Baltic J. Modern Computing, Vol. 5 (2017), No. 2, 164-182
http://dx.doi.org/10.22364/bjmc.2017.5.2.02

Computer Science High School Curriculum in Israel and Lithuania – Comparison and Teachers' Views

Tamar BENAYA[1], Ela ZUR[1],
Valentina DAGIENĖ[2], Gabrielė STUPURIENĖ[2]

[1]The Open University of Israel, Faculty of Mathematics and Computer Science
[2]Vilnius University, Institute of Mathematics and Informatics

tamar@openu.ac.il, ela@openu.ac.il
valentina.dagiene@mii.vu.lt, gabriele.stupuriene@mii.vu.lt

Abstract Computer Science (CS) education in high schools is entering the fifth decade of its existence. Israel is one of the first countries which started to offer CS courses in high schools in the middle of the 1970s. Many European countries joined this process a decade later – Lithuania is among them. Both countries put a lot of effort in developing CS curricula and establishing assessment examinations. Nowadays there has been considerable activities surrounding CS education on all levels therefore we suggest to take a look at the experience of these two countries. In this paper we describe the CS curriculum for high schools in both countries, including a description of the final exams, some statistics regarding students' participation and achievements as well as exam evaluation. We then display the results of a survey, conducted among high-school teachers in both countries regarding teachers' attitudes towards the CS curriculum. We sum up with a discussion comparing curricula and teachers' attitudes in both countries and conclude with some insights from the survey.

Keywords Computer Science Education, K-12, Computer Science Curricula.

1. Introduction

At present, all evidence points to a significant boom in Computer Science (CS) education at the high school level. This boom is most clearly manifested in the shifting education attention from information technology (IT) to CS (Computing or Informatics as it is called in many European countries) and in increasing articles devoted to the questions of CS education in schools.

Education policy makers are becoming inspired by the challenges posed by the CS Teacher Association in USA (Seehorn et al., 2011), Computing at School in the UK (Computing…, 2012) and the Computing Curricula (Joint IEEE…, 2013). The new Computing curriculum in UK puts the subject on an entirely new footing, as the "fourth science" at school. It offers new opportunities for professional development for teachers and better education for students (Computing…, 2012).

So, in the last four decades, there has been considerable activity surrounding CS curricula on all levels: beginning with ACM Curriculum Committee on CS (Atchison et al., 1968) through Computing Curricula 1991 (Tucker et al., 1991) and up to Computing Curricula 2013 (Joint IEEE…, 2013). Notable is the high-school curriculum designed by the special ACM task force (Merritt et al., 1994) and in particular the K-12 curriculum (Tucker, 2003). The goal of the K-12 curriculum, was to create a 4-level curriculum that could be widely disseminated, accessible to every high school student in the US. Its aim was to enable every CS student to understand the nature of the field and the place of CS in the modern world. Students need to understand that CS combines theoretical principles and application skills. They need to be capable of algorithmic thinking and of solving problems in other subject areas and in other areas of their lives. The aim of the most recent curriculum K–12 Computer Science Framework (2016) is dedicated to students who are not just computer users but also computationally literate creators who are proficient in the concepts and practices of computer science.

In light of the recommendations presented above, we can see that different countries developed unique curriculums for high school CS education, see for example ACM Transactions on Computing Education (TOCE) special issue on Computing Education in (K-12) Schools (Tenenberg and McCartney, 2014). Although much had been previously written about CS education at schools, we would like to share Israel's and Lithuania's long experience in teaching CS at high school.

2. The High School CS Curriculum

2.1. The Israeli Case

Teaching CS in Israeli schools was offered since mid-1970s. The current Israeli CS high school curriculum was designed in the early 1990s and first implemented in 1995. One particular principle underlying the curriculum is the interleaving of theoretical principles with application skills. This interleaving notion is specifically termed in the Israeli curriculum as the "zipper approach". A detailed description of the program is given in (Gal-Ezer et al., 1995; Gal-Ezer and Harel, 1999).

In Israeli high schools, every student must study at least one subject in depth, in addition to general studies which include Mathematics, English, History, Literature etc. The highest level of studies is the 5-point (as opposed to 3 or 4-point) program, each point representing 90 class hours. CS is one of the subjects that high school students can select to study in depth. The CS program starts in 10^{th} or 11^{th} grade depending on the high school. The units included in the 5-point program are:

- Foundation of Computer Science 1 and 2;
- Second Paradigm or Application – There are several alternatives for the second paradigm, such as: logic programming, functional programming or low-level programming; and several alternatives for the application area such as: Internet programming, computer graphics or information systems;
- Data Structures;
- Theoretical unit – such as: object oriented programming, computational models and operation research.

2.2. The Lithuanian Case

In Lithuania, the directive to teach informatics came from the Ministry of Education in Moscow in 1985, so after year informatics lessons started in all Lithuania schools. Work of Lithuanian researchers in the field of the methodology of programming was well known in the Soviet Union. Plenty of textbooks on teaching algorithms and programming based on attractive tasks were developed. Methodology on teaching CS at secondary school level was strong and well designed by Lithuanian's researchers.

As a part of the Education Reform in 1997, the Informatics core curriculum went through a major revision and it was expanded from teaching two years to four years (in total 136 hours) with more focus on application and the processing of information (mainly, text processing). In regard with the changed role of the information and communication technologies as well as with the needs of students and school communities, the curricula of all subjects were substantially revised and renewed in 2005: subject title "Informatics" was changed to "Information Technologies (IT)".

