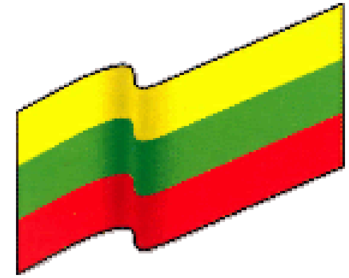
From Algorithms to Computational Thinking in K-12: the Lithuanian Experience

Prof. dr. Valentina Dagienė
valentina.dagiene@mii.vu.lt

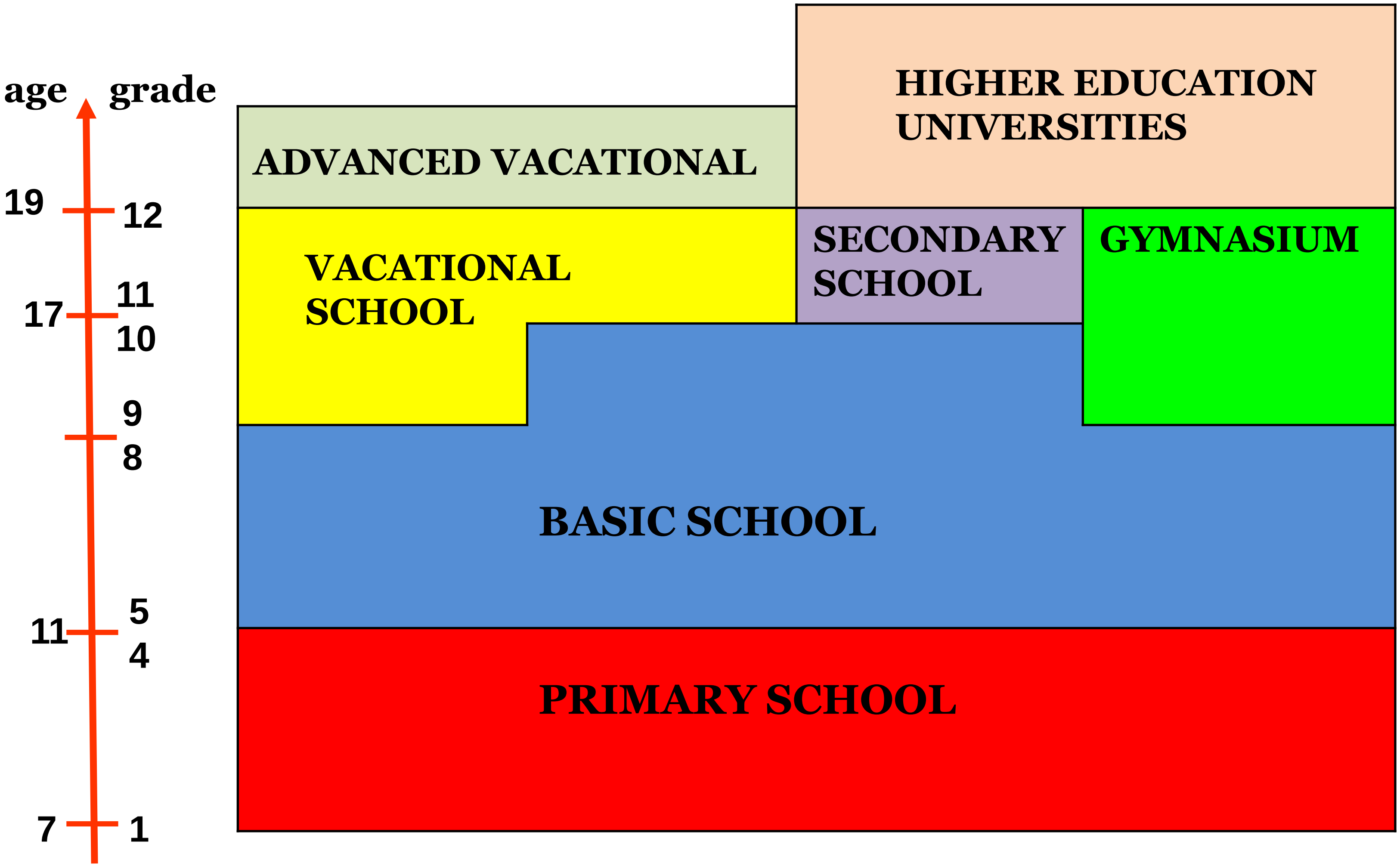
Vilnius University

Lithuania – **LIETUVA**

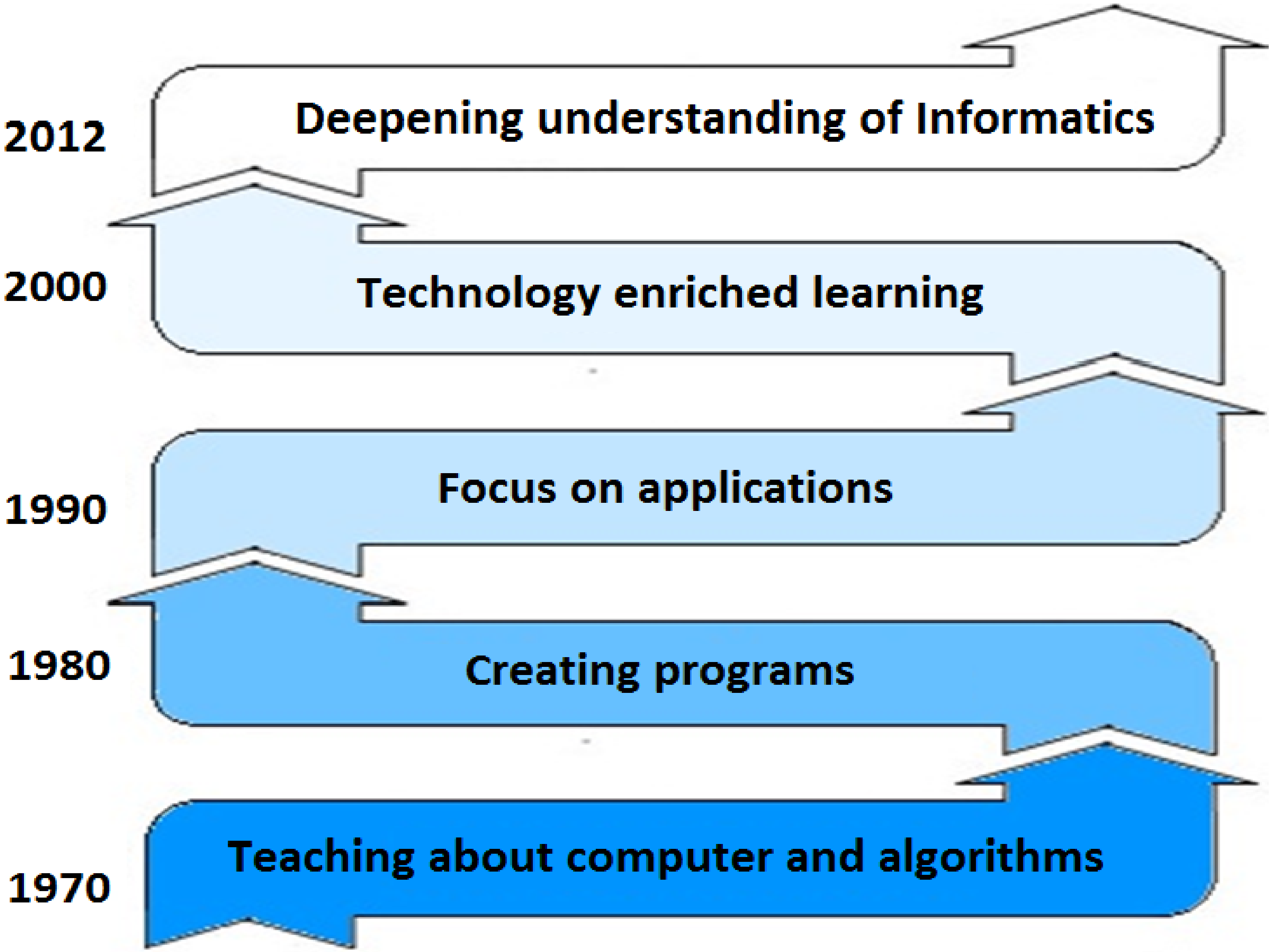


- Territory – 65 300 km²
- Population – about 3 mln.
- Vilnius – about 0,5 mln.
- Currency – Euro (2015)
- Borders: with Belorussia, Latvia, Poland, Russia and Baltic sea

The General Education Structure in Lithuania



Short glance to Informatics/IT at School



„Prehistory“ of teaching programming

- ~40 years ago (in 1975) – the idea of nation wide teaching of programming in schools in Lithuania has emerged.
- Implementation: Teaching material was prepared.
- In 1979–1981 the Experimental School of Programming by Correspondence was organized.
- 34 years ago (January 1981) –

Young Programmer’s School by Correspondence was established officially
Jaunųjų programuotojų mokykla

Pirmoji pamoka

ALGORITMAI

Gyvenime labai dažnai sutinkame iš anksto numatytus nurodymus, kuriuos reikia vykdyti norint atlikti konkretų darbą. Pavyzdžiui, prie telefono automato galima rasti instrukciją, kurioje trumpai ir aiškiai pasakyta, ką reikia daryti, norint paskambinti:

„1. Įmeskite dviejų kapeikų monetą į automato skylę.

2. Nukelkite ragelį ir laukite signalo.

