Sprego as an introductory language to programming

Gábor Csapó[a]

[a]University of Debrecen, Doctoral School of Informatics
csapo.gabor@inf.unideb.hu

Abstract

Teaching programming primarily focuses on imperative, object-oriented, or block programming languages to cover the requirements of the curricula. Most students find text-based languages difficult to learn, and the educational environments childish, or in some cases even useless. Furthermore, research and experience show that students who complete secondary education do not possess the required level of algorithmic and computational thinking skills for effective problem-solving. Considering these, we are convinced we need to rethink how programming should be taught, especially to beginners. Besides programming, data-management is also a mandatory topic in ICT education, which is traditionally taught with surface approaches, focusing on the interfaces and features of applications without considering the underlying algorithms involved.

In contrast with traditional approaches, Sprego (Spreadsheet Lego) is a schema-centric methodology for teaching both data-management and programming, by applying Pólya's concept based problem-solving method. In order to further ease the cognitive load required by these topics – the data-analyses and the building of algorithms –, Sprego is supported with various unplugged and semi-unplugged tools, while coding is carried out on user-friendly spreadsheet interfaces. To motivate students and make them solve real-world problems, authentic sources are collected and presented in classes, in accordance with the students' age, background knowledge, and subjects of interest. With this unique approach to programming, the Sprego methodology provides an opportunity to build up long-lasting knowledge and to develop algorithmic and computational thinking skills. In the present paper, our goal is to demonstrate how Sprego can be utilized as an introductory programming environment and what the advantages are compared to traditional methods.

Keywords: Sprego, functional programming languages, end-user computing

1. Introduction

Computational thinking is considered the fourth basic skill [1], along with the 3Rs (reading, writing, arithmetic). In the educational context, when talking about the development of computational skills, teachers usually refer to the programming topic of ICT (Information and Communications Technology) education. Teaching programming holds a "special place" in ICT education, as it is overemphasized by IT professionals as well as educators, implying that it is the only topic where "serious" computer science happens.

1.1. Programming education

Schools offer a wide range of programming tools to teach the fundamentals and requirements of programming stated in the curricula [2] [3]. These tools range from traditional text-based imperative, object-oriented languages to various forms of visual programming and educational

programming languages (EPLs). The most popular of these languages are C, C#, C++, Java, Pascal, Python, Logo [4], Scratch [5], Alice [6], Lego Mindstorms [7], etc..
Despite the advantages of EPL tools and interfaces, previous research has pointed out that students studying with them could not perform better than students who studied with imperative and object-oriented languages [8] [9] [10] [11].
Notwithstanding, regardless of the tools used, research shows that students who complete secondary education do not possess the level of algorithmic and computational thinking skills on which they can rely effectively in tertiary education [12] [13]. In general, students find the text-based languages difficult to learn and the applications they create and run in the command-line window unmotivating. In contrast, using various EPLs usually results in visually more engaging student productions that stimulate multiple sensory organs [14]. However, we must note that besides this advantage, students usually find these environments childish, in some cases even useless, which means they have a less serious approach to the topic.

1.2. Computer problem-solving

Based on the ACM&IEEE report [15] we can distinguish three knowledge levels:

- Familiarity: "The student understands what a concept is or what it means. This level of mastery concerns a basic awareness of a concept as opposed to expecting real facility with its application. It provides an answer to the question 'What do you know about this?'."
- Usage: "The student is able to use or apply a concept in a concrete way. Using a concept may include, for example, appropriately using a specific concept in a program, using a particular proof technique, or performing a particular analysis. It provides an answer to the question 'What do you know how to do?'."
- Assessment: "The student is able to consider a concept from multiple viewpoints and/or justify the selection of a particular approach to solve a problem. This level of mastery implies more than using a concept; it involves the ability to select an appropriate approach from understood alternatives. It provides an answer to the question 'Why would you do that?'."

