://olympiads.win.tue.nl/ioi/ioi97/ffutwrld/competit.html (accessed on June 10, 2007)

Kenderov, P., and N. Maneva (Eds.) (1989). *International Olympiad in Informatics*. Sofia.

Yovcheva, B., and I. Ivanova (2006). *First Step in Programming with C/C++*. KLMN, Sofia (in Bulgarian).

Kelevedjiev, E., and Z. Dzhenkova (2004). *Algorithms, Programs, Problems. Manual for Beginner's Trainning in Competitions and Olympiads*. Regalia, Sofia (in Bulgarian).

Kelevedjiev, E. (2001). *Dynamic Programming*. Anubis, Sofia (in Bulgarian).

K. Manev is an associate professor at Sofia University, PhD, teaching discrete math and algorithms. Since 1982 he is a member of the Bulgarian NC for olympiads in informatics and leader of NC from 1998 to 2002. He participated in organization of the First IOI and was leader of the Bulgarian team for IOI's and BOI's, president of the SC of two BOI's. Elected member of IC of IOI from 2000 to 2003 and member of IC of IOI from 2005 to 2010 as a representative from the host country of IOI'2009.

E. Kelevedjiev is a research fellow in Institute of Mathematics and Informatics of Bulgarian Academy of Sciences. His field of interests includes algorithms in computer science, operation research, digitization techniques, etc. He is a member of the Bulgarian NC for olympiads in informatics since 1993; leader or deputy leader of the Bulgarian team for many IOI's and BOI's.

S. Kapralov is an associated professor at the Technical University of Gabrovo, doctor of sciences in mathematics, teaching discrete mathematics and programming. He is vice president of UBM, responsible for olympiads in informatics; member of the Bulgarian NC for olympiads in informatics since 2000 and leader of the NC since 2002; leader or deputy leader of the Bulgarian team for many IOI's and BOI's.

Olympiads in Informatics, 2007, Vol. 1, 124–130
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Perspectives on Grading Systems

Martin MAREŠ

Department of Applied Mathematics, Faculty of Math and Physics, Charles University in Prague
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
e-mail: mares@kam.mff.cuni.cz

Abstract. Programming contests often use automatic grading of the submitted solutions. This article describes one such system we have developed for the Czech national programming olympiad, the experiences gathered over the course of its development and also our perspectives on the future of such systems.

Key words: automatic grading, mo-eval, Linux.

1. Introduction

Many programming contests in the world, including the IOI, are based on automatic grading of the submitted solutions. This is accomplished by running them on batches of input data and testing correctness of the output. Time and space limits are usually enforced during the process, which allows to judge by not only the (approximation of) the correctness of the solution, but also its time and space complexity.

Multiple such evaluation systems have been developed, but most of them are used only in a single country and they are usually neither publicly available nor well documented. This seems to be a waste of effort by inventing the same things over and over and also by making mistakes somebody else has already made and understood.

This article is our modest attempt to help mapping the landscape of automatic evaluation of solutions. We will describe the MO contest environment we have designed and developed for several Czech programming contests. We will try to review the experiences gathered over the course of its development and present our perspectives on the future of similar systems.

The current version of the MO system can be downloaded from the web site listed in references. At the time of this writing, it was in use for several years in the Czech national olympiad in programming, at the Czech-Polish-Slovak preparation camps (which serve as a training for both our contestants and the system) and also for testing of student homeworks at our faculty. A new version is currently being finished for the Central-European olympiad in informatics (CEOI 2007) .

2. The Contest System

2.1. *Building Blocks*

The MO contest system consists of the following parts:

- *development environment* – editors, compilers, debuggers and similar tools used by the contestants for writing the solutions. In our case, this is just an appropriately configured Debian Linux system with a couple of packages added;
- *submitter* – this is the user interface of the contest system for contestants. It is primarily used for submitting a finished task solution for grading. Currently, we use a simple command-line utility for the national olympiad and a web-based interface for our classes;
- *evaluator* (also known as a *grader*) – takes care of testing the solutions and imposing limits. This is the core of the whole system;
- *feedback* – presents the results of the evaluator to the users. In IOI-type contests, its only role is to generate the evaluation reports and rank lists, but in general it can provide on-line feedback to contestants and/or spectators;
- *auxiliary services* – printing and similar. They vary from contest to contest and there are out of scope of this paper.

In this article, we will focus on the evaluator part, but we will keep in mind its connections to the other parts.

2.2. *Design Goals*

When we were designing our contest system, we had several basic goals in mind:

First of all, the system should be *flexible.* There are many types of contests ranging from the strictly off-line nature of the IOI to those with fully on-line feedback as the ACM ICPC. The variety of contest tasks and their types of interaction is probably even higher. Therefore we should try to expect the unexpected and make the system highly configurable and modular, so that the usual tasks can be dealt with by setting the parameters, while for the exotic ones we can plug in new modules.

Second, we should make the system *secure.* The submitted solutions can try to attack the evaluation system in various ways (not necessarily intentionally). A thorough study of known attacks has been recently published by M. Forišek. Hence the evaluator must be robust enough and isolate the examined solution from the rest of the evaluation system. This is an instance of the classical problem of running untrusted code, frequently studied as a part of OS security.

Last, but not least, the evaluator should be *as simple as possible* (but of course not simpler) in order to allow easy review of the whole code for security and correctness. Because of this, we have avoided putting any user interface into the evaluator and we prefer trivial input and output interfaces instead, which can be then presented to the user by some other components of the contest system.

We have also decided that having the system work on Linux is enough for our purposes. However, most parts of the evaluator can run on any POSIX-compliant system, the only exception being the sandbox, which makes heavy use of special features of the Linux kernel.

3. The Evaluator

3.1. *Sandbox*

The *sandbox* is the core of the evaluator. Its purpose is to run a program in a controlled environment, where both the interaction with other programs and the consumption of system resources (time, memory and disk space) is limited. We use the sandbox for running the solutions being evaluated, but also (with a relaxed set of restrictions) for compiling them.

The implementation of the sandbox makes use of the ptrace interface of the Linux kernel, originally designed for attaching debuggers to programs. It runs the program within a separate process and asks the kernel to interrupt the process every time it tries to make a system call. The sandbox then examines the parameters of the call and either lets the program continue its execution, or terminates it. For example, allocation of memory is always allowed, opening of files is allowed after inspecting the path to the file, and creating a new process is always forbidden.

This approach has been well studied by the secure system researchers and it has several known drawbacks, mostly related to race conditions in multi-threaded programs (another thread can modify the parameters of a syscall in the small time window between checking the parameters by the monitor and really executing the call). Fortunately, none of these problems apply to our case as no concurrency takes place. Another thing needing some extra attention is that although the contestants are not expected to use any syscalls outside the basic reading and writing of files and allocating memory, the standard libraries they call do use much more. We can however still manage with a simple list of obviously safe syscalls and a handful of easily verifiable exceptions.

Consumption of resources is controlled in the usual ways. We use the ulimit mechanism provided by the kernel whereever it is possible, that is to constrain the memory (address space) allocated by the process and also the maximum number of file handles used. Disk space filled by the program is limited by a disk quota for a special user ID which is used exclusively for the sandbox.

Limitation of execution time is slightly more complex, because it is not obvious what exactly should the time mean. The sandbox allows to measure either the user time of the process (which is accounted by the kernel and includes only timer ticks spent in the specific process in the user mode of the processor, that is, excluding syscalls and interrupts) or the wall clock time (as reported by the real-time clock of the OS). In both cases, the sandbox monitors the state of the timer periodically, kills the process when it exceeds the allowed time and it also checks the exact value of the timer when the program finishes successfully.

We currently use the first method for almost all tasks, since it makes the timing of the program less dependent on other programs running on the same system (but not completely independent, see the discussion below). As the kernel measures the user time by sampling on timer ticks, this method can be circumvented by processes which tend to work in short time quanta and sleep in the meantime, but in our case no syscalls for sleeping should be available.

The second method is sometimes used for interactive tasks, where substantial amounts of time can be spent by waiting for the system's reply and a deadlock is possible when violating the protocol. Proper accounting for communication delays and deadlock detection are obviously preferable, but they are not always easy to perform.

3.2. *The Scripts*

Except for the sandbox, the rest of the evaluator is implemented as a set of scripts for the Bourne shell. This can sound strange at first, but as most of the job is just gluing small parts together, the shell is often a better tool than a "real" programming language.

As we have already mentioned, the design is modular. We have a library of shell functions serving as building blocks for performing the basic tasks: compilation of the contestant's solution, preparing inputs, running it inside a sandbox, fetching and validation of the program's outputs. The evaluator itself (the front-end used by the organizers to process the solutions) is then a simple script which just calls the building blocks in the right order.

All modules also record whatever they are doing in a common log file, which can be later examined by the contestants to understand what they have done wrong.

3.3. *Configuration*

All the building blocks are highly configurable. The configuration variables range from simple settings like the table of compilers and their options for various languages, or time and memory limits, to commands run to perform various tasks (for example verification of correctness of program output).

The configuration files are again shell scripts. Their primary task is to set environment variables corresponding to the configuration settings. First, a global configuration file for the contest is loaded (it usually defines the basic parameters like the compilers and also provides defaults for all other settings), then it can be modified by per-task configuration and finally even individual test cases can override anything (for example, different time limits can be used for different test cases).

3.4. *Task Types*

The standard building blocks know how to handle the usual types of contest tasks. If the task is of one of these types, it is sufficient to set the corresponding parameters in the configuration. You can of course define your own task types by providing a couple of shell functions or scripts.

The most common task type is the *input/output task.* The tested program is given an input file and it produces an output file. Alternatively, communication by standard input and output can be used. A judge program specified in the configuration is then run to check the correctness of the output and its exit code determines the outcome. The default judge is just a call of the diff utility, set up to compare the file with the (unique) correct output, ignoring differences in white space. The judge can also write messages to its standard error output, which become a part of the evaluation log.

Other tasks can be *interactive.* Such tasks are for example games, which communicate on-line with an opponent played by the judge. In this mode, the evaluator runs the tested program and the judge in parallel and it connects the standard output of the program to the standard input of the judge and vice versa. The exit code and the error output of the judge are again used to determine the outcome and log messages. We usually wrap the protocol between the program and the judge in a library linked to the evaluated programs, so that the contestants do not have to take care about details of the communication protocol and proper flushing of I/O buffers.

The third group of standard tasks are the *open-data tasks,* in which the testing data are public and the contestants submit only the output files, which they can obtain in whatever way they wish. In this mode, the modules taking care of compilation and running of the solution are replaced by a simple fetching of the submitted output file and the file is then processed like in an I/O task.

3.5. *Hacks*

The flexibility of a system can be probably best judged by its applications to situations unknown at the time of its design. Here we show a couple of such "hacks".

At one of the previous preparation camps, we had an *approximation task* and the points were determined by the quality of the approximation (as we did not know the exact optimum, the quality was measured relative to the maximum of the best of the contestants' solutions and our program). This does not fit well the structure of the evaluator, because it is run separately for different contestants. However, we can take advantage of the simple interface between the evaluator and the rest of the contest systems – for every contestant, the evaluator creates a simple text file containing points and judge's verdicts for all test cases. Hence we can let the judge check the validity of the output and record its value in the verdict and after evaluating all contestants plug in a simple program, which will read all verdicts and recalculate the scores appropriately.

This method can be also used for *grouping of test cases* (we want to avoid awarding points to programs which always print "No solution", so we combine test cases of roughly the same complexity to groups and award points only if the whole group is answered correctly). We let the evaluator assign points for individual test cases and then a grouping module is run, which calculates the group scores. As the grouping technique is becoming commonplace, this module will be moved to the library of standard modules soon.

Another interesting application is *pitting* the solutions against each other if the task is a two-player game. Instead of playing against a judge provided by the organizers, we simulate a tournament in which all possible pairs of matches are played and then the points are assigned according to the outcome of the tournament. This again does not fit in the framework directly, but we can replace the default interface of the evaluator by a simple program (in fact, it was something like 30 lines of shell script), which will use the standard modules to compile the programs, run them in the respective sandboxes and connect them together through a judge, which will make sure that everybody follows the rules of the game and the communication protocol.

4. Perspectives

Our contest environment has proven itself useful in multiple contests, but it is far from being the final word on the subject. Several questions keep arising and the answers will shape the future contest environments (and also future versions of ours).

4.1. *Time Measurement*

The computers are becoming gradually faster and the traditional one-second resolution of time limits requires still larger input data to distinguish between efficient and slow solutions. Also, the speed of the processor is increasing faster than the speed of disks, so with larger inputs the proportion of time spent by reading the input increases.

The obvious solution is of course increasing the timer resolution and use sub-second timing, but it is not so easy as it might seem, because there is a lot of noise in the time measurements, which suddenly becomes very visible on this scale. Many different factors contribute to the noise, the most important of them being caches (both the code and data caches of the processor and the disk cache of the operating system). The initial contents of caches when the program is started are unpredictable and the algorithms controlling cache operations are usually very complex and they involve hidden variables. For example, most data caches are set-associative and indexed by physical addresses, so the efficiency of the caches is influenced by placement of the pages of memory allocated by the program in the physical address space of the processor. While it is very rare for the cache effects to cause slowdown on the order of magnitude (even this has been observed, but not in a contest task), the effects are large enough to become a significant factor in millisecond time measurements.

We can try to use the standard engineering techniques to deal with the noise: repeat the measurements several times and take a minimum or an average of the values, or try to make the initial state more predictable (e.g., by letting the evaluator pre-read the input file to increase the probability of it being present in the disk cache; by the way, in our evaluator this is a pleasant side-effect of copying the input file to the directory accessible to the sandbox just before running the program). This helps to eliminate the most visible effects, but definitely not all of them, since the physical addresses used in the different testing runs will very likely be correlated.

An interesting approach has been recently suggested by Szymon Acedanski and tested at CPSPC 2006 in Warszawa. Instead of measuring the time, we can count the number of instructions executed in the user mode of the processor. This can be easily accomplished by a simple kernel patch using the performance counters of the processor. The instruction count corresponds to the execution time on an idealized computer and it is not influenced by any cache effects nor other sources of noise, so the resolution can be arbitrarily fine. The only drawback is that it could hide more implementation details than we would wish – for example, this model makes integer addition and floating-point multiplication equally expensive, which might not be desired.