The modules included in the CS curriculum are:

- Short introduction to Programming based on Logo or Scratch – in grades 5 or 6.
- Elements of Algorithms and Programming – for grades 9 or 10. The course is aimed at summarizing and systematizing students' knowledge on algorithms and drawing attention to their application and programming.
- Developing Algorithms and Learning Programming – for grades 11 – 12. The topics include: algorithms, text files, procedure and functions, arrays, strings, records and programming technique.

The teaching process in Lithuania depends very closely on the knowledge and activeness of the teachers themselves. However, the optional modules on programming and related topics are available in high school and in some schools also in lower grades. Especially learning coding became more and more popular among pupils with focus on web design and programming of mobile devices.

2.3. Comparison

Main principles and components of the CS curricula in both countries are similar: more than 2/3 of CS curricula in both countries are devoted to algorithms and programming. In particular, programming can now be interpreted as a component of an emerging new form of literacy (Vee, 2013); as a tool to conceive and create things, to develop creativity (Resnick, 2009); as a way for children to widen their experience and experiment with their own ideas (following, in a sense, Papert's Mindstorms' perspective (Papert, 1980)).

Teaching programming has been designed very carefully in both countries: from developing teaching resources, exercises, handbooks, computer tools to teacher training and deep connection between educators and researchers. As we have noticed teaching programming has been focused on CS concepts and building understanding. Pupils are asked to recognize and use variables, data types and data structures, understand and apply control statements: assignment, condition, repetition and procedures. Rather than increasing various CS concepts both countries chose the way "Less is More": learning less concepts but making more activities and practice.

Beside CS curricula at schools both countries have a long tradition of suggesting to pupils many different outreach activities in the CS field, especially programming clubs and contests such as Olympiads[1] and the Challenge on Informatics and Computational Thinking "Bebras"[2].

2.4. Focus on CS concepts

CS concepts play a central role for understanding fundamentals of computers, information technology, software, hardware and other devices. However, in practice, very often the training of skills in application software is given much more room at schools than to discover and going deeper into concepts of CS (Dagiene and Stupuriene, 2016a). CS education should be taken seriously and combine various forces. To obtain deep understanding of CS concepts, formal lessons are not enough attractive for keeping students' motivation. Students should be encouraged to play with these CS concepts in their everyday life.

Educators agree, that learning fundamental concepts and principles at an early age is very important for a deeper understanding of various computer science topics (Dagiene et al., 2017). There exist some activities based on attractive learning: CS Unplugged[3] (is a collection of free learning activities that teach CS through engaging games and puzzles that use cards, string, crayons and lots of running around); code.org[4] (a blocky coding platform, but have a course that uses a blended learning approach to teaching CS, which means that students learn from a mix of online, self-guided activities and unplugged activities with teacher-led activities that use no computer at all.)

One of the way to introduce CS concepts are short concept-based tasks for different age students. International challenge on Informatics and Computational Thinking Bebras focus on concepts-based tasks. These tasks can include a wide range of concepts within CS including algorithms and programs, both sequential and concurrent; data structures like heaps, stacks and queues; modelling of states, control flow and data flow; human-computer interaction; graphics; etc. The learning and understanding process of CS concepts will come later, actually after practice to solve many of these tasks (Dagiene and Stupuriene, 2016b). Learning through solving small pieces of concept-based tasks, or a flipped learning, fits better to the digital era society. A teacher's role is very important for strengthening the understanding of CS concepts. Teachers can help pupils to clarify task solutions, to explain why it is informatics/CS, to provide resources for readings and discussions.

The challenge has been successfully conducted in the Davidson Institute of Science Education at the Weizmann Institute of Science in Israel. It arose from a partnership among academia, the Ministry of Education, and K–12 educators (Haberman et. al., 2011).

In the next section we describe the matriculation and final exams both in Israel and Lithuania.

[1] http://www.ioinformatics.org
[2] http://www.bebras.org/
[3] http://csunplugged.org/
[4] https://code.org/

3. Matriculation/Final Exams

3.1. The Israeli Case

All high school students are required to take matriculation exams in the main subjects studied in high school. The Israeli matriculation exams are similar to the American AP exams in that they are external nationwide exams. Internal high school exams are used to prepare students for the national matriculation exams. The final grade in the subject tested is calculated as the average of the matriculation exam and an internal grade which is based on the internal exam and the student performance throughout the year. It is important that teachers will be familiar with the matriculation exams in order to prepare their students in the best way possible (Drysdale et al., 2005). The questions in the matriculation exams, like the AP exams, should test the intended concepts accurately, unambiguously, and without bias (Hunt et al., 2002).

The content of the matriculation exam of FCS1 and FCS2 reflects the foundations of algorithmic thinking and programming. The duration of the exam is three hours. The exam is divided into three sections according to the Bloom taxonomy.

- The first section contains 5 mandatory ten point questions which test basic skills such as knowledge and comprehension.
- The second section includes 3 fifteen point questions which the students are required to answer two of them. The questions in this section are application questions which require the students to solve problems to new situations by applying acquired knowledge. The questions in this section may require writing a small program, or writing a sub-program and demonstrating its use or tracing a given program. This section requires the use of sequential and/or nested patterns.
- The third section includes 2 twenty point questions from which the students are required to answer one. The questions in this section require analytic and synthesis skills. This section requires writing a complete program which includes: defining appropriate sub-tasks, defining main variables and data structures and implementing the code including documentation.

