

Today's Concepts of Teaching Computer Science Basics and Occupational Profile of Software Engineer

H. Budzisz, K. Kadowski, W. Suslow

Abstract. In the paper the technique of teaching the basics of Computer Science (CS) has been discussed. The level of knowledge and skills for students beginning CS study has been defined. Starting from this requirements, the process of teaching IT in secondary school was analysed. The role of programming skills was underlined. Methods and tools necessary to reach the proper level of this skills and solve real problems has been discussed.

Key words: education, IT teaching, programming skills, syllabus

Introduction

As in the late 90's [1] the popularity of computer science (CS) courses in the first level studies (BSc.) is still the highest now [2]. Strong interest of the secondary schools graduates in studying computer science has been observed for several years. About 3.5 percent from a total of 1,788,000 students were studying CS during the 2002/2003 academic year. This trend is confirmed by the result of a detailed study [3] that analyzed the changes of course and specialisation preferences of first year students (compared academic years: 1999/2000 and 2002/2003). "The number of the first year students majoring in CS increased about 8.3 thousand" during last three years. For example, about 10.5 percent of students were studying CS in the first level (BSc.) studies.

This situation is a challenge for teachers because the increasing interest in knowledge related to computers and computer technology in secondary school can be observed. But the question is if the teachers are methodically enough prepared to give their students proper knowledge? That question is especially important with respects to that part of graduate students who plan BSc. study in CS and who would like to be competitive on European work market as software engineers. The nature of that occupation is very close to the rule of playing the violin – you cannot master this activity if you do not start in the early childhood. Specialised learning in school is directly concerned with the professional success of a software engineer. Therefore the discussion of conceptual ideas of the teaching process efficiency in that quick changing branch of science is very gainful.

In the author's opinion, the choice of CS teaching strategy in school should be based on the following thesis *"the main task of the basic course of CS is to work out the style of thinking, to form basic habits, skills and views in the information technology without teaching specific technologies and tools"*. A course like that has to be a fundamental basis of the information culture for all students. In a good way it can serve as a basis for the next stages of professional learning in vocational schools and universities.

Integration of IT with didactical process of the school

Teaching CS and Information Technology (IT) in grammar schools and secondary school is realised in Poland using spiral method on all levels of learning. CS and IT education is based on specific classes (classes of Information Technology) and as well as on some knowledge presented in other school subjects. The students can learn IT on the basic and advanced levels and the studies can be concluded by secondary school certificate exam. IT classes in the advanced mathematics-computer science course and mathematics–physics course take from 4 to 8 hours a week in the 3–year educational cycle. Natural connections between IT and other school subjects are shown on Figure 1.

It is easy to see, that having exceptional IT knowledge can not guarantee the success of IT professionals in the future. A good mathematical-logical preparation is necessary. Knowledge

of physics is also useful because it provides a wide mental outlook and a precise look at the task. Another important requirement such as good linguistic knowledge comes as a surprise to CS pretenders. Not only should they be familiar with basics of English (it is historically natural), but also be proficient in their native language. From experience of work with first year students it is possible to say *that there is a distinct correlation between skills in programming and difficulties in talking in natural language.*