We plan to implement this type of timing in our environment to give it more field testing.

4.2. *Inputs and Outputs*

Large input files are necessary not only because of precision of timing, but also when we want to distinguish between similar time complexities, especially between linear and linear-logarithmic one. As already mentioned, this leads to a big fraction of time being spent on parsing of files and it also significantly hurts contestants using the slower I/O libraries (for example, we have seen several tasks which are impossible to solve if one uses C++ streams, because the stream library is unable to read the input within the time limit).

Experienced authors of tasks often take this problem into account and they use tighter encodings of inputs, like describing a tree by its traversal sequence. This helps in some cases, but it is far from being a universal technique. It can also unnecessarily complicate parsing of inputs.

One possibility (again suggested by Szymon Acedanski) is to ask the contestants to use a special library for reading the input, which provides functions for reading values of all standard types. These functions consume input files preprocessed to a special binary format, which saves most of the reading and parsing overhead. However, the contestants have to learn the new functions.

A similar effect with less complications for the contestants could be achieved by making the evaluation system parse the input before the clock starts and providing it to the program in global variables. The program can then access its input data by simply linking with a library. This trick of course leaves out an interesting class of problems – the streaming problems, where the input is larger than available memory and it has to be processed sequentially. On the other hand, we can view such problems as a special case of interactive tasks and also hide the implementation of input and output in a library.

It is not clear if this format of input is the best answer to the problem, but it is definitely worth trying and we plan to experiment with it in our framework in the near future.

References

Forišek, M. (2006). Security of programming contest systems. In *Informatics in Secondary Schools, Evolution and Perspectives*, Vilnius, Lithuania.

Mareš, M. (2007). The MO-eval web site.
`http://mj.ucw.cz/mo-eval/`

M. Mareš is a doctoral student at the Department of Applied Mathematics of Faculty of Mathematics and Physics of the Charles University in Prague, a researcher at the Institute for Theoretical Computer Science of the same faculty, organizer of several Czech programming contests, member of the IOI Scientific Committee and a Linux hacker.

Olympiads in Informatics, 2007, Vol. 1, 131–140
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Tasks at Kyrgyzstani Olympiads in Informatics: Experience and Proposals

Pavel S. PANKOV

International University of Kyrgyzstan
A. Sydykov str., 252, Apt. 10, Bishkek, 720001 Kyrgyzstan
e-mail: pps50ster@gmail.com, pps50@rambler.ru

Timur R. ORUSKULOV

Ministry of Education and Science of Kyrgyzstan
Vostok 5, 14/2, Apt. 3, Bishkek, 720065 Kyrgyzstan
e-mail: toruskulov@mail.ru

Abstract. Many of tasks proposed at Olympiads in Informatics (OI) mean “images” in any sense but in most cases they do not appear evidently. At each OI in Kyrgyzstan, one task is given to present any graphical image, either in text mode or in graphical mode. Such tasks meet the Statute S1.7 of the IOI Regulations “to bring the discipline of Informatics to the attention of young people”, make OIs more attractive for sponsors, can reflect state and national features. Ways to generate and to score such tasks and presentation of some tasks at the Conference is supposed. The history and content of OIs and teaching Informatics in Kyrgyzstan are also described.

Key words: olympiads in Informatics, Kyrgyzstan, history, graphics.

1. Survey of Teaching Informatics in Kyrgyzstan

Informatics (under the traditional name “Foundations of Informatics and Computer Facilities”) is taught in all secondary schools of Kyrgyzstan since autumn 1985. Firstly it was taught “with chalk on a blackboard” and calculators and pupils sometimes visited local computer centers. Further several were put in some schools. Now almost all schools have classes with IBM-compatible computers, some of them linked up with WWW.

By the State standard, now Informatics is taught obligatorily in 7th (1 hour a week), 8th and 9th (2 hours a week) forms. Decision to teach or not Informatics in elder (10th and 11th) forms is given to schools. Many schools, all lyceums and gymnasiums themselves introduce Informatics lessons in elder forms. Now, a new curriculum is elaborated. It provides teaching Informatics in all forms of primary and secondary school, from 1st till 11th ones.

2. Survey of Olympiads in Informatics in Kyrgyzstan

The Olympiads in Bishkek city, the capital of Kyrgyzstan, are conducted since 1985 (annually in January); National ones are conducted from 1987 (annually in March). The contestants on all four levels: I (school); II (district or area); III (city or region); and IV (National) are divided into two groups: the first one includes 10th form schoolchildren (16 years old and younger) and the second one does 11th form schoolchildren (17 years old).

Each of the seven regions, Bishkek city as a capital and Osh city as "a southern capital" send 2 pupils in each group of the IV level. So, 19–20 pupils (including winners of the preceding year) participate in the first group and 17–18 pupils do in the second one.

Because of essential differences between traditional contents of IOI and ones of our Olympiads (see below), Spring camps with final Selective competitions to IOI are conducted for all better contestants (6–8 of the first group and 3–4 of the second one) of the IV level (annually in April).

The IV level is conducted by the Ministry of Education and Science at the Kyrgyz National University or at the Kyrgyz State Technical University with support by sponsors. Spring camps are conducted at the National Computer Gymnasium.

Also, some universities conduct Olympiads and other kinds of competitions in Informatics (irregularly) for their students and for schoolchildren to attract entrants.

Kyrgyzstan participates in IOI since 2000. Our achievements are three bronze medals won by A. Mokhov (IOI 2000), A. Baryshnikov (IOI 2004), I. Goroshko (IOI 2005).

3. Content of Olympiads

The IV level of Olympiad includes two rounds: "theoretical" (by pen and paper; two tasks; 2.5 hours) and "practical" (by computer; three tasks; 2.5 hours).

The theoretical round yields the opportunity to diversify content of Olympiads and to involve items of Informatics which cannot be covered by tasks for computer implementation.

Most of tasks proposed at theoretical rounds may be classified as follows (Pankov *et al.*, 2000).

T1) A goal is described. To write a corresponding algorithm (using operations and conditions which are impossible or too difficult to be implemented at a computer).

T2) To restore (guess) an algorithm and its goal (or possible goals) by examples of results of its work.

T3) To understand (guess) the goal (or possible goals) of a given algorithm (also described with non-formal operations and conditions) and to improve it. The output of the algorithm may be "graphical" also.

We announce that, to obtain a full score, the contestant has to write a (same) algorithm in two languages. There can be a natural language, flow-charts, program in an algorithmic language (possibly, with additional non-formal operations), vast comments to a program, and so on.

REMARK 1. In T3 case, the given algorithm must have a simple goal but could be written very complicatedly, with unnecessary operations and conditions, cycling or infinite (until overflow) branches.

REMARK 2. In T2 and T3 cases, the jury must have their own version about the goal but contestants' different guesses can obtain full score if they are witty and meet the data.

Most of tasks proposed at practical rounds are standard:

P1) A goal and restrictions on input are described. To write a program transforming input into output in a limited time (5 seconds) due to this goal.

In one of three tasks, the output must either be graphical or imitate graphics in a text mode (see the detailed description below).

To diversify scope of tasks, the following types of tasks are also proposed.

P2) To write a program for P1 with restricted means (for example, comparison only letter-by-letter). Hence, the jury is to verify the listing too.

P3) A very slow (possibly, non-formal) algorithm and restrictions on input are described. To write a program being equivalent to it and working in a limited time.

P4) "Black box". A program is given as an exe-file (a contestant may run it as many times as she wishes). To write a program being equivalent to the given one. (It means that the given program is sufficiently simple, does not contain large numbers and complex algebraic expressions).

REMARK 3. At each Olympiad, some tasks must be devoted to state symbols or other features, algorithms in Kyrgyz language, sponsors (their logos, business, addresses etc.), number of the year, events of the year, geography of Kyrgyzstan (Pankov *et al.*, 2003).

Thus, some tasks of our Olympiads demand erudition, knowledge in other subjects, such as physics, chemistry, geography, philology.

Tasks in Selective competitions correspond to traditional scope of IOIs: long arithmetic, combinatory, graphs, discrete optimization, moving along rectangular grid (points with integer coordinates) and polygons on it, embedding of words, cubes with integer sides (Pankov, 2004).

REMARK 4. Simple in sense and interesting tasks for graphs are generated by involving of "moving" objects differing from a point (two points which are prohibited to meet, "train", "worm" etc.).

4. Graphical Tasks

Many of tasks proposed at Olympiads in Informatics (on graphs, rectangular grid, counting of geometrical figures, coverings, packings) mean "images" in any sense, and examples to them are given in a graphical form but the output is textual. For example, see

(Pankov *et al.*, 2005). (Also, while solving a task, a contestant can program an additional graphical output for their own use, to avoid rough mistakes).

We shall not consider such tasks; we mean "graphical" tasks as ones with graphical output. By our opinion, such tasks meet the Statute S1.7 of the IOI Regulations "to bring the discipline of Informatics to the attention of young people". By our experience, such tasks make Olympiads more attractive for sponsors and can reflect state and national features.

By our opinion, tasks themselves ought to contain text only; graphical images can be in examples only.

4.1. *Possible Content of Graphical Tasks*

We propose the following standard and non-standard ways to elaborate graphical tasks (including interactive tasks, animated cartoons).

G1) A simple drawing (an element of state symbols, a sponsor's logo etc.) is described verbally; either its dimensions (numbers) or some characteristic points (on a display) are input.

Such a description of an image is a base for more complex tasks. All motions (transformations) mentioned below must be continuous and slow. "User" means the member of jury verifying the solution.

G1A) Firstly, all display "is covered with snow". If User "erases snow" with a cursor then the image appears. (More generally, the image changes under a cursor).

G1B) The image moves (transforms) due to demanding of the task. The image can also be "larger" than the display; then its parts appear successfully.

G1C) Additionally, firstly User inputs a (very simple) drawing (one or two segments); after input it transforms into the image.

G2) The image is being built successfully of elements (letters, points, segments) input by User arbitrarily until the image is completed; if the input element does not meet the condition then it is rejected.

G3) The image is to reflect any mathematical object being an input and/or a solution of the task.

G4) The initial image presents a base for the task. User inputs a data for the task; the program adds a presentation of an (optimal, close to optimal or arbitrary) solution of the task to the image.

There is also a kind of mathematical tasks which do not demand graphics explicitly but can be solved effectively by graphical and pixel methods rather than mathematical ones:

GD) A geometrical image is defined in any way. Find a distance between two mentioned points with a low accuracy (2–5%).

GA) . . . Find an area of any mentioned part of the image with a low accuracy (5–10%).

4.2. *Composition and Scoring of Graphical Tasks*

To make a task interesting and accessible for contestants of various levels, it may be subdivided into stages and/or alternative subtasks in increasing order of complexity, with corresponding scoring of each stage. Let the total score be 10 points.

EXAMPLE 1. The first stage of a task of type G1B must be a task of type G1 (1–2 points).

EXAMPLE 2. The first stage of a task of type GL or GA may be a task to present the graphical image itself (4–5 points).

EXAMPLE 3. If letters are to be shown then it can be done: in text mode (1 point), with segments (5 points) or with segments and arcs (10 points).

5. Examples of Tasks

Most of tasks given below are very simple. We give them as examples rather than standards.

Task 1 (T2). Any algorithm transforms certain words containing the letters B, M, P. For instance, the algorithm transforms the word PBPBMB into PB, MBMBPBMB into MBMB, MBPBPBMBPB into PB, PBPB into PBPB, MBPBPBPB into PBPB. For certain words, for instance, PPB, MBMP, BBPBPMPB, the algorithm gives “error”. A) What sense may be in this? B) Write such algorithm.

Comments. Contestants gave different “right” answers to A): “algebraic simplifying: $+$B$+$B-B$+$B$=$ $+$B$+$B”; “Annihilation of particles in nuclear physics” etc.

From these examples a contestant cannot guess, what does a word of type PBMBMBPB transforms to? Thus, any non-empty respond demonstrating paying attention to it was considered to be right.

Task 2 (T2). While working the algorithm elaborates a sequence of natural numbers; the last number is the output.

Ex.1. 5000, 2000, 1000, 1500, 1700, 1600, 1550, 1520, 1510, 1515, 1516, 1517, 1516.

Ex.2. 5000, 2000, 1000, 500, 700, 900, 950, 970, 990, 995, 997, 998, 999, 998.

A) What sense may be in this? B) Write such algorithm.

Jury’s version: it is a process of weighing with the traditional (decimal) set of weights: 1, 2, 2, 5, 10, 20, 20, 50, . . ., with rounding down.

From these examples a contestant must guess that an input is a positive number. Also, s/he cannot guess, what is the heaviest weight in the set? Thus, any solution demonstrating paying attention to it was considered to be right.

Task 3 (T3) (2003). A) What goal may this algorithm have? (A possible goal must be defined by the initial (instead of simplified) text of algorithm). B) Is it possible to simplify or improve it (with the same input and output)?

```
Let $X := 90$; $Y := 53$; $T := -201$; Output $X, Y, T$.
M1) Let $T := T + 38$; Input $B$.
If $B > 0$ then let $X := X - 1/2$ else $Y := Y - 1/2$.
If $X < 72$ then let $X := X + 1/2$ and $Y := Y - 1/2$.
If $Y < 42$ then let $X := X - 1/2$ and $Y := Y + 1/2$.
If $X > 72$ or $Y > 42$ then go to M1.
Output $X$, $Y$, $T$; End.
```

Answer B). A version of simplified algorithm:

```
Let $X := 90$; $Y := 53$; $T := -201$; Output $X, Y, T$.
M) Input (any) numbers 58 times.
Let $X := 72$; $Y := 42$; $T := 2003$; Output $X, Y, T$; End.
```

Comments. Analysis demonstrates that values of numbers B do not influence to the final result. But it is stressed in the condition of the task, that the simplified algorithm should have the same input and output, as initial one. Therefore the jury reduced points to those who had missed a statement of type M) in simplified algorithms.

