Croatian Journal of Education
Vol.23; No.3/2021, pages: 819-850
Original research paper
Paper submitted: 20th February 2020
Paper accepted: 28th April 2021
https://doi.org/10.15516/cje.v23i3.3934

Exploring the Achievement and Motivation of Learning a Text Programming Language among Elementary School Students in the Republic of Serbia

Dragan Rastovac[1], Milinko Mandić[1], Violeta Majski[2] and Dragan Cvetković[1]
[1]*Department of Informatics and media, Faculty of Education Sombor, University of Novi Sad*
[2]*Elementary school "Dositej Obradović", Sombor*

Abstract

The introduction of Informatics and Computing as a compulsory subject for students from the fifth to the eighth grade of elementary school, and programming as a basic content of the curriculum represents a major advancement in elementary education in the Republic of Serbia. In this study we conducted research on 58 primary school students in the sixth grade. In the 2016/17 school year, students studied the programming language Scratch, and in the school year 2017/18, they learned the Python programming language. The programming courses took place once a week (45 minutes) over 17 weeks. This study aimed to monitor the flow of learning visual and textual programming language following the new elementary school curriculum in Republic of Serbia, that is, the achievement and motivation of students to continue learning programming. The research instrument used was a questionnaire. The results of this study showed that it was easier for students to master the visual programming language Scratch than the textual programming language Python. However, the research results show that algorithmic way of thinking and motivation to learn by programming a text language are satisfactory, given that students have not had previous experience with it.

Key words: *computational thinking; elementary education; textual programming; visual programming.*

Introduction

In the last decade, there has been an expansion of information communication technology that has resulted in the emergence of new digital devices. Children of pre-school and elementary school age increasingly replace television and picture books with computers, tablets, and smartphones. Consequently, children spend more time playing games on the computer and watching video clips (which can be educational or any other content). However, these digital devices and the Internet are becoming an important learning environment for children. The computer, as one of the digital devices, has an important role in improving education (Salomon, 1985; Lenhart et al., 2001).

Wing (2006) points out that if children want to understand and actively participate in the new digital world, they need to learn "computational thinking". A new national curriculum for computing will enable children around the world to understand and acquire computer skills that they will need in the future. Based on new computer science programs, children will gain knowledge of the structure and operation of computers and develop their ideas using new technology (Berry, 2013).

The aim of the study presented in this paper is to point out the interest of elementary school students towards computer science, programming above all. Since programming can be considered an important part of computer science teaching (Schulte, 2013), it is particularly important to choose the right teaching methodology for students who meet with programming for the first ...time - novice programmers. (Gilmore, 1990; Lahtinen et al., 2005; Kelleher & Pausch, 2005; Robins, 2019). This especially refers to selecting the type of the (first) programming language, i.e., the order and priority of studying the textual and visual programming language. An important aspect of the method of programming teaching methodology is its representation in computer science courses of primary school curricula. Also, bountiful literature deals with the relationship between the algorithmic way of thinking and programming, as well as mathematical knowledge and programming skills, followed by factors which motivate students to learn programming. Therefore, the next theoretical chapter (literature review) is organized in such a way as to give an overview of the papers in relation to these aspects.

Literature review

Programming

If we want to learn programming in the traditional way, it is necessary to know the instructions of programming language, so that the computer understands us. The basic task of programming is learning the programming language command (declarative knowledge) and to enable the use of this information in different ways (procedural knowledge) (Palumbo, 1990). Palumbo (1990, acc. to Pea, 1984), expressed doubts about the statistical dependence between the instructional language and problem-solving methods. In the 1990s, people's interest in programs that use the graphical environment for programming, debugging, etc. began to appear. The emergence of the graphical environment was expected because it is well-known that traditional programming is more complex for learning because it requires a certain capability that not all people have. With the emergence of programming languages designed to allow visual programming, where there is no need to know the syntax of the program as in textual programming languages, interest in them is beginning to expand. Therefore, it is necessary to establish a graphical environment that will be easy to manage and use, and get more people interested in "graphics programming" (Halbert, 1984; Lewis & Olson, 1987; Myers, 1990).

Visual and textual programming languages in schools

One of the most popular visual programming languages is Scratch, which allows children in elementary school to start to acquire the basics of programming on their own and without fear. Scratch was designed by Mitchel Resnick from the Massachusetts Institute of Technology. It is very easy to use because the controls (blocks) are on the screen and they can be fitted with each other and obtain the proper meaning. There is no syntax code or error messages as in textual programming (Resnick et al., 2016). Moreno-León, Robles and Román-González (2015) carried out an overview of papers published in the period from 2007 to 2015 whose field of interest was programming language. The review results indicate that Scratch has often been the subject of much research. Thus, the application of integrating coding and visual blocks programming in Scratch for the 5th and 6th grades of primary school in the period of two school years was researched by Sáez-López et al. (2016). Kalelioğlu and Gülbahar (2014) examined the impact of Scratch on problem solving skills of students in the 5th grade of elementary school. Research results showed that students were interested in programming, but there were no major differences in the problem-solving skills. Also, four undergraduates majoring in computer science underwent training in game programming using Scratch at the University of Washington Bothel to be able to transfer knowledge to students (grades 6 through 8) because it is considered that there are not enough experts in that field (Gruenbaum, 2014). Kalelioğlu (2015) also examined how the code.org site influenced the problem-solving skills in students of both genders in elementary school. The results of the research again showed students' interest in programming, but it did not show differences in thinking among students of different gender. However, there is a slight difference in reflective thinking between genders (in favour of female students).

On the other hand, by learning (textual) programming, children acquire a computer-based mindset that will be helpful in further education. When they write the code, they have continuous conflict with the compiler whenever their code is to be executed. However, they acquire better educational opportunities, and the software encourages them to overcome the impression that writing the code is beyond their capabilities (Duncan, Bell, & Tanimoto, 2014; Tsukamoto et al., 2015). Also, Lopez et al. (2008) show that the skill of writing correlates with the skill of reading the code. One of the simplest textual languages is Python. Python was designed by Guido van Rossum in early 1990s It offers everything that is required from a programming language and it is simple to learn and understand. Perhaps it is enough to mention the quote from the research given in Lindstrom (2005): "My 10-year-old daughter programs her mathematics homework in Python (she just wrote a routine to convert degrees Celsius to degrees Fahrenheit)". Facilitating the application of Python's programming language in introductory courses of computer science was presented by Ranum et al. (2010).

The complexity of the structure of textual programming languages (like C), which is taught at the faculty, can cause disinterest and lack of motivation for students to continue learning programming. In order to facilitate the training of students for C language, students will introduce Scratch as an environmental support, in parallel with traditional lessons (Ozoran et al., 2012). The analysis of the shift from learning visual programming to textual programming language (C # or Java) for students (15 to 16 years old) was examined by Armoni et al. (2015). The authors concluded that the knowledge and experience in programming by students who had been taught Scratch greatly facilitates the learning of a more advanced programming language. They also had a better understanding of the subject than students who had not previously studied Scratch. Erol and Kurt (2017) analyzed the motivation and results achieved by students of the Faculty of Education in Turkey, while learning the programming languages Scratch and C# over 7 weeks. There are also other research papers dealing with comparative studies of the .achievements of children in learning a visual and textual programming language (Hromkovič et al., 2016; Mladenović et al. 2018; Noone & Mooney, 2018). Kölling et al. (2015) presented the transition from learning a visual programming language to textual in elementary school.

Development of algorithmic thinking

According to Kerner (1986), the construction of algorithms and subprograms should be an integral part of learning programming, not just a list of programming languages. Based on the Turtle Graphics Tutorial System (TGTS), the online puzzle-based learning system, Hsu and Wang (2018) believe that it will stimulate student algorithmic thinking. The importance of algorithmic thinking of elementary school-aged children as the basis of learning was recognised by Futschek and Moschitz (2011). They presented the Tim the Train learning scenario that includes material objects based on which tasks are taught in order to learn algorithmic thinking. However, the authors also showed a slight transition from objects to a virtual Scratch/ BYOB environment that makes students feel better about their first programming steps. Ko and Park (2011) point out that primary school students will perform various activities in the programming process, regardless of the outcome. Thus, during this process, problem-solving and logical thinking ability will be improved. According to Kátai (2015) properly set e-learning tools generally have an impact on the development of algorithmic thinking in both science-oriented and humanities-oriented students.

Programming and Mathematics

However, in the implementation of teaching programming, it is necessary to take into account methods, didactics and pedagogy of the related courses such as mathematics. Given the importance of studying algorithms, the subject of mathematics in schools should be directed towards algorithms to familiarize students and make teaching Computer science easier (that is, their basic part of learning programming). Du Boulay (1980) analyzed the problem where students with lower mathematics skills learned the Logo programming language and its application in mathematical models. Clements and Meredith (1993) analyzed the research papers of Hillel (1984), Carmichael (1985), Kull (1986), and Hillel and Kieran (1987), with the conclusion that Logo programming implies certain mathematical contents. Moreover, Pea (1983) and Kurland et al. (1986) also believe that empirical research has not finally shown that Logo improves the way of thinking in children (Lye & Koh, 2014). The analysis confirming that mathematical science courses are appropriate for computing courses was presented also by Churchhouse (1993). Calao et al. (2015) have been researching whether coding improves students' mathematical skills in Math classes and came to positive results. Klymchuk (2017) examines the learning of engineering mathematics using puzzles as one of the pedagogic strategies for improving thinking abilities. Also, learners who understood the programming logic could transfer the acquired knowledge to the learning of other programming languages (Wolz et al., 2009). The review study by Popat and Starkey (2019) points out that learning programming includes mathematical problem-solving, critical thinking and academic skills as well.