These levels show similarities with the concept based problem-solving approach developed by Pólya [16]. However, while Pólya defined four steps (understanding the problem, devising a plan, carrying out the plan and looking back), the ACM&IEEE report merges the first two steps [17]. Currently, ICT education, students, and end-users in general primarily focus on the second of the knowledge levels of the ACM&IEEE report; usage. This practice ignores the first level and makes the third inaccessible. This leads to the conclusion that one of the main problems in ICT education is that it focuses on the interfaces and features, without considering the underlying algorithms involved in the process of computer problem-solving.

1.3. Our goals

In our methodology – Sprego (Spreadsheet Lego) – we take a different approach, both to teaching programming and end-user computing; our approach provides an opportunity to teach programming principles through data-handling, primarily focusing on spreadsheet programs. This novel approach requires the introduction of concept based problem-solving methods with schema-construction to utilize both slow and fast thinking in the most effective ways [18].

2. Sprego

2.1. Introduction to Sprego

Sprego is a methodology to teach data-management in a simplified functional language with a schemata-centric, high-mathability approach [19] [20] [21] [22] [23] [24] [25] [26] [27]. It is based on Pólya's [16] four-step problem-solving approach.
In contrast with traditional approaches in spreadsheeting, where students work with hundreds of problem-specific functions, Sprego relies on a small set of general-purpose spreadsheet functions, entitled Sprego functions [23]. Students analyze the problems, construct their algorithms, then – strictly after these two steps – carry out the coding, and finally do the discussion and debugging [21] [28] [29] [23] [30] [31] [32] [33] [34] [35]. With this approach, the problem-specific functions can be replaced with algorithms, programming, and schemata. The methodology puts a great emphasis on using authentic sources. In contrast with the decontextualized sources presented in textbooks and in online helps [36] [37], authentic sources provide real-life data from contexts that students are familiar with [38]. To support the understanding of the problems, algorithms, multilevel formulas, data-analyses, and the recalling of schemata, Sprego uses various unplugged and semi-unplugged tools [28] [29] [39] [40] [41].
The Sprego methodology can be used without age and platform restrictions, being compatible with the popular spreadsheet management environments in the industry. Through its approach to problem-solving, the methodology builds up long-lasting knowledge and develops students' algorithmic skills and computational thinking [42] [43].

2.2. Programming in Sprego

Building and coding algorithms with Sprego relies on the functional language of spreadsheet environments [44] [45] [46] [47] [48]. Teaching programming with this approach provides an opportunity for beginners to work in a familiar, user-friendly environment combined with the moderate cognitive load of functional programming [45]. These tools make the teaching-learning process more effective, as students do not need to learn the syntax of demanding programming languages and the complicated IDE (Integrated Development Environment) interfaces [49] [50].

2.2.1. Programming concepts

Teaching programming concepts in spreadsheet environments differs from traditional approaches. However, this does not limit the introduction, discussion, and use of fundamental programming elements.
Spreadsheets handle various elementary datatypes, such as string, integer, double, date and boolean. Although the declaration of these datatypes is not explicit, the environment aids the users with automated formatting to show what type of data is stored and presented in cells:

- string: left aligned
- integer, double, date, and time: right aligned
- boolean: center aligned

Due to differences in the type of separator characters in various languages and the automated datatype-recognition of spreadsheet programs, authentic sources can result in unintended datatypes. Integers can be recognized as doubles, dates, or strings, booleans as numbers, dates and times as strings. Furthermore, multiple data stored in a single cell are frequently met. This provides further opportunities to teach more conscious use of datatypes and the conversion between them. Consequently, in the Hungarian version of Microsoft Excel [51] a comma is the