3. Išgirdę ilgą, nepertraukiamą gaudesį, surinkite reikiamą numerį ir laukite atsakomojo signalo.

4. Išgirdę ilgus gaudesius, laukite, kol abonentas atsakys.

5. Išgirdę trumpus, dažnai pasikartojančius gaudesius, pakabinkite ragelį ir išimkite monetą: jums reikalingas abonentas užimtas".

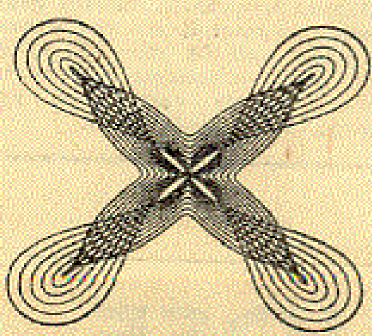
Panašios instrukcijos sudaromos ir uždaviniams spręsti. Pavyzdžiui, dviejų skaičių a ir b aritmetinio vidurkio radimą galima nusakyti nurodymais:

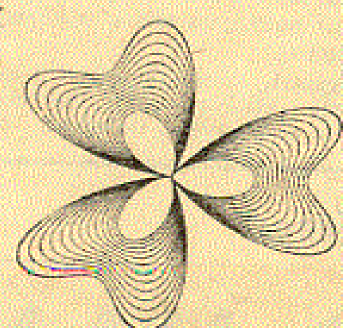
1. Sudėkite duotus du skaičius.
2. Gautą sumą padalykite iš dviejų.

Skambinimas telefonu ir aritmetinio vidurkio ieškojimas — tai du labai skirtingi procesai. Tačiau jie turi ir bendrų bruožų. Abu procesai aprašyti trumpais ir aiškiais nurodymais. Tų nurodymų seka ir sudaro algoritmą. Tiksliai atlikę aprašytus nurodymus, gauname reikiamą rezultatą: paskambiname telefonu, randame aritmetinį vidurkį.

Algoritmu vadinami aiškūs ir visų vienareikšmiškai suprantami nurodymai, nusakantys veiksmų procesą, kaip iš turimų duomenų gauti reikiamą rezultatą. Turimi duomenys vadinami pradiniais. Jie žinomi prieš atliekant algoritmą. Rezultatai dar vadinami galutiniais duomenimis. Jų reikšmės sužinomos atlikus algoritmą. Dviejų skaičių vidurkio algoritmo pradiniai duomenys yra du duoti skaičiai, o galutinis — vidurkis.

