INFORMATICA, 1996, Vol. 7, No. 4, 419–430

TEACHING OF COMPUTER PROGRAMMING BY ELECTRONIC MAIL

Jūratė BULOTAITĖ, Gintautas GRIGAS and Aidas ŽANDARIS

Institute of Mathematics and Informatics
Akademijos 4, 2600 Vilnius, Lithuania,
E-mail: ava@ktl.mii.lt

Abstract. Teaching of computer programming by electronic mail has been held in Lithuanian schools since 1992. School students of upper grades (8 – 12) took part in the experiments. The experiments showed that students cope better with first stages of programming (specifying the problem, etc.) than with the last ones (programming, debugging). The analysis of programming products (made by their peers) by students themselves may be expected to be an important component of learning and teaching.

The outcome of three year work is summarized and discussed. Observations and conclusions are presented. Distance teaching perspectives are discussed.

Key words: competition, distance teaching, programming, teaching of programming.

1. Introduction. Historically, the discipline of Informatics came into schools simultaneously with computers. As a school subject, Informatics is closely related to computer programming. A computer program may be considered as a constructive solution to a problem. Learning about programs develops children's aptitude for problem solving. They learn to analyze, specify, and formulate a problem (Arsac, 1985). Thus, through mastering programs, broader goals are achieved, including problem-solving and thinking skills (Olimpo *et al.*, 1985; Papert, 1980). Programming is more than a technical tool; it is also a tool for thinking, reasoning, and problem solving (Graf, 1985). It is a proper discipline to develop thinking and creative skills of a student (Papert, 1980).

The subject of Informatics is delivered to all Lithuanian school students of higher grades. Computer programming is a part of Informatics. Therefore, practically all school students have a chance to get primary acquaintance with some basic ideas of algorithms and programming in the classes of Informatics.

However, there is a number of advanced students eager to go deeper in programming. They need extra knowledge and skills in computer programming. Those students are scattered in various schools. Many schools cannot offer them any more knowledge in programming than that defined by the curriculum for an average student. The task is to help those advanced students. Such help may be ensured by some form of distance teaching (distance learning).

The first systematic approach of distance teaching of programming was made in Lithuanian Young Programmer's School by correspondence (Dagys, 1994) where land mail is used for communication with school students. The school was based in 1981.

A similar school for young programmers was organized in Russia between 1979 and 1985 (Junerman, 1984). The starting points of these schools were similar. However, the Lithuanian school was oriented more toward algorithmization and problem solving, and this orientation became stronger with time. It is reasonable to assume that the orientation of the Lithuanian school was better suited to distance teaching and perhaps this was among the main reasons for its survival. Teaching of programming by electronic mail naturally extends the work of the Lithuanian Young Programmer's School by correspondence.

Practical activities are more interesting and attractive than theoretical studies for school students, especially for those of lower grades (Papert, 1980). Therefore programming competition has been chosen as a form of teaching. Besides, it is known that elements of competition stimulate learning process (Schwiel, 1985). As the main task for the competition we have chosen the design of a small program.

The result of the work is expressed in an executable computer code. It may be transmitted by electronic mail and then immediately executed. This distinguishing feature of computer programming brings it into a favorable position among other school subjects if an electronic mail is used as a medium for distance learning. This was the main reason for our determination to choose the use of an electronic mail for teaching of computer programming where land mail is used for distance teaching of algorithms.

The preliminary results were originally described in Bulotaitė (1994); Dagienė and Grigas (1993); Dagys and Klupšaitė (1993); Grigas (1994a); Grigas (1994b). Here we'll try to summarize the results and observations derived from 6 sessions during the 3 years.

2. Organization of teaching sessions. Two teaching sessions are organized each school year: the first session in autumn, the second in spring. The second session is concluding. This session is organized as a competition. While the first session is a training session, i.e., preparation for the competition.

Students work in teams. Professional programmers usually work on large projects in teams. In order to make students' work more or less close to that of professionals it was decided to get them work in teams.

The number of students in a team and their age is not restricted. The leader of a team may be a teacher as well as a student.