Factors that affect learning programming

Robins et al. (2003) ask the question: "Is it possible to identify the specific deficits of ineffective novices and help them to become effective learners of programming?" There are a number of factors (motivation, emotional responses, general or specific knowledge) that influence learning how to program. Fincher (1999) states that the styles of learning, abilities and skills can be predictive factors for success in learning programming. Perkovic et al. (2010) emphasize the need for intellectual skills required for programming techniques and applications. Based on the precisely defined skills that should be developed for students who are learning programming, it is necessary to properly design a learning game (Lin & Chen, 2016). According to Bruce et al. (2004), students are pleased to write programs if they know that they will be graded for their work.

Computer science in elementary school

Strategy: computer science and programming

During the 1980s, computer science studied the structure of the computer and its principle of work (hardware, logic, binary, etc.). In the 1990s, schools implemented the working principle as well as the use of computers and its applications, while programming was rarely mentioned. Since 2011, there have been significant changes in relation to e-skills as skills of emerging importance, and some organizations started becoming involved in defining computer science at school with the conclusion that they should be an integral part of the school curriculum (Doyle, 1988; e-skills UK, 2012; Livingstone & Hope, 2011; Crick & Sentance, 2011).

The study of computer science and programming, as well as the introduction of their contents into primary school curricula, were among the first to be carried out in the UK and Australia. One of the basic goals was to better adapt these curricula to children in elementary schools (Duncan & Bell, 2015; Brown et al., 2014; Falkner et al., 2014). Motivation and engagement of elementary school students (aged between 8 and 11) to use the programming language (Scratch) in Scotland was managed by the curriculum. Wilson et al., (2012) present empirical proofs and further guidance on the assessment of the ability to do game-based programming. The after-school club named "Code club" was founded in Great Britain in 2012 to support the elementary school in the field of programming. Students are trained together (creating games in the Scratch program) by volunteer programmers and teachers, each in their area of expertise (Smith et al., 2014).

The model of K-12 computer science curriculum consists of three studying levels, whereby each one corresponds with the students' age (Seehorn, 2011). The first level, K-8, corresponds to students in elementary school in the educational system (primarily school and upper grades) of the Republic of Serbia. A noticeable advantage of the K-12 curriculum, looking at the structure of levels, is one compulsory course in the computer science area (at least) at the elementary level, which was a flaw of the educational system in Serbia (although it is studied to some extent through the courses of Technical and Computer Science Education) (Frost et al., 2009; K12, 2011). Also, the ACM K12 curriculum especially emphasizes the importance of developing problem-solving skills and algorithmic thinking, as well as learning programming languages.

In recent times, most countries have been introducing learning a programming language as an integral part of elementary school education. Elementary school students with their first programming steps learn visual programming language - block programming (Serafini, 2011; Pardamean, 2014; Taheri et al., 2016; Mladenović et al., 2017; Papavlasopoulou et al., 2019; Cheng, 2019). After the course of blocks-based programming, students start to learn the basics of textual programming (Borne, 1991; Ferrariet al., 2016; García-Peñalvoet al., 2016; So & Kim, 2018).

Strategy: computer science and programming in the Republic of Serbia

Computer science was first taught in the subject of Computer science and Computing in the Republic of Serbia (RS) in the 2017/18 school year as a compulsory subject for students from the fifth to the eighth grade. For comparison, Australia, the USA and the UK have initiated computer science as a subject from 2015, mostly from the first grade of primary schools, while the Czech Republic, Denmark, Lithuania, Poland and the Netherlands are in the process of doing so (Fluck et al., 2016). The content of this subject, based on the RS Law on Primary Education (Regulation, 2017), differs from the previous one, as it was optional. Computer science and computing consist of three topics: information communication technology, digital literacy, and computing. Within the topic of computing, the most important novelty is the learning of programming in a visual programming language. The course aims to develop digital literacy as one of the most important skills in the 21st century. Also, all teachers of Computer Science in elementary schools in the Republic of Serbia underwent training prior to implementing the teaching (Petlja Foundation). The greatest benefits are anticipated for students who will see that information technology is not just entertainment: video games, surfing the internet, chat, etc., but it can significantly affect their future educational and business direction.

The new program respects the fact that generations born in the digital age enter the educational process with rich experiences in using technology in their everyday life. Information science and computing, described as in the Regulation (2017), will bring students closer to information technology and teach them how to use technology safely. On the other hand, programming gives students the ability to develop a computer-based way of thinking and solving problems. This is especially important considering that the computer mindset is focused on problem-solving and is applicable in all areas of human activity. According to the *Ministry of Education, Science and Technological Development the Republic of Serbia* (2018), this concept combines decomposition of the problem into smaller parts which are easier to resolve. It furthermore empowers the identification identification of samples and general solutions, generalization and an algorithmic way of solving problems, as well as the evaluation of the solution.

Recently, scientists and experts in the Republic of Serbia have written numerous papers arguing the justification for introducing programming in primary schools. Thus, Ivanović and Antonijević (2020) analyze the current situation and perspectives in this area. Suggestions of what the “School of Tomorrow” should look like and criticisms of the late introduction of the subject of Information Technology and Computers as a compulsory subject in primary schools are given in detail by Hilčenko (2017). Nikolić and Subotić (2018) present a model of support for talented students in the field of programming, while Bujić (2020) analyzed the learning of programming based on digital games.

Despite the growing number of papers in the Republic of Serbia dealing with the importance of introducing programming in primary school curricula, to the best of our knowledge, there is no research in the RS that further studies the best approach to learning programming for beginners, especially the relationship between textual and visual programming languages.

Method

Research goal and problem

The introduction of Information Science and Computing as a compulsory subject for students from the fifth to the eighth grade of elementary school, and programming as the basic content of the curriculum represents a major advancement in elementary education in the Republic of Serbia. The aim of the study is to assess the motivation to learn and students' achievement gained by learning the programming languages. This study compared the analysis of learning the first textual language (Python) in relation to the visual programming language (Scratch) by elementary school students.

Hypotheses

Based on the analysis of relevant literature presented in the theoretical part of the paper, the following hypotheses were tested in this study:

H1: Students have a positive attitude toward learning visual and their first textual the programming language.

H2: Students have a positive attitude toward mastering simple algorithmic tasks while learning programming language.

H3: Students have a positive attitude toward mastering the basic elements (input-output commands, relational operators, control of the flow) while learning programming language.

H4: Students consider that knowledge in mathematics influences better mastery of their first textual programming language.

H5: Differences in students' achievements and their interest in the subject effects the learning of the programming language.

Sample

Fifty-eight students were included in this study. They are students of the sixth grade of „Dositej Obradovic" elementary school from Sombor, Republic of Serbia. The ages of participants ranged from 12 to 13 (53.3% male and 46.7% female). Of the seventy-seven students of the fifth grade in the 2016/17 school year, 58 selected the elective subject Information Science and Computing. In order to comply with the need for research ethics, participation in the research was voluntary, and students' grades did not depend on the test results.

Instrument

Our study utilized the experimental design with a pre-test, post-test and final test. The research instrument used was a questionnaire. The questions that are contained in the pre-test and post-test include the subject matter students have learned in the Scratch and Python courses. However, the questions and their number are defined in agreement with the teacher, regarding the appropriate curriculum and, in some elements, are limited by the age of the students. The final test contains statements relating to motivation and the skills that students have acquired by learning the programming languages. In the pre-test and post-test, participants' knowledge was tested for the following: declaring variable, input and output functions, algorithms, logical operators, conditions, and loops. The students were asked to recognize the type of declaration if the type of number is given, and the methods and types of algorithms (described by the pseudo language and the block diagram). Each question contained a textual description (sometimes a graphic image, Figure 1) and three offered answers (of which only one is correct). The pre-test (and pos-test) consisted of 10 questions, some of them (related to conditional expressions and iteration) are given below.

1. What indicates an example of using the if then command?
IF x≥5
then x + 1
else x-1;

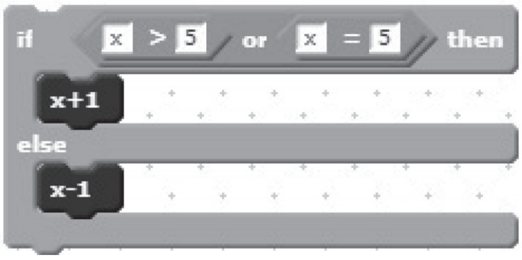


Figure 1. Example if then command

a) If x is greater than 5, increase the number x by 1, and if x is less than 5, decrease the number x by 1.

b) If x is greater than or equal to 5, increase the number x by 1, and if x is less than 5, decrease the number x by 1.

c) *If x is greater than or equal to 5, increase the number x by 1, and if x is less than or equal to 5, decrease the number x by 1.*

2. Using the FOR Command presented example:
FOR i = 1 TO 15
a) prints numbers from 1 to 15.
b) cyclic loop repeats 15 times
c) adds numbers from 1 to 15

The final test contained comparative statements about Scratch and Python (Table 5) and five offered responses (Likert scale).

The following statistical methods were used for processing the collected data: descriptive statistical measures (measures of central tendency, measures of variability, parameters of a distribution), and measures of statistical conclusion (Pearson Chi-Square test and Spearman's relational analysis).