Algoritmai užrašomi įvairiai. Jų užrašymo pavidalas, nurodymų skaičius ir detalumas priklauso nuo to, kam jie skirti, t. y. kas atliks algoritmo nurodymus (spręs uždavinį). **Jeigu norima, kad uždavinį spręstų mašina, jį reikia užrašyti mašinai suprantamu pavidalu. Tokie algoritmai vadinami programomis. Mes ir nagrinėsime programas.** Pradėsime nuo paprastų uždavinių, kuriuos nesunku būtų išspręsti ir be ESM. Palaipsniui uždavinių „svorį" didinsime. Sudarysime ir tokių uždavinių programas, kuriuos be ESM išspręsti būtų per sunku ar iš viso neįmanoma.





Šių fantastinių piešinių autorė — ESM.

The first lesson of JPM
(Young Programmer's School)
published 1981-01-27
in daily newspaper
"Komjaunimo tiesa"

PROGRAMAVIMO KULTŪRA

TRYLIKTOJI PAMOKA

Skyrelį tvarko LTSR MA Matematikos ir kibernetikos instituto jaunesnioji mokslinė bendradarbė Valentina DAGIENĖ

Programuotojai, rašantys neaiškias, griozdiškas programas, mėgsta teisintis, kad programa skiriama kompiuteriui, o ne žmogui. Be abejo, kompiuteriui programos aiškumas nesvarbus — jis mechaniškai atlieka veiksmus ir nesidomi programos vaizdumu. Tačiau kad ir kaip atrodytų keista, didžiausias programų skaitytojas vis dėlto yra žmogus, o ne kompiuteris. Skaitydamas programas, žmogus susipažįsta su kitų programuotojų idėjomis ir patirtimi, mokosi pats sudarinėti programas. Dažnai tenka tobulinti ir pačių su-

mentarais galima paaiškinti ne tik kintamųjų vardus, bet ir atskiras programas dalis, nurodyti, ką vienas ar kitas sakinys atlieka ir panašiai. Komentarus galima įterpti visur tarp atskirų simbolių, **žodžių, skaičių, vardų. Jie suskliaudžiami skliaustais (*ir*).**

Komentarai padeda greitai ir lengvai skaityti programas. Tačiau jais nereikia piktnaudžiauti — komentarai turi būti lakoniški, griežti, trumpai nusakantys pagrindinius dalykus, neužgriozdinantys programos teksto.

Paminėsime dar vieną programavimo kultūros elementą — **programų redagavimą. Redagavimu vadinamas programos teksto išdėstymas popieriaus lape.** Nekyla abejonių, kad žmogui kur kas lengviau skaityti vaizdžiai išdėstytą programą. Be to, tokioje programoje būna mažiau klaidų (pavyzdžiui, sunkiau pamiršti žodį end, jei jis rašomas po

rojo mėnesio pabaigoje prieauglį duos tik pirmoji pora, todėl turėsime tris poras, o dar po mėnesio prieauglį duos ir pradinė pora, ir pora, gimusi prieš du mėnesius. Todėl iš viso bus 5 poros.

Simboliu F(n) pažymėkime triušių porų skaičių, kurį turėsime po n mėnesių. Matome, kad n-ojo mėnesio pabaigoje turėsime tiek porų, kiek jų buvo prieš mėnesį, t.y. F(n−1) ir dar tiek naujų porų, kiek jų buvo prieš du mėnesius, t.y. (n−2)-ojo mėnesio pabaigoje. Kitaip sakant, gausime tokią priklausomybę:

$$F(n)=F(n-1)+F(n-2)$$

Pateiksime programą triušių porų skaičiui po n mėnesių spausdinti.

```
program fibonacci;
   var fn, (* F(n) *)
         fn1, (* F(n—1) *)
         fn2, (* F(n—2) *)
         n, (* mėnesių skaičius*)
         k: integer;
```

```
for s:=1 to n do
   begin
      write (s);
      for d:=1 to n do
         if s mod d=0 then
            write ('+');
      writeln
   end
end.
```

Jis nėra efektyvus, tačiau kadangi iš uždavinio sąlygos nerealu, kad n būtų labai didelis, tai skaičiavimų bus nedaug ir nėra reikalo šiuo aspektu tobulinti programos.

KONTROLINIAI UŽDAVINIAI

13. Duota programa:

```
program atspėk;
   var a, b, c,
      i, j: integer;
begin
   read (n);
   a:=1; b:=1;
   for i:=0 to n do
      begin
         for j:=1 to b do
            write ('*');
         writeln;
         c:=a+b;
         a:=b; b:=c
      end
end.
```

Ką išspausdins kompiuteris, atlikęs šią programą, jei pradinis duomuo yra 7? Kokio uždavinio sprendimas užrašytas šia programa? (7 balai)

The curriculum and content

- Names, variables, values, assignment statement and sequence of statements
- Branches of actions
- Repetition (Loop)
- Program and its running by computer
- Logical values
- Functions and procedures
- Recursion
- Discrete data types
- Real numbers and records
- Arrays
- Programming style
- Program design
- Efficiency

Program reading was considered as important as writing

- Programmer must be able to read and investigate the programs designed by other programmers.
- Program analysis – reading makes a considerable part of home tasks.
- Pupils were asked to modify a program in order to adapt it to another similar problem, to restore the omitted parts of its text, etc.

Development periods of the Young Programmer's School by Correspondence

1. General programming teaching (1981–1986)
2. Learning effectively: differentiation by students' abilities (1986–1993)
3. Intensive teaching of gifted students (1993–1999)
4. Training students for the Informatics Olympiads (1999–2005)
5. Using new media (virtual learning environment) while learning algorithms (since 2005)

What is valuable: Lessons learned

- **Tasks**: Interesting, attractive tasks (small pieces) and problems (connected to real world) for students
- **Flexible learning**: Learning should occur anytime anywhere
- **Learning resources**: Divide learning material in pieces, combine practical problems with theoretical approaches
- **Teaching resources**: Well-prepared material for teachers
- **Face-to-face meetings**: summer camps, teacher training sessions...
- The successful introduction Informatics is determined by a change of mental habit and promoting **co-operation among researchers, policy makers, and teachers**

Logo

- Learning Logo
- Logo contests
- Logo workshops for teachers
- Books for schools (project-based method)
- „Comenius Logo“ learning environment

