Programming Versus Application

Péter Szlávi[1], László Zsakó[1]

[1] Lorand Eötvös University, Faculty of Informatics, Department of Teacher's Training in Computer Science
1117 Budapest, Hungary
{szlavi, zsako}@ludens.elte.hu
http://digo.inf.elte.hu

Abstract. All over Europe including Hungary there has been serious disputes for years about teaching informatics, about its goals and its possible contents. In this field the sharpest question is the problem of teaching programming and/or application. Do you need one or the other? If both, what is their accurate proportion? Why might you need either of them? Which age group should be taught which of them? This article is aimed at finding the answer to these questions.

1 Introduction

There are several fundamental issues in teaching informatics that have not been satisfactorily solved so far. Informatics as a school subject varies from country to country, region to region. There are countries like France where there is not a compulsory school subject called informatics; in this case students are taught IT skills within the framework of other subjects. In many other countries, however, informatics is defined as an independent subject. This article is not aimed at making a choice between the two possibilities or adducing pros and cons. The authors as well as the Hungarian education government of the past 12 years definitely support an informatics as an independent school subject.

Where there is a school subject called informatics, the next question that arises is: which age groups are to learn it. Although this question is not tackled in this essay, either, we would like to state our position. We believe that in each year of primary and secondary education (i. e. years 1 to 12 in Hungary) students need IT skills therefore informatics should be taught as an independent subject minimum from the third year,[1] which we are trying to prove in section *Informatics at school*. On the other hand, we think that in the last two years of secondary education you cannot define a uniform informatics subject. Surely those who want to go on to university and study computer science must learn something different than those who will start work after leaving school.

[1] Moreover, we agree with Márta Turcsányi-Szabó, who launched the first informatics experiment at Hungarian kindergartens, that even in the kindergarten informatics may be an important tool to develop children's skills and stabilize the community [1].

Starting teaching informatics at an early stage is not a unique phenomenon: e.g. in a decree for primary and secondary schools published by the French National Ministry of Education, it is stated that even pupils of elementary schools i.e. age group 6 to 11 are to be encouraged to use computers, certain software, multimedia, electronic mailing and the internet. [2]. As for the English National Curriculum, Information and Communication Technology (ICT) appears at the very beginning of education i.e. for age group 5 to 7 (Key stage 1)! Although there is not an independent school subject called informatics, ICT is expected to be used every day [3]

Regarding Hungary, the National Institute for Public Education worked out a version of informatics as a school subject called Adventures in Informationland [4] for years 1 to 4. Schools themselves can decide whether they are to include it in their curriculum as a compulsory subject or not. The syllabus is structured as follows:

Year 1	**Year 2**	**Year 3**	**Year 4**
Information games (8)	Getting to know our environment (3+3+2)	Signs and codes (4+4)	Information coding (2+3) Sending a message (3+3+2)
Playing with algorithms (4+3+3)	Everyday algorithms (4+5)	Algorithms in our everyday life (4+5)	Using algorithms (3+3)
Getting to know the computer (15)	Making friends with the computer (8+7)	The computer is our help (14)	Creativity with the computer (10)
Making friends with books (2+2)	Visiting the school library (2+3)	Visits to the library (6)	In the library again (5+3)

Therefore, the starting point of this article is: there is a subject called informatics nearly in every year of primary and secondary education. We will not go into a detailed description of what topics should be covered, either. We will only tackle the problem of programming and application.[2]

We would also like to sum up why you need teaching programming and/or application (the Delusions of Informatics Education). Then we will move on to survey the goals of programming and application skills. The next part will be dedicated to the contents of algorithmization and application skills (Fields of Informatics). Finally, we will look into what kind of knowledge and skills should students possess at different ages.

2 The Delusions of Informatics Education [5]

We are trying to enlighten the two fields to be taught, their importance and proportion by describing the faulty extreme views and discussing them with a critical eye.