Each session is divided into phases of two sorts: 1) development, 2) analysis and evaluation. During development phases students work on their own programming products. During analysis students and/or teachers make reviews of the programming products developed during the previous phase. Each session begins with a development phase. The second phase is analysis, the third – the development (improvement) of their own product, etc. The majority of sessions have 4 phases:

1. Development of the first (preliminary) version of the programming product.

2. Analysis and evaluation by students of programming products made by the fixed number (3 – 6) of other teams (their competitors).

3. Development of the final version of the programming product using their own first version of the product, products of other teams obtained at the beginning of Phase 2 of the session for analysis, and reviews of other teams as a result of Phase 2.

4. Analysis and evaluation by teachers of results obtained during all the previous phases, and a discussion (teleconference).

The duration of each phase varies from a day to a week. It depends on the complexity of a programming problem and how busy school students are during the session (i.e., whether they have classes or not), etc.

Main teaching and instructional materials for students were distributed as hard copies by land mail in advance. All information exchange during the sessions was organized by electronic mail.

Numerical data about all 6 sessions are presented in Table 1.

3. Programming problems. The requirements for problem solution (the job of a school student) are formulated so that these solutions would be as

Table 1. Numerical data about all sessions

	Date	Nb. of teams	Nb. of pupils
1	16/11/92 – 12/12/92	7	44
2	2/4/93 – 9/4/93	15	73
3	29/11/93 – 18/12/93	21	102
4	16/5/94 – 29/5/94	14	51
5	21/11/94 – 16/12/94	15	67
6	17/3/95 – 23/3/95	12	50
TOTAL	6 sessions	84	387

close as possible to a real programming product developed by professional programmers. The program has to be accompanied by the documentation, including summary, description of the idea of solution with motivation, the user's guide and the programmer's guide. Of course, a student's product can be very small in size. These requirements are common to development project of any (technical) discipline, i.e., not exclusive to programming.

A single programming problem was presented to students during each session. Here are the shortened formulations of all the used problems:

1. The computer reads Lithuanian text from a data file and converts it to Morse's code (i.e., to the corresponding sounds). The user listens to the sounds and types the text. The task – to design Morse code trainer.

2. There are certain means (e.g., indentation, leaving spaces after certain symbols, etc.) which make program text more readable and understandable. Given a set of such rules. The task – to design a program text editor which would edit the text of the given program according to the rules.

3. Lithuanian language has nine special letters (symbols). Four different code tables with Lithuanian letters are currently used. The task – to design a converter of Lithuanian symbols among different code tables.

4. A telegraph code is such code, when symbols from the original ASCII

table are presented by their original codes, while symbols not present in the table are coded by two symbols from ASCII table. The task – to design a telegraph code for Lithuanian alphabet together with encoding-decoding programs.

5. A much simplified version of chess is defined. The task – to design a program which would imitate the game (i.e., would play with the user) as well as a the strategy of the game.

6. Two players are playing with a dice. The initial situation is described by the starting position of the dice and an integer value n. The players make moves in turns. A move consists of rotating the dice by the degree of 90% and subtracting the number (written on the top of the dice) from n. The player wins if his rival makes n negative. The task – to design a program which would imitate the game (i.e., would play with the user) as well as a the strategy of the game.

The task formulations were not so strict or exhaustive. That was done deliberately in order to leave some room for the students to make their own options and decisions.

Turbo Pascal was used as a programming language.

4. Analysis of the teaching process and observations

4.1. Improvement of scores from session to session. In each session the list of participants changes. We have related the scores of novices and of those who have taken part in previous sessions. Indeed, the average scores of the "old" teams were about 34,8% higher than that of the newcomers.

4.2. Improvement of scores from phase to phase in the same session. Preliminary products of Phase 1 had to be improved during Phase 3. Thus, the difference between the scores of Phases 3 and 1 shows the actual benefit of teams gained during Phase 3. This improvement of the results during one session is shown on the diagram (see Fig. 1). We can see that greater improvements were gained by the teams with average success in Phase 1. Perhaps more successful teams had too little room to improve their scores, however, less successful teams did not manage to improve them. Therefore, we naturally arrived at the conclusion that the maximum learning benefit from the competition was gained by the teams with average results.