decimal character, while in English, the comma serves as the thousand-separator character and the dot as the decimal character. Both teachers and students must be aware of these syntactical rules to be able to handle datatypes correctly.
Reaching the level of abstraction required for understanding the concept and use of variables is a key element in programming education. However, based on experience, students have a hard time understanding this fundamental concept. In spreadsheet interfaces the declaration of variables and assigning values to them is as simple as storing data in cells. Furthermore, spreadsheet environments also make it possible to give custom names to cells and ranges of cells, similar to the explicit declaration of variables and arrays in text-based programming languages.
Calling functions in programming and also in spreadsheeting is based on the concept of function, officially built up in math classes. However, spreadsheet programs offer a much wider opportunity for calling functions than mathematics, by applying n-ary and multilevel formulas of various domains and ranges. This practice is in accordance with modular programming practices, where the method of calling functions is introduced at an early stage of the topic and students learn to write modular programs with pre-written, reusable elements.
The execution order of multilevel array formulas is observable through the built-in formula evaluator in spreadsheet environments; the composite formulas are evaluated, starting from the inside and working outwards. This approach adheres to the pre-defined evaluation and execution order of formula components, which is in contrast with the sequential execution of commands in imperative and object-oriented programming languages. Through the creation of multilevel formulas, students also learn how functions return values, and how other functions use the previously returned values as inputs in programming.
Forming yes/no questions in programming is used to set conditions. In traditional programming practices, the conditions are usually used either in an IF statement or in loops. Setting up conditions in spreadsheets is in accordance with this practice. In spreadsheeting, students form yes/no questions for solving conditional calculation and formatting problems. This emphasizes the evaluation of questions as a separate step which returns true/false logical values, similar to text-based programming languages.
Loops in text-based programming languages are used according to traditional programming principles and are easily identifiable. In traditional spreadsheeting, iterations – repeating actions on a range of cells by copying or array formulas – are not emphasized either by the software supports or the teaching materials. On the contrary, in Sprego the concept of iteration is widely supported by creating array formulas when the problem requires so. This approach is in accordance with one of the newest Google Sheets [52] and Microsoft Excel [51] built-in features, entitled dynamic array formulas [53].

2.2.2. Comparing knowledge items

To further highlight the opportunities for using Sprego for teaching programming, we analyzed the required knowledge items for solving a conditional counting task in both spreadsheets and text-based programming languages. The data table shown in Figure 1 is a list of world heritage sites and their connected values [54]. The task is to count the number of sites that are in a region, stored in I2 (the region in question is stored in cell I2 to avoid hard-coding in the condition).

	A	B	C	D	E	F	G	H	I
1	Year	NameOfTheProperty	Country	Type	Region	Property_ha	ID		
2	1978	Aachen Cathedral	DE	C	EUR	0	3		
3	1978	City of Quito	EC	C	LAC	320	2		
4	1978	Galápagos Islands	EC	N	LAC	14066514	1		
5	1978	Historic Centre of Kraków	PL	C	EUR	150	29		
6	1978	Island of Gorée	SN	C	AFR	0	26		
7	1978	L'Anse aux Meadows National Historic Site	CA	C	EUR	7991	4		
8	1978	Mesa Verde National Park	US	C	EUR	21043	27		
9	1978	Nahanni National Park #	CA	N	EUR	47656	24		
1201	2019	Water Management System of Augsburg	DE	C	EUR	113	1580		
1202	2019	Writing-on-Stone / Áísínaí'pi	CA	C	EUR	1106	1597		

Figure 1: The converted World Heritage table used for the conditional counting task presented in the paper [54].

Knowledge items involved in the problem-solving process are listed in Table 1. We considered the navigation in spreadsheet environments and handling programming IDEs (for example starting new projects, creating an empty program, etc.) as brought-in knowledge. Therefore, items referring to this knowledge are not listed in Table 1.