For the third unit Second Paradigm or Application the students are required to prepare a project according to the units' topic and requirements. The students present their project to external examiners who are usually CS high school teachers from other schools. The students must defend and run their projects and answer questions posed by the examiner.

The fourth and fifth unit (Data Structures and Theory) have a combined 3-hour exam. The first part deals with Data Structures and the second part deals with the Theory unit. In each part the students are presented with four questions from which they must select two.

3.2. The Lithuanian Case

Exams in school cause contradictory feelings. No doubt that having a maturity exam increases the value of the subject. School students and teachers often give more respect to exams than to the process of learning. Besides, it is better for pupils to have more choice for choosing exams. In 1995, the informatics maturity exam was developed. Informatics as a separate subject was taught for many years in Lithuanian schools thus to establish the maturity exam in informatics was a natural process. Discussion on informatics exams has been presented in (Blonskis and Dagiene, 2008).

The main goal of the Informatics exam is to encourage students to take interest in programming. The demand for programmers is considerable. Programming as a creative process is being comprehended by learning to write programs from one's as early as possible youth upwards. Algorithmic and structural thinking skills greatly influence the conception of the exact sciences.

The results of the Informatics exam are being recognized when choosing studies of informatics or informatics-related specialties at Lithuanian universities. Those, who pass the Informatics exam successfully, have wider possibilities to enter CS-related studies in higher education. At the same time, it checks whether student have the aptitude for studying informatics: there are many first year students who quit their studies since they find programming too hard to understand and an uninviting occupation for themselves.

Lithuania's maturity programming exam is an interesting use case of semi-automatic evaluation. Research on exam data demonstrated that this approach is rather effective and still provides good quality evaluation. However, this type of evaluation is still not very popular among CS teachers and the outcome of this use case can be rather interesting for the community.

The Informatics exam consists of two parts: the larger part (75%) is allocated to programming, while the rest (25%) concerns the issues of computer literacy. The programming part consists of test (25%) and two practical tasks (50%). The aim of the programming test is to examine the level of students' knowledge and understanding of the tools required in programming (elements of programming language, data types and structures, control structures, basic algorithms).

The Informatics exam focuses on: knowledge and understanding – 30%, skills – 30% and problem solving – 40%. The problems are oriented towards the selection of data structures and application of basic algorithms to work with the created data structures.

In Lithuanian schools, each subject's exam has its own curriculum, which is more concrete than the general subject's curriculum. The curriculum of Informatics exam closely corresponds to the content of the programming module. Three main fields are emphasized: algorithms, data types and structures, and constructs of a programming language (Table 1).

Table 1. Components of curriculum of Informatics exam

Algorithms	Calculation of sums, product, quantity, and average; Search of max/min value; Data I/O; Sorting; Modify algorithms according to particular data structures.	Programming environment. Technology of structural (procedural) programming. Testing. Program documentations. Arrangement of dialogs. Program writing (style)
Data Structures	Integer, real, char, Boolean, and string; Text file; One-dimensional array; Record; Creating simple data structures.	
Programming Language (Pascal)	Program structure; Documentation; Variables; Assignment; Relational & Logical operations; If statement; Loops; Compound statement; Procedure & function; Parameters & arguments; Standard math procedures & functions; Procedures & functions related to files.	

The main attention is being paid to the abilities to choose the proper data types and data structures, also to the implementation of the algorithms and developing the programs.

An exam is not the best way of teaching students; – it seems to be late. We have noticed something different. The students who intend to take the programming exam choose the programming module a year before and try to follow the exam model while studying. In other words, if a lot of attention is paid to writing programs, if there are many tasks of algorithms and data structure selection in the exam, the students pay much attention to the mentioned points while learning. Therefore, the exam performs an educational function.

The following section presents some statistics regarding students' participation and achievements and exam evaluation.

4. Statistics and evaluation

4.1. The Israeli Case

The data in this section was taken from the Publications of the Israeli Central Bureau of Statistics and from the Science and Technology Department in the Ministry of Education (Publications…, 2013; Sciences…, 2014).

The following statistics refer to the percentage and grades of students studying CS in 2012:

- The percentage of students studying the 5-point CS program in high school is 10.4% and the percentage of students studying only the 3-point CS program in high school is 5%.
- 97% of the students who took the CS matriculation exams passed the exams. The percentage of females who passed these exams is slightly higher than the percentage of males who passed the exams.
- The average final grade of FCS1 and FCS2 exam was 88, while the average final grade of the Data Structures and Theory exam was 80.
- The Theory exam was distributed as follows:
  - 67% selected Computational Models and achieved an average grade of 81.1
  - 12% selected Object Oriented Programming with C# and achieved an average grade of 81.1
  - 8.4% selected Object Oriented Programming with Java and achieved an average grade of 80.6
  - 10.4% selected Computer Systems and achieved an average grade of 76
  - 2% selected operation research and achieved an average grade of 82.5

4.2. The Lithuanian Case

The practical part of the Informatics consists of two tasks – students have to write programs for the given problems. The practical tasks constitute 50% of all points. The main aim is to examine the students' ability to master independently the stages of programming activities, i.e. to move from the formulation of the task to the final result.

