UDC 37.015.311

COMPUTATIONAL THINKING AS A PEDAGOGICAL TOOL FOR UKRAINIAN STUDENTS

Joseph C. Kush
Ph. D., Professor
Duquesne University
Ukraine Fulbright Scholar
Pittsburgh, Pennsylvania, the USA
kush@duq.edu

Abstract. The article focuses on the importance of developing computational thinking of Ukrainian students and the ways computational thinking can be integrated into the curricula of higher educational institutions. The incorporation of enhanced pedagogy into the educational curricula is particularly needed in countries such as Ukraine which is currently undergoing strategic modernization and reform. The author defines the concept of computational thinking as it is used in constructivism learning theory and social-constructivism theories. It is emphasized in the article that computational thinking reduces complex problems into smaller and more manageable problems, which make it easier to solve either using a computer or without technology. A wide range of researches conducted by the scientists all over the world are presented.

The aim of the article is to analyse the implementation of computational thinking as a pedagogical tool for Ukrainian educational system.

The special attention is paid to developing computational thinking of children at an early age. Pattern Recognition is considered to be one component of computational thinking and can be used to teach the process of searching for trends, similarities, differences, or regularities.

The computational thinking instructions can be carried out in the area of the computer science or outside of computer science. Students who learn computer-programming skills as part of a have been shown to experience numerous benefits to their education because computational thinking is a problem-solving skill for all disciplines that can be taught through the integration into a particular content area or alternatively by teaching it as a stand-alone content area.

The author recommends Ukrainian educators to consider the integration of *unplugged* CT activities into their lesson plans. Unplugged curricular activities are implemented without the use of computers and are typically considered as an important first step toward comprehensive CT integration. Unplugged experiences are claimed to serve as a foundational in learning CT because they typically require the least amount of technical knowledge.

Key words: computational thinking; logic-based problem solving; STEM; understanding algorithms.

Problem setting in general. Science, Technology, Engineering, and Mathematics (STEM) jobs have demonstrated unprecedented growth across the globe (Langdon et al., 2011). Within the United States, STEM jobs are growing three times as fast as non-STEM jobs, and graduates working in STEM fields are also experiencing lower rates of unemployment. In a global market, STEM jobs increase innovation and are considered by many to be the most attractive jobs of the future (Langdon et al., 2011).

The latest papers and publications on the problem. Computational thinking (CT) is a set of pedagogical techniques that have been shown to benefit students across STEM areas, independent of their interest or background in technology (Grover & Pea, 2013; Honey, Pearson, & Schweingruber, 2014). The adoption of this pedagogy continues to grow worldwide and was recently mandated in the UK school curriculum. Computational thinking (CT) is a cognitive approach that teaches students to solve problems by incorporating processes that break down problems into a series of steps using a structured and systematic process. Contemporary, constructivism learning theory (Papert 1980, 1987; Papert & Harel 1991) and social-constructivism theories (Vygotsky 1978) underly the concepts of CT (Bers, 2008, Brennan, & Resnick, 2012; Stetsenko, & Arievitch, 2004; Wing, 2006) and the process combines problem solving and critical thinking to create new ideas, by drawing on the concepts fundamental to computer science (Wing, 2006). Despite the obvious relevance of CT to computer science, scholars argue that CT should be taught in curricular areas, outside of computer science, as early as kindergarten (Barr and Stephenson 2011; Yadav et al. 2011).

Computational thinking reduces complex problems into smaller and more manageable problems, which make it easier to solve either using a computer or without technology. The approach is not synonymous with learning a programming language and includes four main components: decomposition, abstraction, analyzation, and algorithms. As stated by Yadav, Hong, and Stephenson (2016), "computational thinking involves breaking down complex problems into more familiar/manageable sub-problems (problem decomposition), using a sequence of steps (algorithms) to solve problems, reviewing how the solution transfers to similar problems (abstraction), and finally determining if a computer can help us more efficiently solve those problems (automation)" (p. 565-566).