[2] Besides there are several important fields like infocommunication, media informatics etc.

Informatics education = teaching users only

According to this, informatics is about being able to use computers (and other devices) properly. This delusion holds that the development of abilities, the improvement of problem solving skills, the practice of problem solving activities and the ability to invent are not part of informatics[3].

This delusion was created as an opposite to the following delusion (informatics education = teaching programming), and it took nearly 10 years to fight it. It is interesting to note that people who fought against this idea were the ones that had earlier fought against the opposite delusion as well.

The supporters of this delusion often say that it is unnecessary to teach programming because only a few students will become programmers. This is nonsense, and can only be accepted by the blind. Some of the questions that illustrate this are as follows:

- *Why do we teach mathematics: do we want everybody to become a mathematician?*
- *If only a small number of students become a historian, why to teach history?*

Mathematics is taught because it improves thinking and other abilities. The role of informatics is very similar to this[4], which means that the teaching of programming can only be justified if in the certain age groups there are such skills and abilities, in the development of which it may play an important role.

Informatics education = teaching programming only

This is the opposite of the previous delusion and it claims everything that is denied by the former. Namely, there is no need for a new form of computer literacy, informatics does not change our everyday life, or at least not in a way that should be taught systematically.

They set the following examples: we are not taught to use the telephone or the television at school, and this aspect is true for all the applications of informatics. Even the starting point of this statement is false: most people use their mobile phones (even the remote control of their TV set) in a primitive, very limited way. Informatics invades our everyday life: even our simple devices are becoming more complex and multi-functional, with a number of opportunities. It is a well-known fact (both in pedagogy and in programming) that beyond a certain extent of complexity, frontal recognition, problem solving techniques become difficult, the process of acquisition slows down and it requires such an extent of abstraction skills and notion recognition to understand the logic of the system that can be developed much slower alone than in a well-constructed learning process.

This delusion appeared when computers could only be used for programming, which was typical in the 80`s after the introduction of personal computers that could be programmed in BASIC language only. Later, when application systems became widespread, this delusion was pushed into the background and

[3] Even the caveman used his invention skills to rise over animals.

[4] For mathematics education it is worth considering the following statement: programming can be an experimental device for mathematics! Nevertheless, pedagogy considers experimentation extremely useful in the process of recognition.

Our world is full of algorithms: we always do algorithms in our everyday life, daily work or while studying. Therefore it is in our own interest to improve our knowledge to understand, execute, even to design algorithms.	nowadays it is supported by mainly informatics experts working in secondary education, whose job is to distribute programming tasks.

To be able to avoid the two extreme delusions (teaching only users vs. teaching only programmers) the best course of action is to follow an advice of the famous Hungarian actor Gyula Kabos: to take a little bit of this and a little bit of that as well.

3 Why Do You Need Both?

Solving application tasks	**Algorithmization, data modelling**
IT tools invade our world. Only those can make good use of the opportunities of the new information society that regularly use these tools. Since they are fairly sophisticated, the stress is not on their routine use but on the knowledge of the opportunities they offer and their creative use. In this field it is important to approach computer use from the problem side, where the question is whether a certain – the given – general program can be applied or not (and we are less interested in the way how it can be used).	In school as well as in your everyday life you keep performing algorithms when filling in data structures – questionnaires, forms – designing action sequences, information-flow processes. This world cannot be fully understood by those who are not aware of the basics of these actions. In your everyday life including school and the various subjects you learn/teach you may have to face several problems that can be – moreover are sensible to be – solved by computers. First students must be able to realise whether a problem or any of its parts can be solved by using IT tools. The next step is solving the problem arisen with the aid of those IT tools.

4 Fields of Informatics [6]

Below we are trying to sum up what we mean by the two important fields of IT knowledge. We are primarily describing the goals and referring to connections with other fields of knowledge. [7]

Solving application tasks	**Algorithmization, data modelling**
This scope of knowledge emphasizes computer application from the point of view of the problem and the question is whether – the given – general program	Algorithmization is an important element because of the development of thinking skills. It is the ability of problem solving; actually not only of solving routine prob-

can be used for problem solving or not (rather than the way how it can be used).