Obviously, the contest system from Olympiads can be useful, but it cannot be used without significant changes. A new automated evaluation system with all the

requirements met was developed. Application of the evaluation operates with packages of solutions. Each solution must be compiled, and then run with several data sets. The answers provided for all these data sets must be compared with the correct one.

The exam may be approached in two ways: on the one hand, it is the evaluation of the results achieved by a student; on the other hand, it could heighten the motivation to learn. Both must be considered when planning the exam. The exam should be prepared so that it measures the competences needed for further studies in CS. The exam is based on the optional module of the basics of programming which consists of four parts: 1) introduction – basic elements of programming; 2) data structures; 3) developing algorithms; 4) testing and debugging programs.

Students should demonstrate understanding of existing code (Lister et al., 2004). According to many years of experience, the exam has settled structure: 30% is allocated to knowledge and understanding skills, and the rest – to problem solving. The problems are oriented towards the selection of data structures and application of basic algorithms to work with the developed data structures.

In the practical part students have to write programs for the given two tasks. The main aim is to examine the students' ability to master the stages of programming activities independently. The first task is intended to examine the students' abilities to write programs of the difficulty described in educational standards. The abilities of students to use the procedures or functions as well as basic data types, to realize the algorithms for work with data structures as well as the abilities to manage with input and output in text files are examined. The second task is intended to examine the students' understanding and abilities to implement data structures. The core of the task is to develop the appropriate structures of records together with arrays. The abilities to input data from a text file to array of records, to perform operations by implementing the analyzed algorithms, and to present the results in a text file are being examined.

Evaluation of the programs submitted to the exam is a very important issue. The National Examination Centre has made a decision to create an automatic evaluation system with all the requirements met. The system consists of several modules responsible for the evaluation of different aspects such as evaluation of the programming style. The development still continues, as the main rules of the exam change step-by-step and new ideas arise for better evaluation (Table 2). One of the latest ideas is to integrate open question answer testing in the same system, by adding C++ as a possibility for the programming part.

Table 2. Evaluation of the program development

Parts of Program Evaluation	% of Points
Testing. Automatic evaluation.	80
Data structures, data reading, actions of calculation, printing of results. ***Evaluated only if*** results of at least one test are incorrect.	80
Obligatory requirements to the program (procedures & functions for single actions are indicated), programming technology, and style.	20

Application of the evaluation operates with packages of solutions. Each solution must be processed as follows: it must be compiled, and then it must be run with several

data sets. The answers provided for all these data sets must be compared with the correct ones.

The evaluator team tries to evaluate the solutions positively. This means that students get points for their effort. For example, correct input/output routines can be assessed by several points. Also, some points can be gained for dividing the program to subroutines, for using complex data structures like the array or record, for writing good comments, for good programming style, etc. These criteria can be easily evaluated by a person, while computer evaluation is not so obvious. This is the reason for manual evaluation of solutions.

The first tasks are easier therefore a larger number of students attempt to solve them. The second tasks are intended to examine the students' understanding and abilities of implementation of record data type. The core of the task is to develop the appropriate structures of records together with arrays. The abilities to input data from text files to arrays containing elements of the record type, to perform operations by implementing the analyzed algorithms and to present the results in a text file are examined. The operations are to be performed only with numerical values. The curriculum does not require operations with character strings, only reading and derivation of such strings are applied.

In order to get maturity certificate students should pass a compulsory mother tongue exam and at least two optional exams. Informatics exam has quite good number of participants, over two thousand (in comparison, Chemistry and Physics exams have around three thousand students each). The number of failed students varies from 2% to 17% (Table 3).

Table 3. Informatics exam in 2011-2016

Year	2011	2012	2013	2014	2015	2016
Attended students	1871	1830	2328	2268	2502	2207
Pass	97.69%	92.73%	82.43%	92.21%	93.17%	93.52%

5. Teachers' Attitudes towards the CS Curriculum

The teachers' attitudes towards the CS curriculum are vital for the success of CS education. It is important that the teachers identify with the curriculum in order for their teaching to be effective.

Due to the vital role of the teachers, we were interested in examining their views. Thus, we conducted a preliminary study among the CS teachers in both countries, in an attempt to learn about their backgrounds, attitudes and opinions towards the CS curriculum. We posed a 12-question questionnaire by e-mail to CS teachers in both countries.

For the Israeli case, the questionnaire was sent to 137 CS teachers, and was answered by 25 of them (18%). This response rate is similar to the response rate in other research in this field in Israel. We want to point out that CS in high school in Israel is an elective subject selected by only 15.4% of the students therefore CS is not taught in all high schools and the total numbers of CS teachers is not very high. We also want to point out that the questionnaire was sent primarily to senior teachers and heads of CS programs in their schools.

For the Lithuanian case, the questionnaire was sent to about 650 IT/CS teachers (usually IT teachers teach CS as well in lower secondary schools), and was answered by 337 of them (about 52%). Results from the questionnaire in depth are presented in paper by Dagiene and Stupuriene (2016a). Below we present the questions from the questionnaire along with a summary of the teachers' answers.

Which CS unit are you teaching?

The Israeli case

Table 4 shows the percentage of respondents teaching each of the units in the CS curriculum. The most popular options for the Second Paradigm or Application unit among the respondents were Internet programming and low-level programming. The most popular option for the Theory unit among the respondents was computational models followed by object oriented programming.