Procedure

The learning of programming languages was implemented according to Kalelioğlu (2015), andErol and Kurt (2017), while the analyzsis of results was based on Kalelioğlu (2015). All students who participated in this research during the school year 2016/2017 learned the programming language Scratch. The course took place once a week (45 minutes) over 17 weeks, in the period from January to May. The students had one school lesson per week (45 minutes), in accordance with the curriculum for elementary school students (as part of elective subject Information Science and Computing). Also, based on the same principle, in the 2017/2018 school year, the Python programming language course was organized from January to May. However, this programming language is being studied as a part of compulsory subject Information Science and Computing. There were no unexpected events or difficulties with the procedure. Figure 2 illustrates the research model of this study. The content of course materials and activities related to the course are presented in Table 1 (Erol & Kurt, 2017).

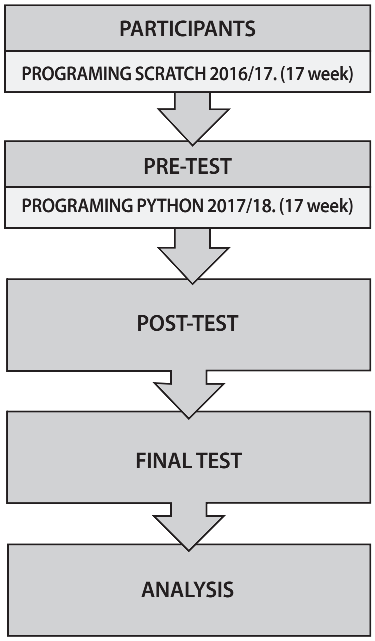

PARTICIPANTS
PROGRAMING SCRATCH 2016/17. (17 week)
PRE-TEST
PROGRAMING PYTHON 2017/18. (17 week)
POST-TEST
FINAL TEST
ANALYSIS

Figure 2. The research model

Table 1
Course content and activities

Work week	Topic
Week 1	Introduction to Python
Week 2	Variables
Weeks 3 & 4	Arithmetic operations
Week 5 - 7	Embedded functions (min, max, abs)
Week 8 & 9	Arrays
Week 10 – 12	Loops
Week 13 – 15	Flow charts
Week 16 & 17	Basic algorithms

Results

The results are given in Table 2, Table 3 and Table 4.

Table 2
Statistics of statements S1 and S4

	Scratch		Python					
	Mean	SD	Mean	SD	Pearson Corr.	Spearman's rho	p	N (valid)
S1	3.31	1.19	3.43	1.30	0.61	0.572	<0.001	51
S4	3.10	1.48	3.10	1.54	0.51	0.5	<0.001	51

Table 3
Scores of pre- and post-test statistics

	χ^2	p	M	SD	N (valid)
I/O instructions	9.68	<0.002	0.68	0.47	56
Relations operator	9.31	<0.002	0.71	0.45	57
Embedded functions (max)	25.83	<0.001	0.85	0.36	57
Algorithms	13.75	<0.001	0.75	0.43	57
Flow control (Repeat,	15.08	<0.001	0.77	0.42	56
For)	9.98	<0.002	0.28	0.45	57

Table 4
Statistics of statements S2, S3 and S5

	Scratch		Python				
	M	SD	M	SD	Pearson χ^2	p	N (valid)
S2	2.78	1.50	3.20	1.27	37.36	<0.002	50
S3	2.78	1.53	2.98	1.75	40.04	<0.001	49
S5	2.92	1.44	2.86	1.30	50.46	<0.001	49

Table 5
Statements S1, S2, S3, S4 and S5

	a: Scratch	b: Python
S1	Learning programming with Scratch is interesting.	Learning programming with Python is interesting.
S2	To master Scratch, I needed knowledge of mathematics.	To master Python, I needed knowledge of mathematics.
S3	While I was learning Scratch, I independently practiced programming (out of school, at home).	While I was learning Python, I indepen-dently practiced programming (out of school, at home).
S4	Learning Scratch made me interested in further studying programming	Learning Python made me interested in further studying programming.
S5	I find it difficult to learn programming through Scratch.	I find it difficult to learn programming through Python.

Discussion

H1: As can be seen in Table 2 (column mean for "Scratch" and "Python" section), students satisfactorily rated studying both programming languages paradigm in terms of how interesting they are. Detailed insight into the obtained results showed that for Scratch column in S1 statements 40 out of 58 students (of which 5 are valid) selected one of the values from 3 to 5 on a Likert scale, while, for Python programming language, 38 out of 58 students (51 valid) chose the same range, i.e. some of the answers denoted as 3, 4 or 5. Also, observing the sameTable and column S4, it can be concluded that students mostly consider that learning textual and visual programming languages can motivate them for further studying programming. Accordingly, based on the answers to statements S1 and S4, hypothesis 1 is confirmed. The confirmation of this hypothesis can also be based on the fact that out of 77 students of the fifth grade in the 2016/17 school year, 58 of them selected the elective subject Science and Computing (within which they were introduced to the basics of information communication technologies and taught the programming language Scratch). These conclusions are consistent with the results obtained in the studies presented in numerous papers (Hromkovič et al., 2016; Mladenović, Boljat, & Žanko 2018; Noone & Mooney, 2018). However, on the basis of the studied literature, one could expect a positive attitude towards learning visual programming language. On the other hand, the results obtained for the students' attitude towards the study of textual programming language can be considered particularly encouraging, given the importance of learning of this type of language, according to Lindstrom (2005) Ranumet al. (2010).

H2: Column „M" in Table 3 presents the mean value of the answers related to knowledge of the five content areas (I/O instructions, Relations operator, Embedded functions, Algorithms, Flow control). Correct answers are denoted with 1, and vice versa; false answers have gotten zero value. Since the majority of students correctly answered questions relating to the algorithm field in the post test, it can be concluded that students have a positive attitude toward mastering simple algorithmic tasks while learning programming language. Therefore, hypothesis 2 is confirmed. The obtained result, although in agreement with most of the literature mentioned in the introductory part, can be considered somewhat different from the research given in the paper by Kalelioğlu and Gülbahar (2014), where no connection was found between problem solving skills and programming using the Scratch programming language.

H3: Analogously, the results related to other thematic areas shown in Table 3 indicate that hypothesis 3 is also confirmed. On the other hand, slightly worse results were obtained for knowledge of iterations (using for loops), which indicates that students have not sufficiently mastered this area using text programming language. This suggests that it is necessary to improve methods for learning iterations. Table three also shows that for all content areas there are statistically significant differences between the pre-test and post-test. Along with the pre-test results, these results imply that the five listed content areas are better mastered by students when using textual programming languages. The obtained confirmation of the hypothesis, which, to the best of our knowledge, has not been researched in modern literature, is the basis for further research. It could involve defining and testing an analogous hypothesis that would encompass other programming topics.

H4: This hypothesis is confirmed by giving answers to statements S2b (Table 5) from the final test. The results from Table 4 (row S2) reveal that students consider that knowledge in mathematics impacts better mastery of their first textual programming language. However, the analysis using the Pearson Chi-Square test showed that there was a statistically significant difference between the answers to these two statements (Pearson's coefficient: 37.36, $p < 0.002$; see Table 4). In this way, students have taken a unique attitude that certain skills in mathematics are required in order to master the textual programming language (Python), and it can be concluded that the hypothesis is confirmed mainly for Python as the first programming language. The obtained results can be considered satisfactory, especially if we take into account that learning programming languages implies mathematical problem-solving (Popat & Starkey, 2019).

H5: Although numerous authors like Bruce et al. (2004), Fincher (1999), Robinset al. (2003) research factors influencing students' propensity for programming, this paper introduces a new approach by analyzing the dependence of interest in learning, propensity for independent learning and difficulty in mastering the language in relation to the two programming paradigms. Thus, the following hypothesis is set: Differences in the achievements of students and their interest in the subject influence learning of the programming language. This hypothesis is confirmed by several points. Firstly by observing the answers to the statements S5a and S5b (Table 5) from the final test shown in the Table 4. Although the students rated these statements considering Python programming language somewhat lower than previous (S1 – S4, Table 2 and 4), and mean values can be considered as satisfactory. Based on students' responses and the obtained results, a significant statistical difference between these two statements can be observed (Pearson's coefficient: 50.46, $p < 0.001$; see Table 4). That is, the students believe that the programming language Scratch is acquired more easily than Python. Secondly – by analysis of the answers to the statements S3a and S3b (Table 5). There is also a significant statistical difference (Pearson coefficient: 40.04, $p < 0.001$; see Table 4) between these two questions. Students who learned Python engaged in practical exercises more independently.

Conclusion

A review of the relevant literature given in this paper shows that the visual paradigm is most appropriate for students learning the first programming language. Also, it turned out that learning the first textual programming language is faster and more relaxed if students have already studied visual programming. Since Scratch and Python are considered among the most popular and simplest visual and textual programming languages, respectively, and are part of the curriculum in the Republic of Serbia, the research of students' knowledge and attitudes in this paper was carried out in relation to these programming languages. A detailed analysis of the current literature pointed to other important aspects related to learning programming among programming novices that needed to be explored (connection with algorithmic thinking, mathematical knowledge, the influence of other aspects, such as independent learning, difficulty, interest, etc.).

The results of this study showed that it was easier for students to master the visual programming language Scratch than the textual programming language Python. Students showed a satisfactory degree of progress in learning the Phyton programming language by acquiring knowledge from using I/O instruction loops and algorithms. Students also began to develop the ability of computer thinking through their active participation in the course of their teaching (by implementing programming languages exercises on a computer).