We do not concentrate on the tool: the computer or the software. Within the frames of Informatics tools it is the hardware, while in the Application systems and Informatics tools part (that is operating system as program system) it is the software that is focused on. There is a similar idea in the Computer-assisted program solving scope of knowledge as well, but while there the choice is made on the basis of the ability of the 'whole' computer, here the software applied is fixed and only its 'philosophy' is to be studied.

We rely on the knowledge of certain notions and skills that belong to other areas of knowledge. Thus the existing or developing ability of algorithmic thinking is a definite advantage (this is meant to be developed within the scopes of knowledge in Algorithmization and Data modelling).

As it has become evident from the facts mentioned above, the teaching material of this area of knowledge does not exclusively belong to informatics as a subject. There are certain important areas that seem clear today, but their number can be increased with the rapid development of informatics:

- word processing: compilation of text documents, publications in traditional and electronic form;
- constructing graphic images and objects: constructing and processing diagrams, graphic figures, photos;
- spread sheeting: arranging data in a table, making calculations;
- database management: storing, arranging, grouping data, making reports;
- presentation: making presentations, electronic notice boards, billboards;
- multimedia design: designing video

lems but those that require a kind of independence, sound judgement, originality and creativity. That means that a basic objective of teaching informatics is to emphasize the systematic planning of problem solving.

We learn to understand the world around us with the help of models. *Programming* can be a useful way of developing modelling ability and making students think logically. Because it has to be formalized, it requires a precise, exact way of thinking. The scene of formalization in program composition is data modelling and algorithmization, thus elements connected to them should be taught here. Formalization should be extended with care: using examples, notions that are appropriate to the age group. We regard it very important to lay emphasis on this in education from the very beginning (e.g. it will be regarding the improvement of abstraction skills, or concerning the effectiveness of computer use).

First, algorithmization is not about computer-assisted execution. In most of the cases, the person who created the algorithm can perform it in his mind as well. It is only then that an automatic machine, the computer, can be made to process the precisely constructed algorithm.

The aim of the application of the computer is to create (new) output data from the input data with the help of programs. That is why teaching data structures and algorithms cannot be separated.

The point is that students should realize that the basis of computer-assisted problem solving is algorithm elaboration (and not coding)! However, the knowledge of programming languages is required to reach this, because programming cannot effectively be taught from books only, students need to try their and audio files, animation.

Many different subject areas should be considered in order to draw up the problems. The many areas of application at a higher level can be studied within the subject informatics. Handling knowledge should be included in a certain subject if the tool is closely linked to a special profession (e.g. CAD).

In this case the stress is on the use of the tool: the program, as a tool is used to solve the task.

programs on the computer, as well. We would like to note that teaching a *programming language should not be the primary aim* in studying programming.

It is important for the students to get acquainted with traditional programming structures independently of programming languages. Therefore this scope of knowledge describes computer-assisted problem solving as **tool improvement**, where the problem solving tool (the program) should be created.

5 Informatics at School

What you will find below is based on the Informatics Section of the Hungarian National Curriculum formed in the past 10 years [8], as well as on the requirements of the school-leaving examination. The authors' ideas were included in the last (2005) version of the Hungarian National Curriculum, as well. [9]