4.3. Improvement of scores of particular parts of programming product during two subsequent development phases. The diagram in Fig. 2 shows the compound parts of the product. Most of the improvements were connected

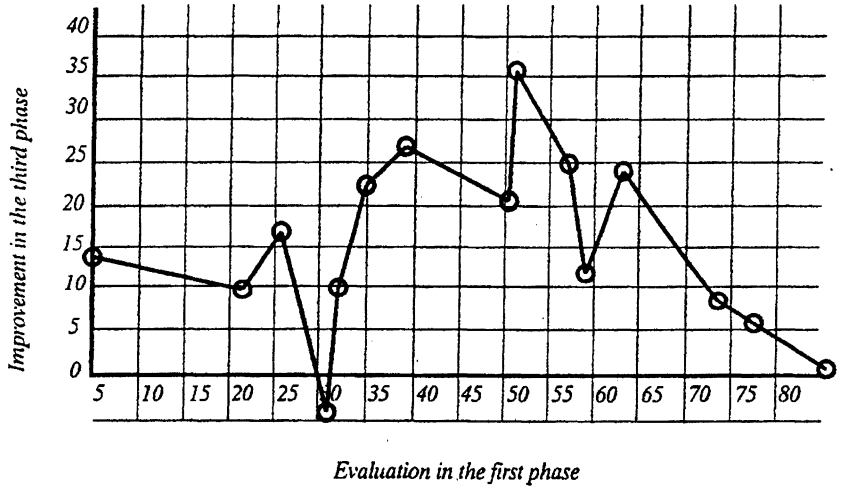

Fig. 1. Improvement of the results from Phase 1 to Phase 3.

with programs in general and with messages produced by a computer during the dialogue with the user. This was due to the fact that everybody could see those messages while executing the program and most of the critique was addressed to them in Phase 2 of the experiment.

Despite numerous improvements, the weakest point in the programs remains the interface between the program (the computer) and the user. The dialogue with the user is not always fluent. In every program one can face situations that leave the user very uncertain, i.e., the user is not supplied with sufficient information as to what he must or may do next (i.e., how to select the next step of the program which is being executed, how to change options of the program, etc.). This situation could by explained by the shortage of appropriate literature available for school students and their teachers.

The programming style of all the programs was more or less satisfactory. All the texts were readable. Most of them were supplied with comments.

In the final version of the programming product all the teams included all the required parts of documentation. The improvements made during Phase 3 were mostly stylistic.

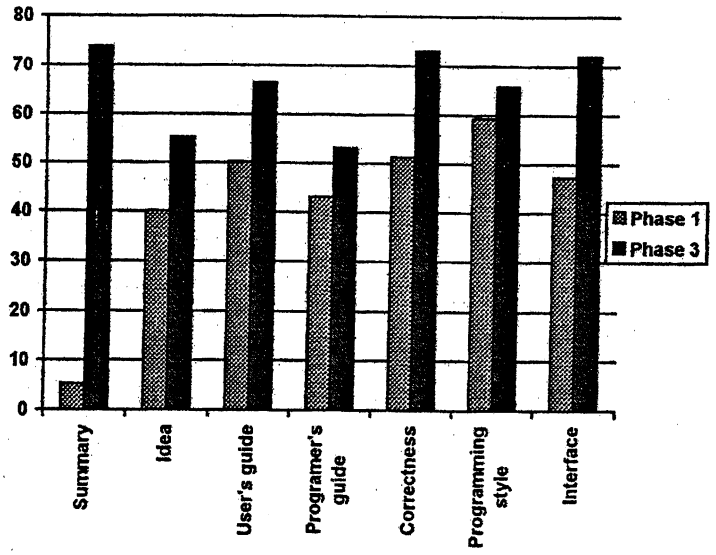

Fig. 2. Improvement of scores of particular parts of programming product.