The results in this study are satisfactory given that the students had no previous experience with the programming of textual language, as well as considering their age, motivation, etc. The introduction of Informatics and Computing as a compulsory subject and programming in elementary schools in the Republic of Serbia is of great significance. Schools were provided with adequate teaching equipment because in most schools the equipment was obsolete. Teachers of Informatics and Computing gain a more important role, i.e., they have the opportunity to bring children into the world of information technology, as their subject is no longer elective..

Future work will focus on expanding research in two ways. One is the inclusion of seventh and eighth grade students. The second is to expand the number of surveyed students, so that the research covers more schools. Also, one more future research direction will be to investigate attitudes of informatics teachers towards research issues presented in this paper.

References

Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From scratch to "real" programming. *ACM Transactions on Computing Education (TOCE)*, *14(4)*, 25. https://doi.org/10.1145/2677087

Berry, M. (2013). Computing in the national curriculum: A guide for primary teachers. *The Chartered Institute for IT* (pp 1-34). Computing at School.

Borne, I. (1991). Object-oriented programming in the primary classroom. *Computers & Education*, *16(1)*, 93-98. https://doi.org/10.1016/0360-1315(91)90048-V

Brown, N. C., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education (TOCE)*, *14*(2), 1-22. https://doi.org/10.1145/2602484

Bruce, C., Buckingham, L., Hynd, J., McMahon, C., Roggenkamp, M., & Stoodley, I. (2004). Ways of experiencing the act of learning to program: A phenomenographic study of introductory programming students at university. *Journal of Information Technology Education: Research*, *3(1),* 145-160. https://doi.org/10.28945/294

Bujić, V. (2020). Digitalne igre kao način učenja programiranja [Digital games as a way of learning programming]. Simpozijum „Tehnika i informatika u obrazovanju: nastavnici za nastavnike", Fakultet tehničkih nauka u Čačku Univerziteta u Kragujevcu, Srbija, 18-20. septembar 2020, 408-413.

Calao, L. A., Moreno-León, J., Correa, H. E., & Robles, G. (2015). Developing mathematical thinking with scratch. *In Design for teaching and learning in a networked world* (pp. 17-27). Springer. https://doi.org/10.1007/978-3-319-24258-3_2

Carmichael, H. W. (1985). Computers, children and classrooms: A multisite evaluation of the creative use of microcomputers by elementary school children. *Final Report.*

Cheng, G. (2019). Exploring factors influencing the acceptance of visual programming environment among boys and girls in primary schools. *Computers in Human Behavior*, *92,* 361-372. https://doi.org/10.1016/j.chb.2018.11.043

Churchhouse, R. F. (1993). Computational mathematics and computer science. *Education and Computing*, *8(4),* 331-337. https://doi.org/10.1016/0167-9287(93)90423-X

Clements, D. H., & Meredith, J. S. (1993). Research on Logo: Effects and efficacy. *Journal of Computing in Childhood Education*, *4(4),* 263-290.

Crick, T., & Sentance, S. (2011, November). Computing at school: stimulating computing education in the UK. *In Proceedings of the 11th Koli Calling International Conference on Computing Education Research* (pp. 122-123). ACM. https://doi.org/10.1145/2094131.2094158

Doyle, S. (1988). *GCSE Computer Studies for You*. Hutchinson Education.

Du Boulay, J. B. H. (1980). Teaching teachers mathematics through programming. *International Journal of Mathematical Educational in Science and Technology*, *11(3),* 347-360. https://doi.org/10.1080/0020739800110306

Duncan, C., & Bell, T. (2015). A pilot computer science and programming course for primary school students. *In Proceedings of the Workshop in Primary and Secondary Computing Education* (pp. 39-48). ACM. https://doi.org/10.1145/2818314.2818328

Duncan, C., Bell, T., & Tanimoto, S. (2014). Should your 8-year-old learn coding? *In Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (pp. 60-69). ACM. https://doi.org/10.1145/2670757.2670774

Erol, O., & Kurt, A. A. (2017). The effects of teaching programming with Scratch on pre-service information technology teachers' motivation and achievement. *Computers in Human Behavior*, *77,* 11-18. https://doi.org/10.1016/j.chb.2017.08.017

e-skills UK. 2012. *Technology insights 2012*. http://www.e-skills.com/research/research-publications/insights-reports-and-videos/technologyinsights-2012/

Falkner, K., Vivian, R., & Falkner, N. (2014). The Australian digital technologies curriculum: Challenge and opportunity. *In Proceedings of the Sixteenth Australasian Computing Education Conference-Volume 148* (pp. 3-12).

Ferrari, A., Poggi, A., & Tomaiuolo, M. (2016). Object oriented puzzle programming. *Mondo Digitale*, *15,* 64. https://doi.org/10.1007/978-1-4842-3171-5_2

Fincher, S. (1999). What are we doing when we teach programming?. In FIE'99 Frontiers in Education. *29th Annual Frontiers in Education Conference. Designing the Future of*

Science and Engineering Education. Conference Proceedings (IEEE Cat. No. 99CH37011 (Vol. 1, pp. 12A4-1). IEEE. https://doi.org/10.1109/FIE.1999.839268

Fluck, A., Webb, M., Cox, M., Angeli, C., Malyn-Smith, J., Voogt, J., & Zagami, J. (2016). Arguing for computer science in the school curriculum. *Journal of Educational Technology & Society,* 19(3), 38-46.

Frost, D., Verno, A., Burkhart, D., Hutton, M., North, K., & Houston, I. S. D. (2009). A model curriculum for K–12 Computer Science Level I objectives and outlines. *Computer Science Teachers Association*. https://d1wqtxts1xzle7.cloudfront.net/47451839/L1-Objectives-and-Outlines.pdf?1469274163=&response-content- .

Futschek, G., & Moschitz, J. (2011). Learning algorithmic thinking with tangible objects eases transition to computer programming. *In International conference on informatics in schools: Situation, evolution, and perspectives* (pp. 155-164). Springer. https://doi.org/10.1007/978-3-642-24722-4_14

García-Peñalvo, F. J., Hughes, J., Rees, A., Jormanainen, I., Toivonen, T., Reimann, D., Tuul, M., & Virnes, M. (2016). Evaluation of existing resources (study/analysis). *TACCLE3 Consortium*, Belgium.

Gilmore, D. J. (1990). Expert programming knowledge: A strategic approach. *In Psychology of programming* (pp. 223-234). Academic Press. https://doi.org/10.1016/B978-0-12-350772-3.50019-7

Gruenbaum, P. (2014). Undergraduates teach game programming using Scratch. *IEEE Computer*, *47(2),* 82-84. https://doi.org/10.1109/MC.2014.49

Halbert D. C. (1984). *Programming by Example*. (*Doctoral dissertation).* Berkeley: University of California.

Hilčenko, S. (2017). School of tomorrow. *Media, Culture and Public Relations,* 8(1), 88-93.

Hillel, J. (1984). Mathematical concepts and programming skills acquired by 8-year-olds in a restricted Logo environment. *Logo 84 conference: Cambridge*, MA.

Hillel, J. & Kieran, C. (1987). Schemas used by 12-year-olds in solving selected turtle geometry tasks. *Recherches en Didactique des Mathématiques*. 8(1-2), 61-103.

Hromkovič, J., Kohn, T., Komm, D., & Serafini, G. (2016). Combining the power of python with the simplicity of Logo for a sustainable computer science education. *In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 155-166). Springer. https://doi.org/10.1007/978-3-319-46747-4_13

Hsu, C. C., & Wang, T. I. (2018). Applying game mechanics and student-generated questions to an online puzzle-based game learning system to promote algorithmic thinking skills. *Computers & Education*, *121,* 73-88. https://doi.org/10.1016/j.compedu.2018.02.002

Ivanović, Đ., & Antonijević, M. (2020). E-učenje-stanje i perspektive u Republici Srbiji, XXVI Skup *Trendovi razvoja: Inovacije u modernom obrazovanju* [*Development trends: Innovations in modern education*].

", Kopaonik, February 16 - 192020, 187-190.

K-12 (2011). *Computer Science Standards*. http://csta.acm.org/Curriculum/sub/K12 Standards .html

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code. org. *Computers in Human Behaviour*, *52,* 200-210. https://doi.org/10.1016/j.chb.2015.05.047

Kalelioğlu, F., & Gülbahar, Y. (2014). The effects of teaching programming via Scratch on problem solving skills: A discussion from learners' perspective. *Informatics in Education,* 13(1), 33-50.

Kátai, Z. (2015). The challenge of promoting algorithmic thinking of both sciences-and humanities-oriented learners. *Journal of Computer Assisted Learning, 31(4),* 287-299. https://doi.org/10.1111/jcal.12070

Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR), 37(2),* 83-137. https://doi.org/10.1145/1089733.1089734

Kerner, I. O. (1986). Computer awareness by teaching informatics. *Education and Computing, 2(1-2),* 133-135. https://doi.org/10.1016/S0167-9287(86)91271-9

Klymchuk, S. (2017). Puzzle-based learning in engineering mathematics: Students' attitudes. *International Journal of Mathematical Education in Science and Technology, 48(7),* 1106-1119. https://doi.org/10.1080/0020739X.2017.1327088

Ko, Y., & Park, N. (2011). Experiment and verification of teaching fractal geometry concepts using a Logo-based framework for elementary school children. *In International Conference on Future Generation Information Technology* (pp. 257-267). Springer. https://doi.org/10.1007/978-3-642-27142-7_30

Kölling, M., Brown, N. C., & Altadmri, A. (2015). Frame-based editing: Easing the transition from blocks to text-based programming. *In Proceedings of the Workshop in Primary and Secondary Computing Education* (pp. 29-38). ACM. https://doi.org/10.1145/2818314.2818331

Kull, J.A. (1986). Learning and Logo, in young cChildren and microcomputers, P.F. Campbell and G.G. Fein, Editor. (pp. 103-130.). Prentice-Hall. Englewood Cliffs, NJ.

