

# Computer Science Education

in Poland


Maciej M. Sysło



Warsaw School of Computer Science

[syslo@ii.uni.wroc.pl](mailto:syslo@ii.uni.wroc.pl); <http://mmsyslo.pl/>

# Short history of computers in education in

- 1965: first regular classes in high schools in Wrocław on numerical methods and programming in Algol – there was no ICT (Uni. of Wrocław; Elliott 803) 
- 1970: beginning of teachers' training (on main frames)
- 1985: first formal CS curriculum/syllabus for high schools, approved by the Ministry of National Education – programming in Logo
- 1993-2008: several updated CS curricula
- 1993-2017: several changes in the school system



When a new government came to power in 2015 and planned to completely change our school system, we were ready with a new CS curriculum for all students in K-12. Fortunately, there were „computer classes” at each level and there were also teachers teaching in these classes.

# The curriculum 2017: Unified Aims for ALL Level

1. **Understanding and analysis of problems based on logical and abstract thinking, algorithmic thinking, and information representations.**
2. **Programming and problem solving using computers and other digital devices** – designing algorithms and programs, organizing, searching and sharing information using 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; taking various roles in group projects; equity.
5. **Observing law and security principles and regulations** – respecting privacy of personal information, intellectual property, data security, netiquette, and social norms; positive and negative impact of technology on culture, social life and security.

# algorithmic thinking *versus* computational thinking

Peter Denning 2009: *Computational thinking has a long history within CS. Known in the 1950s and 1960s as "algorithmic thinking,"*

- Define a problem situation, including data [*abstraction*], the goal and the results
- Formulate a plan for solving the problem – separate subproblems [*decomposition*] and indicate connections between them
- Choose a way to solve the problem:
  - design an algorithm [*algorithmic thinking*]
  - use an existing program or program a solution method in a selected programming language [*implementation, programming*]
- Analyze the correctness of the algorithm and its implementation [*debugging*], and assess its complexity, test the program [*testing*]
- Complex projects solve in a team [*collaboration*]
- Choose and solve problems from various school subjects [*generalization*]

Mental tools of computational thinking used in problem solving approach,



# Conclusions

---

## Concepts:

- unplugged informatics
- mental tools of computational thinking

are present indirectly in the curriculum and in the classroom practices prior to defining and introducing them into study and research

In our practice we do not directly refer to these approaches by their actual names, even today, they are naturally used

No classes on computational thinking, by name – students learn how to use its mental tools while solving problems.

# classical unplugged extended...



## Informatics for „smyk” – I4S

A package of 25 interactive applications, mainly for young children in grades 1-3:

Materiały dla ucznia

1. Pierwsze kroki ...		2. Codzienne czynności ...		3. Chodzenie w różnych kierunkach ...		4. Artystyczne kreski ...		5. Chodzenie w różnych kierunkach i... ...	
6. Jak dobrze nam idzie? ...		7. Sudoku I ...		8. Sudoku II ...		9. Sudoku III ...		10. Sudoku IV ...	
11. Memory ...		12. Posegreguj ...		13. Połącz w pary ...		14. Wprawki matematyczne ...		15. Piramidy działań matematycznych ...	
16. Łamigłówki z zapalkami ...		17. Symetria ...		18. Połącz kropki ...		19. Edytor graficzny I ...		20. Edytor graficzny II ...	
21. Flagi ...		22. Mondriany ...		23. Łapanie liter ...		24. Tangramy ...		25. Labirynty ...	

# Sudoku – CT from k-3 ...

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 ...

Proste Sudoku – kwadrat łaciński

Uzupełnij poniższy kwadrat tak, aby każda figura występowała dokładnie raz w każdym wierszu i w każdej kolumnie – jest to tak zwany kwadrat łaciński.


✓ ↻ 🔑

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 ...

Proste Sudoku – kwadrat łaciński

Uzupełnij poniższy kwadrat tak, aby każda figura występowała dokładnie raz w każdym wierszu i w każdej kolumnie – jest to tak zwany kwadrat łaciński.


✓ ↻ 🔑

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 ...

Sudoku ze zwierzętami

Rozwiąż poniższe Sudoku – w każdym wierszu, w każdej kolumnie i w każdym zaznaczonym kwadracie 2 x 2 powinny się znaleźć różne zwierzęta.


✓ ↻ 🔑

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 ...

Sudoku III

Rozwiąż następujące Sudoku – w każdym wierszu, w każdej kolumnie i w każdym zaznaczonym kwadracie 3 x 3 powinny się znaleźć wszystkie liczby od 1 do 9 i to tylko raz.

2		4	1	5		8	9	3
9	1		6	8	4	7	2	5
7	8	9	2	3	1			4
5	8		7	1	9		3	6
1	4	6	2	3		5	7	9
	7	9	5	4	6		1	8
8	2		3	6	5		4	1
4	3	1	8	9		6	5	7
6	9		4	7	1	3		2

✓ ↻ 🔑

## Concepts:

- abstraction – no matter what we arrange: fruits, animals, figures, numbers, **the rules are important**
- decomposition – into rows, columns, squares
- algorithm – the order of steps
- generalisations – more involved tasks

# Algorithmics and programming unplugged, 1997

Contents of the book *Pyramids...* Each chapter is characterized by **CS topics** it deals with and **CT tools** applied in solving the related problems.

May help teachers in developing PCK.

Chapters	CS topics, CT tools
Add a pinch of salt to taste – are recipes algorithms	CS topics: precision of algorithmic steps CT tools: approximation, uniqueness, cook versus computer
How the pyramids were built	CS topics: calculations CT tools: algorithm
Social games	CS topics: who is the idol? leader election. CT tools: reduction by elimination
The efficiency of Russian peasants in multiplication – how to simplify your life	CS topics: binary system, fast multiplication CT tools: multiplication by decomposition
Recursion – how to use what we know, how to "dump the work" to a computer	CS topics: generating consecutive digits of a number CT tools: recursion, positional representation of numbers
Fibonacci numbers – how to be perfect	CS topics: Fibonacci numbers in science CT tools: recursive thinking, fast calculations
Filling vessels using the Euclid algorithm	CS topics: Euclid algorithm CT tools: geometric interpretation, diophantine equation
Prime numbers and composite numbers	CS topics: prime and composite numbers CT tools: algorithm, testing whether a number is prime
Clock arithmetic – benefits of residuals	CS topics: modular arithmetic CT tools: fast calculations on large numbers
Searching in ordered and unordered sets – about the benefits of taking care of order	CS topics: searching in ordered sets CT tools: binary search, divide and conquer
Finding stable relationships – dancing couples, marriages	CS topics: stable matching CT tools: greedy strategy
Do we always gain from greediness?	CS topics: the change making problem, leaving the maze CT tools: greedy algorithm
Small trees – fast vending machines and short codes	CS topics: Huffman compression, fast vending machines CT tools: greedy approach, trees
Backtracking search	CS topics: the queens problem, leaving the maze CT tools: backtracking, brute force
Dynamic programming	CS topics: dynamic programming CT tools: optimization by dynamic programming





# Algorithmic thinking

Unplugged as a part of **algorithmic thinking, problem solving, computational thinking**:

- Define a problem situation, including data, the goal and the results.
- Formulate a plan for solving the problem – separate sub-problems and indicate connections between them.
- Choose a way to solve the problem:
  - design an algorithm.
  - use an existing program or program a solution method in a selected programming language.
- Analyze the correctness of the algorithm and its implementation, and assess its complexity, test the program.
- Complex projects solve in a team.
- Choose and solve problems from various school subjects.

Thank you for your attention