Solving application tasks	**Algorithmization, data modelling**
Years 1 to 4	
The main goal of the first four years is making friends with computers and laying the foundations of a future ”harmonic” relationship. Thus what you need here are playful programs and tasks; on the other hand, it is important in other fields of science, as well. Nevertheless, playfulness cannot become dominant and for its own sake; its direct goal i.e. why you are using the computer/program should remain absolutely clear. Playful programs usually do not mean traditional computer games though their introduction into the learning process should not be ruled out in advance, either. For this age group, application skills could primarily mean image and music editing. The very first application field – leading from the kindergarten to school – may be the use of the so-called *stamping programs* (similar to children's real rubber stamps). Drawing at this level means fusing	Pupils must be able to formulate algorithms and to execute simple everyday algorithms (morning routine, crossing a street etc.). In order to form these skills, teachers are free to choose their own devices (e.g. Logo-tortoise, robot games, Lego etc.), but they should use a variety of them. Pupils must become aware of the fact that each step of the algorithms must be unambiguously executable and it is not the device that counts. They must realise everyday objects can be described with data, some of which are numbers, others are texts or others (e.g. colour, drawing, music etc.). They must be able to tell the difference between them. Here they should learn the basic concepts of orientation, directions and the measurability of distances. This goal can mainly be achieved by algorithmic games. Data can be sorted: numbers ascend-

simple line drawings and patches (colours and textures). If pupils combine drawing, music and some text, they can prepare simple multimedia displays and animations. When using music applications, they can play the music, do some simple editing or write their own music.

They can prepare invitation cards to a birthday party, carnival posters, classrooms decorations etc. Naturally, teachers' guidance is essential here. It is just as vital that pupils create their own compositions on a traditional medium so that they can take them home and show them to their parents and friends.

It is important to note that good IT applications can highly develop pupils' manual, coordination, calculation, reading and writing skills.

ing/descending while texts and words alphabetically. In maths classes there are several manual data processing tasks that you can later make use of when writing algorithms e.g. When teaching colours and geometric forms you can ask a question such as what is more numerous? (counting), Is there a red triangle? (decision making), Select the shapes bordered with straight lines only (selection), Group the shapes by their colours (grouping), Sort them ascending by their size (sorting) etc.

Years 5 to 6

For the 10 to 12 year olds the scope of application possibilities widens.

The most widespread task types here are the ones related to their school and home life like creating, printing, storing and correcting documents in accordance with the interests of this age group.

These are mainly drawings: figures accompanied with a little text such as invitation cards to birthday parties and carnivals, greeting cards, school and class badges, various kinds of posters, playing cards, token money etc. They can prepare the layouts of flats, their classroom, schoolyard etc. on their own. We can state a basic principle: when compared to the previous age group, the difference is a bit more text in the documents created.

The word-processing tasks of the next age group can be introduced by the manual text editing methods: assembling a text from ready-made parts, cutting and pasting as well as swapping parts, making drafts etc.

In these tasks it is sensible to make

Pupils must be able to formulate precise algorithms and to design simple everyday algorithms (morning routine, the recipe of making tea, crossing a street etc.).

Importantly, they are to create algorithms that they can act out themselves but at the same time they must be executable by computers, as well. The best area for this purpose is moving and drawing (Logo).

They are to formulate what is meant by an algorithm (they can be reduced to steps but the steps themselves are also algorithms; executable with a fixed order of execution; something happens to something at each step)! Relying on their abstraction skills, they should be able to divide data from algorithms that “operate” on them.

They should realise that you use three types of elements when constructing an algorithm:

- each of the atomic steps must be executed (in the given order),
- one of the atomic steps should be