Thinking, playing, and learning are the occupational activities for young learners to apply in their daily lives – in school as well as outside the classroom. However, thinking, playing, and learning do not often happen in a traditional classroom (Papert, 2005). CT makes it possible for young learners to play while thinking and learning and they can learn without even realizing that learning is occurring. Unfortunately, however, many classroom teachers are not taught how to teach with these techniques (Basawapatna, Koh, Repenning, Webb, & Marshall, 2011; Ottenbreit-Leftwich, Glazewski, Newby, & Ertmer, 2010).

The incorporation of enhanced pedagogy into the educational curricula is particularly needed in countries such as Ukraine which is currently undergoing strategic modernization and reform. In 2002 the Ukraine Cabinet of Ministers put forward a reform strategy in the country's basic education system: "National Doctrine of Development of Education in Ukraine in the XXIst Century". This report indicated that the quality of education in the country was not currently being measured and indirect assessments suggested persistent problems. Further, the report indicated that other than establishing a national context for education, Ukraine has not altered the pedagogical nature of the schooling system since Soviet times. The report continues:

The curriculum is relatively theoretical, and pedagogy is skewed towards memorization of factual knowledge.

Putting aims of the article (tasks). The aim of the article is to analyse the implementation of computational thinking as a pedagogical tool for Ukrainian educational system.

Results of the study. When applied across diverse content areas, computational thinking influences how students approach and solve problems. In providing multiple ways to approach problems, computational thinking helps to ensure success for the problem-solver (Sneider, Stephenson, Schafer, & Flick, 2014). Providing students with the tools to find new or unique methods to solve problems will also strengthen students' confidence in their academic abilities. Clearly, educators should continually work towards instilling this sense of urgency within their students, (Malyn-Smith, Coulter, Denner, Lee, Stiles, & Werner, 2010).

Computational thinking allows learners to create projects including games, animated stories, online news shows, book reports, music videos, science projects, tutorials, and simulations (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010). Pattern Recognition is one component of computational thinking and can be used to teach the process of searching for trends, similarities, differences, or regularities. Pattern recognition is an incredibly powerful computational thinking tool in data analysis that can lead to abstraction and problem solving. CT curricula can utilize technological resources including tablets and computers but also incorporate numerous "unplugged" activities. Asking students to identify the best route to drive home – in normal traffic, when there is an accident, or when the roads are icy is one popular unplugged activity. This example can also be adapted to other routine activities such as selecting the fastest market check-out lane (e.g., one person with a lot of items, vs. several people with a few items, vs. a customer with a lot of children).

Children who are exposed to CT instruction at an early age have been shown to be better at problem solving, decision-making, and computational thinking skills (Flannery, Silverman, Kazakoff, Bers, Bontá, & Resnick, 2013). Additionally, children who learn CT techniques go through a similar process as children learning a second language, with these skills leading them to become increasingly fluent with new technology. Having achieved fluency, children will better be able to express themselves and start expressing new ideas.

Students who learn computer-programming skills as part of a STEM curriculum have been shown to experience numerous benefits to their education because computational thinking is a problem-solving skill for all disciplines that can be taught through the integration into a particular content area or alternatively by teaching it as a stand-alone content area (Czerkawski, 2016). For example, children may not fully grasp the purpose of why they do math, when they are focused on the process of creating formulas for their projects. Additionally, children are becoming more familiar, and sophisticated with formal systems and are interacting with themselves and doing hands-on activity by thinking (Papert, 1980, 1993). Even for children who do not secure STEM-related employment, the inclusion of STEM curriculum in education will allow students to develop literacy in Science, Technology, Engineering, and Math and the critical thinking skills that are demonstrated by scientists, mathematicians, and engineers (Honey, Pearson, & Schweingruber, 2014; Voskoglou & Buckley, 2012). CT provides young learners opportunities to create while expressing their thoughts, beliefs, and feelings in digital environments (Resnick et al., 2009; Wing, 2006; Wing, 2008, a, b).

