EDUCATION FOR DEMOCRATIC CITIZENSHIP: THEORY AND TEACHING PRACTICE

Session 7/8 (3rd): Education for Democratic Citizenship and Human Rights Education: Cross-curricular implementation

Broadening participation in Informatics

Agoritsa Gogoulou

Given the ubiquity of the computing field in society, in a variety of scientific and work fields and facets of everyday life, it is of growing importance that all students have the opportunity to learn informatics. However, reports reveal the underrepresentation of women and racial and ethnic minorities in computer science (CS) courses (Bottela et al., 2019; Google LLC & Gallup, Inc., 2016). In many countries the term "digital divide" is used to denote the difference in access to computers and the internet between different demographic groups. Many of these divisions are based on a person's income, race, and whether they live in urban or rural areas. As far as the gender gap in science and technology disciplines is concerned, this can be located before the age of 15 years and tend to persist, throughout the secondary and postsecondary years constituting a serious issue to fundamental notions of fairness and equity.

Students are generally unconvinced that CS is important for them to learn. Female students are particularly skeptical about the merits of computer science education and remain less likely to express interest than male students do in both learning computer science and in pursuing careers in the field (Google LLC & Gallup, Inc., 2020). Teachers are on the front lines of society's efforts to engage students, to create learning experiences that convey Information and Communications Technology (ICT) affordances and promote equality of opportunity. It is important that all students have the opportunity to learn ICT skills during their primary and secondary education, as these skills become increasingly important in many areas of life. Digital literacy skills not only make it possible for students to engage, create and innovate in an increasingly technology-fueled society but they also prepare them for evolving computing occupations. The Economist Intelligence Unit (EIU) report entitled "Driving the skills agenda: Preparing students for the future", stresses that problem solving is the most important skill for students' future. High value is also given to team work and communication. Activities should encourage students to identify a problem and generate potential solutions through discussion and evaluation.

Computational Thinking

A core function of school education is preparing students to be capable and informed citizens. Having a digitally competent population is significant in order to confront pressing global issues such as climate change, resource depletion, and inequality around the world. Understanding of CS and the nature of computation has become a necessity. Douglas Rushkoff in 2010, in his influential book "Program or be Programmed" mentions

> "*When human beings acquired language, we learned not just how to listen but how to speak. When we gained literacy, we learned not just how to read but how to write, and as we move into an increasingly digital reality, we must learn not just how to use*

Co-funded by the Erasmus+ Programme of the European Union

ΙΔΡΥΜΑ ΚΡΑΤΙΚΩΝ ΥΠΟΤΡΟΦΙΩΝ IKY

STEP UP-DC Students' Practices for Democratic Culture

programs but how to make them. …. We teach kids how to use software to write, but not how to write software. This means they have access to the capabilities given to them by others, but not the power to determine the value-creating capabilities of these technologies for themselves."

There is a growing recognition that to live and function in a digital society, to solve problems and to be creative and innovative, individuals need to develop Computational Thinking (CT) (Namukasa et al., 2015). CT is "an approach to solving problems, designing systems and understanding human behaviour that draws on concepts fundamental to computing" (Wing, 2006). The term "Computational Thinking" was first introduced in 1980, by Computer Scientist and Educator, Seymour Papert, in his pioneering book "Mindstorms. Children, Computers, and Powerful Ideas" (page 4):

"In this book I discuss ways in which the computer presence could contribute to mental processes not only instrumentally but in more essential, conceptual ways, influencing how people think even when they are far removed from physical contact with a computer."

Computational thinking is a structured way to solve problems. As described by Jeannette Wing, (2006), computational thinking has the following features:

Conceptualizing, not programming: computer science is not computer programming. Coding is simply one expression of computer science concepts and problems.

Fundamental, not a rote skill: it's a fundamental skill as every person needs to know (or should know) how to solve problem to participate in society. On the other hand, computers support rote tasks and computer-based solution designed by persons.

A way that humans, not computers, think: it's a way human beings think about the world and its problems.

Complements and combines mathematical and engineering thinking: Computational thinking includes math and engineering. It's not a subset of either discipline. Computer scientists leverage math and engineering to develop solutions that may go beyond the limits of either way of thinking.