Parts of documentation concerning the program text itself (the programmer's guide and the user's guide) were written better than those more sophisticated texts concerned with the first phases of the development of a programming product (description of an idea of the solution, summary).

These results lead to the suggestion to improve teaching of general knowledge instead of facts.

4.4. Reviewing of programming products. The analysis of students' programming products by the peers leads to considerable improvements in these products. This may be explained by the fact that reviewing exerts influence on the product in two ways. One way is direct and obvious: the author of the work learns about mistakes and shortcomings in his work from reviews. The second is indirect: when the author of the product becomes a reviewer of another realization of the same product, he gets an opportunity to compare his own product with the products under review and to see the mistakes and shortcomings in his own product better.

In the first experiments of reviewing some students as well as teachers complained that it was not interesting to read works of their peers. Only after one or two sessions most of participants realized how useful for them was reviewing.

4.5. Collaborative work of students. The number of students in team was not restricted. It ranged from 1 to 12. An average number was 4,7. Analysis of results indicated no relation between the size of a team and scores (success) gained by that team. This shows that school students fail to work together in one team. Perhaps collaborative work for students is still unusual and difficult. It could be seen the lack of coordination in the teams. This was also reported by teachers–team leaders during discussions. More collaborative work in teams is desirable in classes.

4.6. Age of the students. Students from 7 to 12 forms (age from 13 to 18 years) have participated in sessions. The analysis of the results indicated no relation between the age of the students and their scores (success). The success of younger students comparable with that of older ones leads to the conclusion that good results may be also achieved by learning programming in lower grades and by themselves.

4.7. Motivation. Students very like ranking their teams. They ask to do it in the training session as well. So the elements of competition are very desirable in teaching.

4.8. Data transmission costs. A new component of expenditure in teaching by electronic mail is payment for using telephone lines. An average amount of information transmitted during the whole teaching session from the coordinator to one team was approximately 93K bytes, and that in the opposite direction was 27K bytes. The total amount was 120K bytes. The transmission rate varied from 300 up to 2400 bps dependently on the interference in the telephone lines. The total transmission time of this amount of data was about 10 minutes. Seven letters would be necessary for an equivalent exchange of information by land mail. The payment for one minute of the conversation by phone in the daytime in Lithuania in 1992 was equal to the cost of postage stamps for a single letter. At this moment 3 minutes of interurban telephone calls equals to the cost of postage stamps for a letter. So the economics is in favor of an electronic mail.

However, distribution of all permanent teaching materials by land mail as hard copies is preferable. The reasons are other. Cost of typographically printed materials is lower than that of materials printed by computer in school and the quality is higher.

It is also valid that data transmission costs do not make the essential part of the total expenditure.

5. Conclusions and discussion towards future work. Distance teaching experiments showed that school students cope with first stages of programming (specifying the problem, developing methods and algorithms for problem solution) better than with last stages (programming and debugging), in particular, with the development of a fluent dialogue between the computer and the user. This can be explained by the fact that in our schools more attention is paid to the teaching of algorithms and less attention is paid to programming and especially to programming practice.

School students get basic knowledge of algorithms in general classes of Informatics. In some schools they get some acquaintance with programming too. However, they write programs only for themselves to carry out some calculations. The results of calculations (not the program itself as a product) are the target.

There was another attitude in the distance teaching sessions. Students were to elaborate a programming product for the user. This product (program text together with documentation) was set as a goal. Their activities during the programming practice to some extent simulated the work of professionals. It was obvious the need of additional literature on the design of the dialogue between the computer and the user, as well as on the testing of the programs. More exhaustive teaching materials including textbooks, tutorials, examples of programming problems together with the samples of their solutions are desirable.

The analysis of products by students themselves may be expected to be an important component of learning and teaching. However, the teaching methods need some corrections to make the analysis more interesting and attractive for school students.

More investigations and special attention are needed in further experiments. Distance teaching based on the electronic mail is a proper medium for such experiments because it ensures a fast exchange of information.