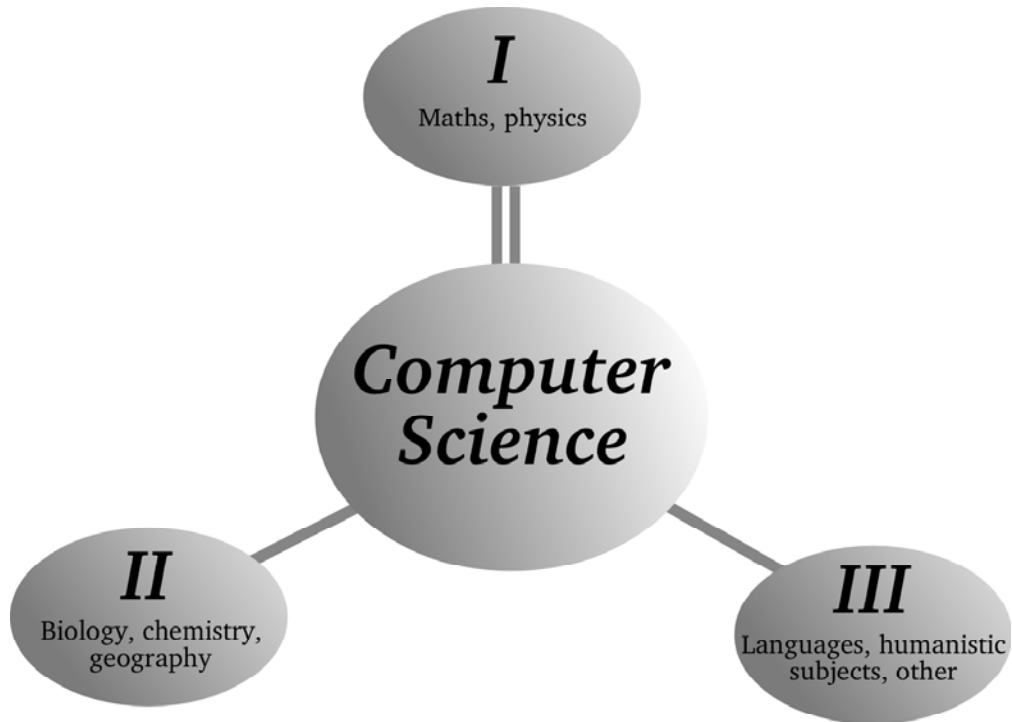


Figure 1. Correlation between IT and other subjects at school (I - group of subject with double-direction correlation; II - group of subjects with one-direction correlation which is afield of work of programmers, for example modelling or statistic; III - group of subjects with one-direction correlations wich use multimedial tools of information technology and computer science).

Authors analysed the requirements for CS studies of leading Polish educational institutions. Attention was paid not only at obligate knowledge requirements but also at indirect hints at minimum knowledge. Beginning level of knowledge and skills for students beginning CS study is described as following:

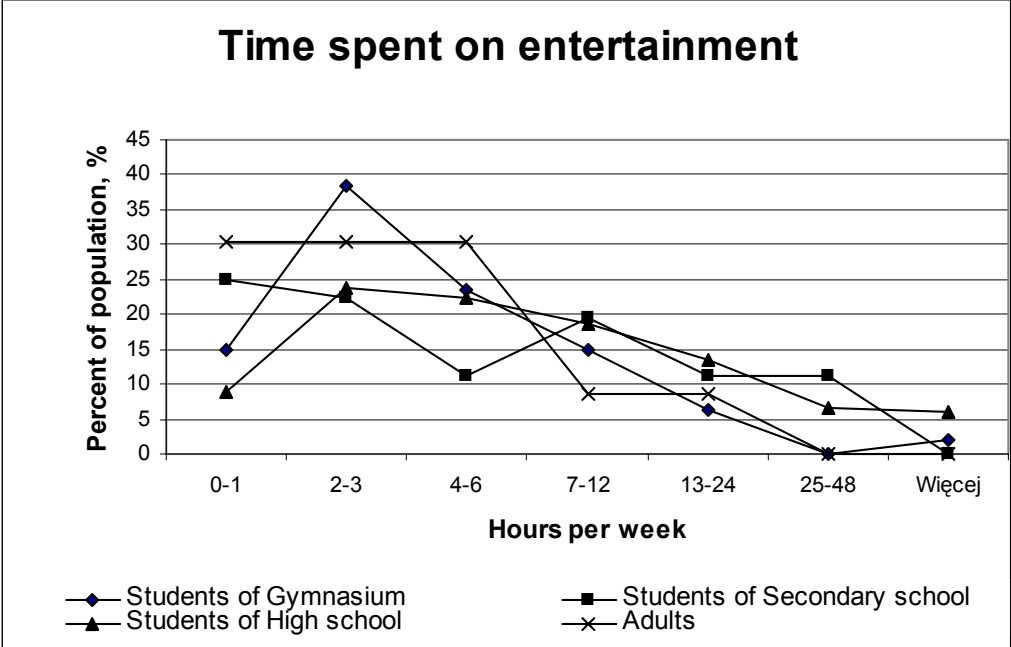
- knowledge of basic elements of one of modern high level programming language
- basic structural programming skills
- knowledge of English
- knowledge of mathematics and logics

Other merits such as creativity, fantasy, persistence, intuition, inventiveness, good memory and communication skills are very welcome.