Table 4. Israel - percentage of respondents teaching each of the CS units

CS Unit	% of Respondents
Foundation of Computer Science 1 and 2	88%
Second Paradigm or Application	64%
Data Structures	76%
Theory	68%

The Lithuanian case

Table 5 shows the percentage of respondents teaching each of the units in the CS curriculum. Only 27% of the respondents teach informatics in high school. The rest teach IT in lower grades.

Table 5. Lithuania - percentage of respondents teaching each of the CS units

CS unit	% of Respondents
Basic Knowledge on Informatics	3.6%
Basic Knowledge on Information Technology	69.4%
Introducing Algorithms and Programming	5.6%
Developing Algorithms and Learning Programming	21.4%

For how many years have you been teaching Informatics?

The Israeli case

- 25% of the respondents have been teaching CS for 3 to 6 years.
- 75% of the respondents have been teaching CS for more than 6 years.

The Lithuanian case

- 3% of the respondents have been teaching Informatics for less than 3 years.
- 7% of the respondents have been teaching Informatics for 3 to 6 years.
- 90% of the respondents have been teaching Informatics for more than 6 years.

Are you in charge of Informatics or Information Technology in your school?

The Israeli case
52% of the teachers were heads of the CS program in their schools.

The Lithuanian case
86% of the teachers were heads of the Informatics program in their schools.

What is your Informatics/CS education?

The Israeli case

- All of the respondents, but one, have at least an undergraduate degree. Half of them have an undergraduate degree in CS, 28% of them have an undergraduate degree in CS Education and 17% of them have an undergraduate degree in unrelated field.
- 60% of the respondents have a graduate degree and mainly in education. 21% of them have a graduate degree in CS.
- 80% of the respondents have a teaching certificate. 80% of them have a CS teaching certificate.

The Lithuanian case
All of the respondents have at least an undergraduate degree. 52% of them have an undergraduate degree in CS or Informatics and 12% have an undergraduate degree in unrelated fields. 36% of the respondents have a graduate degree.

Rank from 1 to 4 your satisfaction with the Informatics curriculum (1 – very satisfied, 4 – not satisfied).

The Israeli case
The vast majority of the respondents (79%) claimed that they were very satisfied or satisfied with the informatics curriculum and only one respondent claimed that she was no satisfied with the informatics curriculum. Most of the respondents felt that the curriculum puts too much emphasis on algorithmic thinking. Half of the respondents claimed that the curriculum is missing new technology topics such as android programming, gamming, etc.

The Lithuanian case
Only 22% of the respondents claimed that they were satisfied or very satisfied with the informatics curriculum, while 66% of the respondents claimed that they were not very satisfied with the informatics curriculum and 11% claimed that they were not satisfied at all.
Some of the respondents' comments were:

- IT must be compulsory in primary school from 3rd grade (now compulsory from 5th grade) emphasizing security of internet, passwords, etc. including robotics and programming with Scratch. Creation of presentation should be introduced in 5th grade because a lot of teachers in other subjects required it in early years (now it is taught in 7th grade).
- More programming in 7th and 8th grade, and more hours for teaching, because in 7th and 8th grade there are half lessons per week, and the rest of the time is designated for integration with other subjects (but in reality it does not work).
- Many topics and theory are taught in 9th and 10th grade, but a lot of students don't remember anything, because in 8th grade they have a break from informatics.

- The curriculum needs more programming and algorithmic thinking topics.
- Informatics must be compulsory throughout secondary school.
- Maturity exam is very difficult for students who do not have programing in early years because a part of the exam is designed for programming.
- Our tutorials are ten years old and there are topics, such as floppy disk which are not relevant.

Do you think the Informatics Curriculum needs to be updated? If so, Suggest in what directions.

The Israeli case

Also most of the respondents claimed that they were satisfied with the Informatics curriculum. Some of them made the following comments:

- There is too much emphasis on technique and not enough on algorithm.
- Object first is too abstract for beginners and she suggests introducing objects after operations.
- The curriculum should introduce advanced topics such as machine learning, graphics, computer networking and data mining.
- The curriculum is missing operating systems, computer systems and file manipulation.
- One of the respondents said that she would like to have more professional textbooks and another said that the curriculum should be updated every few years and not continuously.

Suggestions related with CS units:

- First and second unit (Foundation of Computer Science 1 and 2) – 77% of the respondents claimed that the students find the unit interesting or very interesting and the rest claimed that the students find it partially interesting.
- Third unit (Second Paradigm or Application) – 65% of the respondents claimed that the students find the unit interesting or very interesting. Most of the respondents claimed that Internet programming, information systems and low-level programming are the most important options for the students' CS education while logic programming, functional programming and computer graphics are less important.
- Fourth unit (Data Structures) – 74% of the respondents claimed that the students find the unit interesting or very interesting. One respondent recommended adding inheritance and polymorphism to this unit.
- Fifth unit (Theory) – 82% of the respondents claimed that the students find the unit interesting or very interesting. All of the respondents claimed that object oriented programming is the most important option for the students' CS education. 64% claimed that computational models are also important or very important for the students' CS education while operation research, computer systems and assembly and parallel and distributed programming are less important.

The Lithuanian case

Some of the respondents' comments were as follows:

- Nowadays there are a lot of modern technologies and it is important to present them to students.
- More and more jobs require computer literacy and therefore computers, information and internet should be introduced to students.

- We need to introduce topics such as: 3D printers, game creating apps, databases and programs working with sound.

Do you feel that you have enough professional support from other Informatics teachers or workshops etc.?