Ideas, not artifacts: computational thinking is not about output, it's the ideas that lead to creation.

For everyone, everywhere: computational thinking is available to all people, whether they use technology or not, whether their solutions require technology or not.

Computing at School organization outline their view for CT in school in the guide "Computational Thinking: a guide for teachers" as follows (Csizmadia et al., 2015):

Computational thinking is a cognitive or thought process involving logical reasoning by which problems are solved and artefacts, procedures and systems are better understood. It involves:

- the ability to think algorithmically
- the ability to think in terms of decomposition
- the ability to think in generalizations, identifying and making use of patterns
- the ability to think in abstractions, choosing good representations

- the ability to think in terms of evaluation

The main arguments for teaching CT topics are (Duncan & Bell, 2015):

- Preparing students for future endeavours in computing
- Giving students the confidence and capacity to be more than just users of digital technology
- Increased diversity in the computing field
- Facing any stereotypes of computing
- Encouraging early development of general CT skills

Learning computer science (CS) helps people better understand technology-enabled world. It provides students with skills that are broadly applicable and positions them for high-demand jobs. Computer-based problem solving learning experiences prepare students for success across sectors, equip them will core life skills, and provide a foundation for citizenship in today's world. Aheer and her colleagues (2020) developed a cross-institution activity as part of an Internationalization at Home (IaH) initiative to expose first year computer science students to the concept of computing for social good in an international context. The aim was to explore how differences in culture can influence students' perceptions and approaches to computing for social good. They were introduced to the use of computing for social good and how to better understand another culture. The analysis revealed that the students had much more in common with each other than they had differences. The majority felt they were more similar to their peers of the other culture than they were different. Despite their cultural difference, students of different cultures can have similar ideas about computing for social good.

CT has to do with problem solving, on the basis of computer based programs. When dealing with problem solving in a computer-based environment, the following dispositions are present and enhanced:

- The ability to communicate and work with others to achieve a common goal or solution
- The ability to think of various problems, to deal with open ended problem, to formulate problems in a way that enables the use of a computer and digital devices to solve them
- Confidence in dealing with complexity and in investigating various solutions
- Tolerance for ambiguity and ability in decision making

Some effort has been made to establish operationalizable frameworks for CT. These frameworks focus on the constructs that emerge from existing game-based and media-narrative programming environments. For example, Brennan and Resnick's (2012) defined a framework of CT which categorizes CT into three dimensions taking into account programming in SCRATCH:

- The *concepts* designers engage with in programming, i.e. sequences, loops, parallelism, events, conditionals, operators and data
- The *practices* designers develop as they engage with the concepts, i.e. forming iterations, testing and debugging, reusing and remixing, abstracting and modularizing
- The *perspectives* designers form about the world around them and about themselves, i.e. connecting modules/parts, setting new goals, questioning.

In a more general sense, these three dimensions include

- *computational concepts*: the fundamental concepts students engage with as they program or engage in CT oriented practices-such as algorithmic thinking, decomposition, abstraction, parallelism, and pattern generalization
- *computational practices*: the actual practices students develop as they encounter and engage with the concepts; this includes collecting and sorting data, designing and remixing computational models, debugging simulations, documenting one's work, and collaboratively breaking down complex problems
- *computational perspectives*: the perspectives students form about the world around them and about themselves as they comprehend these concepts and engage in such practices; sense how systems function, how they can be improved

It is considered necessary to teach CT outside the area of computer science and start relevant activities even from the age of kindergarten (Fessakis, Gouli, & Mavroudi, 2013). Teachers may provide both interactive and independent activities to gently challenge children and help awaken their sense of self-expression, confidence and creativity. Much of the content in preschool and kindergarten is taught with hands-on manipulatives, games, and songs, and with thoughtful planning, young children can engage in such activities and develop computational thinking in age-appropriate ways. For example, at this young age, activities can begin to scaffold an understanding of algorithms, sequencing, events, conditionals, and repeat loops.

The skills, attitudes, and approaches that make up CT are fundamental, universal, transferrable, and particularly appropriate and useful for the computer age. CT is a combination of disciplined mental habits, attitudes of endurance, and essential soft skills. Learning computational thinking can benefit students both economically and academically.