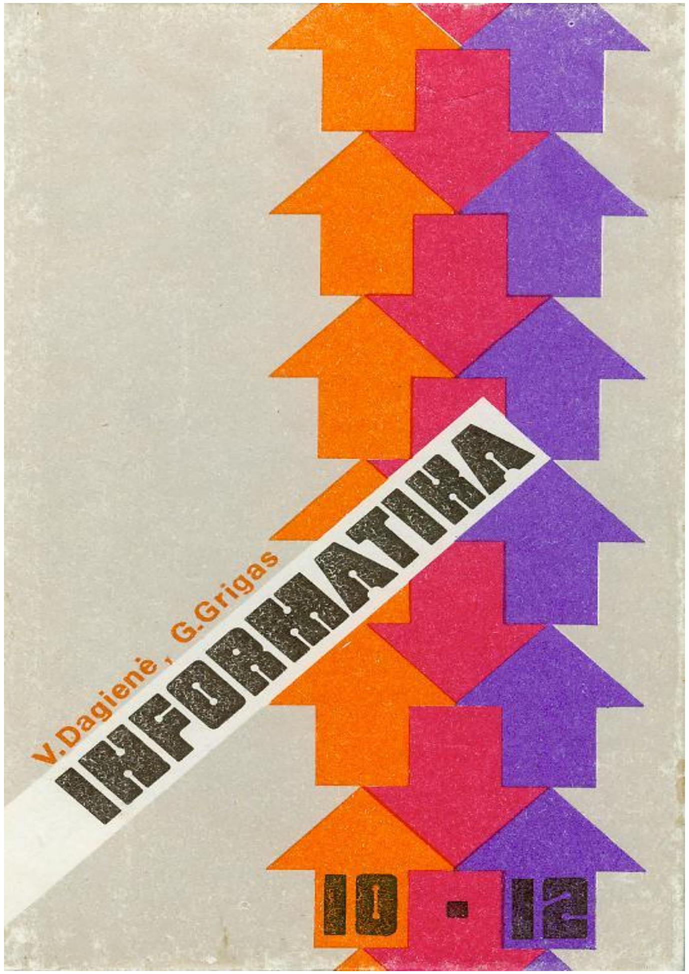
Valentina Dagiene and Seymour Papert
Hanoi, Vietnam, 2006

Informatics in schools of Lithuania: 1986

- The official beginning of informatics as a subject in schools of Lithuania is dated back to 1986
- Main idea: introduce each pupil to a computer
- *Programming is the second literacy* - prof. Andrei Ershov, a founder of the Siberian School of CS
- We translated the textbook and added a chapter on Pascal programming language

Informatics in schools of Lithuanian: 1991

- In 1991 the first curriculum for teaching informatics in secondary schools was developed
- An original Lithuanian textbook of Informatics was written just after Lithuania has regained independence
- The course was lectured for two years in school-leaving grades 10th – 12th of upper secondary schools composing it in different ways



Informatics in schools of Lithuanian: 1997

Since 1997 the teaching of informatics essentially changed:

- the compulsory course for the 9th and 10th grades was introduced
- Informatics remained compulsory for the 11th and 12th grades as well
- the possibility to take advanced optional modules was provided

Informatics in schools of Lithuanian: 2005

- Since 2005 the basics of informatics and IT as a separate subject has been introduced to grades 5 and 6
- Changing name: Informatics to Information Technologies or IT
- IT – a compulsory subject at 5 –10 grades
  - 1 hour per week (35 hours per year) for grades
    - 5 and 6
    - 7 or 8
    - 9 and 10
- Optional modules for grades 11 and 12 – programming, data base, desktop publishing – for upper secondary school

Lower Secondary School, grades 5-10

5-6 Grades

- Handling information by computer
- Drawing with computer
- Text processing and formatting
- Internet and services
- Moddeling with computer

7-8 Grades

- Handling information by computer
- Text processing and formatting
- Data handling and spreadsheets
- Presentation

9-10 Grades

- Handling information by computer
- Text processing and formatting
- Internet and services
- Data handling and spreadsheets
- *Introduction to programming*
- *Desktop publishing*
- *Webpage design*