A) As the final value of T is a year (2003 A.D.), we may suppose that the initial value of ' is any historical date (201 B.C.). It is possible to recollect, that it is the date of first known mention of ethnonym "Kyrgyz". Hence, the algorithm shows the movement of Kyrgyzes at time from Altai Mountains up to Tien Shan Mountains, in general south-west direction. Then X, Y are, likely, coordinates (geographical coordinates). One may guess, that X is longitude and Y is latitude [we have chosen the center of Khakassia as the initial point and Talas valley as the final point of wandering]. Inputs of numbers B denote orders or historical circumstances pushing people in various directions (once during each generation).

As much is unknown in the history of Kyrgyzes, this algorithm, irrespective of input numbers (defining coordinates of intermediate points) yields the same output.

Task 4 (GL). Solar disk S with forty uniformly radiating beams is placed in the center of the Kyrgyzstan national flag, on the red background. An image of red "tyundyuk" T is placed inside S. The ratio of length L of a flag to its width W is equal to $5 : 3$. The diameter D of forty-beams circle is equal to $3/5 * W$, the diameter E of S is equal to $3/5 * D$. The diameter F of T is equal to $1/2 * D$.

REMARK. An image of "tyundyuk" (top window of transportable felt house – "yurta") consists of thin ring and six bent lines crossing the ring.

Let us to mark and renumber some points at the flag:

The middle of a left edge of a flag is #1; the end of the left beam is #2, the left edge of Sun is #3, the right edge of "tyundyuk" is #4, the end of the right beam is #5; the bottom edge of Sun is #6, the top right corner of a flag is #7.

To write a program to find the area of a flag with an error less than 1% if A) given the number K ($K \leqslant 5$) and the distance H between the center of a flag and the point

#K (positive number), or B) given two numbers $K \neq J$ ($K, J \leqslant 7$) and the distance H between the point #K and the point #J (positive number).

Comment. Although the goal is to find an area, this task is of type (GL) because the length of the flag defines its area evidently.

The simplest solution by means of computer. Choose size and arrangement of the flag on the coordinate plane arbitrarily; find its area G_1; calculate coordinates of the listed points (denote them as $P[K]$, $K = 1..7$). To solve A) and B) uniformly, denote the center of the flag as $P[0]$. Given two integers $K \neq J$ in [0..7] and $H > 0$, calculate H_1 :=(distance between $P[K]$ and $P[J]$), and the answer $G := G_1 * (H/H_1)^2$.

For example, let $P[0] := (0,0)$; $L := 2$. Then $W = 2 * 3/5$, $D = 3/5 * W$, $E := 3/5 * D$, $F := 1/2 * D$; $G_1 := L * W$; $P[1] := (-1,0)$; $P[2] := (-D/2,0)$; $P[3] := (-E/2,0)$; $P[4] := (F/2,0)$; $P[5] := (D/2,0)$; $P[6] := (0,-E/2)$; $P[7] := (1,W/2)$.

Task 5 (G1B) Suppose that we are above the Earth and we can see Bishkek and Istanbul only through clouds (as two little figures on a homogeneous background). What shall we see (how will the view change) if we move (at our will):

A) downwards (up to clouds);

B) either to East or to West and as in item A);

C) either to North or to South and as in items A) and B)?

REMARK. “Istanbul” is mentioned because there are some Kyrgyz-Turkish lyceums in Kyrgyzstan and their directorate “Sebat” supports Olympiads in Informatics.

Task 6 (G4) (2002). Let Lake looks like an isosceles triangle, the basis of the triangle (northern coast) is 190 km and height (width of Lake) is 60 km. Village is located on northern coast of Lake at 20 km from the western corner. Horse runs with speed of 20 km/hour and swims with speed of 10 km/hour. Write a program: A) to show Lake and Village; B) to enable User to show any point on the coast of Lake; C) to draw the fastest way for Horse or D) to show the motion (in scale of 1 hour =1 sec.) of Horse from this point up to Village along such way.

Scoring: 1 point for A); 2 points for B); 3 points for C); 7 points for D).

Comments. 2002 was the year of Horse. The task reflects a historic fact. This village was named after Horse which had crossed the Issyk-Kul lake in XVIII century.

Task 7 (G1–GA). On midday, near the Ataturk-Alatoo University [a sponsor], a photographer realized that one of mountain peaks will be soon seen exactly on the center of the Sun [element of the insignia of Kyrgyzstan]. He wishes to choose a colored photo for the newspaper “Vecherniy (Evening) Bishkek” [a sponsor]. The seen diameter of the Sun is half degree.

Given the number of seconds passed after “touching” the mountainside by the Sun. Write a program A) performing the sight of the Sun and the mountain; B) calculating the percent of the seen part of the Sun’s disk (with the accuracy 10%).

Present the mountain as a symmetrical right angle. The seen diameter of the Sun disc is half degree.

Comments. Beginning of solution of A). The Sun passes its diameter during $24 * 0.5/360\text{hours} = 120$ seconds. It moves horizontally (on midday) and from left to right (in Kyrgyzstan). When the Sun's disk "touches" the mountainside the "distance" between the center of the Sun and the peak is $60\sqrt{2} = 85$ seconds.

Mathematical solution for B) is complicated but the contestant can do as follows. Choose an acceptable diameter D of the Sun's disk (50–70 pixels); count the number N of pixels in it and include N into the program; after performing the image on a display count the number N_1 of pixels in the seen part of the Sun's disk (the number of yellow pixels in the square with the side D circumscribing the Sun's disk); output the number $N_1/N * 100$ (percents).

Task 8 (G4). Given a natural number N ($13 \leqslant N \leqslant 200$), and two natural numbers L, M ($1 \leqslant L < M \leqslant 10$). Arrange N points with integer coordinates on a plane in such a way that the number of distances between pairs of them that are in the interval $[L, M]$ to be as large as possible. The program must give the following output:

A) a file with:

1) N lines with the list of numbers and coordinates of N points. Each line contains number of point (from 1 to N), its X-coordinate and Y-coordinate. All points must be different. The coordinates must be natural numbers less than 800.

2) One line with number (denote it as K) of pairs of points having distance in the interval $[L, M]$.

3) Corresponding K lines with the list of these pairs of points. Each line contains the number of the pair and the numbers of points in pair (the second must be greater than the first). All pairs must be different.

B) Arrangement of these points replaced by little stars on a display.

Scoring: $1 + 9*$(NumPairsInYourAnswer/NumPairsInBestAnswer)^2 rounded down.

(Such a task was submitted to IOI'2003 but was not accepted).

Comment. This task looks like a continuous one but is discrete.

6. Proposals and Conclusion

At IOI'2006, in Merida Zide Du, President of the IOI, proposed the idea of involving graphics into IOI. Certainly, most of types of graphical tasks mentioned above do not suit responsible international competitions. We propose tasks of type G3 (only task each day). If it is possible, images would be related to any reality: reminiscences of the host country or sponsors. Also, knowledge and skills necessary to fulfill the task must be obvious for contestants.

The following items are to diminish subjectivity and non-automation in scoring.

i) The score for a graphical presentation is fixed: about 30% points; other 70% points are distributed in a common way.

ii) These 30% points are given alternatively: 30 or 0.

iii) The quantity and content of graphical images are fixed: the first image presents the initial data and the second one presents the result.

iv) The only test (G-test below) in the task for graphical presentation is to be as simple as possible.

v) The initial data for the G-test have to be announced as fully as possible.

vi) The size and other parameters of a graphical image in the G-test are to be described in details, for instance “600×400 pixels etc.”

vii) If the goal of task is an optimization then the contestant’s program for the G-test must give a result not less than 50% of the best result if it is known or of the best result of all contestants (i.e., the score for the test itself may be zero, but the score for the graphical presentation may be 30% points).

viii) The following scoring procedure is proposed:

- while submitting the task the contestant announces whether the program has the option to generate graphical images (only within the range announced in v));
- if the G-test is passed (may be, not with the full score) then the Scoring Program shows the graphical images to two or three members of jury. If they (independently) confirm that these graphical images are proper or not proper then the Scoring Program adds or does not add 30% points; if their opinions are different then after this procedure they gather together and discuss all (only few) ambiguous programs (still they do not know contestants-authors of these programs).

We hope that graduated implementation of graphical tasks into Olympiads of top levels will make them more interesting for young people and attractive for prospective sponsors. Also, demonstration of best works can decorate closing ceremonies and publications on Olympiads.

References

Pankov, P.S., T.R. Oruskulov, G.G. Miroshnichenko (2000). *School Olympiads in Informatics* (*1985–2000 years*). Bishkek (in Kyrgyz & Russian).

Pankov, P.S., T.R. Oruskulov, G.G. Miroshnichenko (2003). *Olympiad tasks in Informatics, devoted to Kirghiz statehood, history of Kyrgyzstan and Great Silk road*. Bishkek [in Kyrgyz & Russian].

Pankov, P.S. (2004). Preparing for international Olympiads in Informatics. *Bulletin of the Kyrgyz National University*, **6**(4), 52–54.

Pankov, P., S. Acedanski, J. Pawlewicz (2005). Polish flag. In *The 17th International Olympiad in Informatics* (*IOI’2005*). *Tasks and Solutions*. Nowy Sacz, pp. 19–23.

P. S. Pankov (1950), doctor of physical-math. sciences, prof., corr. member of Kyrgyzstani National Academy of Sciences (KR NAS), is the chairman of jury of Bishkek City OIs since 1985, of National OIs since 1987, the leader of Kyrgyzstani teams at IOIs since 2002. Graduated from the Kyrgyz State University in 1969, is a main research worker of Institute of Mathematics of KR NAS, a manager of chair of the International University of Kyrgyzstan.

T. R. Oruskulov (1954), candidate of pedagogical sciences, is the deputy chairman of jury of Bishkek City OIs and of National OIs since 1988, the deputy leader of Kyrgyzstani teams at IOIs since 2005. Graduated from the Frunze (now Bishkek) Polytechnic Institute in 1977, worked in Institute of Automatics of KR NAS, Kyrgyz Research Institute of Pedagogy, studied at Post Graduate office of the USSR Academy of Pedagogy in 1988–1991, was a professor of Kyrgyz Academy of Education, works in the II Project of Education of the Asian Development Bank at Kyrgyzstani Ministry of Education and Sceince.

Olympiads in Informatics, 2007, Vol. 1, 141–148
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Computer Science Contests in Germany

Wolfgang POHL
Bundeswettbewerb Informatik
Ahrstr. 45, 53175 Bonn, Germany
e-mail: pohl@bwinf.de

Abstract. In Germany, the preconditions for running a successful Computer Science contest for secondary school students are not perfect. However, many contests that are related to the area of Computer Science are run by as many different organizations. The Federal Contest in Computer Science (German: Bundeswettbewerb Informatik, short: BWINF), a task-based contest with two homework rounds and a symposium-style final round, is the most important, nation-wide contest in the field. BWINF office is also responsible for Germany's IOI team selection and participation. In addition, in recent years the office has been running several other projects to popularize Computer Science and promote talents in this field.

Key words: computer science contests, tasks, grading, teacing informatics.

1. Introduction

1.1. *Preconditions for Contests*

There are three important observations to be made when looking at the preconditions for running contests for (secondary) school students. First and foremost, the responsibility for the school system is not held by the federal government, but by the governments of the 16 federal states. Some basic properties of the school system hold nation-wide, but especially in the details and for non-standard subjects like Informatics, the situation may vary significantly between the states. Many educational initiatives are run in a single state only, and there are many state-wide student contests, often in parallel and sometimes even in conflict with nation-wide contest activities. Recently, the German constitution was changed to make sure that the state governments have full responsibility for school education. The federal government was left with some influence on academic education and was allowed to continue its activities in the area of promotion of the gifted. Hence, the federally supported nation-wide contests can be continued.

Second, during several decades many teachers and schools in Germany focussed on leveling education to the intellectual capabilities of the majority of students. At the same time, promotion of the gifted almost was a non-issue. Only in recent years, this situation has improved. In addition, many schools are striving to develop and demonstrate their special profile. This has led to an increased popularity of contests accessible to many students, like the "Math Kangaroo". However, these contests do not naturally lead to participation in more advanced contests like national olympiads.

Third, the lack of a centralized contest organization led to the birth of a huge variety of contests, many of which are organized by non-governmental institutions or industry companies. In such an environment, it is often difficult for teachers to recommend good contests. Teachers complain about a “contest flood” they can no longer cope with. Many company contests are organized only as a public relations activity, but typically this is well hidden. In addition, contests with government funding cannot compete with the awards and prizes that contests with funding from companies or private institutions can offer.

In summary, participation in a nation-wide, government-funded contest are in no way a standard activity for German school students. Therefore, only very few contests see really high participation rates.

1.2. *Informatics at School*

The subject of Informatics is an optional subject in most federal states. Exceptions are seen in three of the 16 states only. Informatics is mainly taught in the final years of Grammar School, but nevertheless it can be difficult to choose Informatics as a subject that will become part of the final school exam “Abitur”. Due to the low importance of Informatics, there are only few teachers with a genuine education in Informatics; most Informatics teachers have a supplementary education in Informatics. This situation given, it is no surprise that Informatics is not one of the popular subjects at school. Among girls, it is even one of the least popular subjects at all.

1.3. *Contests in Computer Science*

In the next section, we will describe “Bundeswettbewerb Informatik” (short: BWINF; Engl.: Federal Contest in Computer Science; see also BWI), a nation-wide contest, run by the German Informatics Society together with the Fraunhofer association of information and communication technology research institutes. It is mainly funded by the federal government and hence the “official” German contest in the field. However, it is by far not the only contest for German secondary school students related to Computer Science: There are multi-media contests, software project contests run by industry or non-profit institutions, and several robot construction and programming contests (like Robocup Junior). However, BWINF is probably the contest with the highest reputation; BWINF winners receive a university scholarship from a national foundation. The contest next important to BWINF is the combined Mathematics/Computer Science section of “Jugend forscht” (Young Scientists). In this competition, participants present their own inventions or research projects. For a more detailed overview (in German) about related contests, see (Pohl, 2005).

2. Bundeswettbewerb Informatik

In 1980, Informatics was not yet established as a school subject. This gap was to be filled with a nation-wide contest. Its first issue was called “Youth Programming Contest”;

the winner[1] was celebrated at an IFIP congress in Lausanne in 1989. In terms of Pohl (2006), the contest began as project contest; participants were asked to describe their own programming projects. The third contest was the first one to be called “Bundeswettbewerb Informatik”, and from the fourth contest, the contest model was changed to the one that is still being applied today.