Kurland, D. M., Pea, R., Clement, C., & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research, 2(4),* 429–458. https://doi.org/10.2190/BKML-B1QV-KDN4-8ULH

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *Acm Sigcse Bulletin, 37(3),* 14-18. https://doi.org/10.1145/1151954.1067453

Lenhart, A., Simon, M., & Graziano, M. (2001). The Internet and education: Findings of the Pew Internet & American Life Project. https://files.eric.ed.gov/fulltext/ED457849.pdf

Lewis, C., & Olson, G. (1987). Can principles of cognition lower the barriers to programming?. *In Empirical studies of programmers: second workshop* (pp. 248-263). Ablex Publishing Corp.

Lin, C. H., & Chen, C. M. (2016). Developing spatial visualization and mental rotation with a digital puzzle game at primary school level. *Computers in Human Behavior, 57,* 23-30. https://doi.org/10.1016/j.chb.2015.12.026

Lindstrom, G. (2005). Programming with python. *IT professional, (5),* 10-16. https://doi.org/10.1109/MITP.2005.120

Livingstone, I., & Hope, A. (2011). Next Gen: Transforming the UK into the world's leading talent hub for the video games and visual effects industries. *National Endowment for Science, Technology and the Arts (NESTA)*, London, UK.

Lopez, M., Whalley, J., Robbins, P., & Lister, R. (2008). Relationships between reading, tracing and writing skills in introductory programming. *In Proceedings of the fourth international workshop on computing education research* (pp. 101-112). ACM. https://doi.org/10.1145/1404520.1404531

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. *Computers in Human Behavior, 41,* 51-61. https://doi.org/10.1016/j.chb.2014.09.012

MESTD (2018). *Ministry of Education, Science and Technological Development the Republic of Serbia.* http://www.mpn.gov.rs/informatika-i-racunarstvo-obavezan-predmet-za-ucenike-petog-razreda/

Mladenović, M., Boljat, I., & Žanko, Ž. (2018). Comparing loops misconceptions in block-based and text-based programming languages at the K-12 level. *Education and Information Technologies*, *23(4),* 1483-1500. https://doi.org/10.1007/s10639-017-9673-3

Mladenović, M., Krpan, D., & Mladenović, S. (2017). Learning programming from Scratch. *The Turkish Online Journal of Educational Technology*, *2,* 419-427.

Moreno, J., & Robles, G. (2016). Code to learn with Scratch. In A systematic literature review. *IEEE Global Engineering Education Conference, EDUCON* (pp. 150-156), Abu Dhabi.

Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. RED. *Revista de Educación a Distancia, (46),* 1-23.

Myers, B. A. (1990). Taxonomies of visual programming and program visualization. *Journal of Visual Languages & Computing*, *1(1),* 97-123. https://doi.org/10.1016/S1045-926X(05)80036-9

Nikolić, M., & Subotić, S. (2018). Ilustracija modela podrške darovitosti u oblasti programiranja [The illustration of the support to giftedness model in the programming field]. Tematski zbornik sa 23. Okruglog stola "Darovitost i kreativni pristupi učenju". Vršac: Visoka škola strukovnih studija za obrazovanje vaspitača „Mihailo Palov", 287-292.

Noone, M., & Mooney, A. (2018). Visual and textual programming languages: a systematic review of the literature. *Journal of Computers in Education*, *5(2),* 149-174. https://doi.org/10.1007/s40692-018-0101-5

Ozoran, D., Cagiltay, N., & Topalli, D. (2012). Using Scratch in introduction to programming course for engineering students. In 2nd International Engineering Education Conference (IEEC2012) (Vol. 2, pp. 125-132).

Palumbo, D. B. (1990). Programming language/problem-solving research: A review of relevant issues. *Review of educational research*, *60(1),* 65-89. https://doi.org/10.3102/00346543060001065

Papavlasopoulou, S., Giannakos, M. N., & Jaccheri, L. (2019). Exploring children's learning experience in constructionism-based coding activities through design-based research. *Computers in Human Behavior*, 99, 415-427. https://doi.org/10.1016/j.chb.2019.01.008

Pardamean, B. (2014). Enhancement of creativity through Logo programming. *American Journal of Applied Sciences*, 11(4), 528. https://doi.org/10.3844/ajassp.2014.528.533

Pea, R. (1983). Logo programming and problem solving. *In American Educational Research Association*. Montreal, Canada.

Pea, R. (1984). Symbol systems and thinking skills: Logo in context. *Preproceedings of the 1984 International Logo Conference*. Cambridge, MA: Massachusetts Institute of Technology.

Perković, L., Settle, A., Hwang, S., & Jones, J. (2010). A framework for computational thinking across the curriculum. *In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (pp. 123-127). ACM. https://doi.org/10.1145/1822090.1822126

Petlja Foundation. *Obuka za nastavnike informatike u okviru programa „Nastava programiranja u petom i šestom razredu [Training for ICT teachers within the frame of "Programming class in the fifth and sixth grade" programme]*. https://petlja.org/n/pocetak-obuka-za-nastavnike-informatike-petog-i-sestog-razreda

Popat, S., & Starkey, L. (2019). Learning to code or coding to learn? A systematic review. *Computers & Education*, *128,* 365-376. https://doi.org/10.1016/j.compedu.2018.10.005

Pravilnik (2017). Pravilnik o izmenama i dopunama pravilnika o nastavnom planu za drugi ciklus osnovnog obrazovanja i vaspitanja i nastavnom programu za peti razred osnovnog obrazovanja i vaspitanja ("Sl. glasnik RS – Prosvetni glasnik", br. 6/2017).

Ranum, D., Miller, B., Zelle, J., & Guzdial, M. (2006). Successful approaches to teaching introductory computer science courses with Python. *In ACM SIGCSE Bulletin* (Vol. 38, No. 1, pp. 396-397). ACM. https://doi.org/10.1145/1124706.1121465

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52(11),* 60-67. https://doi.org/10.1145/1592761.1592779

Robins, A. V. (2019). Novice programmers and introductory programming. *The Cambridge handbook of computing education research*, *1*, 327-376. https://doi.org/10.1017/9781108654555.013

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer science education*, *13(2),* 137-172. https://doi.org/10.1076/csed.13.2.137.14200

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two-year case study using "Scratch" in five schools. *Computers & Education*, *97,* 129-141. https://doi.org/10.1016/j.compedu.2016.03.003

Salomon, G. (1985). Information technologies: What you see is not (always) what you get. *Educational Psychologist*, 20(4), 207-216. https://doi.org/10.1207/s15326985ep2004_5

Schulte, C. (2013, November). Reflections on the role of programming in primary and secondary computing education. *In Proceedings of the 8th Workshop in Primary and Secondary Computing Education* (pp. 17-24). ACM. https://doi.org/10.1145/2532748.2532754

Seehorn, D. (2011). *K-12 computer science standards–revised 2011: The CSTA standards task force.* ACM, October.

Serafini, G. (2011). Teaching programming at primary schools: Visions, experiences, and long-term research prospects. *In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 143-154). Springer. https://doi.org/10.1007/978-3-642-24722-4_13

Smith, N., Sutcliffe, C., & Sandvik, L. (2014). Code club: Bringing programming to UK primary schools through scratch. *In Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 517-522). ACM.

So, M. H., & Kim, J. (2018). An analysis of the difficulties of elementary school students in Python programming learning. *International Journal on Advanced Science, Engineering and Information Technology, 8(4-2),* 1507-1512. https://doi.org/10.18517/ijaseit.8.4-2.2720

Taheri, S. M., Sasaki, M., Chu, J. O., & Ngetha, H. T. (2016). A study of teaching problem solving and programming to children by introducing a new programming language. *The International Journal of E-Learning and Educational Technologies in the Digital Media (IJEETDM)*, *2(1),* 31-36. https://doi.org/10.17781/P001972

Tsukamoto, H., Takemura, Y., Nagumo, H., Ikeda, I., Monden, A., & Matsumoto, K. I. (2015). Programming education for primary school children using a textual programming language. *In Frontiers in Education Conference (FIE)*, 2015 IEEE (pp. 1-7). IEEE. https://doi.org/10.1109/FIE.2015.7344187

Wilson, A., & Moffat, D. C. (2010, September). Evaluating Scratch to Introduce Younger Schoolchildren to Programming. In *PPIG* (Vol. 1, No. 1, pp. 1-12).

Wilson, A., Hainey, T., & Connolly, T. (2012). Evaluation of computer games developed by primary school children to gauge understanding of programming concepts. *In 6th European conference on games-based learning (ECGBL)* (pp. 4-5).