Computer Science (CS) Unplugged

The term CS Unplugged was introduced by Tim Bell, Mike Fellows and Ian Witten at the University of Canterbury in New Zealand, when they shared an online book, called "Computer Science Unplugged: Off-line activities and games for all ages", aiming to attract students to study computer science in high school and post-secondary education and to enhance subject knowledge. Unplugged activities can be implemented without the use of computers. CS Unplugged activities are kinesthetic, engaging, and above all emphasize that computer science is about problem solving, and not synonymous with programming. They also make "abstract concepts both tangible and visible". Unplugged activities can be a powerful way to introduce students to computing concepts, to help them improve their problem solving skills, to make them collaborate and express their thoughts about computer issues, to increase students' interest and motivation for the CS field (Namukasa, 2015). Unplugged activities present fundamental concepts of Computer Science such as algorithms, artificial intelligence, graphics, information theory, human computer interfaces, programming languages, and so on.

The unplugged approach differs from Papert's constructionism as Papert recognizes programming both in physical and computer-based environment, the major factor for constructing knowledge. However, both approaches consider concrete activities that aim to teach complex CS ideas to children. CS concepts are not simplified, but instead made accessible with practical experiences. Also, unplugged activities generally have children

using their bodies or the physical manipulation of objects to perform various functions and come to a conclusion (Bell & Lodi, 2019).

By using unplugged activities, younger learners, girls and novices feel more comfortable, they may change their attitude and approach computers in order to learn programming or other tools and concepts.

In CS Unplugged there are many links to Computational Thinking. Teaching Computational Thinking through CS Unplugged activities, the students learn how to:

- describe a problem
- identify the important details needed to solve this problem
- break the problem down into small, logical steps
- use these steps to create a process (algorithm) that solves the problem
- evaluate the solution

These skills are transferable to other subject matters area, but are particularly relevant to developing digital systems and solving problems using the capabilities of computers.

The CS unplugged activities have been successfully used in classrooms and outreach programs. Children seem to love the fact that they can get involved and try the activities themselves. Carmichael (2008) used the activities as part of a camp designed to get girls interested in Computer Science. She found that the activities increased students understanding of the concept and of its relation to Computer Science. Carmichael noted that "after the CS Unplugged activity for this topic was finished, they seemed to feel more comfortable with the connection, particularly because of the discussion included in the activity." Renate Thies and Jan Vahrenhold, from the Technical University of Dortmund in Germany, investigated the suitability of CS Unplugged activities for use in a classroom. They used CS Unplugged activities to teach half the students, and used alternative tools for the other half of the students. Their findings showed CS Unplugged activities were equally effective in transferring knowledge as there was no significant difference in achievement between the group who learned with CS Unplugged activities and the group who learned with alternative materials. Additionally, the researchers studied the impact of using CS Unplugged activities in different grade levels, and found that the activities had a significant positive impact when used with middle school classes (Rodriguez, Rader & Camp, 2016).

A pioneering resource for unplugged activities is Computer Science (CS) Unplugged www.csunplugged.org. Other websites include www.cs4fn.org, and teachinglondoncomputing.org, which provide resources for teachers who are looking to include unplugged computational thinking activities into their curricula. There are many "unplugged" activities that aren't necessarily based on csunplugged.org, usually called kinesthetic activities. The term "unplugged" is sometimes used to refer to the curated activities on the open-source CS Unplugged website (csunplugged.org), but in other contexts refers to any activity relating to computer science carried out away from a computer.

The main characteristics of the Unplugged activities are (Nishida et al., 2009):

No computers: computers are not used directly in the activities. This facilitates the execution of the activities in any place and CS background is not a prerequisite for participation.

Games: The activities are generally based around a game or challenge, so that children see them as play, which leads to interest, curiosity and motivation. For example, the surprise of the magic trick attracts the interest of the students, and their desire to play the role of the magician keeps their interest to understand the background knowledge.

Kinaesthetic: As physical objects are used, such as cards, papers, whiteboard, the children are engaged in kinaesthetic activities. Learning is achieved by moving around, by physically engaging the people and objects around. For example, the presence of an obstacle on the playground makes students move, discuss and think of ways to proceed.

Student and collaborative centred: Students are actively involved. The activities generally promote interaction with other students, and encourage students to discover answers by trial and error. Working with other students encourages teamwork and communication.