2.1. *BWINF Procedure*

There is an office consisting of a full-time manager (that’s me) and a half-time secretary, paid to run the contest, organize and execute IOI (and CEOI) training and participation. The resources of the office are pretty much consumed by organizational, administrative and public relation work – since it is not standard for German students to participate in contests, each contest needs advertising, given the situation that there is such a huge quantity of contests and competitions. Furthermore, there is a volunteer committee for task creation and selection, and a steering committee with representatives from the organizing institutions as well as the federal and state ministries. In the first two rounds, the jury is composed of students (some of them former contestants), who are paid a small fee. In the final round, committee members and experts from both school and academia are called into the jury.

The contest is organized in three rounds. Contest task sheets and posters are sent to all secondary schools, and the BWINF office also tries to announce the contest in the media. Students register by sending their solutions. They work at home, individually or in (preferably small) teams. In general, five tasks are given, and if three of them are solved, the students qualify for the second round. In this round, students still work at home, but must solve the problems on their own. The best of the second round (ca. 30 people) are invited to take part in the final round, which is organized as a meeting. The participants of this final round are interviewed individually by jury members, and, on two days, need to work in a group on one problem each day. Mainly because it is a homework contest, it is run during a whole year; often, the new contest starts before the final round of the running contest has been completed.

2.2. *Characterizing BWINF*

First, it is most easy to say what BWINF is *not*: In contrast to the International Olympiad in Informatics, its regional variations like CEOI, Baltic OI, etc., and many national olympiads (confer Pohl, 2004), it is not a contest with programming exams and a thematical focus an algorithmics. and to a task contest with long-time homework rounds, manual jury grading and submission of both executable programs and written solution descriptions. Following the categories that were suggested by Pohl (2006a) to describe contests in the area of Computer Science, it has the following properties:

Scientific Area As a contest for secondary school students, BWINF needs to cater for the possible Computer Science expertise of those students. As already mentioned,

[1]The first winner, Otfried Schwarzkopf, now is professor for theoretical Computer Science in Korea.

Informatics is not an obligatory subject in German schools and not very popular as well, there is no standard level of expertise that can be required. As we know, many contestants are self-taught only, while the students of a few specialized schools learn about Informatics on a high level already. Therefore, the contest needs to set its own standards, but at the same time should consider the most important topics of school Informatics. As a consequence, BWINF has basic algorithms as its core area, but also deals with information modeling, data bases, simulation processes, language and text processing, and other areas of Computer Science.

Style BWINF is a task contest; in all rounds, tasks are given.

Duration In the first two rounds, contestants have about two and four months, resp., to work on the tasks. In the first round, five to six tasks are given, and solutions to at least three need to be submitted in order to qualify for the second round. In the second round, three tasks are given, two of which the contestants need to select and work on. The third round, the final, is different: On each of two days, contestants work in groups of four for about four hours on one single task. In addition, each individual contestant has two interviews, each with one jury member.

Grading In all rounds, there is manual grading by jury members. In the first two rounds, jury members follow given grading schemes. For each task, there is a grading scheme that defines a set of crucial issues. For each such issue, its weight and influence on the grading are prescribed. In the final round, jury members need not follow a grading scheme. They translate their assessment of the contestants' performance into a numerical score. In the jury meeting however, where winners are chosen, there is still time for discussions about individual contestants; the addition of scores is regarded as the foundation for the final decisions only.

Submission In the first two rounds, contestants submit written descriptions of their solutions. In the task sheets, a certain structure is recommended for these descriptions: abstract solution ideas first, then a documentation of the most important source code components, followed by examples that demonstrate the program's abilities. Also, source code at least of the main parts of the solution program (no IDE-generated code, no GUI code) needs to be submitted in print. This is to ensure that grading can be performed without executing the solution programs; resource constraints of the grading process do not allow for a genuine testing process. In the task sheets, it is explicitly mentioned that grading may rely on the written material only. The final round, again, is different: Contestants are graded individually based on how they perform during interviews and group work; groups give presentations of their results, but also in an overall weaker group, a single member may have made excellent contributions.

Divisions BWINF is not structured into divisions. In each round, there is only one task set for all contestants. However, there is a slight exception to that rule: In the first round, one task is identified as "junior task"; this task is supposed to be somewhat easier than the others and aims at attracting younger students to the contest. Therefore, it must not be chosen by contestants who are more than 16 years old.

2.3. *25 BWINF Contests – Observations Made*

In 2006, the 25th Bundeswettbewerb Informatik was started. On the occasion of this anniversary, a report about the development of the contest was published (Pohl, 2006b). Several central observations were stated:

- the average participant is a male student in grade 12 of a grammar school;
- female participation is very low, but has slightly increased (!) in recent years to about 5 percent (see Fig. 1);
- while in the early years of the contest, BWINF typically had between 1000 and 2000 participants per year, there is now a fairly stable annual participation rate of about 700 students (see Fig. 2);
- in some federal states, especially from the Eastern part of Germany, participation is

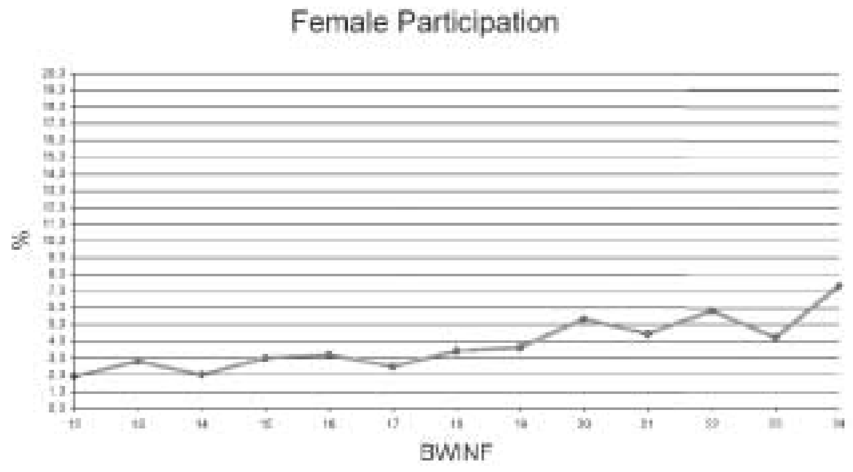


Fig. 1. Female participation rate from 12th to 24th BWINF.

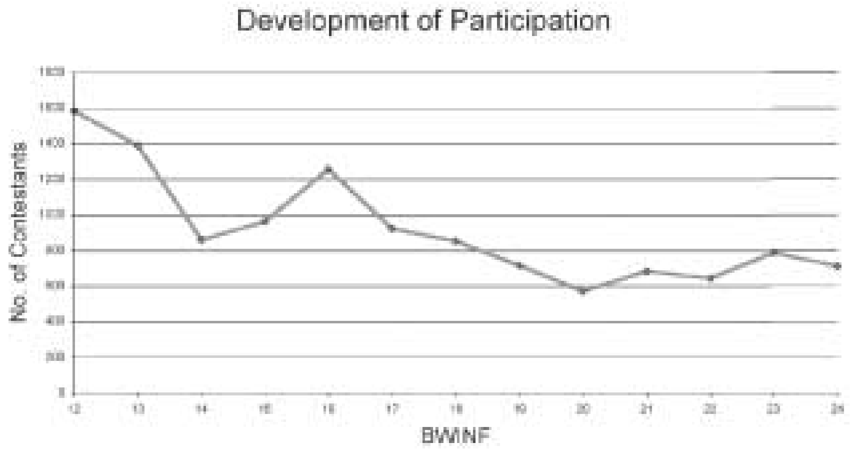


Fig. 2. Number of participants in the first contest round from 12th to 24th BWINF.

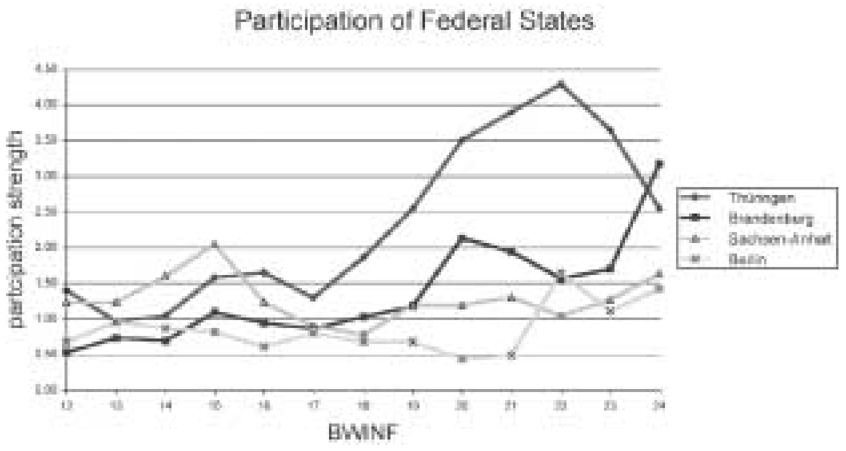


Fig. 3. Federal states with strong participation in BWINF.

very strong, but in the states with large population, participation is relatively weak. This assessment is based on the relation between the state's share in BWINF participants and its share in the overall number of grammar school students in Germany. For instance, a state with a 10 percent share in BWINF participants and a 5 percent share in high school students has a participation strength of 2. Fig. 3 shows the development of participation strength values (from 12th to 24th BWINF) of the states with the highest such values of recent years. Note the significant changes in recent years.

3. IOI Qualification

The BWINF office also is responsible for selecting the German team to participate at the International Olympiad in Informatics (IOI). In addition, German participation at CEOI is supported by the federal government. Since 2001, there has been a delegation at Baltic OIs, too, but Baltic OI participation and (for the first time in 2007) organization must be funded by non-government sponsors.

About a dozen IOI candidates are selected from the final round of BWINF based on if they satisfy IOI age requirements. Some other candidates are selected from the "Jugend forscht" contest mentioned at the beginning. Since both contests are very different from IOI-style contests, three training camps are organized to introduce the candidates to IOI-style problem solving. The training, including the tasks for training exams, are prepared by the BWINF office. Moreover, many former German IOI participants contribute significantly to our IOI training, as well as to organizing olympiads like CEOI and BOI in Germany.

Note that the whole selection process takes place in the year after the final round of BWINF. So, German students go to IOI one year after finishing the national contest and even two years after entering it. As a consequence, many German IOI contestants can participate in IOI only once. In recent years, we have seen a higher number of young contestants in the BWINF final round, and therefore have had a few more two- and even three-year participations in IOI. Since the national contest activities do not give experience in IOI-style contests, OI-experience is a very important factor for IOI-success of German contestants. In addition, the achievements of German IOI participants are very much dependent on their personal ambition. After two years of first national contest and second IOI training, IOI contestants tend to regard their qualifying for the IOI team as main achievement; international comparison is of lesser importance. In such cases, delegation leaders and coaches have difficulties to keep the motivation high.

At IOI, Germany typically sees a good overall team performance, with medals for all or almost all team members. Gold medals, however, are rare; in 18 IOIs, 8 gold medals were won by 8 German contestants.

4. Further Activities

For many years, the BWINF office has been suggesting to extend its activities in popularizing Computer Science among young people and promoting talents in that area. With many German students lacking a solid education in Computer Science, non-school education offers are needed. In 2006, proclaimed as “Year of Computer Science” by the German Federal Minister of Education and Research, the BWINF office was granted the needed resources to run the project “Einstieg Informatik” (First Steps into Computer Science, see (Pohl *et al.*, 2006)), to develop and apply approaches both to teach ideas of Computer Science to children and to provide interested youngsters with information about how to learn more about the subject. The project can be regarded as success: Within 10 months, the project web site attracted about 110.000 visitors. The project gave presentations and shows at public events with an overall number of more than 600.000 visitors, and addressed teachers at conferences with an overall number of about 2000 participants.

Using the resources of the “Einstieg Informatik” project, the first German participation in the International Beaver Contest (Dagiene, 2006) took place in December 2006. Although the resources were not sufficient advertising the contest in the large, more than 2100 students participated. Table 1 gives more detailed information about the participation, divided by grades and gender. Germany’s participation in the Beaver initiative shall

Table 1
Participation in the first German Beaver contest 2006

Grades	**Participants**	**Girls**		**Boys**	
		Number	***Percent***	***Number***	***Percent***
All	2126	698	32.83%	1428	67.17%
5–8	959	394	41.08%	565	58.92%
9 and 10	479	133	27.77%	346	72.23%
11–13	688	171	24.85%	517	75.15%

be continued, while “Einstieg Informatik” was unfortunately discontinued with the end of the “Year of Computer Science”.

5. Conclusion

In spite of the many and diverse contests in the area of Computer Science that students can participate in, universities have seen a strong decrease in the number of students of Computer Science and related subjects. Contests alone are not sufficient to promote a subject and its talents. The BWINF office is currently looking for support of its plans to continue promotional activities like those of “Einstieg Informatik” and set up a virtual community of talents in Computer Science as central entry point for school students who are interested in the subject. Only if such activities are successful, participation in the BWINF contest can be increased again, which in the end might also lead to greater international success of German IOI contestants.

References

BWINF home page. `http://www.bwinf.de/`.
Dagiene, V. (2006). Information technology contests – introduction to computer science in an attractive way. *Informatics in Education*, **5**(1), 37–46.
Pohl, W. (2004). National computer science contests. `http://www.bwinf.de/old-page/olympiade/national-contests.pdf`.
Pohl, W. (2005). Informatik-Wettbewerbe in Deutschland. *LOG IN*, **133**, 10–23.
Pohl, W. (2006a). Computer science contests for secondary school students: Approaches to classification. *Informatics in Education*, **5**(1), 125–132.
Pohl, W. (2006b). Wettbewerb im Silberglanz. *LOG IN*, **26**(141/142), 10–13.
Pohl, W., K. Kranzdorf, and H.-W. Hein (2006). First steps into computer science – the German project Einstieg Informatik. In V. Dagiene and R. Mittermeir (Eds.), *Information Technologies at School: Selected Papers of the 2nd International Conference “Informatics in Secondary Schools: Evolution and Perspectives”*, Institute of Mathematics and Informatics, Vilnius, pp. 62–70.