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49(3),* 33–35. https://doi.org/10.1145/1118178.1118215

Wolz, U., Leitner, H. H., Malan, D. J., & Maloney, J. (2009). Starting with scratch in CS 1. *ACM SIGCSE Bulletin*, *41(1),* 2-3. https://doi.org/10.1145/1539024.1508869

Dragan Rastovac
Department of Informatics and media
Faculty of Education Sombor
University of Novi Sad
Podgorička 4, 25101 Sombor, Serbia
rastovacd@gmail.com

Milinko Mandić
Department of Informatics and media
Faculty of Education Sombor
University of Novi Sad
Podgorička 4, 25101 Sombor, Serbia
milinmand@gmail.com

Violeta Majski
Elementary school "Dositej Obradović", Sombor
Slaviše Vajnera Čiče bb, 25101 Sombor, Srbija
violeta.majski@gmail.com

Dragan Cvetković
Department of Informatics and media
Faculty of Education Sombor
University of Novi Sad
Podgorička 4, 25101 Sombor, Serbia
dcveles@gmail.com

Istraživanje postignuća i motivacije učenja tekstualnoga programskog jezika među učenicima osnovnih škola u Republici Srbiji

Sažetak

Uvođenje informatike i računarstva kao obveznoga predmeta za učenike od petog do osmog razreda osnovne škole te programiranja kao osnovnoga sadržaja kurikula predstavlja veliki iskorak u osnovnom obrazovanju u Republici Srbiji. U ovom radu provodeno je istraživanje na N = 58 učenika osnovnih škola šestog razreda. U školskoj godini 2016./17. učenici su učili programski jezik Scratch, a u školskoj godini 2017./18. učili su programski jezik Python. Tečajevi programiranja realizirali su se jednom tjedno (45 minuta) tijekom 17 tjedana. Cilj je ove studije pratiti tijek učenja vizualnoga i tekstualnoga programskog jezika sljedeći novi OŠ kurikul u Republici Srbiji te istražiti postignuće i motivaciju učenika za nastavak učenja programiranja. Korišteni istraživački instrument bio je upitnik. Rezultati ovoga istraživanja pokazali su da je učenicima bilo lakše svladati vizualni programski jezik Scratch nego tekstualni programski jezik Python. Međutim, rezultati istraživanja pokazuju da su algoritamski način razmišljanja i motivacija za učenje programiranjem tekstualnoga jezika zadovoljavajući s obzirom na to da učenici s tim nisu imali prethodno iskustvo.

Ključne riječi: *osnovno obrazovanje; računalno razmišljanje, tekstualno programiranje; vizualno programiranje.*

Uvod

U posljednjem desetljeću došlo je do ekspanzije informacijske komunikacijske tehnologije što je rezultiralo pojavom novih digitalnih uređaja. Djeca predškolske i osnovnoškolske dobi sve više zamjenjuju televiziju i slikovnice računalima, tabletima i pametnim telefonima. Slijedom toga, djeca provode više vremena igrajući igre na računalu i gledajući videoisječke (koji mogu biti edukativni ili slični bilo kojem drugom sadržaju). Navedeni digitalni uređaji i internet postaju važno okruženje za učenje djece. Računalo kao jedan od najznačajnihih digitalnih uređaja ima veoma važnu ulogu u poboljšanju obrazovanja (Salomon, 1985; Lenhart, Simon, i Graziano, 2001).

Wing (2006) ističe da ako djeca žele razumjeti i aktivno sudjelovati u novom digitalnom svijetu, važno je da ona ovladaju „računalnim razmišljanjem“. Novi nacionalni kurikul za računalstvo omogućit će djeci širom svijeta da shvate i steknu računalne vještine koje će im trebati u budućnosti. Na temelju novoga programa informatike, djeca će steći znanje o strukturi i radu računala i razvijati svoje ideje koristeći novu tehnologiju (Berry, 2013).

Cilj je studije, predstavljene u ovom radu, istražiti interes učenika osnovnih škola za informatiku ponajprije za programiranje. Budući da se programiranje može smatrati važnim dijelom nastave Informatike (Schulte, 2013), posebno je važno odabrati pravu metodologiju poučavanja za studente koji se prvi put susreću s programiranjem - programere početnike (Gilmore, 1990; Lahtinen, Ala-Mutka, i Järvinen 2005; Kelleher i Pausch, 2005; Robins, 2019). To se posebno odnosi na izbor tipa (prvog) programskoga jezika te na redoslijed i prioritet proučavanja tekstualne i vizualne programske paradigme. Važan aspekt metodologije poučavanja programiranja je njezina zastupljenost u tečajevima informatike u osnovnoškolskim programima. Također, brojna se literatura bavi odnosom između algoritamskoga načina razmišljanja i programiranja, kao i matematičkim znanjem i vještinama programiranja te koji čimbenici motiviraju učenike na učenje programiranja. Stoga je sljedeće teorijsko poglavlje (pregled literature) organizirano na takav način da daje pregled radova u vezi s navedenim aspektima.

Pregled literature

Programiranje

Ako programiranje želimo naučiti na tradicionalan način, potrebno je znati instrukcije programskoga jezika, tako da nas računalo „razumije“. Osnovni zadatak programiranja je učenje naredbi programskoga jezika (deklarativno znanje) i omogućavanje upotrebe tih podataka na različite načine (proceduralno znanje) (Palumbo, 1990). Palumbo (1990) je na temelju rezultata danih u radu Pea (1984) izrazio sumnju u statističku ovisnost između instrukcija programskoga jezika i načina rješavanja problema. Devedesetih godina dvadesetog stoljeća počeo se javljati interes ljudi za jezike koji koriste grafičko okruženje za programiranje, uklanjanje pogrešaka itd. Pojava grafičkoga okruženja bila je očekivana jer je poznato da je tradicionalno programiranje složenije za učenje budući da zahtijeva određenu kognitivnu sposobnost koju nemaju svi ljudi. Pojavom programskih jezika dizajniranih da omogućuju vizualno programiranje, pri čemu nema potrebe za detaljnim poznavanjem sintakse (za razliku od tekstualnih programskih jezika), zanimanje za njih počinje se širiti. Stoga je bilo potrebno osmisliti grafičko okruženje kojim će se lako upravljati i koje će se lako koristiti te zainteresirati što više ljudi za „grafičko programiranje“ (Halbert, 1984; Lewis i Olson, 1987; Myers, 1990).

Vizualni i tekstualni programski jezici u školama

Jedan od najpopularnijih jezika vizualnoga programiranja je Scratch koji djeci u osnovnoj školi omogućuje da samostalno i bez straha steknu osnove programiranja. Scratch je dizajnirao Mitchel Resnick s Massachusetts Institute of Technology. Vrlo je jednostavan za upotrebu jer su kontrole (blokovi) na zaslonu i mogu se međusobno prilagoditi i dobiti odgovarajuće značenje. Ne postoji sintaksni kod ili poruke o pogreškama kao u tekstualnom programiranju (Resnick, Maloney, Monroy-Hernández, Rusk, Eastmond, Brennan, i Kafai, 2009; Wilson i Moffat, 2010; Moreno i Robles, 2016). Moreno-León, Robles i Román-González (2015) izvršili su pregled objavljenih radova iz područja programskih jezika za razdoblje od 2007. do 2015. Rezultati pregleda pokazuju da je Scratch bio predmet mnogih istraživanja. Tako su primjenu integriranja vizualnih blokova i kodiranja u Scratchu za 5. i 6. razred osnovne škole u razdoblju od dvije školske godine istraživali Sáez-López, Román-González i Vázquez-Cano (2016). Kalelioğlu i Gülbahar (2014) ispitali su utjecaj programa Scratch na vještine rješavanja problema učenika 5. razreda osnovne škole. Rezultati istraživanja pokazali su da su studenti zainteresirani za programiranje, ali nije bilo većih razlika u vještinama rješavanja problema. Važno je istaći i da su četvorica dodiplomaca na računalnim smjerovima Sveučilišta Washington Bothel prošla obuke programiranje igara koristeći Scratch kako bi mogla prenijeti znanje učenicima (od 6. do 8. razreda), jer je procijenjeno da u tom području nema dovoljno stručnjaka (Gruenbaum, 2014). Kalelioğlu (2015) je također ispitao kako je mrežna platforma code.org utjecala na vještine rješavanja problema kod učenika oba spola u osnovnoj školi. Rezultati istraživanja ponovno su pokazali zanimanje učenika za (vizualno) programiranje. Istraživanje primjene stranice code.org nije pokazalo razlike u razmišljanjima među učenicima različitoga spola. Međutim, postoji mala razlika u reflektivnom razmišljanju među spolovima (u korist učenica).

S druge strane, učeći tekstualno programiranje, djeca stječu „računalni način razmišljanja" koji može biti koristan, kasnije, u daljnjem obrazovanju. Dok pišu kôd, u neprestanom su „sukobu" s kompajlerom kad god se njihov kôd izvršava. Međutim, stječu bolje obrazovne mogućnosti, a softver ih potiče da prevladaju dojam da je pisanje kôda iznad njihovih mogućnosti (Duncan, Bell, i Tanimoto, 2014; Tsukamoto, Takemura, Nagumo, Ikeda, Monden, i Matsumoto, 2015). Također, Lopez, Whalley, Robbins i Lister (2008) pokazuju da vještina pisanja tekstualnoga programskog kôda korelira s vještinom čitanja kôda. Jedan od najjednostavnijih tekstualnih jezika je Python. Python je dizajnirao Guido van Rossum početkom 1990. godine. Nudi sve što nam je potrebno od programskoga jezika i jednostavan je za učenje i razumijevanje. Možda je dovoljno samo spomenuti citat iz istraživanja danog u Lindstrom (2005): „Moja 10-godišnja kći programira domaću zadaću iz matematike na Pythonu (upravo je napisala rutinu za pretvaranje stupnjeva Celzijevih u stupnjeve Fahrenheita)". Olakšavanje primjene Pythonova programskoga jezika u uvodnim tečajevima informatike predstavili su, među ostalima, i Ranum, Miller, Zelle i Guzdial (2010).