Upper Secondary School, grades 11-12

General course

- Formatting texts
- Spreadsheet
- Presentation
- Internet security, ethics

Extended course

- **General course**

\+

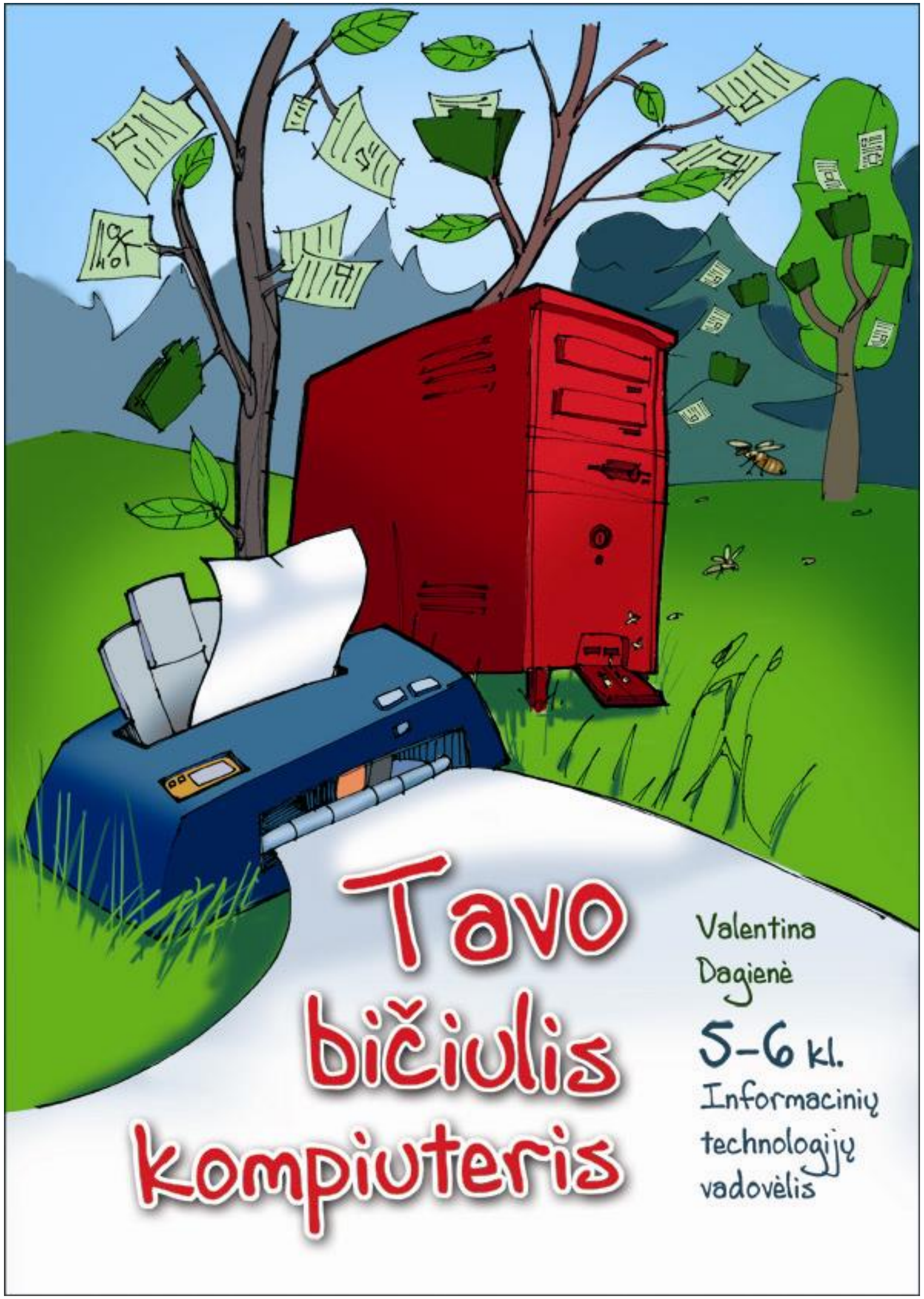
- *Programming*
- *Desktop publishing*
- *Data Base developing and management*

9-10 Grades

- *Introduction to programming*
- *Desktop publishing*
- *Webpage design*

IT in 5–6 grades

Themes, subthemes	IT hours	Subjects, integration is addressed to
Introduction to computer application	**10**	
Principles of computer use	**6**	
Drawing with computer	**4**	Art; 10
Text and keyboard	**14**	Mother tongue; 10
Internet and electronic mail	**10**	Mother tongue; 4 Foreign language; 10
Modeling (*Logo* or *Scratch*)	**24**	



Textbook for 5-6 grades

- Introduction to computer
- Drawing
- Text processing
- Internet and emails
- Modeling (Logo)

Contents of Informatics and IT subjects

9-10 grades (*Compulsory course*)	**11-12 grades** (***Optional*** *course*)	**11-12 grades** (***Advanced*** *modules*)
Computer (principles of the work)	Advanced elements of text editing	Data base
Text processing	Presentation	Multimedia
Information (basics of information handling)	Web and email	Programming
Algorithms (main concepts and commands)	Social and ethical issues of using IT	
	Spreadsheet	

Matriculation in Lithuania

- All high school students are required to take matriculation exams in main subjects studied in high school: at least two and not more than 6.
- Informatics (IT + Programming) is one of them
- The exams are external nation wide exams - National Examination Centre is responsible.
- Evaluation of an exam is by points from 16 to 100.

Informatics exam structure

Two parts	Questions	Scores
Computer literacy tasks	1-2 multiply choice questions 2-3 short answer questions 2-3 open-ended questions 1 task with text processing 1 task with spreadsheet	10 20 20
Programming tasks	2 or 3 practical tasks to be programmed	50

Components of curriculum of programming exam

Algorithms	Data structures	Control structures
Calculation of the sums (of product, quantity, and arithmetical average). Search of the maximal (minimal) value. Data input/output. Data sorting. Modification of algorithms according to the particular data structures.	Integer and real, char, boolean, and string . Text file. One-dimension array. Record. Development of data structures.	Program structure. Comments. Variables. Assignment and sentence. Logical operations, if statement. Loops. Compound statement. Procedure and function. Parameters and arguments. Standard files.
Programming environment. Technology of procedural programming. Testing. Program documentations. Arrangement of dialog. Program writing (style)		