W. Pohl was educated in computer science, and received a PhD in 1997 from the University of Essen, Germany. For many years, he investigated the use of artificial intelligence techniques for the improvement of interaction between humans and machines. In 1999, he changed position and perspective by becoming executive director of the German Federal Contest in Computer Science. Among his responsibilities is to coach the German IOI team and lead the German IOI delegation. Now his interest lies in improving computer science contests, establishing new ones, and work on diverse other projects, everything in order to popularize computer science among youth. From 2003 to 2006, he was elected member of the IOI International Committee, and briefly held the position of executive director of IOI in 2006.

Olympiads in Informatics, 2007, Vol. 1, 149–164
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Increasing the Appeal of Programming Contests with Tasks Involving Graphical User Interfaces and Computer Graphics

Pedro RIBEIRO
Departamento de Ciência de Computadores, Faculdade de Ciências, Universidade do Porto
Rua Campo Alegre, 1021/1055, 4169-007 Porto, Portugal
e-mail: pribeiro@dcc.fc.up.pt

Pedro GUERREIRO
Departamento de Informática, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa
Quinta da Torre, 2829-516 Caparica, Portugal
e-mail: pg@di.fct.unl.pt

Abstract. Programming contests should be capable of being appealing to both the contestants and the general public. We feel that the use of graphical user interfaces and computer graphics could help achieve this goal, providing new ways of viewing the task. We describe experiments we made with games (Tic-Tac-Toe, Snake and Ataxx, an Othello-like game), which were made available to students with graphical components, and discuss the results. We also present a simple graphic library where simple drawings can be made and show how it can be used in a programming contest environment. We then conclude by revisiting some past IOI problems, suggesting ways to enhance them with graphical components.

Key words: programming contests, graphical user interfaces, computer graphics.

1. Introduction

One of the main goals of programming contests is to draw the attention of the public to the contestants, to their achievements and to their extraordinary capabilities in the art and science of programming. Therefore, the more popular our programming contests become the closer we will be of reaching that goal. On the one hand, we need the public to understand or at least to have a reasonable idea of what the problems are about; on the other hand, we must design the tasks in a way that is appealing to the contestants, and especially to the newcomers, so that we can gather more and more participants, and so that our contests cause a greater impact on society. As observed in (Dagiene, 2006) the tasks are the keystones of contests. Their attractiveness to the contestants and public is vital. Therefore, anything we can make to improve this aspect will benefit our programming contests.

Traditional tasks in the International Olympiad in Informatics (IOI) usually involve simple text input and output. Even with improvements such as output only tasks or reactive tasks, the interaction between the program and the outside world is always done in a purely textual way. No other ways of observing the programs working exist, other than looking at the input and the output, which typically are just a bunch of numbers or strings. This approach lacks almost any attractiveness factor, and is meaningless to the general public, since it normally needs some pre-knowledge of the problem statement in order to minimally appreciate what the program is doing.

We have been experimenting with tasks that use graphical user interfaces and computer graphics in preliminary contests, in secondary schools in Portugal, and also in competitive assignments in our programming courses at the university. We believe that this is a domain that can be used to make the contests more attractive and more fun. The use of graphics could dramatically improve the manner in which people visualize the tasks, the test cases and the program outputs, creating a whole new level of ways to observe a programming contest. It also presents the non-informatics audience with an opportunity to perceive and more duly appreciate what the contestants have accomplished. Even the contestants could sometimes benefit of seeing how their program works in a different way. This graphic part should not take precedence over the educational and algorithmic component of the tasks, of course, but we claim that both these aspects can be merged, thus paving the way to interesting new problems.

In this paper we will present some of those experiments, discussing some of its advantages and disadvantages, and we will show how they could be adapted to programming contests like the IOI.

2. Graphical User Interfaces in Programming Tasks

Many IOI tasks are interactive, in the sense that there is an agent reading the output of the contestant's program, which, in turn, has to read the output by the agent. Typically the dialogue is carried out at the console, on line by line basis. Although this is sufficient in the strict setting of problem solving, we believe that in some cases the task could be made more appealing by having the agent respond graphically and by giving visual feedback to the moves by the program. In fact, the task not only becomes more appealing and more fun, but also the visual clues may indeed help the abstract reasoning that leads to the solution.

We have been experimenting along these lines in our own introductory programming courses, at secondary level and at university level. In these courses, some of the labs are competition-like: the students are given a task, they have to write a program for it, and submit it to an IOI-like automatic judge. The automatic judge we use is called Mooshak (Leal and Silva, 2003). It runs a sequence of test cases and informs "Accepted", "Wrong Answer", "Time Limit Exceeded", etc. In principle, "Accepted" means that all test cases passed, which is fine for class use. In competition mode, partial points are possible.

Enhancing competitions tasks with a graphical interface does not cause any penalty on the automatic judge, who still has to check text output only. The extra burden is a responsibility of the agent, and does not interfere greatly with the contestant's job.

We will now describe three of those experiments. The first is the well-known Tic-Tac-Toe problem, and uses an elementary form of ascii-graphics. The second is “Snake”: in this case, the snakes are represented by colored blocks on the screen. The third is an Othello-like board game, and input was done via the mouse, not via the keyboard. In all these cases, the games act as platforms for the programming tasks, providing immediate visual feedback to the students. This visual feedback offers new possibilities of reasoning about the program and checking its result.

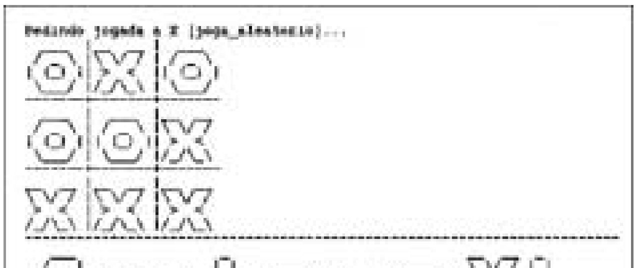
2.1. *Tic-Tac-Toe*

Tic-Tac-Toe is a well-know game and programming it is a typical exercise. We used it in an introductory C programming course for secondary students. The task was to create an agent with artificial intelligence for the game, in a way that agents could play against one another. We organized a tournament among all valid players and in order to validate the contestant submission we used black-box automatic evaluations with Mooshak.

Communication between agents is accomplished in a standard way, as in traditional IOI input/output programs. Each agent receives on the standard input the current state of the game, i.e., the configuration of the board, and writes the chosen move on the standard output. Each move is triggered by making a new call to the student program. This functions almost like an IOI test run, in the sense that the program is called from scratch for each move, without any type of memory between program calls. Each game is processed in the following way: the server initiates the game, with a circular list of the programs that are playing; then automatically calls the next program from the list, feeding it with correspondent input, reading its output and continuing in the same fashion until the game is over, where it communicates the final result.

In this example, the graphical component is built using character strings. It is a naïve design, but provided adequate visualization of the game. And, being very simple, it could be passed to the students, so that they could also learn from it. Fig. 1 shows a snapshot of this component in action.

This apparently simple scheme proved to be very addictive to the students, who had an immediate form of seeing the results of their programming effort. The students also used



```
Pedindo jogada a X [joga_aleatorio]...

Ganhou o X!

 O programa de X era "joga_aleatorio"!!
```

Fig. 1. Graphical component of the Tic-Tac-Toe server (with Portuguese text).

the server to show their families and friends what they were doing in the programming course, thus helping explain to the "outside world" what programming is about. During the tournament, the games were projected in the classroom to a small audience, in an enticing way that could not be achieved if we had stuck to the elementary form of input and output.

This was precisely what we wanted to happen, and it corresponded perfectly to our aim of trying to show the "outside world" what programming can be.

2.2. *Snake*

"Snake" is another popular game. Each player controls a snake in a grid environment, striving to make it eat as much food as possible, without hitting the walls or other snakes (including itself). Each time a snake eats an "egg", its size increases by one. This game also provides a good platform for programming tasks.

We used this game in a Logic Programming course at the university, very much like we had used Tic-Tac-Toe for secondary school students. The students now had to program an artificial intelligence for a single snake and the final goal was to make a tournament with two available modes: all against all and solitary. In the first case, the winner is the last surviving snake; in solitary mode, the winner is the snake that eats the eggs fastest.

Communication between the server and programs driving the snakes was similar to the Tic-Tac-Toe program, and the server operated in the same way.

The graphical component was more sophisticated. It was designed for Linux, using XLib (Gettys and Scheifler, 2002). It displays a window in which all snakes, eggs and other food items are represented. Fig. 2 shows an example.

In this experiment, students only had the duration of a single lab (2 hours) for programming their snake agent. This matches contest time constraints.

Again, the experiment was very successful, and we could observe pride and joy on the face of some students when the final tournament was projected in the classroom. Some even brought in their friends (with no programming background), who came to just watch and have fun.

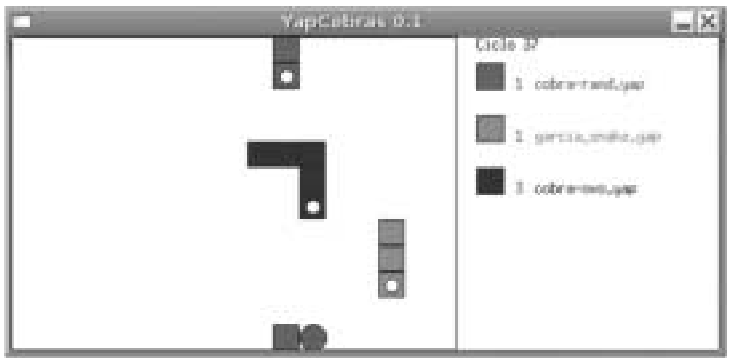


Fig. 2. Graphical component of the Snake server (with Portuguese text).

Out of curiosity, it should be noted that we also used Mooshak automatic evaluation for Logic Programming (Prolog), not only in this experience but also in other exercises. In fact, Mooshak is very configurable and we make extensive use of it in our courses, for different programming paradigms, both at introductory and advanced levels.

2.3. *Ataxx*

Our third experiment was more ambitious. The task was a more complete programming project that took a few weeks to complete. We chose Ataxx, a board game similar to Othello. For a better understating of the challenges involved, we will briefly explain the game rules, but a more detailed explanation can be seen on (Beyrand, 2007). Ataxx is a 2 player game played on a 7×7 grid board. The players start with two pieces each, on opposite corners of the board and take turns to play. Each time, we can make two types of moves: we either add a new piece, by making it appear on an adjacent square to an already existent piece, or we move a piece, by making it jump at most two squares. After each move, pieces that are adjacent to the piece that was added or that moved turn to our color. In the end, the player with more pieces wins.

Like before, the main goal was to make an agent capable of playing the game automatically. This was a Logic Programming task, to be programmed in Prolog, and the communication was handled directly, by calling a logic predicate with the game state as input. The game state was given as a list representing the board and an integer representing the available time. This second parameter was necessary because the agents had a time limit for their moves.

We implemented Ataxx using a general architecture where a text-only server was responsible for implementing the game, its rules, and calling the agents. This server could be called to automatically run a game in the fastest possible way, in batch, with no human interaction. On the other hand, a Graphical User Interface (GUI) could be attached to the text-server, making use of standard input/output interaction. This module is independent from the game itself. Fig. 3 shows a basic view of the architecture we used.

The text server functions basically as the previous ones. It initiates the game, calls the programs for the players in the right order, and announces the final result when the game is over.

The GUI itself is more refined that the previous ones. We used Java Swing (Project Swing) for it and included a clickable interface to be used by humans playing the game. This way, it was possible to have computer vs. computer, human vs. computer and human vs. human matches. Actually, this possibility may be used by the students to become more

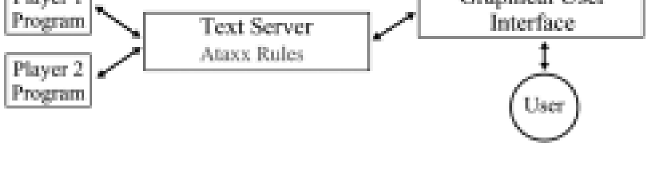


Fig. 3. Ataxx experience architecture.

familiar with the game and devise the strategies that they will later implement. The GUI included animations for the piece captures and provided ways to automatically run a full game, advance turn by turn, undo moves and to load/save game logs. Fig. 4 displays a snapshot of the GUI in action.

For guiding the students in developing their solutions, we designed a sequence of tasks, each of which was handled as a competition problem: students should complete each task and submit it to the automatic judge Mooshak for validation. For example, we had a task for computing the list of adjacent squares, for checking whether a move was valid, and for generating the list of all possible moves, given a board description. We held several "mini-tournaments" where students could participate in order to see how their agents were doing, and also just for the fun of it.

In the end, we had 47 functional programs to grade. Some of them were quite good players, constantly beating known Ataxx programs, by a large margin. For grading, we first created a set of agents, with which we made an initial estimate of the agent's "merit". With this we could identify, for example, the programs that were not even able to win against a random opponent. After the initial analysis by the grading agents, we grouped the programs in three different leagues, according to their strength, and then, in each league we carried out a large-scale tournament, in which all played against all. The final score was a function of the ranking in the tournament. In the end, the best programs were announced in the department, and their authors publicly recognized.

This experiment was very successful and the students were strongly motivated by it. The graphic component, including the animations, had a great impact, and we know that some students used it to show their work to their non-informatics friends. The Ataxx project also had a positive influence on the results in the course. In the survey we carried at

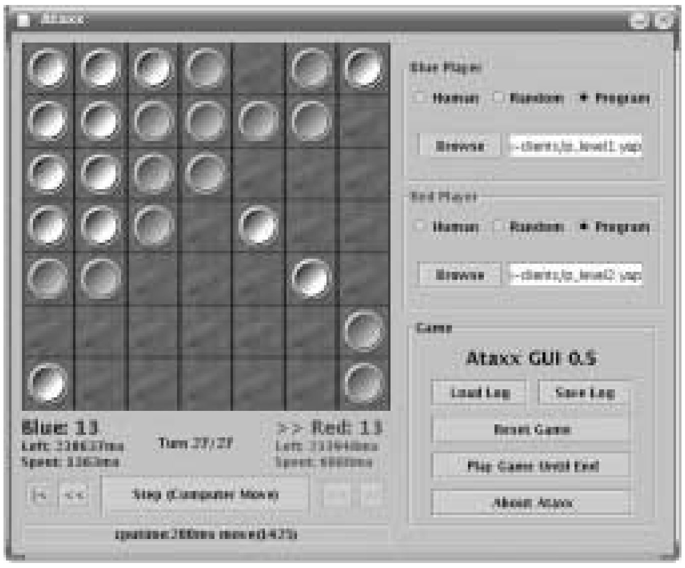


Fig. 4. Snapshot of the graphical user interface for Ataxx.

the end of the semester, the course was rated highly, due in large part to this programming task.