Low-cost and easy implementation: The activities are easy to prepare and use only inexpensive equipment, much of which can be found in a school. Most require paper and pencil, perhaps cards, string, chalk, whiteboard markers.

Story-telling context: Elements of fantasy and story-telling are used to engage students. Problems are presented as part of a story rather than as an abstract mathematical challenge. This can draw younger students into the activity, and emphasizes the value of creativity rather than dry learning.

Since CS Unplugged activities are fast to deploy and do not require expensive specialist equipment, they have also been used as the basis of connecting computer science with other disciplines. At primary school level connections with maths, physical education, literacy, creative writing, art are pursued by the teachers. Moreover, cs unplugged activities were used to connect computational thinking to physics and music (Bell & Vahrenhold, 2018). For example, sorting networks can be integrated with topics that students are exploring in other areas; they might be used to compare dates in history, words in alphabetical order, note pitches in music, or numbers written in a foreign language. Such activities provide motivation for students to repeatedly compare the values that they are learning, and to see them in various situations. At the same time, they are becoming familiar with a computational model and fundamental CS concepts.

CS Competitions

CS competitions are widely used to promote interest in Computer Science at various levels of education.

Bebras

Bebras is an international initiative aiming to promote Informatics (Computer Science, or Computing) and computational thinking among school students at all ages. The challenge is performed at schools using computers or mobile devices. Problems are designed to introduce the participants to concepts and methods that are typical to computer science. Solving a Bebras task requires reading abilities specific to the written content of a problem statement and abilities in the use of interactive artifacts.

As interest in competitions essentially depends on problems, attraction, invention, tricks, surprise should be desirable features of each problem presented to competitors. The problems have to be selected carefully (Dagienė & Futschek, 2008). In 2007, some active

members of the Bebras Organizing Committee proposed the following topics for the Bebras contests on informatics and computer literacy:

INF Information comprehension
Representation (symbolic, numerical, visual)
Coding, encryption
ALG Algorithmic thinking
Including programming aspects
USE Using computer systems
e.g. search engines, email, spread sheets, etc.
General principles, but no specific systems
STRUC Structures, patterns and arrangements
Combinatorics
Discrete structures (graphs, etc.)
PUZ Puzzles
Logical puzzles
Games (mastermind, minesweeper, etc.)
SOC ICT and Society
Social, ethical, cultural, international, legal issues

Criteria for good Bebras tasks that are used by the International Bebras Organizing Committee include: the tasks …

are independent from specific IT systems
No specific operating system, programming language or application software is taken for granted. All system specific terms must be explained within a task.

have easy understandable problem statements
A problem statement should be presented as easy as possible: easy understandable wording, easy understandable presentation of the problem (maybe use of pictures, examples, embedded in a proper story, use of a simulation or an interactive solving process), a problem statement should never be misleading.

are politically correct
Contain no gender, racial or religious stereotypes.

should be funny
Some sort of excitement or fun should be provoked by a good task or by solving the task.

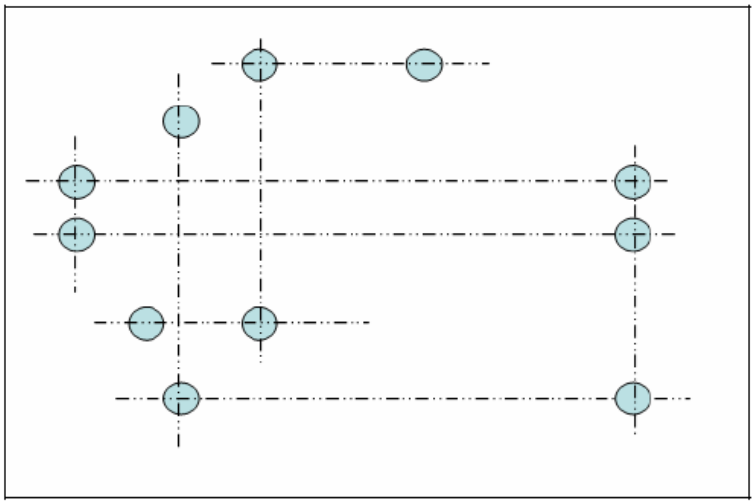
In particular, the students have to solve a series of tasks of three different difficulty levels. Each task takes between 1 to 4 minutes to be solved.