Sprego	Text-based programming
Small set of syntactic rules	Demanding syntactic rules
Declaring variables and arrays: built in	Declaring variables and arrays: explicit
Coding	Coding
General purpose functions: built-in	General purpose functions: standard libraries
Calling functions	Calling functions
Handling files: opening	Handling files: I/O functions/statements
Standard I/O: typing, automated	Standard I/O: I/O functions/statements
Conditions: yes/no questions	Conditions: yes/no questions
IF() function	IF statement
SUM() function	+ operator
Multilevel functions	Multilevel functions
Creating array formulas	Using loops with arrays
Indexing	Indexing

Table 1: Knowledge items required for solving a conditional counting task using the Sprego methodology and text-based programming approaches.

In Table 2 codes are presented for both the Sprego and text-based programming solutions of the problem introduced in this section. Note, that we chose a simplified, fictional language to represent the text-based approaches, and that the solution only slightly differs in other programming languages.

Sprego	Text-based programming

{=E2:E1202=I2}
{=IF(E2:E1202=I2,1)}
{=SUM(IF(E2:E1202=I2,1))}
	#include <stdio.h>
	int main (){
	  int count=0, i;
	  string searchregion="EUR", regions[250];
	  for(i=0;i<250;i++){
	  if(regions[i] == searchregion) count = count + 1;
	  }
	  printf("%d",count);
	}

Table 2: Example codes for solving a conditional counting task in Sprego and a text-based, simplified, fictional language.

The lists of items show that solving the problem using traditional programming approaches requires more advanced items. In contrast, Sprego works with simpler, easier to understand items, making the task easier to code and solve for beginners. Comparing the two lists of knowledge items presented in Table 1 and the two codes in Table 2 shows that introducing and solving fundamental programming problems is possible at an earlier stage in spreadsheet management.

3. Summary

In this paper, we presented Sprego as a methodology to teach programming through data-handling in spreadsheet environments. We discussed the knowledge items used in both spreadsheets and imperative, text-based languages for solving conditional counting tasks. Based on the comparison of the required knowledges items, we can conclude that Sprego coding is less demanding than traditional imperative and object-oriented languages; consequently, it is more effective to teach programming for beginners as an introductory language. Further work includes the teaching of the programming topic with Sprego in K-12 ICT education and the measurement of its effectiveness in developing the students' algorithmic and computational thinking skills compared to text-based programming languages.

4. Acknowledgements

This work was supported by the construction EFOP-3.6.3-VEKOP-16-2017-00002. The project was supported by the European Union, co-financed by the European Social Fund.

5. References

[1]. WING, J. M., Computational thinking, *Communications of the ACM,* 49(3) (2006), pp. 33–35.

[2]. OFI, National base curriculum, In Hungarian: Nemzeti alaptanterv, Oktatáskutató és Fejlesztő Intézet, Budapest (2012), http://ofi.hu/sites/default/files/attachments/mk_nat_20121.pdf. Retrieved on: 2016.10.22.

[3]. OFI, Central curriculum framework for year 9-12 students, In Hungarian: Kerettanterv a gimnáziumok 9-12. évfolyama számára. Oktatáskutató és Fejlesztő Intézet, Budapest (2013), http://kerettanterv.ofi.hu/03_melleklet_9-12/index_4_gimn.html. Retrieved on: 2016.11.09.

[4]. KÖNCZÖL, T., IMAGINE is here!, In Hungarian: Itt az IMAGINE!, http://logo.sulinet.hu/, Retrieved on: 2019.12.29.

[5]. LIFELONG KINDERGARTEN GROUP, Scratch - Imagine, Program, Share (2019), https://scratch.mit.edu. Retrieved on: 2019.12.10.

[6]. CARNEGIE MELLON UNIVERSITY, Alice - Tell Stories. Build Games. Learn to Program (2019), https://www.alice.org. Retrieved on: 2019.12.11.

[7]. LEGO, Home - LEGO.com (2019), https://www.lego.com/en-gb/mindstorms?ignorereferer=true. Retrieved on: 2019.12.11.