We now use this program as an example of student projects that we present when prospective secondary school students visit us. Needless to say, without the graphical component, that would not be possible. The Ataxx server and related resources are publicly available at (Ribeiro, 2007).

3. Simple Graphics for Simple Drawings

Many problems in IOI competitions are geometric, and would become more dramatic with a graphical output. Take, for example, problem Joining Points from IOI 2006 in Merida, Mexico (IOI Secretariat). The goal is to join pairs of *green* points and pairs of *red* points according to certain rules. The output is merely the sequence of pairs of points, given by their position in the input files. Of course this is enough, as far as algorithmics go, but the problem would be more appealing and less abstract if the task was to actually draw the figure that comes out as the points are joined. Indeed, it might happen that by actually watching the drawing appear on their screens, albeit in the wrong form in their early experiments with the problem, more contestants would be able to figure out the correct strategy.

This example suggests that two additional aspects need be considered when designing contest problems with a graphical output: we need to agree on a common graphical library and we must be able to automatically judge the graphical output.

The graphical library is necessary because students use different languages and operating systems, and we must ensure that no environment gives an unfair advantage to contestants using it. On the other hand, professional graphical libraries are too vast for the needs of common problems and it would be a waste of time trying to completely master them, in preparation for IOI. By providing a simple graphical library, we could achieve an informal competition "standard" that would be followed worldwide. As a side effect, it could eventually spread out and be used in general for teaching programming in general.

In programming competitions such as the IOI and ACM-ICPC (ICPC site) most tasks have text output. Automatic judging is performed either by directly comparing the output of the contestants program to the official solution or by running a validating program, specific to each problem, that checks whether the output is correct. This second solution is necessary when there are many solutions and it is not adequate to pick one of them as a representative. We anticipate this will be the common case with graphical programs: the programs must draw a picture but it does not matter which parts of the figure are drawn first. Therefore, except for very simple cases, graphical programs will be judged via ad-hoc tester programs.

We will now present the sketch of simple graphical library along the lines we have discussed. It has been tested within our own programming courses, where most of the programming assignments are contest-like.

3.1. *Classes for Figures*

The first class in our library is class Point, representing points in the two dimensional plane. It is useful directly in many problems and it is used in many other classes: Circle, Polygon, Polyline, Segment, etc. For example, in the Joining Points problems that we mentioned, we are given two arrays of Points. Actually, there are green points and red points, which means perhaps we should handle them via subclasses for colored points. The result is an array of segments. If from a pure algorithmic perspective, segments are just pairs of points, and this point of view would be enough for solving the problem as presented, the graphical variant that we suggest would be better handled by using a class for colored segments as well.

All these classes – Point, Circle, Polygon, Polyline, Segment – and also, the classes for colored points and colored segments, represent figures, and this leads us to an abstract class Figure, from which the others derive. Class Figure is another fine example, because it declares as pure virtual functions those functions that all figures must implement and also declares and implements abstractly a number of template functions in terms of the pure virtual ones. These will be available without further effort in all derived classes. Here is a sketch of the class declaration, in C++:

```
class Figure {
public:
    // basic functions
    virtual void Scale (double fx, double fy) = 0;
                                       pre IsDeformable () || fx == fy;
    virtual void Translate (double dx, double dy) = 0;
    virtual void Rotate (double angle) = 0; // pre IsRotatable ();
    // preconditions
    virtual bool IsRotatable () const;
    virtual bool IsDeformable () const;
    // template functions
    virtual void RotateAround (double angle, const Point& p);
    ...
};
```

The three basic functions, Scale, Translate and Rotate, are pure virtual. The template functions have default implementations or are programmed in terms of those three. Each class derived from Figure will provide an implementation for the three basic functions and inherit the remaining functions. Boolean functions IsRotatable and IsDeformable are used as abstract preconditions for commands Rotate and Scale (Meyer, 1997). They are defined here returning true: descending classes whose objects cannot be rotated (for example, rectangles with horizontal and vertical sides) must redefine IsRotatable; descending classes whose objects cannot be scaled differently in along the two axes (for example, circles) must redefine IsDeformable.

3.2. *Clone and Draw*

Sooner or later, we will need containers of polymorphic figures. In C++ these can be implemented as vectors of pointers to Figure. When adding a figure to a container, we prefer not to manipulate the pointers explicitly. Therefore, each class derived from Figure must provide a virtual function Clone that creates a dynamic copy of its object. The situation would be simpler in Java, where the objects have reference semantics. Still, when adding a figure to a composite figure, we do not want to add a reference to the argument: we want to automatically create a copy of the argument and store the copy, so the argument and the composite figure can evolve independently. Therefore we also need to clone in Java, and this should be done avoiding the complications of the built-in clone function (Bloch, 2001).

For graphics, we want each Figure to be able to draw itself, and thus class Figure must declare a function Draw. The case for function Draw introduces another issue: where will the drawing be made? In other words, when we write, we write on a stream; when we draw, where do we draw? In C++, when we write a value of type double on a stream, it is the stream that decides how the number will be written: fixed point or scientific notation, how many decimal places, etc. Likewise, when we draw, there must be some structure (i.e., some class) that "decides" how to draw: which color, which line thickness, etc. Well, that structure will be called canvas and we posit an abstract class Canvas to represent it.

Function Draw should also be pure virtual in class Figure:

```
class Figure {
public:
   // ...
   virtual Figure* Clone () const = 0;
   virtual void Draw (CanvasPtr) const = 0;
};
```

Please note that the argument of Draw is of type CanvasPtr. This is a typedef that represents a pointer to Canvas.

3.3. *Classes for Canvases*

A canvas is an object where figures are drawn. Canvases will be implemented in terms of an underlying graphics library, and will effectively shield the users from having to deal with that particular library. Instead, they will only have to learn a few functions: those provided by class Canvas, the abstract class for canvases:

```
class Canvas {
public:
   virtual void DrawPolygon (const std::vector<Point>& p) = 0;
   virtual void DrawRectangle(const Point& p1, const Point& p2) = 0;
   virtual void DrawEllipse (const Point& p1, const Point& p2) = 0;

   virtual void DrawPoint (const Point& p) = 0;
   virtual void SetColor (const Color& c) = 0;
   virtual void SetPenWidth (double t) = 0;
   virtual void SetFilled (bool b) = 0;
   // and the corresponding getters ...
};
```

There are several Draw functions, all with arguments of type Point or std::vector<Point>. These functions handle the common cases, but a few more could be added. Then we have functions for setting the color, the pen width and the font, and the corresponding getters. Function SetFilled determines that the figure will be filled with the current color if the argument is true, or not if it is false.

Class Canvas is abstract. Each concrete class that derives from class Figure will implement its function Draw in terms of the re-sources provided by the abstract class Canvas. The library provides four concrete canvases: CanvasText, CanvasBasic, CanvasExtensible and CanvasEuclidean. When drawing, the user must select one of these canvas classes.

CanvasText is actually not a graphical canvas: it is linked to a stream, and figures are drawn by writing their description on the stream. This textual output can be processed by a viewer program, but we may use it for unit testing (SUnit site) and in the realm of programming contests, we need it for automatic evaluation.

CanvasBasic makes a one to one mapping between window coordinates and the coordinates of the figure, but leaves the origin on the lower left corner of the window, and has the y-axis growing upwards. It is useful in simple cases and it is the base class for the other CanvasExtensible and CanvasEuclidean.

CanvasExtensible is associated to a form and is extensible in the sense that the constructor sets the extension of the larger side of the form. The x-axis runs through the bottom of the window, from left to right, and the y-axis runs through the left side, from bottom to top. If, for example, the extension is 300, and the window is wider than tall, then the visible y-range is zero to 300, and the visible x-range is from zero a number greater than 300, computed so that x units and y units measure the same on the screen.

CanvasEuclidean is similar to CanvasExtensible, but allows for negative coordinates in both axes. More precisely, users can specify the x-range and the y-range arbitrarily.

Even though pictures provide rich visual feedback for the workings of the program, situations occur, typically when debugging, where we want to observe precisely the sequence of elementary strokes that make up the drawing. In principle, this can be achieved directly by simply changing the canvas where the figure is drawn from one of the graphical ones to CanvasText. We have observed that this technique is very effective.

We already mentioned that CanvasText is for automatic evaluation. We have tested this in the programming assignments. The idea was to validate some new transformations, before they were applied in the figure that was being composed. This validation was performed by the automatic judge that we use, Mooshak, on a program drawing on an object of type CanvasText, wrapping the output console. Once the drawing was accepted, students could switch to one of the graphical canvases.

3.4. *Example*

To illustrate the use of our library, let us consider a program to draw simple fractals that was used in a programming assignment with automatic judging. A fractal is a polygon; therefore, class Fractal inherits from class Polygon:

```
class Fractal: public Polygon {
private:
    // some data members for fractals
public:
    virtual void Read (std::istream& input);
    virtual Fractal Next() const;
    virtual void Transform();
};
```

Function Read reads the fractal data from an input stream; function Next yields the next fractal in the generation process; function Transform assigns the result of Next to the target object; function Draw does not appear because it is inherited from Polygon. Here is a simple test function that reads the description of a fractal from the standard input, transforms it twice and "draws" it on the standard output using a CanvasText object:

```
void TestFractal () {
    Fractal f;
    f.Read (std::cin);
    f.Transform(); f.Transform();
    Canvas* t = new CanvasText (std::cout);
    f.Draw (t);
    delete t;
}
```

This test function is part of a console application. In order to draw graphically, we need a form application, but the main difference from the console application is that now we draw on a graphical canvas that wraps the form, not the console. Typically, this happens in a member function of the form class.

3.5. *Composite Figures*

The fractal example that we mentioned before was very simple: there is a single polygon that must be drawn. In general, we want more complicated figures. Anyway, we can still use exactly the same technique provided we can handle composite figures, and we can do that using the Composite design pattern (Gamma *et al.*, 1994): a composite figure is a figure that is made up of a sequence of figures:

```
class FigureComposite: public Figure {
private:
    std::vector <Figure *> components_;
public:
    // Constructors
    // Inherited functions to be redefined

    virtual void Put (const Figure& f);
};
```

Function Put is the only novelty: it adds another figure to the composite figure. Scaling, translating, rotating, or drawing a composite figure means scaling, translating, rotating or drawing each of the components.

We mentioned that the solution for the joining points problem is an array of pair of points. Actually, we could use a composite figure, made up of colored segments.

3.6. *Colored Figures*

From what we have described, we may infer that each figure is drawn using the current value for the color, the pen width, and filled attributes. This implies that the components of a composite figure are all drawn with the same color, same pen width, etc. We do not want that, in general. On the contrary, we need colored figures, which are drawn according to their own properties, not the properties of the canvas.

This means that for each figure in our class library we need a colored version. We can accomplish that using inheritance, of course: for example, class ColoredPolygon inherits from Polygon and adds data members and functions to deal with the color and other graphical attributes.

This solution, however, is not very attractive: it would duplicate the number of classes we have to maintain. A better approach that we can use with C++ relies on generic programming. We define a template class Colored<T> and the instantiate generically with subclasses of Figure, as needed. Here is Colored<T>:

```
template <class T>
class Colored: public T {
private:
    Color c_;
    int w_; // pen width;
    bool f_; // filled
public:
    Colored (const T& t, Color c = Color::black, int w = 1,
                     bool f = false):
       T (t), c_(c), w_(w), f_ (f) {
    }

    virtual Colored<T>* Clone() const {
       return new Colored<T>(*this);
    }


    virtual void Draw (CanvasPtr x) const {
       Color c1 = x->GetColor ();
       double w1 = x->GetPenWidth ();
       bool f1 = x->GetFilled ();
       x->SetColor (c_); x->SetPenWidth (w_); x->SetFilled (f_);
       T::Draw(x);
       x->SetColor (c1); x->SetPenWidth (w1); x->SetFilled (f1);
    }
};
```

Note that template class Colored<T> inherits from its formal argument. Hence, Colored<Polygon> inherits from Polygon, and exhibits normal polygon behavior, to which it adds color and pen width, as required.

This property of C++, of allowing generic classes to inherit from its template argument, is uncommon, but offers an elegant solution to our problem. Note that this approach could not be applied to Java or Eiffel, for example, even though these languages cater for generic programming as well. In Eiffel or Java, the solution would be to design a class Colored, inheriting from Figure, with a component of type Figure, representing the object whose graphical properties we want to control. This is also an interesting situation (also possible in C++, of course) that corresponds more closely to the design pattern Decorator (Gamma *et al.*, 1994).

As an example of this technique at work, observe a class representing the flag of Sweden: a cross made of two yellow rectangles on top of a blue rectangle:

```
class Sweden: public FigureComposite {
public:
    Sweden ():
       FigureComposite () {
       Rectangle r(Point(0, 0), Point(10, 8));
       Colored<Rectangle> rc (r, Color::blue);
       Rectangle v(Point(3, 0), Point(5, 8));
       Colored<Rectangle> vc (v, Color::yellow);
       Rectangle h(Point(0, 3), Point(10, 5));
       Colored<Rectangle> hc (h, Color::yellow);
       Put(rc); Put(vc); Put(hc); Scale(100, 100);
    }
};
```

This concludes the presentation of our simple graphical library.

4. Graphical Tasks

Having discussed our experiments in the area, we will now show how some past IOI tasks could have been graphically enhanced using the approach we advocate. With these kinds of problems, the IOI tasks could more easily be brought to the general media, like the press and the television, which are inherently graphical biased.