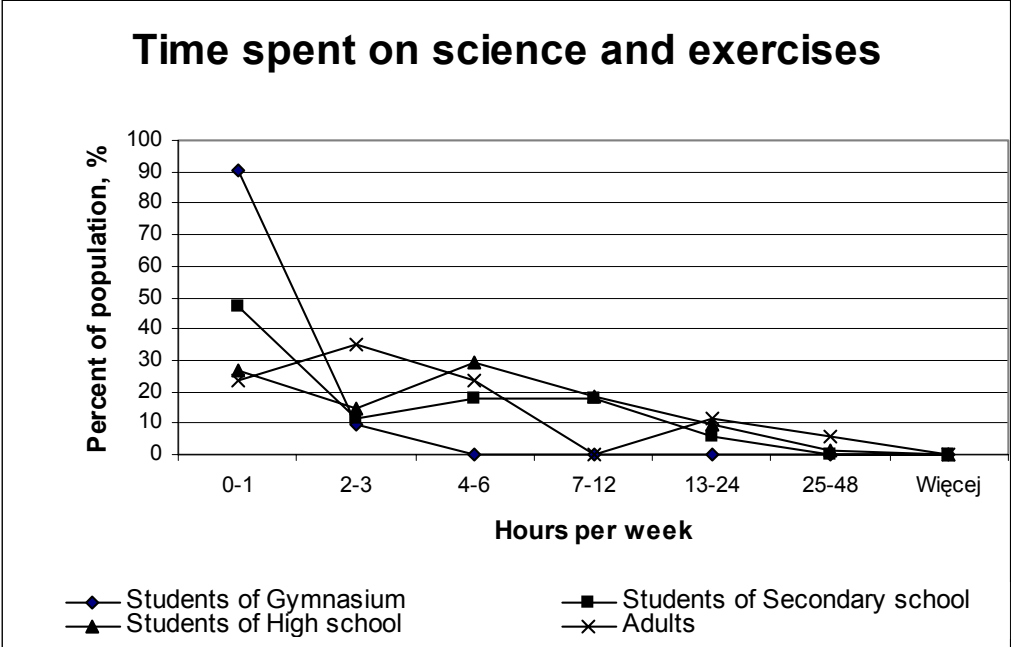
The role of computer programming in school didactics

Computers were converted from calculate machines to easy to use universal tools that can be used for work and entertainment without having any programming skills. It is shown, for example, with the results of a research, which explored the level of information culture of society understood in a broad sense. The information culture is not only a level of practical computer skills, but also a way of perception, a level of intellectual readiness. Figure 2

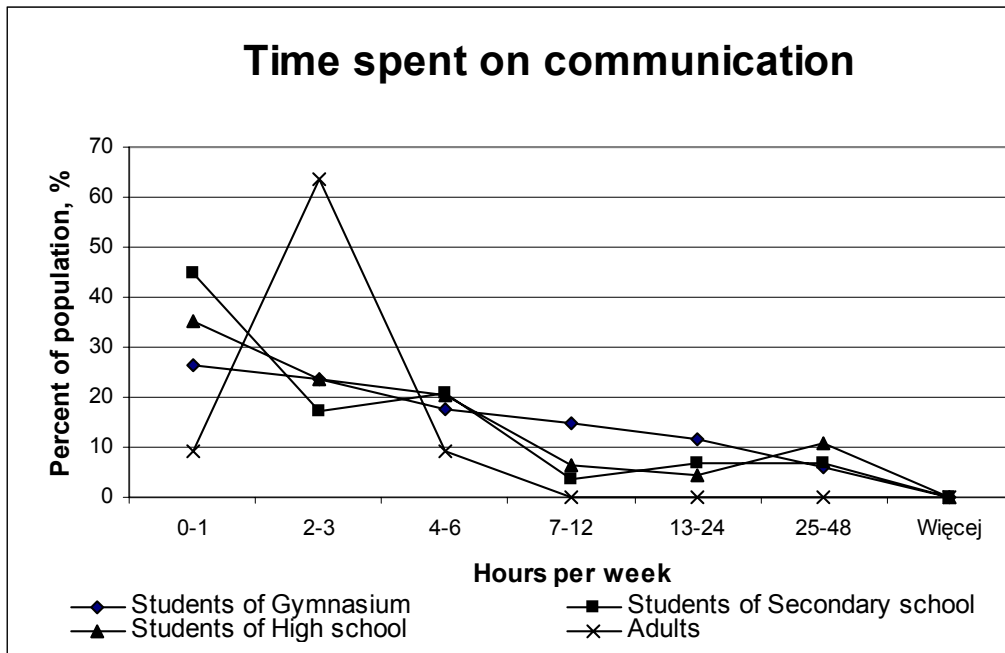
features some statistics, obtained by the authors from questionnaires of young people and adults in the Central Pomeranian region. It is easy to observe that computer has become a popular tool of work, learning and entertainment. The Internet is perceived as a good source of information. The results lay stress on some delicate differences in a level of information culture in different age groups, which are natural consequences of the process of getting older and reshaping thoughts and of the fact that IT technologies appeared not too many years ago.