The Israeli case

Most of the respondents (76%) feel that they have enough professional support. Many of them pointed out that they gain support from workshops, discussion groups and blogs organized by the National Teachers Center[5] and the Science and Technology Office in the Ministry of Education[6] and also from other CS teachers.

The Lithuanian case

- A bit less than half of the respondents (45%) feel that they have enough professional support. They pointed out that Informatics teachers have a group on Facebook on which they share their experiences.
- 55% of the respondents feel that they do not have enough professional support. They claimed that many courses for teachers are not free and are given only in the capital. They also claimed that they need more courses for advanced teachers, more practical courses as opposed to theoretical courses.

Which programming language in your opinion is most appropriate for teaching Foundation of CS 1 and 2 (Israel) / Introducing Algorithms and Programming module (Lithuania) and which programming language for getting familiar with basic knowledge on Informatics?

The Israeli case

Table 6 shows the percentage of respondents supporting different programming languages as most suitable for teaching Foundations of Computer Science 1 and 2. Some of the respondents' comments were: That the language should be chosen according to the industry requirements; Python's environment is easy to learn; Java is open source.

Table 6. Percentage of respondents supporting each programming language for teaching Foundations of CS1 & 2

Programming Language	% of Respondents
Java	60%
C#	48%
Python	12%
C	8%
C++	4%
Visual Basic	0%

Table 7 shows the percentage of respondents supporting different programming languages as most appropriate for getting familiar with basic knowledge on Informatics.

[5] http://cse.proj.ac.il
[6] http://edu.gov.il

The respondents who supported Scratch claimed: that it is friendly language with easy syntax and fast results and does not require abstract thinking; that it enables to combine games with algorithmic thinking; that it is not threatening; that it interests both girls and boys; that it gives a good foundation without too much theory; Scratch has a Hebrew version and therefore it is more appealing to the young children.

Table 7. Percentage of respondents supporting each programming language for teaching basic knowledge on Informatics

Programming Language	% of Respondents
Scratch	60%
C#	20%
Java script	16%
Java	8%
Visual Basic	8%
Logo	4%
Other	12%

The Lithuanian case

Table 8 shows the percentage of respondents supporting different programming languages as most suitable for teaching "Introducing Algorithms and Programming". Some of the respondents' comments were:

- If you know C++, you can pass an exam better, and this programming language is necessary for university.
- For 5^{th} to 8^{th} grade could be Scratch, for 9^{th} to 10^{th} grade – Python, for 11^{th} to 12^{th} grade – Python, Java and C++

Table 8. Percentage of respondents supporting each programming language for teaching algorithms and programming

Programming Language	% of Respondents
C++	58.8%
Scratch	13.4%
Python	4.5%
C#	3%
Java	2.1%
Visual Basic	1.8%
C	0.9%
Other	15.7%

Table 9 shows the percentage of respondents supporting different programming languages as most appropriate for getting familiar with basic knowledge on Informatics. The respondents who supported Scratch claimed that there is a lot of useful information and lessons on the Internet such as code.org tool. The respondents who supported Logo claimed that a lot of senior teachers know LOGO, so sometimes it is not so easy to

change the programming language because teachers have their lesson plans and they do not want to change them.

Table 9. Percentage of respondents supporting each programming language. for teaching basic knowledge

Programming Language	% of Respondents
Logo	32.6%
C++	28.2%
Scratch	23.1%
C#	1.8%
C	1.5%
Visual Basic	0.9%
Java	0.6%
Other	11.3%

Do you think basic knowledge on Informatics should be compulsory for every student in the education system (Similar to foreign language)?

The Israeli case

A bit more than half of the respondents (57%) claimed that basic knowledge on Informatics should be compulsory for every student in the education system. Some of the respondents' comments supporting Informatics as compulsory subject were:

- Algorithmic thinking is important for problem solving in other areas in life.
- Algorithmic thinking is as important as Mathematics and contributes to the understanding of technology in the modern world.
- Algorithmic thinking contributes to logical thinking, abstraction and analytic capabilities and modular thinking.
- Today programming is a skill that is important for everyone.

Some of the respondents' comments against Informatics as compulsory subject were:

- There is a wide variety of students and not all them will find the subject interesting.
- Informatics is suitable only for the science students.
- It is recommended to teach computer application and not Informatics.
- It is recommended to teach higher level Mathematics rather than Informatics.

The Lithuanian case

Many of the respondents think that basic knowledge on Informatics should be compulsory for every student in the education system for the following reasons:

- Informatics is prevalent in daily life therefore it is important to learn it.
- Physics explains the laws of reality and informatics explains the laws virtual reality.
- We need to think algorithmically and analyze the data from daily life.

Starting from which grade do you think algorithm and programming should be introduced?

The Israeli case

Half of the respondents though that algorithm and programming should be introduced as early as the 5th or 6th grade. A third of the respondents thought that it should be introduced in the 7th grade and the rest thought that it should be introduced in the 9th or 10th grade.

Several respondents commented that early introduction of algorithm and programming will develop thinking skills in later classes. One respondent said that children are already exposed to algorithmic thinking through the use of technology and computer games therefore it should be easy to start teaching formally at an early age.

Few of the respondents that supported the 7th grade claimed that students at that age are mature enough to cope with algorithm thinking.

The Lithuanian case

About 40% of the respondents though that algorithms and programming should be introduced as early as the 5th or 6th grade. About 40% of the respondents thought that it should be introduced in the 7th or the 8th grade.