Složenost strukture tekstualnih programskih jezika (poput jezika C), koji se predaju na fakultetima (i ranije u osnovnim te posebno, srednjim školama), može uzrokovati nezainteresiranost i nedostatak motivacije za studente da nastave učiti programiranje. Kako bi olakšali obuku učenika za jezik C, studentima je paralelno s klasičnom nastavom, uveden Scratch kao potporno okruženje (Ozoran, Cagiltay, i Topalli, 2012). Analizu prelaska s učenja vizualnoga programiranja na tekstualni programski jezik (C # ili Java) za učenike (od 15 do 16 godina) ispitali su Armoni, Meerbaum-Salant i Ben-Ari (2015). Autori su zaključili da znanje i iskustvo studenata programiranja koji su poučavali Scratch uvelike olakšava učenje naprednijega programskog jezika. Također su bolje razumjeli tu temu od učenika koji prethodno nisu učili Scratch. Erol i Kurt (2017) su analizirali motivaciju i rezultate koje su postigli studenti Pedagoškog fakulteta u Turskoj, dok su tijekom 7 tjedana učili programske jezike Scratch i C #. Brojni su i drugi istraživački radovi koji se bave komparativnim studijama postignuća djece koja su stekla učenjem vizualnoga i tekstualnoga programskog jezika (Hromkovič, Kohn, Komm, i Serafini, 2016; Mladenović, Boljat i Žanko 2018; Noone i Mooney, 2018). Kölling, Brown i Altadmri (2015) predstavili su prijelaz s učenja vizualnoga programskog jezika na tekstualni u osnovnoj školi.

Razvoj algoritamskoga mišljenja

Prema Kerner (1986), razvoj algoritama i potprograma trebao bi biti sastavni dio učenja programiranja, pored izučavanja samih programskih jezika. Brojni radovi se bave razvojom algoritamskoga razmišljanja. Tako, korištenjem mrežnoga sustava učenja temeljenom na zagonetkama i Tutorskom sustavu grafike „kornjača“ (TGTS), Hsu i Wang (2018) vjeruju da će potaknuti učeničko algoritamsko razmišljanje. Važnost algoritamskoga razmišljanja djece kao osnove učenja u osnovnoškolskoj dobi shvatili su Futschek i Moschitz (2011). Predstavili su takozvani scenarij učenja *Tim the Train* koji uključuje materijalne objekte na temelju kojih se poučavaju zadatci za podizanje razine algoritamskoga razmišljanja. Autori su također pokazali blagi prijelaz s objekata na virtualno Scratch / BYOB okruženje zbog čega se studenti osjećaju bolje u svojim prvim programskim koracima. Ko i Park (2011) ističu da će učenici osnovnih škola izvoditi razne aktivnosti u procesu programiranja, bez obzira na ishod. Tako će se tijekom ovoga procesa poboljšati sposobnost rješavanja problema i logičkoga razmišljanja. Prema Kátai (2015) i pravilno dizajnirani alati za e-učenje, općenito, utječu na razvoj algoritamskoga mišljenja kako kod „znanstveno orijentiranih“, tako i kod „društveno orijentiranih“ učenika.

Programiranje i matematika

Pri realizaciji nastave programiranja potrebno je uzeti u obzir metode nastave, didaktiku i pedagogiju povezanih područja, poput matematike. S obzirom na važnost proučavanja algoritama, predmet matematičkoga obrazovanja u školama trebao bi biti usmjeren na algoritme kako bi se učenicima približila i učinila lakšim nastava Informatike (osnovno njezin najvažnjiji dio – programiranje). Du Boulay (1980) je analizirao problem gdje su učenici s nižim matematičkim vještinama naučili programski jezik Logo i njegovu primjenu u matematičkim modelima. Clements i Meredith (1993) su analizirali istraživačke radove autora: Hillel (1984), Carmichael (1985), Kull (1986), Hillel i Kieran (1987) zaključivši da programiranje Logo jezika podrazumijeva određene matematičke sadržaje. Štoviše, prema autorima Pea (1983), Kurland, Pea, Clement, i Mawby (1986), Lye i Koh (2014) empirijska istraživanja konačno nisu pokazala da Logo poboljšava način razmišljanja kod djece. Analiza da su tečajevi matematičke znanosti prikladni za tečajeve računalstva predstavljena je i u Churchhouse (1993). Calao, Moreno-León, Correa i Robles (2015) istraživali su poboljšava li kodiranje matematičke vještine učenika na satima Matematike i došli do pozitivnih rezultata. Klymchuk (2017) u svojem istraživanju ispituje učenje inženjerske matematike koristeći zagonetke kao jednu od pedagoških strategija za poboljšanje sposobnosti razmišljanja. Također, Wolz, Leitner, Malan i Maloney (2009) zaključuju da bi učenici, koji su razumjeli programsku logiku, mogli stečeno znanje prenijeti na učenje drugih programskih jezika. Tako i pregledna studija Popat i Starkey (2019) navodi da učenje programiranja uključuje matematičko rješavanje problema, kritičko razmišljanje i akademske vještine.

Čimbenici koji utječu na učenje programiranja

Robins, Rountree i Rountree (2003) postavljaju pitanje: „Je li moguće identificirati specifične nedostatke neučinkovitih novaka i pomoći im da postanu učinkoviti učenici u programiranju?" Brojni su čimbenici (motivacija, emocionalni odgovori, općenito ili specifično znanje) koji utječu na učenje programiranja. Fincher (1999) navodi da stilovi učenja, sposobnosti i vještine mogu biti prediktivni faktori uspjeha u učenju programiranja. Potrebu za intelektualnim vještinama neophodnim za programerske tehnike i aplikacije navode i Perković, Settle, Hwang i Jones (2010). Lin i Chen (2016) ističu da je potrebno pravilno osmisliti „igru za učenje" na temelju točno definiranih vještina koje bi trebali razviti učenici koji uče programiranje. Prema Bruceu, Buckinghamu, Hyndu, McMahonu, Roggenkampu i Stoodleyu (2004), studenti sa zadovoljstvom pišu programe ako znaju da će dobiti ocjene za svoj rad.

Računalstvo u osnovnoj školi

Strategija: informatika i programiranje

U računalnoj znanosti 1980-ih proučavana je računalna struktura i njezin princip rada (hardver, logika, binarni brojevi itd.). Devedesetih su škole provodile princip rada kao i upotrebu računala i njegovih aplikacija, dok se programiranje rijetko spominjalo. Od 2011. došlo je do značajnih promjena u vezi s rastućim značajem e-vještina te su neke organizacije uključene u definiranje „školske" informatike zaključujući da bi ona trebala biti sastavni dio školskoga programa (Doyle, 1988; e-vještine UK, 2012; Livingstone i Hope, 2011; Crick i Sentance, 2011).

Izučavanje informatike i programiranja te uvođenje njihovih sadržaja u osnovnoškolske kurikule provodile su među prvima Velika Britanija i Australija. Jedan od osnovnih ciljeva bio je bolje prilagoditi ove nastavne planove i programe djeci u osnovnim školama (Duncan i Bell, 2015; Brown, Sentance, Crick, i Humphreys, 2014; Falkner, Vivian, i Falkner, 2014). Motivacija i angažman učenika osnovnih škola (u dobi između 8 i 11 godina) za korištenje programskoga jezika (Scratch) u Škotskoj definiran je nastavnim planom i programom, temeljenog na igrama. Wilson, Hainey, i Connolly (2012) predstavljaju empirijske dokaze i daljnje smjernice za procjenu sposobnosti programiranja koristeći igre. Klub za predškolske ustanove pod nazivom „Code club" osnovan je u Velikoj Britaniji 2012. godine kao podrška osnovnoj školi u području programiranja. Studente zajedno poučavaju (kreiranje igara u programu Scratch) programeri i učitelji volonteri, svaki u svojem području stručnosti (Smith, Sutcliffe, i Sandvik, 2014).

Model kurikula za informatiku K-12 sastoji se od tri razine studija, pri čemu svaka odgovara određenoj dobi učenika (Seehorn, 2011). Prva razina, K-8 odgovara učenicima u osnovnoj školi u obrazovnom sustavu Republike Srbije. Primjetna prednost kurikula K-12, gledajući strukturu razina, je (barem) jedan obvezni kolegij iz područja računalstva na razini osnovne škole, što je bila mana obrazovnoga sustava u Srbiji (iako se donekle proučavala kroz tečajeve tehničkoga i informatičkoga obrazovanja) (Frost, Verno, Burkhart, Hutton, North, i Houston, 2009; K12, 2011). Također, nastavni plan i program ACM K12 posebno naglašava važnost razvijanja vještina rješavanja problema i algoritamskoga razmišljanja, kao i učenja programskih jezika.

U posljednjem razdoblju većina zemalja uvodi učenje programskoga jezika kao sastavni dio osnovnoškolskoga obrazovanja. Učenici osnovnih škola s prvim programskim koracima uče vizualni programski jezik blok-programiranje (Serafini, 2011; Pardamean, 2014; Taheri, Sasaki, Chu, i Ngetha, 2016; Mladenović, Krpan, i Mladenović, 2017; Papavlasopoulou, Giannakos i Jaccheri, 2019; Cheng, 2019). Nakon tečaja programiranja zasnovanog na blokovima, učenici počinju učiti osnove tekstualnoga programiranja (Borne, 1991; Ferrari, Poggi i Tomaiuolo, 2016; García-Peñalvo, Hughes, Rees, Jormanainen, Toivonen, Reimann, i Virnes, 2016; So i Kim, 2018).