Acknowledgments. The work was supported by the UNESCO under contract CII 408 018.2.

Many thanks to Miss Audronė Klupšaitė, and Mr. Rimas Misevičius for helping in evaluating students' programs.

REFERENCES

Arsac, J.J. (1985). Teaching Informatics in high schools: a French experiment. In K. Duncan, D. Harris (Eds.), *Computers in Education.* Elsevier Sc. Publ. B. V., IFIP, North-Holland. pp. 669–703.

Bulotaitė, J. (1994). Distance teaching of programming in Lithuania. *Proceedings of European Distance Education Network (EDEN) Conference,* Tallinn, Estonia, pp. 179–181.

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

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

Dagys, V. (1994). The work principles of Lithuanian Young Programmers School by correspondence. *Proceedings of European Distance Education Network (EDEN) Conference,* Tallinn, Estonia. pp. 182–184.

Graf, K.D. (1985). Computer science/Informatics – A challenge to mathematical education and vice versa. In K. Duncan and D. Harris (Eds.), *Computers in Education.* Elsevier Sc. Publ. B. V., IFIP, North-Holland, pp. 669–703.

Grigas, G. (1993). An experiment of computer programming practice by e-mail. *Interpersonal Computing and Technology: an Electronic Journal for the 21st Century,* **1**(2), 1–10.

Grigas, G. (1994a). Impact of the electronic mail on teaching of computer programming. In K. Brunstein, E. Raubold (Eds.), *13th Word Computer Congress 94,* Vol. 2. Elsevier Science B.V., North-Holland. pp. 651–655.

Grigas, G. (1994b). Distance teaching of Informatics: motivations, means, and alternatives. *Journal of Research on Computing in Education,* **27**(1), 19–28.

Junerman, N.A. (1984). All-union school of programming by correspondence. *Informatics,* **1**(1), 156–166.

Olimpo, G., D. Persico, L. Sarti and M. Tarell (1985). An experiment in introducing the basic concepts of Informatics. In K. Duncan and D. Harris (Eds.), *Computers in Education.* Elsevier Sc. Publ. B. V., IFIP, North-Holland. pp. 669–703.

Schwiel, A. (1985). Evaluating and improving informatic education in secondary schools by means of programming contests, *Computers in Education.* Sci. Publ. B.V., IFIP, Elsevier. pp. 25–30.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas.* Basic Books, New York.

Received July 1996

J. Bulotaitė graduated from Vilnius University Applied Mathematics department in 1993. Currently she works as an assistant in the Institute of Mathematics and Informatics in the Department of the Programming Methodology. Her research interesets include distance learning, programming methodology.

G. Grigas received the Degree of Doctor of Technical Sciences from the Kaunas Polytechnic Institute (Kaunas, Lithuania) in 1970. He heads the Department of Systems Programming at the Institute of Mathematics and Informatics. His research interests include abstract data types, propgramming methodology and teaching.

A. Žandaris graduated from Vilnius University Applied Mathematics department in 1991. Currently he is making postgraduate studies in the Institute of Mathematics and Informatics and Vytautas Magnus University. His research interesets include distance learning, logical programming, recognition of natural languages.

PROGRAMAVIMO MOKYMAS KOMPIUTERIŲ PAŠTU

Jūratė BULOTAITĖ, Gintautas GRIGAS ir Aidas ŽANDARIS

Nuo 1992 m. Lietuvos mokyklose buvo vykdomas programavimo mokymo kompiuteriniu paštu eksperimentas. Jame dalyvavo 8–12 klasių moksleiviai. Eksperimentas parodė, kad moksleiviai lengviau įveikia pradinius programinio produkto kūrimo etapus (uždavinio formulavimas), negu tolesnius etapus (programos derinimas, testavimas). Eksperimento metu moksleiviai turėjo analizuoti ir vertinti kitų moksleivių sukurtas programas. Eksperimento analizė parodė, kad tai yra labai svarbi mokymo dalis, į kurią verta atkreipti dėmesį planuojant tolimojo mokymo programas.