The international interest in CT has resulted in many countries introducing CT either as a stand-alone course or through cross-curricular integration. For example, England has made the teaching of coding a part of the national curriculum in elementary schools and mandatory starting in the first grade (Berry 2013). Finland introduced a mandatory elementary CT curriculum in 2016, and Estonia has had a similar curriculum in place since 2013. The adoption of a CT curriculum in Ukraine can help to address to the internal calls for educational reform as well as better poise the country as it works toward European Union acceptance.

Conclusions of this research and prospects for further studying in this direction. As a first step, Ukrainian educators should consider the integration of *unplugged* CT activities into their lesson plans. Unplugged curricular activities are implemented without the use of computers and are typically considered as an important first step toward comprehensive CT integration. Barriers such as limited access to technological resources are potentially avoided, particularly for novices and younger students (Curzon 2013; Nishida et al. 2009). Unplugged experiences often serve as a foundational in learning CT because they typically require the least amount of technical knowledge. One purpose of unplugged experiences is to introduce preliminary and overlapping concepts related to CT that can then be explored in a more sophisticated way, either conceptually or technologically, during the other experience.

Numerous websites offer resources that can assist Ukrainian educators with unplugged CT adoption. Such activities involve logic games, cards, or physical movements that are used to represent and understand computer science concepts such as algorithms or data transmission. For example, McOwan and Curzon, (2008, 2009) provide resources for exposing students to computational thinking through magic tricks. Their activities show what algorithmic thinking is, separate from coding itself. For example, magic tricks require that students can specify the steps precisely and in the right order even though they are for a human (magician), not a computer to follow. A second activity, the game of 20 questions, can demonstrate how a divide and conquer approach leads to faster and more efficient algorithms. This activity incorporates the concept of decomposition in the divide and conquer solution and can also extend to the evaluation of performance. A final example of an unplugged CT activity is the sorting of clothing based on properties and attributes (e.g., color, body part, fabric) using a simple if-then decision tree structure. Sorting is an example of a computational algorithm whereby properties or attributes form the rule for defining an object. Further, this activity can demonstrate that there can be more than one algorithmic solution to a problem and that different algorithms can be more or less efficient. These examples are in no way meant to be exhaustive and do not highlight the powerful and feature-rich, *plugged*, tablet and internet-based, CT resources such as Scratch or Code.org. Rather, these examples are intended to stimulate exploration by Ukrainian students, parents, and educators into the stimulating world of computational thinking.

REFERENCES

1. Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads, 2(1),* 48–54.
2. Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011). Recognizing computational thinking patterns. *ACM Technical Symposium on Computer Science Education* (pp. 245–250). Dallas, TX: ACM Press.
3. Berry, M. (2013). *Computing in the national curriculum. A guide for primary teachers*. Bedford: Computing at School.
4. Bers, M. (2008). *Blocks to robots: Learning with technology in the early childhood classroom*. New York, NY: Teacher's College Press.
5. 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, Vancouver, Canada.
6. Curzon, P. (2013). *cs4fn and computational thinking unplugged*. WiPSE '13 Proceedings of the 8th Workshop in Primary and Secondary Computing Education, 47–50.
7. Czerkawski, B. C. (2016). Classroom implementations of computational thinking: Examples from education majors. *Proceedings of the World Conference on E-Learning*, USA, 151–156.
8. Flannery, L.P., Silverman, B., Kazakoff, E.R., Bers, M.U., Bontá, P., & Resnick, M. (2013). Designing Scratcher's: Support for early childhood learning through computer programming. *In Proceedings of the 12th International Conference on Interaction Design and Children* (IDC '13). ACM, New York, NY, USA, 1–10.
9. Grover, S., & Pea, R. (2013). Computational Thinking in K-12: A review of the state of the field. *Educational Researcher, 42(1),* 38–43.
10. Honey, M., Pearson, G., & Schweingruber, H. (2014). *STEM integration in K-12 education: Status, prospects, and an agenda for research*. Washington, D.C.: National Academic Press.
11. Langdon, D., McKittrick, G., Beede, D., Khan, B., & Doms, M. (2011). *STEM: Good jobs now and for the future* (ESA Issue Brief #03-11). Retrieved from U.S. Department of Commerce Economics and Statistics Administration website: http://www.esa.doc.gov/Reports/stem-good-jobs-now-and-future
12. Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The programming language and environment. *ACM Transactions on Computing Education (TOCE), 10(4),* 1–15.
13. Malyn-Smith, J., Coulter, B., Denner, J., Lee, I., Stiles, J., & Werner, L. (2010). Computational thinking in K-12: Defining the space. In D. Gibson & B. Dodge (Eds.), *Proceedings of the Society for Information Technology & Teacher Education International Conference*, USA, 3479–3484.
14. McOwan, P. W., & Curzon, P. (2008). *The Magic of Computer Science*. Queen Mary, University of London.
15. McOwan, P. W., Curzon, P., & Black, J. (2009). *The Magic of Computer Science II: Now we have your attention*. Queen Mary, University of London, 2009.