Examples of tasks

The following task is an algorithmic thinking (ALG) task, since a solution strategy incorporates strategies to find all different ways of building bridges. The possibility of interactively building bridges that are counted automatically and that can also be reset allows a sort of game-based learning.

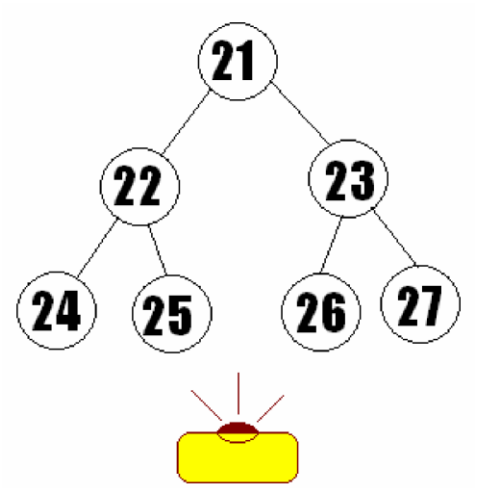
Beaver discovered a number of islands in a lake and decided to build bridges to connect them. While building bridges Beaver follows the rules: the bridges

must be built keeping the directions East-West and North-South and they shouldn't overlap each other. Help Beaver to build as many bridges as possible. Use the mouse to connect pairs of islands.



The following task focuses on the concept of "Heap". The task provides a suitable story that makes the understanding of a "Heap" more intuitively and more easily.

To make a group photo of 7 beavers it is necessary that the smaller beavers stand in front and the larger beavers in back. Unfortunately the beavers stand in a wrong order. In the graphics below those beavers are connected by a line where the back beaver should be larger than the front beaver. The only operation to rearrange beavers you can do is exchanging any two beavers of the group.
What is the minimum number of exchange-operations, that after all, the beavers are ready for taking picture?
Please perform a minimum number of exchange-operations by clicking on pairs of beavers.



Co-funded by the Erasmus+ Programme of the European Union

ΙΔΡΥΜΑ ΚΡΑΤΙΚΩΝ ΥΠΟΤΡΟΦΙΩΝ IKY

STEP UP-DC Students' Practices for Democratic Culture

The Hour of Code (HoC)

HoC was launched in 2013 as an activity planned in the Computer Science Education Week, in collaboration with big names of the software industry (Microsoft, Google, Apple, Bill Gates, Mark Zuckerberg,. . .). Code.org is the organising body of the Hour of Code. It was founded in 2012 and its vision is every student in every school should have the opportunity to learn computer science, increase the participation of women and other underrepresented students to computer science. Code.org aims to bring computer programming into the mainstream dialogue and raise national awareness. Although the main HoC offer is based on an online activity, it does exist also in an "unplugged" version. Surprisingly, the unplugged alternative is rather different from the interactive one as it proposes different tasks focusing more on methodological issues than on coding.

The "Hour of Code" activities are game-based. Students can learn computer science basics by playing a game. The "Hour of Code" tutorials teach students how to utilize problem-solving skills and logic to win the games. The "Hour of Code" tutorials are on-line, web-based, and work on computers or mobile devices. The "Hour of Code" tutorials are designed for all ages and are available in over 45 languages (Du, Wimmer, & Rada, 2018). Many online computer programming portals, including Tynker, Lightbot, Codecademy, Scratch, and Khan Academy, provide free online tutorials to teach students (ages 4–104) basic programming concepts in one hour. There are options for every age and experience level, from kindergarten and up. Also, the HoC activities are self-guided.

The Hour of Code is only the first step for students to learn that computer science is fun and creative. The findings of this study show that the participants became much more interested in programming after they tried the tutorials and expressed an interest to know more about coding. Millions of the participating teachers and students have decided to go beyond one hour - to learn for a whole day or a whole week or longer, and many students have decided to enroll in an entire course (or even a college major) as a result.

The HoC activities nurture problem-solving skills, logic, and creativity. The HoC creates an opportunity for every student (boys and girls) of all backgrounds to try computer science together worldwide (Wilson, 2014). Code.org regularly produces new materials and programs to improve diversity in computer science.