Some of the comments were:

- The concept of algorithm could be explained from 5th grade.
- Easy algorithms could be introduced in early years and it must be a continuous process.
- It is important to show students that informatics it is not only word processing.

6. Summary and Discussion

The changing global context due to the impact of ICT is redefining the type of literacy and skills that are needed. Such skills are not only technical but also cognitive and they involve high-order thinking. The importance of new skills has started to receive considerable political interest throughout Europe (Informatics Education, 2013). These are new challenges for researchers to concentrate their attention to this field.

Informatics education in schools does not clear up the myths about CS and most of the students in high schools graduate with no clear answers to the popular statements formulated as "relations": CS = programming, CS = IT (ICT), CS = computer literacy, CS = a tool for studying other subjects, CS ≠ scientific discipline. The White Paper by the CSTA (Stephenson et al., 2005) lists a number of challenges and requirements that must be met if we want to succeed in bridging the gaps in education and improve education in informatics as a CS discipline:

- students should acquire a broad overview of informatics;
- informatics instruction should focus on problem solving and algorithmic thinking;
- informatics should be taught independently of application software, programming languages and environments;
- informatics should be taught using real-world problems;
- informatics education should provide a solid background for the professional use of computers in other disciplines.

As we can see from the above sections, each country developed a unique curriculum for high school CS education. Israel introduced CS in high school in the mid-1970s

while in Lithuania Informatics was introduced a few years later. Both curricula are continuously being developed and updated according to recommendation of important interest groups and international organizations such as ACM, CSTA and UNESCO. In both countries there is a strong involvement of researchers from the academia in curriculum development and implementation.

Both countries focus on the fundamentals of algorithms and programming and data structures. In Israel it is implemented in the first 2 units Foundation of Computer Science 1 and 2 and in the fourth unit Data Structures. In Lithuania it is implemented in the optional module on programming for high school. Israel's program includes additional modules including an additional paradigm or application and a theoretical unit.

From the survey results we found that most of the respondents in both countries are senior teachers who have been teaching CS for more than 6 years and a large percentage of them (86% in Lithuania and 52% in Israel) head the CS program in their schools. About half of the teachers in both countries have an undergraduate degree in CS and good number of teachers (36% in Lithuania and 60% in Israel) also has a graduate degree. Therefore, we value their responses which are based on many years of experience and insight on CS education.

In Israel, the vast majority of the respondents (79%) claimed that they were very satisfied or satisfied with the informatics curriculum while in Lithuania 77% of the respondents claimed that they were not very satisfied with the informatics curriculum. Some of the comments were that they would like more emphasis on programming, modern technologies and new learning materials.

Most of the respondents in Israel felt that they have enough professional support while in Lithuania only about half felt the same. They mentioned that they would like more practical advanced courses and pointed out that teachers do not take the courses because they are only in the capital and are not free.

In Israel most of the respondents support Java and C# as an appropriate programming language for teaching Fundamentals of CS while in Lithuania most of the teachers support C++.

In Lithuania most of the respondents claimed that basic knowledge on Informatics should be compulsory for every student in the education system while in Israel only about half felt the same. Half of the respondents in Israel and about 40% in Lithuania thought that algorithms and programming should be introduced as early as the fifth or sixth grade and the preferred programming language in Israel was Scratch while in Lithuania the preferred programming languages were Logo, C++ and Scratch.

From this research we can conclude that the teachers feel that it is important for the curriculum to be continuously updated and developed according to advances in the field and that it is important to develop teacher support programs and to introduce algorithmic thinking as early as 5^{th} or 6^{th} grade.

References

Atchison, W. F., Conte, S. D., Hamblen, J. W., Hull, T. E., Keenan, T. A., Kehl, W. B., Viavant, W. (1968). Curriculum 68: Recommendations for academic programs in computer science: a report of the ACM curriculum committee on computer science. *Comm. of the ACM, 11*(3), 151-197.

Blonskis, J., Dagienė, V. (2008). Analysis of students' developed programs at the maturity exams in information technologies. In *International Conference on Informatics in Secondary*

Schools-Evolution and Perspectives, pp. 204-215. Springer Berlin Heidelberg.

Computing at School Working Group, (2012). A Curriculum Framework for Computer Science and Information Technology. http://www.computingatschool.org.uk/data/uploads/Curriculum%20Framework%20for%20CS%20and%20IT.pdf

Dagiene, V., Stupuriene, G. (2016a). Informatics Concepts and Computational Thinking in K-12 Education: A Lithuanian Perspective. *Journal of Information Processing*, *24*(4), 732-739.

Dagienė, V., Stupuriene, G. (2016b). Bebras - a sustainable community building model for the concept based learning of informatics and computational thinking. *Informatics in Education-An International Journal*, 15(1), 25-44.

Dagienė, V., Sentence, S., Stupuriene, G. (2017). Developing a Two-Dimensional Categorization System for Educational Tasks in Informatics. *Informatica*, 28(1), 23-44

Drysdale, S., Hromcik, J., Reed, D., Hahne, R. (2005). The year in review: changes and lessons learned in the design and implementation of the AP CS exam in Java. In *ACM SIGCSE Bulletin* (Vol. 37, No. 1, pp. 323-324). ACM.

Gal-Ezer, J., Beeri, C., Harel, D., Yehudai, A. (1995). A high school program in computer science. *Computer*, *28*(10), 73-80.