[8]. KALELIOGLU, F., GÜLBAHAR, Y., The effects of teaching programming via Scratch on problem solving skills: a discussion from learners' perspective, *Informatics in Education*, 13(1) (2014), pp. 33–50.

[9]. MEERBAUM-SALANT, O., ARMONI, M., BEN-ARI, M., Habits of programming in scratch, *16th annual joint conference on Innovation and technology in computer science education, ACM* (2011), pp. 168–172.

[10]. FRANKLIN, D., HILL, C., DWYER, H. A., HANSEN, A. K., IVELAND, A., HARLOW, D. B., Initialization in scratch: Seeking knowledge transfer, *47th ACM Technical Symposium on Computing Science Education, ACM* (2016), pp. 217–222.

[11]. CLIBURN, D. C., Experiences with the LEGO Mindstorms throughout the undergraduate computer science curriculum, *36th Annual Frontiers in Education Conference* (2006), pp. 1–6. IEEE

[12]. BIRÓ, P., CSERNOCH, M., MÁTH, J., ABARI, K., Measuring the level of algorithmic skills at the end of secondary education in Hungary, *Procedia - Social And Behavioral Sciences*, 176 (2015), pp. 876–883.

[13]. CSERNOCH, M., BIRÓ, P., MÁTH, J., ABARI, K., Testing Algorithmic Skills in Traditional and Non-Traditional Programming Environments, *Informatics in Education*, 14(2) (2015), pp. 175–197. DOI=http://doi.org/10.15388/infedu.2015.11.

[14]. SHAMS, L., SEITZ, A. R., Benefits of multisensory learning, *Trends in cognitive sciences*, 12(11) (2008), pp. 411–417.

[15]. ACM & IEEE, Computer Science Curricula: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science, *ACM Press, and IEEE Computer Society Press*, New York (2013). DOI=http://dx.doi.org/10.1145/2534860.

[16]. PÓLYA, G., How To Solve It. A New Aspect of Mathematical Method. 2nd edition, 1957, *Princeton University Press*, Princeton, New Jersey (1954).

[17]. CSERNOCH, M., BIRÓ, P., Computational problem-solving, In Hungarian: Számítógépes problémamegoldás, *TMT, Tudományos és Műszaki Tájékoztatás, Könyvtár- és információtudományi szakfolyóirat*, 62(3) 2015, pp. 86–94.

[18]. KAHNEMAN, D., Thinking, Fast and Slow. Farrar, Straus and Giroux, New York (2011).

[19]. BARANYI, P., GILANYI, A., Mathability: emulating and enhancing human mathematical capabilities, *4th IEEE International Conference on Cognitive Infocommunications, CogInfoCom* (2013), pp. 555–558. DOI=http://doi.org/10.1109/CogInfoCom.2013.6719309. IEEE.

[20]. BIRÓ, P., CSERNOCH, M., The mathability of computer problem solving approaches, *6th IEEE International Conference on Cognitive Infocommunications, CogInfoCom* (2015), pp. 111–114. DOI=http://doi.org/10.1109/CogInfoCom.2015.7390574. IEEE.

[21]. BIRÓ, P., CSERNOCH, M., The mathability of spreadsheet tools. *6th IEEE International Conference on Cognitive Infocommunications, CogInfoCom* (2015), pp. 105–110. DOI=http://doi.org/10.1109/CogInfoCom.2015.7390573. IEEE.

[22]. CHMIELEWSKA, K., GILÁNYI, A., ŁUKASIEWICZ, A., Mathability and Mathematical Cognition, *7th IEEE International Conference on Cognitive Infocommunications, CogInfoCom* (2016). DOI=http://doi.org/10.1109/CogInfoCom.2016.7804556. IEEE.

[23]. CSERNOCH, M., Programming with Spreadsheet Functions: Sprego, In Hungarian: Programozás táblázatkezelő függvényekkel – Sprego. Műszaki Könyvkiadó, Budapest (2014).