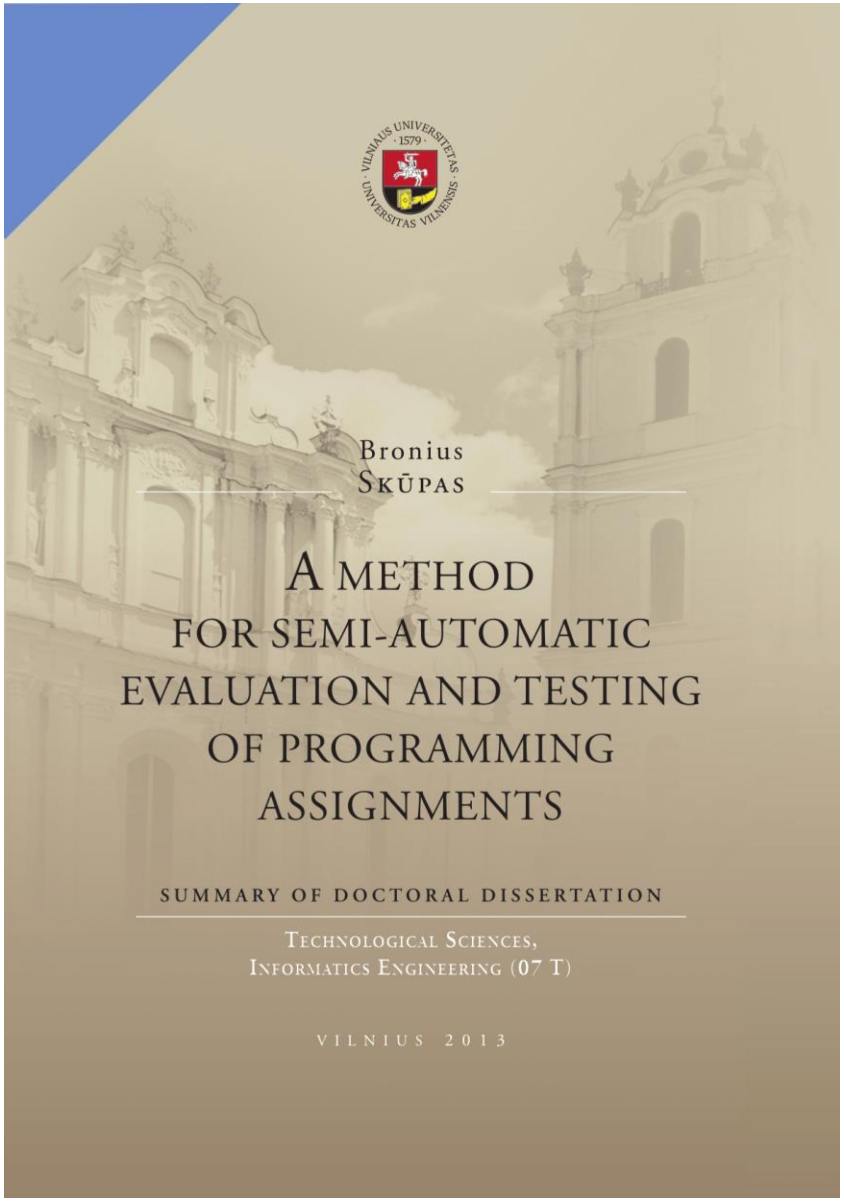
Programming tasks

- A first task is intended to examine the students' abilities
  - to use the procedures or functions,
  - to use the data types,
  - to realize the algorithms for work with data structures,
  - to manage with input and output in text files.

- A 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. Students usually are asked:
  - to input data from the text file to arrays containing the elements of record type,
  - to perform operations by implementing algorithms,
  - to present the results in a text file.

Evaluation system

- Both tasks intentionally requires to write batch style programs, as they are more suitable for blackbox testing.
- Semi-automatic evaluation system with blackbox testing is developed for evaluation of programs.
- Exam evaluation system has different requirements than systems used in programming courses or programming contests



Evaluation schema of a programming task

First task: evaluation criteria	Points	Comments
Tests	20	If the program provides correct outputs to all tests.
Correct reading from file	4	Evaluated only if the program scores no points for the tests.
The result is outputted correctly	2	
The function, which calculates the number of chess sets that can be collected from the pieces brought by the students is created	5	
Other functions, procedures (if there are ones) and the main program are correct	9	
The data type of array is declared correctly	1	Always evaluated.
The function is crated	1	
Meaningful names of the variables. Program parts are commented, spelling is correct.	1	
Programming style is consistent, no statements for working with the screen.	2	
Total	**25**	

IT and Informatics in school education of Lith ia

Informatics introduced in our curriculum in 1986 for HS has never been removed !!!

IT/Informatics as a stand-alone subject from grades 5 to 12 !!!

Infor nd IT education
g related to comp rs
in school

17 - 19

Tertiary education – Universit

Upper – ool

Secondary

Informat T advanced –
odules

11 - 17

Lowe
middle or basi

IT and Informatics for all
with elements of algorithmics

7 - 11

Pri ucation

integrated

From 2015: Modern Informatics /IT for all children with elements of programming

6

Pre-school year

Modern curriculum: Informatics and IT

1. Understanding and analysis of problems based on logical and abstract thinking, algorithmic thinking, algorithms and representations of information.
2. Programing and problem solving by using computers and other digital devices – designing and programming algorithms; organizing, searching and sharing information; utilizing computer applications.
3. Using computers, digital devices, and computer networks – principles of functioning of computers, digital devices, and computer networks; performing calculations and executing programs.
4. Developing social competences – communication and cooperation, in particular in virtual environments; project based learning; group projects; equity.
5. Observing law and security principles and regulations – respecting privacy of personal information, intellectual property, data security, netiquette; positive and negative impact of technology on culture, social live and security.

Supporting activities

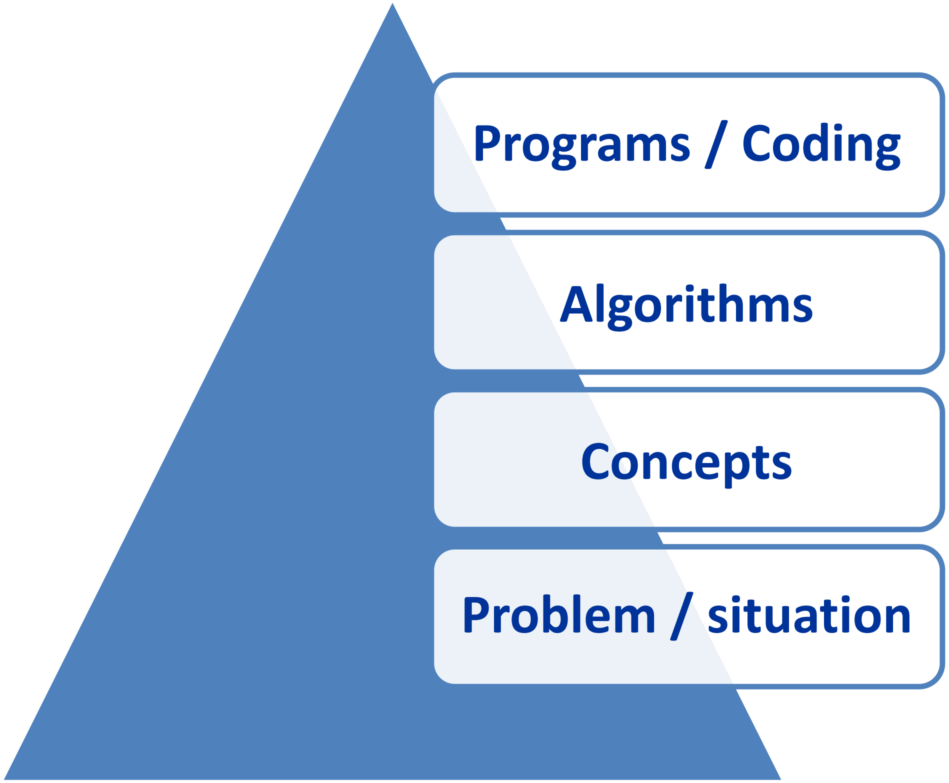
- Teacher preparation:
  a teacher is the most important „technology”!
- Standards, evaluation and support in a classroom
- In-service training at universities – based on standards
- Web service – materials, MOOCs
- Comments to the curricula of other subjects – how to use computational thinking in solving problems
- PBL and flipped learning – extra hours of school learning
- Gamification
- Contests: *Bebras*, Olympiads…
- Informatics oriented tasks in national school tests