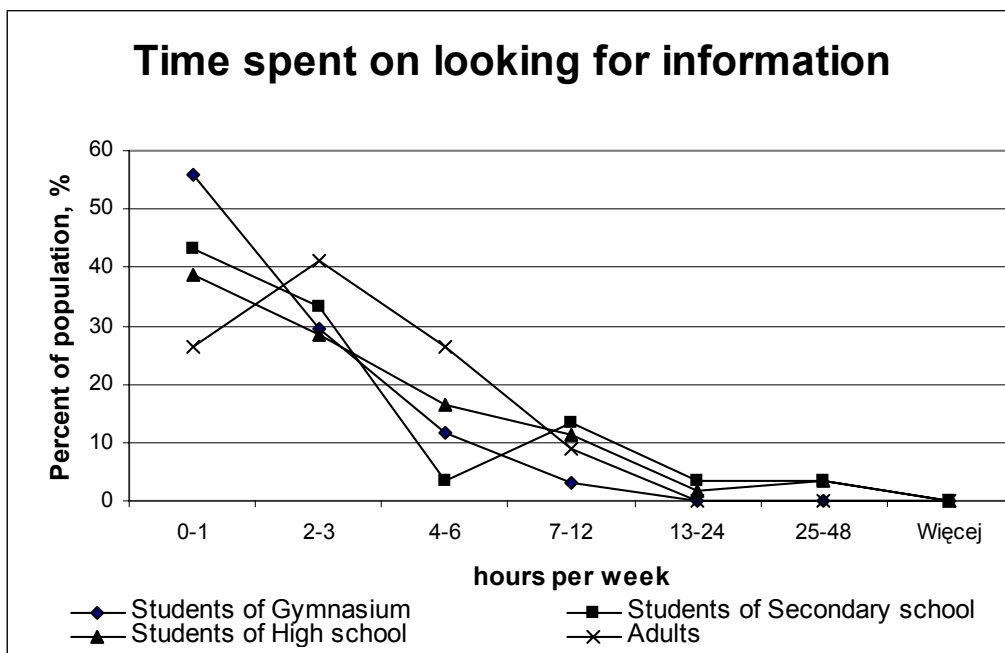
a)



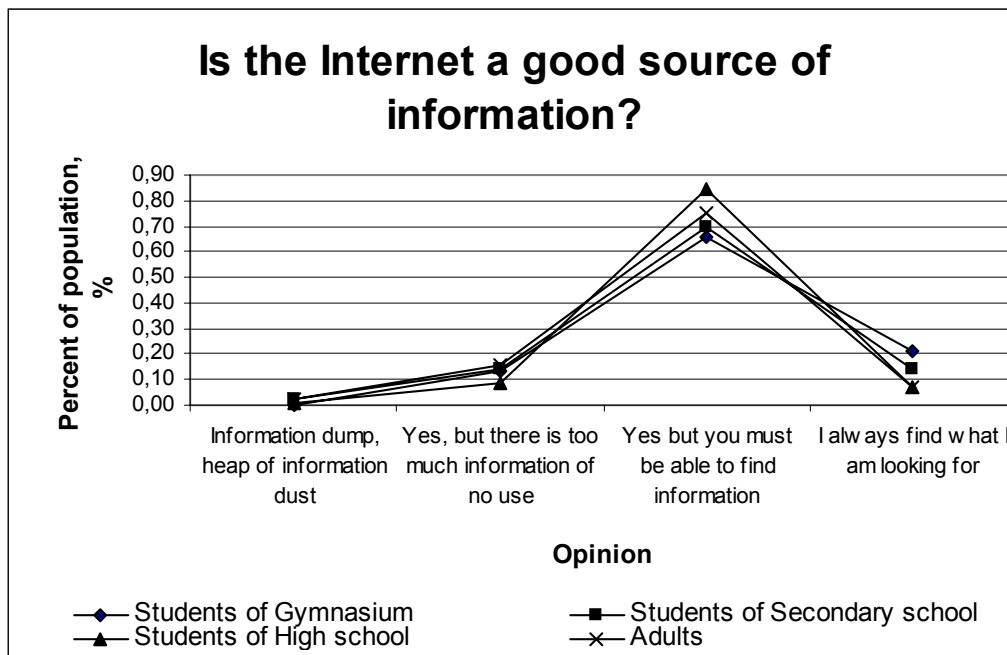
b)



c)



d)



e)

Figure 2. Some results of research of computer culture

Some solid guidelines for teaching IT technologies are collected in the special consolidated ACM and IEEE commission report published on December 15, 2001. This document called CC2001 should serve as a basis for study plans and syllabuses of educational institution around the world. At the same time every particular syllabus should take into consideration specific character of national educational system and modern trends in IT. Core skills which are outlined in CC2001 describe expectations for the level of knowledge of an IT specialist and can be accepted as main criteria that determine the essence of syllabus.

The thing is that, historically, the majority of computer science courses have concentrated first of all on the forming of students' skills in the field of computer programming (programming-first introduction). The main reason of that opinion is that CS as a didactical discipline was formed when the programming courses existed for wide groups of people. Programming is the basis for advanced courses for specialists and this is the reason why it was decided it should also be the main subject matter of study in CS.