We will now give list of some of those problems. Note that we select at least one problem per year since 1994, as an indication that this approach applies widely. All task statements can be seen on (IOI Secretariat):

- *IOI'94, task "The Clocks"*: compute the minimum number of moves to turn 9 clock dials to 12 o'clock. A GUI could have been provided, and it could graphically show the moves happening, with animations. Since the output of the tasks was the set of moves that should be made, nothing had to be changed. Generally speaking, almost all search problems could be made visual if the states can be represented visually in a simple suggestive manner (for example, *MagicSquares from IOI'96* is similar).
- *IOI'95, task "Packing Rectangles")*: Find the smallest enclosing rectangle into which other four rectangles may be fitted without overlapping. Instead of numbers, the output could be really graphical, with drawing commands, giving a new insight into how are the rectangles being packed, and which kind of algorithm is being used.
- *IOI'96, task "A Game"*: A 2-player game where players take turns to collect numbers from the ends of a sequence of integers, winning the best sum of numbers. A fully functional GUI could have been made, which would also allow the player to make games between two versions of his program. Generally speaking, almost every 2-player game that appeared in IOI could have a GUI (for example, *The Game of Hex from IOI'97* could also have a GUI).
- *IOI'97, task "Stacking Containers"*: A crane operates several containers, for storage and removal, in a 3-dimensional world. A GUI could have been used to show the effects of the algorithms used, giving a visual depiction of the simulation. Generally speaking, almost every problem that is related to a 3D world, can have a

visually improved interface (another example is *The Toxic iShongololo, also from IOI'97*).

- *IOI'98, task "Starry Night"*: Detect similar clusters of stars in the sky. This is an example with a great potential public appeal if it used a graphical GUI to show the stars.
- *IOI'99, task "A Strip of Land"*: Find a rectangular region for an airport with the largest area subject to width and height difference constraints. A GUI could show in 3D the area to calculate, giving an attractive view of the task.
- *IOI'00, task "Building With Blocks"*: Find the way to decompose a 3D solid in a minimum number of different small blocks. Once more, the 3D nature of the task could be really used, and a GUI would even allow the contestants to have a better understanding of the problem.
- *IOI'01, task "Ioiwari game"*: A Mancala like 2-player game. Do we need to say more? In this year, there was also another 2-player game task (*Score*).
- *IOI'02, task "Xor"*: An output only-problem where one has a *xor* operation to transform a blank screen into a figure. A GUI would have permit a visual depiction of the solution, but would also give the contestants a whole new way of trying to make "manual solutions" in their heads (that could have algorithmic nature).
- *IOI'03, task "Seeing the Boundary"*: Count how many fence posts one can see from a determined position with some rocks obscuring posts. This is a very graphical problem, and a GUI could have helped the contestants to see exactly why a determined input gives the correspondent output.
- *IOI'04, task "Polygon"*: An output-only problem where we must do a reverse Minkowski sum on a polygon. Even the name of the task is telling us it could use graphics. A GUI could have shown much more clearly the structure of the problem and allow for different approaches to the task. The graphical library could have been used to draw the polygon, instead of simple text output.
- *IOI'05, task "Rectangle Game"*: A two-player game where players take turn to divide the rectangle. All that has been said before about games can be applied.
- *IOI'06, task "Joining Points"*: This problem was already approached before (on chapter 3) and it constitutes a paradigmatic example of a task with almost everything in favour for using a graphical approach.

5. Conclusion

IOI is an algorithmic competition. The emphasis has been, from the beginning, on problem solving by designing appropriate algorithms that solve the proposed tasks. Typically, these tasks accept their data from the console and display the result of the computation also at the console, using numbers and strings. This is understandable, since we really want to focus on algorithm development, and input and output are there merely for being able to validate the submissions. Besides, when IOI started, almost 20 years ago, the available programming environments were very crude, compared to what is common nowadays. Therefore, other possibilities were not even contemplated.

At present, when we review the problem sets of past editions of IOI, we cannot help noticing that some have a strong graphical inclination, but it remains implicit and solutions are not meant to exploit it. This is a pity, since proper use of graphics in programming competitions might make them more attractive and more fun for the contestants, and also more understandable to the general public. This second aspect is very important, since one of the goals of programming competitions is to "shine the spotlight" on the students and on the great programming feats they are able to accomplish.

Graphics can be brought into the competition in two ways, at least. In some tasks the problem may be specified in term of controlling an agent whose behavior is represented graphically; we offered examples of this case in chapter 2. Other tasks may actually have graphical output, instead of plain text output; we explained how this can be approached in chapter 3. In either case, the graphics need not superfluously burden the task. On the contrary, they may actually help the contestant devise the correct strategy for solving the problem.

Complementarily, graphics open the way to a new kind of task, the "tournament" task, as illustrated by the Ataxx example in sub-chapter 2.3. Indeed the possibility of a tournament in IOI was already mentioned in (Opmanis, 2006), as way of improving the contest. In tournament tasks, points would be awarded as a result of a series of matches to be held live, before an audience. The players are the programs for that task, written by the contestants during the competition. A preliminary validation may carried out, in the conventional way, which could also be used to form "leagues" grouping programs whose performance falls within given ranges. This kind of tasks and the associated staging would bring an extra level of excitement to the competition, and could easily be followed by the public in general, thus boosting the IOI spirit.

References

Beyrand, A. (2007). *Ataxx !!*. `http://www.pressibus.org/ataxx/` (accessed May 2007).
Bloch, J. (2001). *Effective Java*. Addison Wesley.
Dagiene, V. (2006). Information technology contests – introduction to computer science in an attractive way. *Informatics in Education*, **5**(1), 37-46.
Gamma, E., R. Helm, R. Johnson and J. Vlissides (1994). *Design Patterns*. Addison-Wesley, Reading, Mass.
Gettys, J., and R.W. Scheifler (2002). Xlib – C Language X Interface. X Consortium Standard. X Version 11, Release 6.7 Draft.
ICPC site, The ACM-ICPC International Collegiate Programming Contest.
`http://icpc.baylor.edu/icpc/` (accessed May 2007).
IOI, International Olympiads in Informatics. `http://www.ioinformatics.org/` (accessed May 2007).
IOI Secretariat. `http://olympiads.win.tue.nl/ioi/` (accessed May 2007).
Leal, J.P., and F. Silva (2003). Mooshak: a Web-based multi-site programming contest system. *Software Practice & Experience*, **33**(6), 567–581.
Meyer, B. (1997). *Object-Oriented Software Construction*, 2nd Ed. Prentice Hall.
Opmanis, M. (2006). Some ways to improve olympiads in informatics. *Informatics in Education*, **5**(1), 133–124.
Project Swing.
`http://java.sun.com/j2se/1.5.0/docs/guide/swing/` (accessed May 2007).
Ribeiro, P. (2007). *Ataxx Server and GUI*.
`http://www.dcc.fc.up.pt/~pribeiro/ataxx/` (accessed May 2007).

P. Ribeiro is currently a PhD student at Universidade do Porto, where he completed his Computer Science degree with top marks. From 1995 to 1998 he represented Portugal at IOI-level and from 1999 to 2003 he represented his university at ACM-IPC national and international contests. During those years he also helped to create new programming contests in Portugal. He now belongs to the Scientific Committee of several contests, actively contributing new problems. He is also co-responsible for the training campus of the Portuguese IOI contestants and since 2005 he has been deputy leader for the Portuguese team. His research interests, besides contests, are data structures and algorithms, artificial intelligence and distributed computing.

P. Guerreiro is an associate professor of Informatics at Universidade Nova de Lisboa. He has been teaching programming to successive generations of students, using various languages and paradigms for the over 30 years. He has been involved with IOI since 1993. He is also the current director of the South-Western Europe Regional Contest, within ACM-ICPC, International Collegiate Programming Contest. He is the author of three popular books on programming, in Portuguese. His research interests are programming, programming languages, software engineering and e-learning.

Olympiads in Informatics, 2007, Vol. 1, 165–174
© 2007 *Institute of Mathematics and Informatics, Vilnius*

Development and Exploration of Chinese National Olympiad in Informatics (CNOI)

Hong WANG
Tsinghua University
100084, Beijing, China
e-mail: wanghong@tsinghua.edu.cn

Baolin YIN
Beijing University of Aeronautics and Astronautics
100083, Beijing, China
e-mail: yin@nlsde.buaa.edu.cn

Wenxin LI
Peking University
100871, Beijing, China
e-mail: lwx@pku.edu.cn

Abstract. This article presents a general overview of the historic development, exploration and practice of CNOI during the past 23 years. It includes: 1) some historical data recording the development of CNOI; 2) main contest activities organized by the Scientific Committee and Competition Committee of NOI of CCF, and some relevant management experiences; 3) the selection mechanism for the best contestants of CNOI; 4) the development and characteristics of a testing and evaluation system; 5) the development and characteristics of a visible team competition; 6) training of contestants and teachers, and the improvement and perfection of competition rules.

Key words: contest organization, contestant selection, evaluation system, visible contest.

1. Introduction

"The popularization of knowledge of computers should begin from children", said in 1984 the former leader Deng Xiaoping, General designer of China's reform and opening policy.

The China Computer Federation organized the 1st China Computer Programming Contest for Youth and Children in 1984, afterwards named the National Olympiad in Informatics (NOI). By the year 2006, a total of 23 NOI contests had been held successfully. As one of the earliest countries to participate the International Olympiad in Informatics (IOI), China has been present at every IOI contest since the first one, IOI1989. The effects of informatics olympiads have been demonstrated in promoting the popularization and raise of information technology in China's middle and high schools, which has and will play an important role in the cultivation and selection of talents in informatics. The following data records the development of CNOI:

- in 2000, the 12th International Olympiad in Informatics (IOI2000) was successfully held in Beijing;
- during the past 18 IOI contests (to the end of the year 2006), Chinese contestants won a total of 42 Gold Medals, 17 Silver, and 11 Bronze;
- during the IOI contests held in the consecutive three years 2004-2006, all the Chinese contestants that participated in the contests were awarded Gold Medals;
- the year 2004 was the 20th Anniversary of China National Olympiad in Informatics, and the China National Symposium on Computer Education was held with the official proceeding published;
- 2006 NOI Contest included the first Visible Team Competition;
- contestants enrolled to join the CNOI Contests in 2006 reached nearly 80,000;
- the first Yearbook of Chinese National Olympiad in Informatics (CNOI2006) was officially published in 2006.

2. Main Contest Activities and Selection Mechanism of CNOI

2.1. *Main Contests and Activities of CNOI*

The NOI Scientific Committee and Competition Committee, under the guidance of the China Computer Federation, is responsible for the technical organization and management of NOI contest. A number of activities are run every year which are aimed at middle and high school students for enrichment and competition in computer programming. The main contests and activities are given in the Table 1.

2.2. *Selection Mechanism for the Best Contestants*

The CNOI establishes a strict selection mechanism and rules for the excellent contestants. It is based on multi-contests and paper defence. It guarantees that the best contestants can be selected by the mechanism.

Take the selection of IOI2007 China team as example. The process begins from NOI2006 during July of 2006. The top 20 contestants of NOI2006 become the members of the National Training Team (NTT) for the IOI2007. This is the first contest that forms part of China's formal IOI team selection process. Then we arrange 4 contests and structures in order to select the best 4 contestants from these 20. Each part has a different weight or score. These structures and their proportions are respectively:

Homework: 5%

NOI Winter Camp Testing and Competiton: 25%

Paper presentation and oral defence: 10%

China Team Selection Competition (CTSC): 60% (two contests, 30% each time).

Besides the score, we also check the comprehentive character and English proficiency of contestants. A personal statement and letter of commitment is requested for the contestants.

Table 1
Main contests and activities of CNOI

Activity	**When**	**What**	**Participantes and Size**
* China National Olympiads in Informatics (NOI)	July–August	Two day competition (5 hours for 3 tasks/each day) and one week activity similar to IOI. A team competition was added in NOI2006. The top 20 contestants from NOI form National Training Team (NTT) for the IOI of next year (candidates for the China team)	5 contestants each province. Totally 150 pers.
* NOI Summer Camp.	At the same time with NOI	The competition and activity is the same as NOI.	4 contestants each province. Totally 120 pers.
* National Olympiad in Informatics in Province (NOIP)	October–November	Preliminary competition: a multiple-choice / short-answer competition. Final competition: 3 hours for 3 tasks. Contestants are divided into middle and high school group	Nearly 80000 contestants participanted in NOIP2006.
Homework practicing and training for NTT	August–Jan.	Training and practicing by homework. Contestants will submit their solutions, and someone will give instant feedback.	20 contestants of NTT
* NOI Winter Camp	Jan–Feb	An intense one week training and a five-hour competition. The formal contestants will also participate in an oral paper presentation and defence $(10' + 5')$	Formal: 20 contestants of NTT Informal: 4 contestants each province
* China Team Selection Competition (CTSC) for the IOI	May	The final China Team Selection Competition. Two day competition (5 hours for 3 tasks each day) is similar to the IOI. The top six contestants will participante in an oral defence. The best four contestants will form China team for the IOI.	Formal: 20 contestants of NTT Informal: 3–4 contestants each province
* APIO (Asia Pacific Informatics Olympiad)	The 2nd Satuday of May	China Regional Competition of APIO held in one place organized by CCF. It is in parall with the CTSC from the year 2007. We took China Regional Competition of APIO as the first day competition of the CTSC in 2007.	20 contestants of NTT; top 50 of the last NOIP; and 1 contestant each province
Training before the IOI	August	Two training competitions with ACM/ICPC contestants for one week before the IOI	4 contestants of China team
Teacher and coach training	1–2 times each year	One week training course for algorithm design and programming skills, also including competition organization rules and evaluation system, etc.	Teachers, coaches and officials of provinces

3. The Development and Characteristics of an Evaluation System (Arbiter)

In order to guarantee the correctness and efficiency of evaluating the contestants' programs, the Scientific Committee of the Chinese National Olympiad in Informatics (SC of NOI) authorized the Group of Advanced Information Technology, Beijing University of Aeronautics & Astronautics (GAIT, BUAA) to develop an official evaluation tool, the Arbiter, seven years ago. After six years use in various competitions organized by the SC of NOI and similar events, the system has been proven to be a stable, comprehensive and trustworthy tool. Arbiter provides support in every phase of a competition, including task design, contest environment preparation, program evaluation, scoring and ranking, statistics and data backup.