References

Aheer, K., Bauer, K., & Macdonell, C. (2020). Internationalizing the Student Experience Through Computing for Social Good. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 434-440).

Bell T. & Vahrenhold J. (2018). CS Unplugged—How Is It Used, and Does It Work?. In: Böckenhauer HJ., Komm D., Unger W. (eds) *Adventures Between Lower Bounds and Higher Altitudes. Lecture Notes in Computer Science*, vol 11011. Springer, Cham. https://doi.org/10.1007/978-3-319-98355-4_29

Bell, T. & Lodi, M. (2019). Constructing Computational Thinking Without Using Computers. *Constructivist foundations, Vrije Universiteit Brussel, Special Issue "Constructionism and Computational Thinking"*, 14 (3), 342-351.

Botella, C., Rueda, S., López-Iñesta, E., & Marzal, P. (2019). Gender Diversity in STEM Disciplines: A Multiple Factor Problem. *Entropy*, *21*(1), 30. MDPI AG. Retrieved from http://dx.doi.org/10.3390/e21010030

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association* (pp. 1-25). Vancouver, Canada.

Carmichael, G. (2008). Girls, Computer Science, and games. *SIGCSE Bull.*, 40(4), 107–110.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C. & Woollard, J. (2015). Computational thinking A guide for teachers. Technical report. Computing at Schools, 2015.

Dagienė, V., & Futschek, G. (2008). Bebras international contest on informatics and computer literacy: Criteria for good tasks. In *International conference on informatics in secondary schools-evolution and perspectives* (pp. 19-30). Springer, Berlin, Heidelberg.

Dee, T. & Gershenson, S. (2017). Unconscious Bias in the Classroom: Evidence and Opportunities. Mountain View, CA: Google Inc. Retrieved from https://goo.gl/O6Btqi.

Du, J., Wimmer, H. & Rada, R. (2018). "Hour of Code": A Case Study. *Information Systems Education Journal*, *16*(1), 51.

Duncan, C. & Bell, T. (2015). A Pilot Computer Science and Programming Course for Primary School Students. In *Proceedings of the 10th Workshop in Primary and Secondary Computing Education - WiPSCE '15*, ACM, pp. 39-48.

Fessakis, G., Gouli, E., & Mavroudi, E.(2013). Problem solving by 5–6 years old kindergarten childrenin a computer programming environment: A case study. Computers & Education, 63, 87–97.

Google LLC & Gallup, Inc. (2016). Diversity Gaps in Computer Science: Exploring the Underrepresentation of Girls, Blacks and Hispanics. Retrieved from https://news.gallup.com/reports/196331/diversity-gaps-computer-science.aspx

Google LLC & Gallup, Inc. (2020). Current Perspectives and Continuing Challenges in Computer Science Education in U.S. K-12 Schools. Retrieved from https://services.google.com/fh/files/misc/computer-science-education-in-us-k12schools-2020-report.pdf

Namukasa, I. K., Kotsopoulos, D., Floyd, L., Weber, J., Kafai, Y. B., Khan, S., et al. (2015). From computational thinking to computational participation: Towards achieving excellence through coding in elementary schools. In G. Gadanidis (Ed.), *Math + coding symposium*. London: Western University.

National center for women & Information Technology – NCWIT (2018). Computer Science Is for Everyone: A toolkit for middle and high schools to increase diversity in computer science education. Retrieved from https://www.ncwit.org/resources/computer-science-everyone-toolkit-middle-and-high-schools-increase-diversity-computer

Nishida, T., Kanemune, S., Idosaka, Y., Namiki, M., Bell, T., & Kuno, Y. (2009). A CS unplugged design pattern. *ACM SIGCSE Bulletin*, *41*(1), 231-235.

Rodriguez, B., Rader, C., & Camp, T. (2016). Using student performance to assess CS unplugged activities in a classroom environment. In *proceedings of the 2016 ACM conference on innovation and technology in computer science education* (pp. 95-100).

Wilson, C. (2014). Hour of code: we can solve the diversity problem in computer science. *ACM Inroads*, *5*(4), 22-22.

Co-funded by the Erasmus+ Programme of the European Union