The described approach to CS is not accepted by CC2001 and the document states that CS is not the same as programming. In the opinion of authors of CC2001, students do not have enough knowledge of conceptual basic knowledge of CS. Additionally, programming courses focus mainly on syntax and specific functions of a specific programming language and that is the reason of learning relatively unimportant details and deficiency of algorithmic models. The courses often concentrate on writing codes of programs during introduction to programming and that gives students an illusion of overrate skills. The idea of teaching CS with programming as a main course gives a notion that programming of computers is the only way to solve the problems, but at the same time these problems can be solved using a growing number of tools accessible at the market that are effective enough. The study of modern tools can be advantageous for students of other courses, different from CS.

However, in the opinion of authors, educational model based on programming helps to form algorithmic skills for solving problems in any practical sphere. This model can be used for learning basics of CS in the course of specialized education. Most didactical models of teaching programming skills in school are built on studying the structure of language by using examples of programs. The thinking of students develops from concrete operators, procedures

and functions. Unfortunately, this way determines thinking by ready-made patterns and learning of solving standard problems. It would cause many troubles with real problems in the future.

The methods of creation and consolidation of programming skills are known very well. But not every teacher accepts these innovations. Solving the open (non – standard) educational problems can be a good teaching idea for CS biased classes. That kind of didactics can be called “from top” because at the beginning total analyse of the problem not the program has to be done. The results of the analysis have to be formed as a group of written associations and roles (principles) useful for solving problems. Afterwards synthesis of specific elements can be realised as an application written in a specific programming language. Basically, it means that teaching to analyse problems is better than teaching to code ready-made algorithms.

Applications written during classes do not have any practical value now because they lose in a comparison to applications included in standard packages. “Programming for programming” does not make sense and, what is more, the consolidation of the necessary skills takes too much work. Teachers know that and they try to work out different ways of motivation for studying programming. One of proposals is realised by using visual and animated possibilities of graphical programming environments. Objects of that environment and easiness of operating them and their properties give students pleasure and the work goes smoothly (students have a sense of virtual belonging; it seems to them that they are in a virtual reality). A logical alternative is a rejection of exercises in programming and changing of methodical direction towards preparation of advanced users.

In the author’s opinion, classical programming approach, unfortunately, remains the only way nowadays to prepare software engineers who would be competitive on the market. The trend for teaching basic computer skills with orientation on use has no chance to survive. A student who writes codes of programmes for console in the text mode has better technical imagination and he will be a better programmer in the future. Basing on the analysis of published materials [...], three models of teaching of programming in the secondary school can be reached:

- programming as a theoretical discipline without teaching any specific programming system(environment),
- teaching programming with the help of a language built for didactical reasons and oriented on teaching basic programming skills,
- concentrating on one or a few specific and popular languages.

The first way has serious hardships because rejection of a programming language in CS teaching gives no possibility to use specific tools. Demonstrations and practical exercises turn out to be impossible. The second model is usually used in the beginning as an introduction to programming. Special programming languages have been made and they are extremely simplified and correlated to the level of knowledge of young students (LOGO in USA, SMR in GB, Rapira and E in Russia).

As for the third model, specific programming languages are not suitable for didactical goals and they do not fully reflect modern concepts of programming. It is natural because programming languages are concentrated on narrow practical fields. Therefore, all standard languages consist of too many technical details and that is why difficult to use in the process of teaching. This is the reason why it is not possible to work out one logically closed subject.