[24]. CSERNOCH, M., Algorithms and schemata in informatics education II, In Hungarian: Algoritmusok és sémák az informatika oktatásában II (2016), http://tanarkepzes.unideb.hu/szaktarnet/kiadvanyok/algoritmusok_es_semak_2.pdf. Retrieved on: 2016.01.25.

[25]. HERMANS, F., How to teach programming (and other things)?, *Strange Loop*, St. Louis, MO (2019), https://www.youtube.com/watch?v=g1ib43q3uXQ. Retrieved on: 2019.11.22.

[26]. MERRIËNBOER, J.J.G. VAN, SWELLER, J., Cognitive Load Theory and Complex Learning: Recent Developments and Future Directions, *Educational Psychology Review*, 17(2) (2005), pp. 147–177. DOI=http://doi.org/10.1007/s10648-005-3951-0.

[27]. SKEMP, R., The Psychology of Learning Mathematics, *Lawrence Erlbaum Associatives*, New Jersey, USA (1971).

[28]. BIRÓ, P., CSERNOCH, M., Unplugged tools for building algorithms with Sprego, *International Conference on Education and New Development, END2017* (2017), pp. 401–405.

[29]. BIRÓ, P., CSERNOCH, M., Semi-unplugged tools for building algorithms with Sprego, *The Turkish Online Journal of Educational Technology*, INTE 2017 November (2017), pp. 946–957, http://tojet.net/special/2017_11_2.pdf. Retrieved on: 2017.12.11.

[30]. CSERNOCH, M., BIRÓ, P., Sprego programming, *Spreadsheets in Education (eJSiE)*, 8(1) 2014, http://epublications.bond.edu.au/cgi/viewcontent.cgi?article=1175&context=ejsie. Retrieved on: 2015.04.03.

[31]. CSERNOCH, M., BIRÓ, P., Sprego, End-user Programming in Spreadsheets, *PPIG 2017, 28th Annual Workshop*, 2017.07.01.–2017.07.03., Delft, Netherlands (2017).

[32]. CSERNOCH, M., BIRÓ, P., ABARI, K., MÁTH, J., Programming oriented spreadsheet functions, In Hungarian: Programozásorientált táblázatkezelői függvények, *XIV. Országos Neveléstudományi Konferencia*, 2014.11.06.–2014.11.08., Debrecen, Hungary (2014).

[33]. CSERNOCH, M., SIMON, K., BRÓSCH, É., KISS, É., I learned spreadsheet management with Sprego, In Hungarian: Spregoval tanultam táblázatkezelést. *INFO Éra Konferencia*, NJSZT, Zamárdi (2014).

[34]. WALKENBACH, J., WILCOX, C., Putting basic array formulas to work (2003), http://office.microsoft.com/en-us/excel-help/putting-basic-array-formulas-to-work-HA001087292.aspx?CTT=5&origin=HA001087290. Retrieved on: 2014.08.17.

[35]. WILCOX, C., WALKENBACH, J., Introducing array formulas in Excel (2003), http://office.microsoft.com/en-us/excel-help/introducing-array-formulas-in-excel-HA001087290.aspx. Retrieved on: 2014.08.22.

[36]. CSERNOCH, M., Thinking Fast and Slow in Computer Problem Solving, Journal of Software Engineering and Applications, 10(1) (2017), http://file.scirp.org/pdf/JSEA_2017012315324696.pdf. Retrieved on: 2017.07.10.

[37]. ANGELI, C., Teaching Spreadsheets: A TPCK Perspective, Improving Computer Science Education (2013), pp. 132–145. Routledge, New York and London.

[38]. CSERNOCH, M., DANI, E., Data-structure validator: an application of the HY-DE model, *8th IEEE International Conference on Cognitive Infocommunications, CogInfoCom* (2017), pp. 197–202. IEEE.