Challenges

- How to motivate and engage students through K -12, for 12 years, e.g. learning programming requires constant practice
- The role of coding – programming
- When and how to switch from visual to textual programming?
- Visual – for beginners, non-professional
- Textual – for those who seriously think about CS – we don't want to loose them

The curriculum – general comments

Informatics ≠ programming
Concepts before tools, before programming



There are plenty of ways to introduce/teach informatics concepts ... without computers:

- ❖ CS unplugged
- ❖ *Bebras* tasks

When appropriate, we can extend unplugged CS by adding ... a computer

❖ Grades 7-9 – focus on real world problems and applications which are meaningful for pupils
❖ Grades 10-12 and vocational schools – CS/ICT specializations

The curriculum – the role of programming

- ❖ Remember: Informatics ≠ programming
- ❖ How to use extra curricular coding activities (e.g. the Hour of Code) in the classroom?

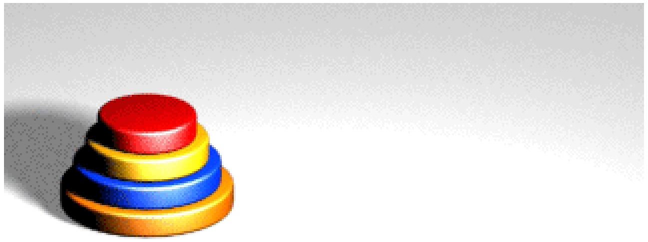
Programming

- ❖ Programming is a tool, not a goal
- ❖ Which programming language? – there are 3000
  - – any, which can be used to introduce and illustrate concepts
  - – introduce new constructs when needed
  - – a program is a message for a computer and also to other people
  - – different languages different programming methods
  - – visual versus textual languages and programming

Methods of introducing Informatics concepts

- Use all three forms of activities:
  - visual learning: pictures, objects, abstract and physical models, ...
  - auditory learning: exchange ideas, discussions, group work, ...
  - kinesthetic learning – physical activities
- Learn/teach in environments of three stages:
  - cooperative games and puzzles that use concrete meaningful objects – discovering concepts: Bebras tasks, The Hour of Code
  - computational thinking about the objects and concepts – algorithms, solutions
  - programming – Scratch, The Hour of Code, Logo
- *Bebras* tasks – the source of problem situations
- The Hour of Code – introduction to (visual) programming with puzzles

The Hanoi Towers



- ❖ The Hanoi Towers story
- ❖ *In the beginning*: ask kids to play and try to find „an algorithm” and calculate the number of moves for different numbers of rings
- ❖ *Expected*: algorithms and tables with the number of moves
- ❖ *Then*: kids play with (against) a computer program
- ❖ *Finally:* they verify initial findings
- ❖ *Extra (MS, HS)*: recursive solution, minimum number of moves

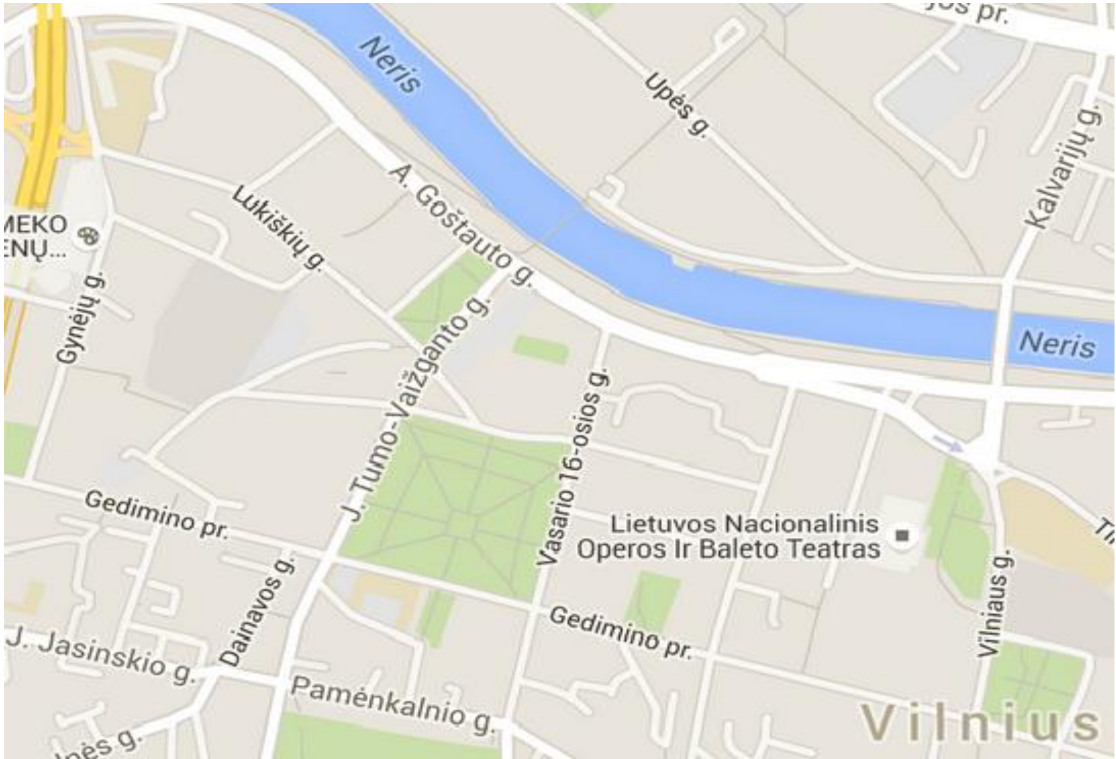
Concepts:

- ❖ game
- ❖ algorithm
- ❖ efficiency (complexity)
- ❖ recursion

Shortest path – introduction

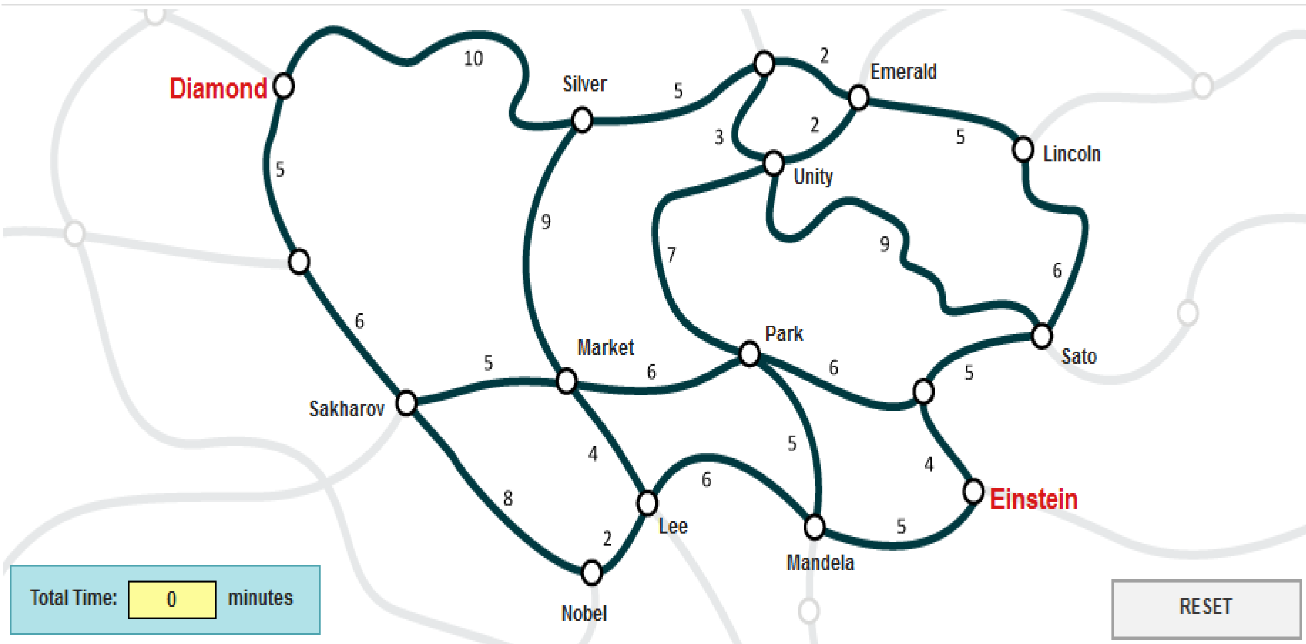
Kids are working with real situation – motivates them:

- Computer: Find your house and your school on the *Google* map. Find your way to/from school
- Find shortest paths (distance and time) to/from school by different transportation means: on foot, by bicycle, by car, public transportation
- Paper and pencil: Table to compare which is the shortest path (time/distance) to school?



Shortest path – PISA task

From Einstein to Diamond it takes 31 min – which way?



Concepts:

- graph models
- algorithm
- greedy approach
- shortest paths
- Dijkstra's algorithm
- symmetry

Typical approach, a greedy type: the nearest neighbor method. It doesn't work!

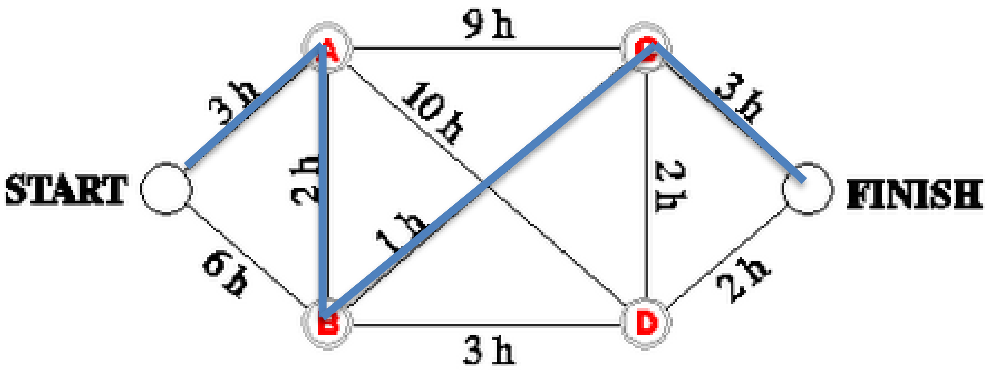
However it works when you go from Diamond to Einstein !!!

Think: Dijkstra's algorithm is a greedy method and optimal

Shortest path – Beaver task

From START to FINISH

Find the quickest path from START to FINISH. The hours in the figure indicate how much time must be spend to travel between the adjacent stations.



START – A – B – C – FINISH

START – A – C – FINISH

START – B – C – FINISH

START – B – D – C – FINISH

Conclusions

With the modern Informatics/IT curriculum:

- Students acquire a broad overview of informatics/IT and applications.
- Teaching informatics focuses on problem solving and CT.
- IT/Informatics is taught independently of application software, languages, environments – students are free to make their own choice.
- IT/Informatics is taught using problem situations coming from school subjects and real-world applications.
- IT/Informatics education provides a background for the professional use of computers in other disciplines.
- Students experience a solid foundation in CT through problem solving with computers
- Students experience that programming is a creative process.
- Students learn how to collaborate on projects.
- IT/Informatics enables innovation also in other fields.
- PBL and flipped methods contribute to personalization of learning.

Thanks you for your attention!

valentina.dagiene@mii.vu.lt