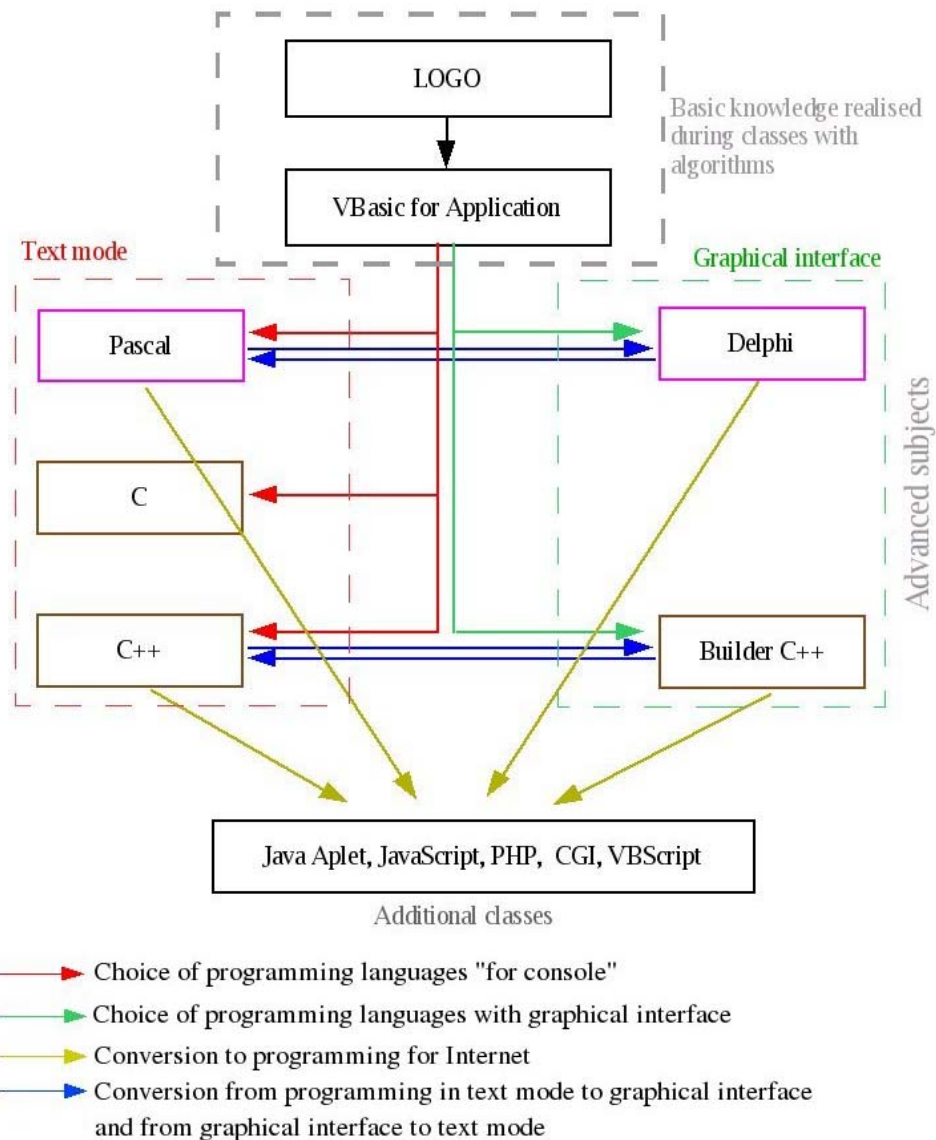


Figure 3. Choice of the programming language for teaching the basics of computer science (introduction in computer sciences).

Joining models 1 and 3 can be a good compromise. It is good to remember that languages such as Pascal or Basic were at first created as didactical languages and later they have become standard programming languages. Theoretically, the usage of Basic and Pascal in schools have some essential faults. All versions of Basic (except for VB.NET version which is problematic to use at school) are not object-oriented. Historically, they were created for DOS and are not supported by modern operating systems such as Windows 2000 and Windows XP. Pascal is a typical language, but, except for Object Pascal in Delphi package, it is not object-oriented either and all object's properties were built on basis of pure Pascal. Therefore, it is impossible to realize real object-oriented style of programming on Pascal. Only Java and C# can represent the concept of object-oriented programming and support object-oriented programming methods on the level of structure of the application. These languages represent competitive platforms Java 2 Enterprise Edition and .NET Framework.

It would be a good idea to refer here to a very emotional text of one of the authors of the article (K. Kadowski – teacher of CS in the secondary school), who commented on

classification of educational models in the following way: “I usually use the third model in correlation with the first one. I have achieved the best results by teaching basics of computer studies at the beginning of the course. At first students must get good knowledge of chosen operating system, its properties and structure. Later it is time to teach algorithms and how to solve different problems. I start with standard ones at the beginning: Euclid’s algorithm, factorial, sorting, stocks, calculations on matrixes, and so on. Then I form problems and real problematic situations, for example, in physics. Next, mathematical problems have to be solved. After that, we try to build an algorithm of a problem in natural language and in a flow of chart notation or schema, and finally we write an application using a chosen language. Students already have basic knowledge of object-oriented programming after course of MS OFFICE with VB for application in Excel. They have already finished a few classes of programming in C/C++: I/O functions, types of data and data structures, procedures and functions, working on files. In my opinion, the challenge plays a great role. If a student is solving a really problematic situation – not an imaginary problem, he identifies himself with it and it is highly probable that he will solve it. Students like that usually send me an e-mail with questions for help even at 11 p.m. Work goes quickly and we have concrete results in a short period of time. Skills’ formation is important, but it has to be formed by teachers. I pay attention to clear syntax, using tips, different types of data and variables, using many functions and classes if it is necessary. I try to make students flexible and ready to solve problems on academic level. That is the reason why my students have no problems with programming. They cannot be taken by surprise when faced with Java or Prolog. They can accommodate to a new situation and this is the most important thing. They have enough knowledge to use any language after short introduction to it. Additionally, such notions as object, properties, class, matrix, recurrence and polymorphism are not strange to them.”