pupils play the key part. However, it is true again that it is essential for the teacher (language and art teachers) to participate in the process of creation and teach the children the aesthetic and formal concepts i.e. the teacher's primary task is to add ”theory” and to introduce the ”methods” and the opportunities they offer.	chosen and then executed, • the atomic step should be executed iteratively. Since the steps of an algorithm could be other algorithms, and they can be named, the concept of procedure can be evolved. [10] According to György Pólya: “If one observes it more closely, one may notice that the solutions of many problems actually consist of procedures, processes of actions and sequences of appropriately related operations i.e. a modus vivendi.” [11] It is worth introducing this age group the tools and methods of manual data management as they can meet tables and diagrams within the framework of other school subjects. For instance, their first data managing tool is their school report book. They also encounter tables of competitions and might want to sort the rows of the table by scores etc. The descriptive concept of sorted an unsorted data. Sorting the same data set by various aspects. Systematic manual sorting (e.g. creating alphabetical order by pasting).
Years 7 to 8	
As for teaching 13 to 14 year olds the philosophy does not differ greatly from those descripted above. Maybe just there should be less teachers' guidance (the length and detailedness of the first presentation). The students are already capable of independently using the computer and the programwarehouse of the school to solve the above mentioned tasks. Typically, teachers can set tasks to be solved with the help of a computer as homework. Tasks related to their school and home life like creating, printing, storing and correcting documents and tables in accordance with the interests of this age group. Here the stress is laid upon text	Students must be able to write down input and output data necessary for information management and assign them to each other. (At this important stage of abstract thinking, they are able to select details of actions from the purpose of the action and handle them independently; i.e. they can operate with actions as a “black box”.) They must be able to analyse the output data from a given point of view and use them for a given purpose. Making students understand the concept of data: they must tell apart scalar (number, character etc.) and compound data (array, table, text etc.) For this age they must be able to consciously apply the principle of refinement step by step. They must be able to

documents which students should often attach figures, sometimes even tables. A possible task may be preparing a school newsletter, schedules of a summer camp, timetables, business cards, invitations to school competitions, their schedule and results. Before they start creating the documents, it is worth teaching them about the aesthetics and typography a text (relief effect, page setting, dividing content units, the role of highlighting etc.). A table should first appear within a text file; relying on this, students can compute some typical values and illustrate the data in the table with a diagram. You can also include searching in public computer information systems (task banks, cultural programmes etc.). Students must be able to collect information, paste it into an existing database and then select it from the database and process it. Relying on text and graphic documents creating an electronic noticeboard or a slide show.	independently formulate, name, parameter sub-algorithms and use them in constructing algorithms. As the technique of step-by-step refinement is an important problem solving principle not only in the field programming, understanding and learning it may be of great use for every one. Understanding the tools of algorithmic abstraction (procedures, functions and recursion), realising their usefulness and using them. Now the algorithmic structures learnt visually by experience at the previous stage should be used consequently and precisely. You may also introduce their common names widespread in computer science: sequence, branch, loop and procedure.
Years 9 to 10	
In the curriculum of the 15 to 16 year olds playfullness is diminishing giving way to “reality”: in the problems to be solved there are more data and their relations. These tasks require another kind of “creativity”: collecting data and exploring their relations i.e. modelling also organically belongs to the problem at a basic level. Tasks related to their school and home life like creating, printing, storing and correcting documents tables and databases in accordance with the interests of this age group. Majority of the written materials are texts such as letters, essays etc. A wider application of tables and diagrams created based on them, computation and plotting of simple statistic data	Atomic and compound data (set, array, record, file, stack, row, graph), types of file (sequential and non-sequential), for task types (sum, decision, selection, linear search, count, maximum selection, at least one type of sorting) and realising them on the computer. Understanding and systematic use of the tools of algorithmic abstraction (procedures and functions). A basic requirement is that the programs written should be quite expressive i.e. provide enough information as well as a way to enable a dialogue between the user and the computer. Students must understand that a program is a product and its writer is a product-making craftsman. They must also realise its consequences.

and evaluation of physical and chemical measurements with spreadsheets. A possible task may be planning and calculating the finances of a school trip, the evaluation of school competitions/ championships with spreadsheets.

Beyond spreadsheets, it is worth mentioning the opportunities of GIS applications such as inserting maps and completing them with data.

Defining and using databases closely related to their everyday life like keeping records of their cassettes or CDs, making their own telephone directory as well as technical and school-subject related databases.

With the aid of text and graphic documents, students might be able to make an interactive electronic newsboard or an information board.

They should become familiar with the basic rules of data modelling and understand that a database is not simply a file but a planned structured system of data and their relations.