Gal-Ezer, J., Harel, D. (1999). Curriculum and course syllabi for a high-school CS program. *Computer Science Education*, *9*(2), 114-147.

Haberman, B., Averbuch, H., Cohen, A., Dagiene, V. (2011). Work in Progress - Initiating the Beaver Contest on Computer Science and Computer Fluency in Israel. In 41 st ASEE/IEEE Frontiers in Education Conference. October 12-15, 2011, Rapid City, South Dakota (p. 1-2). ©2011 IEEE. URL http://fie-conference.org/fie2011

Hunt, F., Kmoch, J., Nevison, C., Rodger, S., Zelenski, J. (2002). How to develop and grade an exam for 20,000 students (or maybe just 200 or 20). In *ACM SIGCSE Bulletin* (Vol. 34, No. 1, pp. 285-286).

Informatics education: Europe cannot afford to miss the boat (2013). Report of the joint Informatics Europe & ACM Europe Working Group on Informatics Education. URL http://www.informatics-europe.org/images/documents/informatics-education-europe-report.pdf

Joint IEEE Computing Society/ACM Task Force on Computing Curricula. (2013). Final Report. URL http://www.acm.org/education/CS2013-final-report.pdf

K–12 Computer Science Framework, (2016). URL http://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf

Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., Simon, B. (2004). A multi-national study of reading and tracing skills in novice programmers. In *ACM SIGCSE Bulletin* (Vol. 36, No. 4, pp. 119-150).

Merritt, S. M., Bruen, C. J., East, J. P., Grantham, D., Rice, C., Proulx, V. K., Wolf, C. E. (1993). ACM model high school computer science curriculum. *Commun. ACM*, *36*(5), 87-90.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

Publications of the Israeli Central Bureau of Statistics (2013) at: URL http://www.cbs.gov.il/reader/shnaton/shnatonh_new.htm?CYear=2013&Vol=64

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Kafai, Y. (2009). Scratch: programming for all. *Comm. of the ACM*, *52*(11), 60-67.

Science and Technology Office in the Ministry of Education (2014) at: URL http://cms.education.gov.il/EducationCMS/UNITS/MadaTech/csit

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D. O'Grady-Cuniff, D., Boucher Owens, B., Stephenson, C., Verno, A. (2011). CSTA K-12 Computer Science Standards. CSTA Standards Task Force.

Stephenson, C. et al. (2005). The New Education Imperative: Improving High School Computer Science Education. Final Report of the CSTA Curriculum Improvement Task Force, CSTA, ACM.

Tenenberg, J., McCartney, R. (2014). Editorial: Computing education in (k-12) schools from a cross-national perspective. *ACM Transactions on Computing Education (TOCE)*, *14*(2), 6.

The Royal Society. Shut down or restart? (2012). The way forward for computing in UK schools. The Royal Society. URL https://royalsociety.org/~/media/educationcomputing-in-schools/2012-01-12-computing-in-schools.pdf

Tucker, A. B., Barnes, B. H., Aiken, R. M. (1991). Computing Curricula 1991; a summary of the ACM/IEEE-CS Joint Curriculum Task Force report. *Comm. of the ACM*, *34*(6), 68-85.

Tucker, A. (2003). A Model Curriculum for K--12 Computer Science: Final Report of the ACM K--12 Task Force Curriculum Committee.

Vee, A. (2013). Understanding computer programming as a literacy. *Literacy in Composition Studies*, *1*(2), 42-64.

Authors' information

Tamar Benaya holds a M.Sc. in Computer Science from Tel-Aviv University. She is a faculty Member of the Computer Science Department at The Open University of Israel. She designed and developed several advanced undergraduate Computer Science courses and workshops, and she serves as a course coordinator of several courses. She also supervises student projects. She is a lecturer of Computer Science courses at The Open University of Israel. Her research interests include Distance Education, Collaborative Learning, Computer Science Education, Computer Science Pedagogy and Object Oriented Programming.

Ela Zur is involved in the Israel IOI project since 1997, and repeatedly served as a deputy leader. She holds a PhD Degree in Computer Science Education from Tel-Aviv University. She is a faculty member of the Computer Science Department at the Open University of Israel. She designed and developed several advanced undergraduate Computer Science courses and workshops, and currently serves as a course coordinator of several courses. Her research interests include Distance Education, Collaborative Learning, Computer Science Education, Computer Science Pedagogy, Teacher Preparation and Certification and Object Oriented Programming.

Valentina Dagienė is professor and principal researcher at Vilnius University Institute of Mathematics and Informatics. She has published over 200 scientific papers and more than 50 textbooks in informatics for high schools. She has been working in various expert groups and work groups, organizing the Olympiads in informatics among students, also engaged in localization of software and educational programs, e-learning, and problem solving. She is an Executive Editor of international journals "Informatics in Education" and "Olympiads in Informatics". She has participated in several EU-funded R&D projects, as well as in a number of national research studies connected with technology and education.

Gabrielė Stupurienė is a doctoral student at Vilnius University Institute of Mathematics and Informatics at the Department of Informatics Methodology. She has been working with the Bebras challenge since 2010. As a Master student she worked on Conceptualisation of Informatics Fundamentals through the Bebras Tasks of earlier years. Her main research focus is developing informatics concepts based educational model for schools.

Received April 5, 2017, revised May 9, 2017, accepted May 10, 2017