The question that remains important nowadays is the following: what programming language should we choose in the beginning? Dejkstr in his great book¹ “The Discipline of Programming” wrote: “The least noticeable feature of every tool is its influence upon human habit-forming...When this tool is a programming language, it exerts its influence upon our way of thinking whether we want it or not”. The problem of choice of programming language is closely related to modern trend – we cannot teach only the technology from the past. Algorithmic way of creating applications must be replaced by object-oriented programming. It can be seen that all modern programming environments for Internet (future of IT) is based on using objects: JavaScript, PHP, ActionScript, Java Applet, etc.

We agree that modern teaching requires object-oriented approach. There is no detailed sequence of operations in this approach even though the languages have procedural programming mechanisms. A rich user interface of these languages gives programmer good conditions to work: Smalltalk, Delphi, Visual Basic, and C++ Builder. There is the same hardness in defining the border between high level programming systems and tools in this situation. That kind of application gives non specialists a possibility of programming advanced applications. Modern professional object languages such as C# or Java are too complicated for secondary school level. Visual Basic has great advantages in the beginning of teaching process because visual mode of programming lets mind work with a tool and at the same time it has a wide group of function and procedure suit with support of encapsulation and polymorphism that gives a part-realisation of object-oriented programming. Another advantageous feature of VB is his usability with MS Office package. Some educational projects can be based on that.

Can the Object Pascal be an alternative for beginners? During the deliberation on choice of language we can see that there are two directions of evaluation of programming languages:

¹ E.W. Dijkstra A Discipline of Programming. Prentice Hall, 1976

procedural and declarative. High level procedural programming languages can be divided into structural (one operator takes a whole algorithmic structures, for example Pascal or C) and operating (operations are separated as in Basic). The public description of operations that has to be done and the solution that is described by the way of achievement by procedure that brings itself to specified order of concrete operations is an advantage of procedural programming. The clear representation of operations, and thus order, during the programming process are needed. It means that procedural programming bases on algorithmic thinking and Pascal can be a great didactical tool for improving that way of thinking.

The declarative languages were developed for the needs of fifth-generation machines during the 80's. They are divided into two groups – functional languages (the programme describes counting of functions that often are a composition of more simple functions; recurrence is the basic element of those languages, for example Lisp) and logical languages (the programme does not describe operations but variables and configuration between them, for example Prolog). A functional programme just like a procedural programme describes a series of operations that have to be done. But the way of creating this kind of programme requires rather mathematical than algorithmic way of thinking. There is no need to describe operation in a traditional logical programming. There is no need for algorithmic thinking, there is no dynamics that is hidden in sorting of data mechanism. Finally, it is clear that this group is not suitable for secondary schools.

In conclusion, we would like to add that teaching of future professional programmers only in one language can create a narrow outlook limited by the categories of that language. The solution can be teaching in different environments and making students describe solutions of problems in natural language. This approach is confirmed by the common belief which states that those who do not speak well in their own native language cannot write texts in artificial language. A good way is to begin to work with object-oriented language because it is fun for children as it takes little time to get results, but at the same time the situation, in which a learner perceives needs to move to programming “on black screen”, should be created.

It seems that both types of languages should be used in schools with math-informatics bias classes. Work in console in C/C++ and work in Builder C++ or Kylix in Linux environments give clear comparison of programming techniques and set a goal to get different ways of solving the same problem. There are students who are better in pure C, but there are students who prefer programming in a graphical environment. These are the results of their innate predispositions. We value the work of both groups, but of course, professionals are particularly fond of the first group.

Method of projects as a didactical tool

Didactical tools that form didactical process obviously influence amount of information remembered by student and growing of thinking process. Thanks to the tool such as computer CS has superiority in realisation of visualisations of models, but not only that. Rather schematic notice of solution of the problem accepted in informatics is a great tool that can maintain a role of mind-map. Students and teachers use mind-map, decision tree methods, maps of states that in a visual way elucidate the problem and ways of solving it.

Good basic skills can not be worked out only during obligatory classes. Practically it can be seen that forming of skills of the high school graduates are most effective when the method of projects and participation in competitions are used. Permanent skills are developed by students during the work on solving open-problems.

Activating methods have a huge influence on student's progress and success. According to pyramid of knowledge adoption published by Z. Czaiński and Z. Wojtowicz, the best results in learning can be observed when different cognitive methods are used. Students acquire abilities and knowledge at most by participation in discussion or teaching other students.

These students learn how to form their own thoughts precisely, how to listen attentively, how to take valuable arguments supported by examples, how to analyse facts, how to use experiences of others, how to exchange views and work in a team. It is a good practice if a problem is related to student and he/she independently tries to solve it or cooperates with others in a team.