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

17. Ottenbreit-Leftwich, A. T., Glazewski, K. D., Newby, T. J., & Ertmer, P. A. (2010). Teacher value beliefs associated with using technology: Addressing professional and student needs, *Computers & Education*, *55(3),* 1321–1335

18. Papert, S. (2005). Teaching children thinking. *Contemporary issues in technology and teacher education*, *5(3),* 353–365.

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

20. Papert, S. (1987). *Constructionism: A new opportunity for elementary science education.* Retrieved August 1, 2016. Retrieved from http://nsf.gov/awardsearch/showAward?AWD_ID=8751190.

21. Papert, S., & Harel, I. (1991). *Constructionism*: New York: Ablex Publishing.

22. Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Computational thinking in high school science classrooms. *The Science Teacher, 81*(5), 53–59.

23. Stetsenko, A., & Arievitch, I. M. (2004). Vygotskian collaborative project of social transformation: History, politics, and practice in knowledge construction. *The International Journal of Critical Psychology*, 12 (4), 58–80.

24. Voskoglou, M. G., & Buckley, S. (2012). Problem solving and computational thinking in a learning environment. *Egyptian Computer Science Journal*, *36(4),* 28–45.

25. Vygotsky, L. (1978). Interaction between learning and development. *Readings on the development of children*, *23(3),* 34–41.

26. Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49(3),* 33–35.

27. Wing, J. M. (2008a). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society, 366,* 3717–3725.

28. Wing, J. M. (2008b). Five deep questions in computing. *Communications of the ACM, 51(1),* 58–60.

29. Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *Tech Trends, 60,* 565–568. doi: 10.1007/s11528-016-0087-7

30. Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). Introducing computational thinking in education courses. *SIGCSE, 11*, 465–470.

КОМП'ЮТАЦІЙНЕ МИСЛЕННЯ ЯК ПЕДАГОГІЧНИЙ ІНСТРУМЕНТ УКРАЇНСЬКИХ СТУДЕНТІВ

Джозеф Куш
Доктор філософії, професор
Університет Дюкейн
м. Пітсбург, штат Пенсільванія, США
kush@duq.edu

Аннотація. Стаття висвітлює актуальність розвитку комп'ютаційного мислення студентів, а також шляхи включення проблеми розвитку комп'ютаційного мислення до навчальних програм дисциплін. Особливої актуальності проблема розвитку комп'ютаційного мислення набуває сьогодні в Україні, яка стоїть на шляху стратегічної модернізації та реформації системи освіти.

У статті визначено зміст поняття «комп'ютаційне мислення» в контексті конструктивістської теорії навчання і соціо-конструктивістської теорії. Автором проаналізовано цілу низку досліджень науковців різних країн світу щодо шляхів розвитку комп'ютаційного мислення.

Автор надає рекомендації українським педагогам використовувати на уроках такі види діяльності для розвитку комп'ютаційного мислення, які не передбачають використання комп'ютера, планшета або смартфона, так звані «*роз'єднані*» завдання. Ефективність таких завдань автор вбачає у зменшенні необхідності формувати вміння працювати з сучасною комп'ютерною технікою.

Ключові слова: комп'ютаційне мислення; логічне вирішення проблем; STEM; розуміння алгоритму.

Матеріали надійшли до редакції 26.02.2019 р.