Arbiter runs on various versions of the Linux operating system, including Redhat, Debian and Ubuntu. The system is based on a LAN with C/S architecture. The server is an independent computer communicating via a LAN with the clients of the contestants' PCs through out the contest. A contest can be divided into three stages: preparation, contesting and evaluation. In the preparation stage, the server cooperates with the clients to establish and update the contest settings, construct the language environments and contestants' accounts on the client PCs, and issue the contest data. During the contesting stage, the Arbiter clients on the contestants' PCs monitor and control the network communications of the PCs, and filter out forbidden packets according to the preset regulations. In the evaluation stage, the server commands the clients to evaluate the contestants' programs locally and then collects the results to form the score list, ranking list, and various analyzing forms as required.

Arbiter has the following characteristics and advantages:

- Flexibility and Efficiency

  Arbiter enables the administrator to configure and control the contest in an easy manner. A contest is composed of a number of tests, and each test contains a number of tasks. A task can be of several types, such as a standard program, a program interacting with a library, or a results only task. A task can be evaluated with a number of items of evaluating data, each with configurable weight. The contestants are allowed to develop their programs. The administrator is able to modify the configuration of the system at any stage in case some client PCs are not working. Since the programs are evaluated in parallel on the client PCs, the evaluation will be finished very fast. Arbiter is highly adaptable to the hardware environment. In case there are not enough client PCs with a unique hardware configuration, the administrator is able to appoint a number of PCs with the same configuration as the evaluation machines so that the timing will be unique and impartial for all the contestants. Furthermore, as the evaluation is done on the contestants' PCs, there is no need for the contestants to submit their programs to the server. This avoids the network traffic jam, as happens on most of the Web base evaluation systems.

- Safety and Security

  Much attention was paid during the design of Arbiter to processing the contestants' results safely and securely. Before the evaluation starts, the contestants' programs and data are uploaded to the server. A backup copy is also stored locally in a system directory. The backup will be used during the self check by the contestants. Arbiter safeguards the data with encryption and access controls, even the legal users are not permitted to access the data files directly. All the data operations must be done by using a system operation tool with authorization control.

In some contests, the contestants are allowed to access some strictly specified websites, while other network communications are forbidden. In order to control the network communications, the Arbiter clients monitor and filter the network communications on contestants' PCs. The filtering is based on the source and destination IPs and the type of the protocol of the packets. Only the pre-specified network communications are allowed.

- Independent to the Language Environments

Arbiter is independent of the programming language environments. The compliers and relevant tools are specified by the administrator during the preparation stage when the contest is under configuration. The command line options can also be specified at the same time for each operation. This enables Arbiter to meet new requirements in the future by supporting various programming languages and different types of the tasks, provided the relevant compilers and tools are available.

- Accuracy in Timing

The system can account and control the execution time of the program being evaluated in the 5ms time slice. The results show that Arbiter performs very stable evaluation and accurate timing with an error rate of less than 0.01% and timing error of less than 10ms.

- Self-Adaptive and Efficient Network Communications

In order to work in different network environments, Arbiter automatically explores and analyzes the topology of local networks. Consequently the best schemes are adopted to transfer data between the server and the clients. If the network supports broadcasting with a low packet drop rate, broadcasting mode will be used to send the data from the server to the contestants' PCs. This mode will provide high speed for data transfer. A typical test shows that 100MB data can be sent to over 100 PCs in less than 20s over a 100M Ethernet. If the PCs are cascaded in the network and broadcasting is forbidden, or the packet drop rate of the network is above a threshold, the system adopts a cascaded mode P2P data transfer when sending data from the server to the PCs, in order to transfer the data correctly with relatively high speed. In both modes, application layer checking will be done, and data will be re-transferred if there is any mistake.

- Data Import and Export

In order to meet the requirements by users for using other tools, such as MS Office, to process and display the contest data, Arbiter stores its data in the standard formats of CSV and PostScript. The data import and export functions are provided, and therefore all configuration files, contestants' information files, and score files can be easily imported and exported.

- Easy to Use

  The server of Arbiter is GUI based and running in an interactive mode, while the clients are running in a daemon mode without any direct interaction with the administrator. All functions are shown in the user area of the GUI of the Arbiter server. Any principal function can be selected within 3 clicks of the mouse buttons with clear guiding information. As both the server and the client are statically linked with the libraries, they are independent of the library versions on the target system.
- Comprehensive Functions

  The Arbiter system is accompanied by a series of supporting tools: a task verifying tool for checking the correctness of the testing data and the score evaluating plug-ins, a seat appointing tool for deploying the contestants over the PCs, and a user account and password generating tool. While both the seat appointing tool and the account and password generating tool are for the contest administrator to set a contest, the task verifying tool is for the task creators to test the evaluating points, the standard programs and time limits.

An aggregative evaluating tool has also been developed in order to meet requirements where a LAN based contest environment cannot be set. In this case, all programs are collected via other media, such as USB disks and email. The aggregative evaluating tool will run after the contestants' programs are collected and stored in a specified file hierarchy, and the evaluation and statistics will be performed in the same way as the Arbiter server.

4. The Development and Characteristics of a Visible Team Competition

In order to make the programming contest more interesting, attractive, and more understandable to the public, the Scientific Committee of the Chinese National Olympiad in Informatics (SC of NOI) authorized the Group of ACM/ICPC (International Collegiate Programming Contest) team of Peking University to develop a new style of programming contest where the running steps of the programs can be viewed on the screen. In 2006, we extended an open source software "Dominate Continent" and developed a contest named "risk" which is similar to the Java Challenge in ACM/ICPC. The difference is that "risk" accepts programs written in any programming language other than just Java in the Java Challenge. Details of "risk" are given below.

4.1. *What is the Meaning of "Visible"*

In the contest, we have several programs competing with each other to dominate as much land as they can. When the programs are running, the map is shown on the screen, see Fig. 1. There are four teams competing on the land. Each team is represented by a unique color. The teams play in turns. In each turn, teams play in a certain sequence. The play result of each step is shown on the map. In Fig. 1, the black circle with number "6" represents the army of team "PNJ_Y" and it is attacking and winning one of its neighbor land. Fig. 2 show the four teams in Fig. 1 competing on other maps.

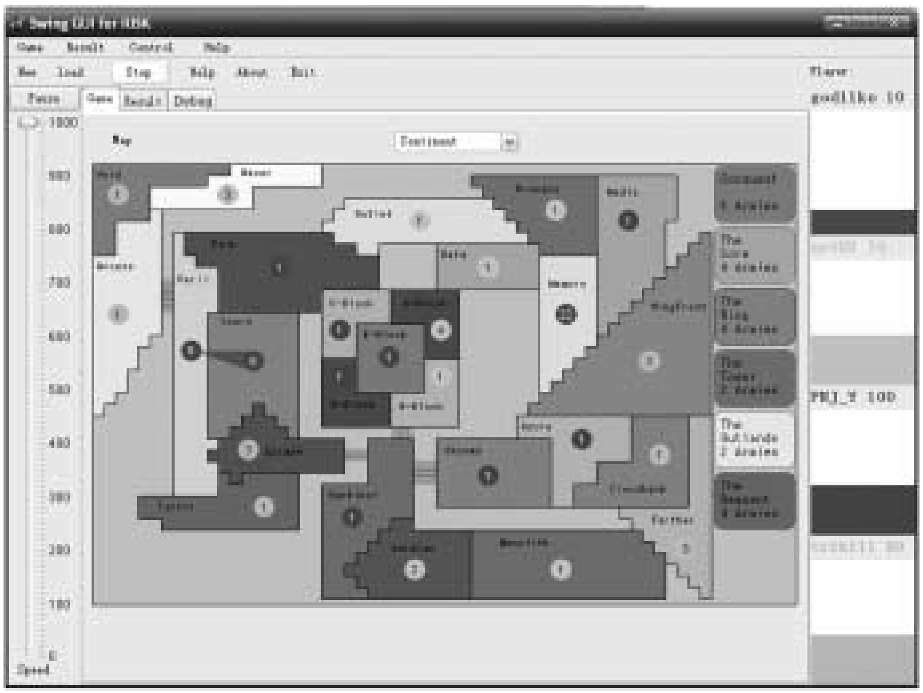


Fig. 1. Four teams (godlike, notHK, PNJ_Y, trikill) are competing on the land.

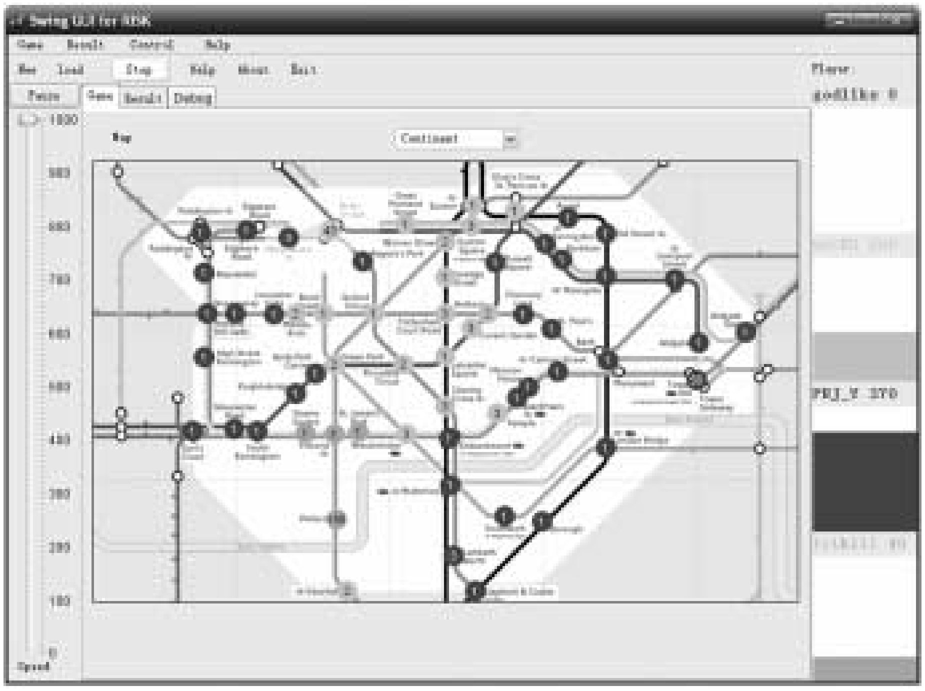


Fig. 2. Four teams competing on a train network.

4.2. *How to Organize the Contest*

The contest includes three stages, the first stage is round-robin; the second stage is qualifying; the third stage is final.

The round-robin has several rounds. In each round the system divides the teams into some small groups randomly. Each group has a separate game. In the same round, every group uses the same map. Maps in different rounds are different. Each game lasts three minutes. After a game, each attended team gets some points, which is equal to 10 times the number of countries it occupied, when the game ends. Each team's points are accumulated as a total score.

In the qualifying, the system divides the teams into small groups according to their scores in the round-robin. The top teams will not be placed into the same group. Each group plays for several rounds and only the champion is put through to the final.

The final still has several rounds, but only one group. The score in the final stage will be the final score.

The information is announced three months before the contest. All the provinces are invited to attend the contest. Each province may organize one team which may include at most five students. One week before the contest, all teams should submit their programs for the **Round-Robin** stage contest. Thereis a break between round-robin and qualifying, and a break between qualifying and final. Teams are permitted to modify their source code at any time, but only permitted to resubmit their source code during the breaks.

4.3. *Hardware and Software Used in the Competition*

Server: WinXP, JRE1.5, Python2.4.3,Tomcat5.5, FPC(Free Pascal Compiler)2.0.2, Dev C++ 4.0

Client: WinXP,JRE1.5, FPC(Free Pascal Compiler)2.0.2,Dev C++ 4.0

4.4. *Description of the Task*

Initial Army Placement

Every player rolls dice,to decide the order of play. Starting with the first player, everyone in turn places one army onto any unoccupied territory. Continue until all territories have been claimed.

Each player in turn places one additional army onto any territory he or she already occupies. Continue in this way until everyone has run out of armies. There is no limit to the number of armies you may place onto a single territory.

Playing

Whoever placed the first army takes the first turn.

Each player's turn consists of three steps, in this order:

1) getting and placing new armies;

2) attacking, if you choose to, by rolling the dice;

3) fortifying your position.

At the beginning of each turn, new armies you'll add to your territories based on

1) the number of territories you occupy;

2) the value of the continents you control;

3) the value of the matched sets of RISK cards you trade in;

4) the specific territory pictured on a traded-in card.

The above 4 steps are automatically calculated by the game and it will display how many armies you can place for that turn.

The task isribed in `http://162.105.81.202/noip/noip_game/game/Risk` _1.0.8.5.zip

5. Conclusion

This article gives some historic data recording the development and exploration of CNOI, and main contest activities organized by SCNOI and CCNOI of CCF. The selection mechanism for the best contestants has been proven to be effective. In addition, an evaluation system with some features, and a visible team competition used by NOI2006 are also presented respectively.

References

China Computer Federation (Eds.) (2007). *The Yearbook of Chinese National Olympiad in Informatics 2006* (*CNOI2006*), Henan Publisher Group of China.

A Summary of Chinese NOI Development Forum (2005). The file of SCNOI and CCNOI of CCF. Oct. 22–23, Beijing, China.

The Competition Rules and Measurements of Chinese NOI (2005–2006). The file of SCNOI and CCNOI of CCF.

Wang, H., B. Yin (2006). Visualization, antagonism and opening – towards the future of the IOI Contest. In *1st Workshop on Computer Science Competitions Reform*, Jan., Germany.

H. Wang received his PhD degree from the Department of Computer Science and Technology, Tsinghua University in 1993. He is currently an associate professor at Department of Computer Science and Technology, Tsinghua University. He also serves as the chairman of Scientific Committee of National Olympiad in Informatics of CCF.

B. Yin received his PhD degree from the Department of Artificial Intelligence, the University of Edinburgh in 1984. He is currently a professor at the School of Computer Science and Technology, Beihang University. He also serves as the vice chairman of Scientific Committee of National Olympiad in Informatics.

W. Li received her PhD degree from the Department of Computing, the Hong Kong Polytechnic University in 2004. She is currently a professor at Department of Computer Science and Technology, Peking University. She serves as a member of Scientific Committee of National Olympiad in Informatics of CCF. She is also the coach of ACM/ICPC Peking University team.

Olympiads in Informatics

Volume 1 2007

1822-7732(2007)1;1-G