In the authors' opinion, project method in small groups (2-4 students) is a natural and one of the best ways of teaching CS. It should be noted, that nowadays very few computer programmers work alone, the majority of programming projects are carried out with contribution of specialists from many different fields. It can be put into practice in didactic classes with students. According to the project method, a student must solve a problem and to do it he has to present the problem to the team, discuss the results, and outline the activities which have led to the successful solution of the problem. The student faced with such a task utilizes following senses: sight (solution, algorithm), hearing (discussion) and acting. The obvious advantage of such projects is a natural cooperation of team members and the necessity of referring to knowledge of different kinds, apart from knowledge of computer programming. The topics for the computer studies syllabus can be derived from other subjects taught at the same time.

A programme which tests English grammar knowledge or backs up exercises in science gives a ground for integrated processes of training. Other sources for computer applications might be born out of the real needs of school everyday life: computerization of the library or creation of an electronic school newspaper. Projects of that kind can be perceived in terms of satisfaction which the results of programming give their creators; they do help the school, because somebody needs them.

From my own practice –the project method in groups very often leads to ‘brainstorming’, which is a good method of instant solving of difficulties. It turns out that students have different knowledge of a particular subject and they help one another heading for solution. An important aspect of solving computer studies problems is an insight which each and every participant brings in; alone we might not be able to detect sometimes obvious errors. Supporting additional classes of computer studies might be just another way of enhancing the knowledge of students taking matriculation. Some schools, mainly due to their teachers' passion, decide to run additional courses for students. Contemporary in the majority of cases the main area of activity is concentrated around creating web sides, which eventually leads students to enrich their knowledge in IT technologies and applications for Internet. There are also classes devoted to preparation to ECDL². There are very few classes that are dealing with programming and that prepare students to contests and develop their knowledge which would enable them to study CS at polytechnics and universities. An opinion issued by one of the lecturers of Akademia Górniczo-Hutnicza in Kraków may testify as an evidence of a very high level of participants of additional interest classes in computer studies organized in high school.

The opinion goes as follows: “... She received an ‘A’ passing grade and she was allowed not to take the exam thanks to that, she exposed a high degree of knowledge of the material and she displayed an overwhelming interest in computer related studies” and “I very highly value the standards she met in the course of high level schooling, never ever have I detected any shortages in her knowledge, very often I was nicely surprised by not typical dealing with exercises” and “The standards she achieved in my classes prove that 90% of success in higher education institutions (at least in the beginning) calls for good habit formations and excellent

² European Computer Driving Licence, <http://www.ecdl.com.pl/>

preparation gained in high school”³.

Conclusion

The technique of teaching the basics of computer science (introduction in a speciality) can create the base of the future occupation if the learning process at school is organized with use of modern creative approaches. Such profound preparation is necessary for the future software engineers. The mass teaching of bases CS should be directed on formation of information culture of the pupils.

Nowadays we wrestle with contradictions during the teaching of the principles of programming. On one hand, we should teach so that the long term-memory would allow restoring programming skills quickly even after a long break in the learning process. A big number of repeated exercises are indispensable for that. Unfortunately, a lot of time is needed for solidifying behaviour patterns and procedures and the stocks of that time are very limited in the school. On the other hand, a bigger emphasis can be placed on creative problem solving that promotes intellectual development, but it does not form the patterns of behaviour so needed at work.

The tactical aims, as for example could be teaching of basic universal algorithms or practice of using common tools, take pretty much time and at the same time they are only the introduction into the solving of real problems that allow the evolution of a future programmer. The strategical aim is still the forming of the skills of solving real problems using computer applications. This is the basic purpose of problem method that is the only way of finding a way out of the outlined contradiction. It is easy to see that „training of master” should be realized in short-term and dense units by problem method according to Mastery Learning idea. There must be a precise connection between different stages of training, so that a student could move to the next level after passing the previous level with at least a B (Good) grade. That kind of connection would function in a proper way only if we have reliable ways of knowledge and skill testing at different stages of learning.

Literature

1. Olo Sawa “Popularność wydziałów informatycznych wciąż rośnie”, Computerworld Polska, nr 26-1997.
2. Zbigniew Pendel “Najpopularniejsze kierunki studiów 2003”, Gazeta Wyborcza, 04-12-2003.
3. Najpopularniejsze kierunki w roku 2003/2004 według Ministerstwa Edukacji Narodowej i Sportu, <http://www.menis.gov.pl/szk-wyz/kierunki/kierunki.htm>
4. Computing Curricula 2001: Computer Science. Final Report (December 15, 2001). The Joint Task Force on Computing Curricula, IEEE Computer Society, Association for Computing Machinery, 240 s. (<http://www.computer.org/education/cc2001/>)

³ Opinion on the student of teachers of AGH University of Science and Technology, Krakow. (M. Woloszyn, L. Adamus and W. Karas)