[39]. CSAPÓ, G., SEBESTYÉN, K., Educational Software for the Sprego Method, *The Turkish Online Journal of Educational Technology*, INTE 2017 October (2017), pp. 986–999, http://www.tojet.net/special/2017_10_1.pdf. Retrieved on: 2019.08.20.

[40]. GULÁCSI, Á., DIENES, N., 3D Software Environment for Educational Sprego Programming, *The Turkish Online Journal of Educational Technology*, INTE 2018 November (2) (2018), pp. 837–844, http://www.tojet.net/special/2018_12_3.pdf. Retrieved on: 2019.02.12.

[41]. SEBESTYÉN, K., CSAPÓ, G., CSERNOCH, M., Visualising Sprego Inequality Problems With 2D Representations, *The Turkish Online Journal of Educational Technology*, INTE 2018 November (2) (2018), pp. 888–898, http://www.tojet.net/special/2018_12_3.pdf. Retrieved on: 2019.08.20.

[42]. CARR, N., The shallows: What the Internet is doing to our brains, WW Norton and Company, New York (2011).

[43]. NATIONAL RESEARCH COUNCIL, How people learn: Brain, mind, experience, & school: Expanded edition, National Academies Press, Washington (2000).

[44]. SESTOFT, P., Spreadsheet technology, Version 0.12 of 2012-01-31, IT University (2011).

[45]. BOOTH, S., Learning to program: A phenomenographic perspective, Acta Universitatis Gothoburgensis, Gothenburg, Sweden (1992).

[46]. KO, A. J., ABRAHAM, R., BECKWITH, L., BLACKWELL, A., BURNETT, M. M., ERWIG, M., SCAFFIDI, C., LAWRANCE, J., LIEBERMAN, H., MYERS, B. A., ROSSON, M. B., ROTHERMEL, G., SHAW, M., WIEDENBECK, S., The State of the Art in End-User Software Engineering, *ACM Computing Surveys*, 43(3) (2011), pp. 1–44.

[47]. JONES, S. P., BLACKWELL, A., BURNETT, M., A User-Centred Approach to Functions in Excel, *International Conference on Functional Programming* (ICFP'03) (2003), Uppsala.

[48]. GARRETT, N., Textbooks for Responsible Data Analysis in Excel, *Journal of Education for Business*, 90(4) (2015), pp. 169–174. DOI=http://doi.org/10.1080/08832323.2015.1007908.

[49]. KÁTAI, Z., OSZTIÁN, E., VEKOV, G. K., Promoting computational thinking by artistically enhanced algorithm visualization, *INFODIDACT 2015 Informatika Szakmódszertani Konferencia* (2016), Zamárdi, pp. 1–12, https://people.inf.elte.hu/szlavi/InfoDidact16/Manuscripts/KZOEVGK.pdf. Retrieved on: 2019.08.20.

[50]. OSZTIÁN, E., KÁTAI, Z., VEKOV, G. K., Multi-Dimensional Expansion of Algo-Rythmics, *The Turkish Online Journal of Educational Technology*, INTE 2017 November (2017), pp. 573–578, http://www.tojet.net/special/2017_11_2.pdf. Retrieved on: 2019.02.12.

[51]. MICROSOFT, Office 365 Login | Microsoft Office (2019), https://www.office.com/. Retrieved on: 2019.12.08.

[52]. GOOGLE, Google Sheets: Free Online Spreadsheets for Personal Use, https://www.google.com/sheets/about/. Retrieved on: 2020.01.20.

[53]. MICROSOFT, Dynamic array formulas in non-dynamic aware Excel, https://support.office.com/en-us/article/dynamic-array-formulas-in-non-dynamic-aware-excel-696e164e-306b-4282-ae9d-aa88f5502fa2. Retrieved on: 2020.01.20.

[54]. UNESCO WORLD HERITAGE CENTRE, World Heritage List (2019), http://whc.unesco.org/?cid=31&mode=table. Retrieved on: 2019. 07. 23.