An objective of data processing is that the data entered in one way could be queried from an other point of view. Data query is a creative-analytical process based on exploring data relations. An essential feature of modelling that you are aware of what kind of information will the ready program be able to provide.

Students should understand the concept of database and the related elements (file, record and field). They should know that the logical and physical ways of data representation are different.

Years 11 to 12

As for teaching IT to 17 to 18 year olds, you may mainly come forward with application programs that best suit the profile of the given education institution. This primarily holds for vocational schools and partly for the students of general schools that do not want to continue their studies at higher education institutions after leaving school but want to acquire some more profound, more special IT skills that facilitate their success at the labour market.

Regarding those that go on to university, less stress is laid on this field: e.g. it contains the application of spreadsheets for solving mathematical problems.

Here – and partly at the previous age group – teachers should once again come forward with drawing, image editing and image processing tasks, now at a higher level, relying on the more serious mathematical knowledge students have acquired in the meantime. With the aid of the above, students might be introduced to multimedia design.

Atomic and compound data (set, array, record, file, stack, row and graph), file management, relational data structures. Basic algorithms for task types (sum, decision, selection, search, count, maximum selection, at least one type of sorting). Recursion in the world of tasks, data and algorithms. Algorithm designing techniques. [12]

Program writing as a process of production (defining a task, designing, encoding, testing, debugging, efficiency and quality testing and documentation).

With the help of text and graphic documents, they may also try making presentations as a new area of applications.

6 Conclusion

We do hope that surveying the fields of application based ICT and programming instruction in parallel, we managed to show their importance and the role they play in education. Their proportion is greatly affected by students' skills, interest and the type of school they attend. Based on the above, we believe that in an informatics for everyone the proportion of knowledge about algorithmization should be at least a third and at most half of the time devoted to teaching.

References

1. Turcsányi-Szabó, M: Approaching Arts through Logo. Sixth European Logo Conference, Budapest, Hungary, pp20-23 August, 1997
2. Nouvelles Technologies. (1999) Mise á niveau informatique en classe de seconde – rentrée 2000. Bulletin Officiel du ministère de l'Education Nationale et du ministère de Recherche. N25 du 24 juin 1999. 1177–1181.
3. The National Curriculum for England (1999): Information and Communication Technology. Qualifications and Curriculum Authority. London. (http://www.nc.uk.net/download/IKT.doc)
4. Körösné Mikis, M.: Kalandozások Információországban, (Adventures in Informationland) http://www.oki.hu/oldal.php?tipus=cikk&kod=oktatas-korosne
5. Szlávi, P., Zsakó, L.: Delusions in informatics education. Teaching Mathematics and Computer Science, Vol. 2., No. 1., pp151-162, 2004.
6. Szlávi, P., Zsakó, L.: Informatics as a particular field of education. Teaching Mathematics and Computer Science, Vol. 3., No.2, pp283-294, 2005.
7. Zsakó, L.: Teaching Informatics in Hungary. The IOI'96 NewsLetter, No 2, pp5-6, No 3, pp5-6, No 4, pp5-6, 1995.
8. Turcsányi-Szabó, M., Ambruszter, G.: The past, present, and future of computers in education – the Hungarian image, International Journal of Continuing Engineering Education and Life-Long learning., UNESCO, 2001 Volume 11, Nos 4/5/6.
9. The National Curriculum for Hungary (2005), http://www.om.hu/main.php?folderID=391
10. Hvorecky, J., Kelemen, J.: Algoritmizácia, elementárny úvod. ALFA, Bratislava, 1983.
11. Polya, Gy.: Mathematical Discovery on understanding, learning and teaching problem solving. John Viley & Sons Inc, New York, 1962.
12. Kátai, Z.: "Upperview" algorithm design in teaching computer science in high schools, Teaching Mathematics and Computer Science, Vol. 3., No.2, pp221-240, 2005.