Strategija: računalstvo i programiranje u Republici Srbiji

U Republici Srbiji (RS), informatika se predaje na predmetu Informatika i računarstvo prvi put od školske 2017./18. godine kao obavezni predmet za učenike od petog do osmog razreda. Za usporedbu, Australija, SAD, UK pokrenule su informatiku kao predmet od 2015. godine, uglavnom od prvih razreda osnovnih škola, dok su Češka, Danska, Litva, Poljska i Nizozemska još uvijek u procesu uvođenja informatike u osnovne škole (Fluck, Webb, Cox, Angeli, Malyn-Smith, Voogt, i Zagami, 2016). Sadržaj ovoga predmeta u RS, zasnovan na Zakonu o osnovnom obrazovanju i obrazovanju RS (Pravilnik, 2017), razlikuje se od prethodnoga koji nije bio obvezan. Informatika i računarstvo sastoje se od tri teme: informacijska komunikacijska tehnologija, digitalna pismenost i računarstvo (Pravilnik, 2017). U okviru teme računalstva najvažnija novost je učenje programiranja na vizualnom programskom jeziku. Cilj tečaja je razviti digitalnu pismenost kao jednu od najvažnijih vještina 21. stoljeća. Također, svi nastavnici informatike u osnovnim školama u Republici Srbiji prošli su obuku prije nego što su započeli s realizacijom nastave (Petlja fondacija). Najveće koristi trebali bi imati učenici koji će vidjeti da informacijska tehnologija nije samo zabava: videoigre, *surfanje* internetom, *chat* itd., već može značajno utjecati na njihov budući obrazovni i poslovni smjer.

Novi program poštuje činjenicu da generacije koje su rođene u digitalno doba dolaze u obrazovni sustav, tako da većina već ima bogato iskustvo u korištenju tehnologije u svom svakodnevnom životu. Informatika i računarstvo, opisani kao u Pravilniku (2017), približit će učenike informacijskoj tehnologiji i naučiti ih kako ih sigurno koristiti. S druge strane, programiranje daje učenicima sposobnost da razviju računalski način razmišljanja i rješavanja problema. To je posebno važno s obzirom na to da je računalni način razmišljanja usredotočen na rješavanje problema i primjenjiv je u svim područjima ljudskoga djelovanja. Prema (MESTD, 2018), računalni način razmišljanja kombinira raščlanjivanje problema na manje dijelove koji se lakše rješavaju, identifikaciju uzoraka i opća rješenja, generalizaciju i algoritamski način rješavanja problema, kao i ocjenu rješenja.

U posljednjem razdoblju znanstvenici i stručnjaci u RS napisali su brojne radove ističući opravdanost uvođenja informatike, a time i programiranja, u osnovne škole. Tako se u radu Ivanović i Antonijević (2020) analizira trenutačno stanje i perspektive na ovom području. Prijedlozi kako bi trebala izgledati „Škola sutrašnjice" i kritike na račun kasnog uvođenja predmeta Informatika i računarstva kao obveznog predmeta u osnovne škole dani su detaljno u Hilčenko (2017). Nikolić i Subotić (2018) predstavljaju model potpore nadarenim učenicima u području programiranja, dok je Bujić (2020) analizirao učenje programiranja temeljeno na digitalnim igrama.

Unatoč sve većem broju radova u RS-u koji se bave važnošću uvođenja programiranja u nastavne programe osnovnih škola, prema našim saznanjima, u RS-u ne postoji istraživanje koje dalje proučava najbolji pristup učenju programiranja za početnike, posebno odnos između tekstualnih i vizualnih programskih jezika.

Metoda

Cilj i problem istraživanja

Uvođenje Informatike i računalstva kao obveznoga predmeta za učenike od petog do osmog razreda osnovne škole te programiranja kao osnovnog sadržaja kurikula predstavlja veliki napredak u osnovnom obrazovanju u Republici Srbiji. Cilj studije je procijeniti motivaciju za učenje i postignuće učenika, stečena učenjem programskih jezika. Istraživanjem, predstavljeno u ovom radu, uspoređivano je učenje tekstualnoga jezika (Python) u odnosu na jezik vizualnoga programiranja (Scratch) kod učenika osnovne škole.

Hipoteze

Na temelju analize relevantne literature, predstavljene u teorijskom dijelu rada, u ovoj su studiji ispitane sljedeće hipoteze:

H1: Učenici imaju pozitivan stav prema učenju vizualnoga i svojega prvog tekstualnog programskog jezika.

H2: Učenici imaju pozitivan stav prema svladavanju jednostavnih algoritamskih zadataka tijekom učenja programskoga jezika.

H3: Učenici imaju pozitivan stav prema svladavanju osnovnih elemenata (ulazno-izlazne naredbe, relacijski operatori, kontrola protoka) tijekom učenja programskoga jezika.

H4: Učenici imaju pozitivan stav da znanje iz matematike utječe na bolje svladavanje njihova prvoga tekstualnog programskog jezika.

H5: Razlike u postignućima učenika i njihov interes za predmet utječu na učenje programskoga jezika.

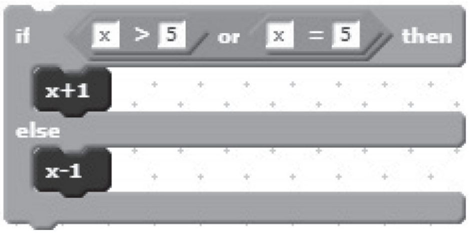
Uzorak

U ovo je istraživanje bilo uključeno pedeset i osam učenika. Oni su učenici šestog razreda osnovne škole „Dositej Obradović" iz Sombora, Republika Srbija. Dob sudionika kretala se od 12 do 13 godina (53,3 % dječaka i 46,7 % djevojčica). Od sedamdeset i sedam učenika petog razreda u školskoj 2016./17. godini, njih 58 odabralo je izborni predmet Informatika i računarstvo. Kako bi se udovoljilo potrebi za istraživačkom etikom, svako sudjelovanje u istraživanju bilo je dobrovoljno, a ocjene učenika nisu ovisile o rezultatima ispitivanja.

Instrument

Naša studija koristila je eksperimentalni dizajn s: predtestom, posttestom i završnim testom. Korišteni instrument istraživanja bio je upitnik. Pitanja sadržana u predtestu i posttestu uključuju nastavni materijal koji su učenici izučavali na tečajevima Scratcha i Pythona. Pitanja i njihov broj definirana su u dogovoru s učiteljem, a u vezi s odgovarajućim nastavnim planom i programom. Ona su u nekim aspektima ograničena dobom učenika. Završni test sadrži izjave o motivaciji za učenje programiranja i vještinama koje su učenici stekli učenjem programskih jezika. U testovima prije i poslije pohađanja tečajeva, znanje sudionika provjereno je u odnosu na sljedeće teme: deklariranje varijabli, ulazne i izlazne funkcije, algoritmi, logički operatori, uvjeti i petlje. Učenici su, tako, trebali prepoznati: vrstu deklaracije ako je naveden tip brojne varijable, kao i metodu i vrstu algoritama (opisanih pseudojezikom i blok-dijagramom). Svako je pitanje sadržavalo tekstualni opis (ponekad i grafičku sliku, Sl. 1) i tri ponuđena odgovora (od kojih je samo jedan točan). Predtest (i posttest) sastojao se od 10 pitanja, a neka od njih (vezana uz uvjetne izraze i ponavljanje) navedena su u nastavku.

1. *Što ukazuje sljedeći primjer upotrebe naredbe if_then?*
   IF x≥5
   then x + 1
   else x-1;
   a) Ako je x veći od 5, povećajte broj x za 1, a ako je x manji od 5, smanjite broj x za 1.
   b) Ako je x veći ili jednak 5, povećajte broj x za 1, a ako je x manji od 5, smanjite broj x za 1.
   c) Ako je x veći ili jednak 5, povećajte broj x za 1, a ako je x manji ili jednak 5, smanjite broj x za 1.



Slika 1. Primjer naredbe if_then

2. *Koristeći naredbu FOR predstavljeni primjer:*
   FOR i = 1 TO 15
   a) ispisuje brojeve od 1 do 15
   b) ciklična petlja se ponavlja 15 puta
   c) zbraja brojeve od 1 do 15

Konačni test sadržavao je usporedne izjave o Scratchu i Pythonu (Tablica 5) i pet ponuđenih odgovora (Likertova skala).

Za obradu prikupljenih podataka korištene su sljedeće statističke metode: deskriptivne statističke mjere (mjere središnje tendencije, mjere varijabilnosti, parametri raspodjele) i mjere statističkoga zaključka (Pearson Chi-Square test i Spearmanova relacijska analiza).

Postupak

Učenje programskih jezika provedeno je prema Kalelioğlu (2015), Erol i Kurt (2017), dok se analiza rezultata temeljila na Kalelioğlu (2015). Svi učenici koji su sudjelovali u ovom istraživanju tijekom školske 2016./17. godine učili su programski jezik Scratch. Tečaj se realizirao jednom tjedno (45 minuta) tijekom 17 tjedana u razdoblju od siječnja do svibnja. Učenici su imali jedan školski sat tjedno (45 minuta) u skladu s Nastavnim planom i programom za učenike osnovne škole (kao dio izbornoga predmeta Informatika i računarstvo). Također, na istom principu, u školskoj godini 2017./18., od siječnja do svibnja organiziran je tečaj programskoga jezika Python. Međutim, ovaj se programski jezik izučava kao dio obveznoga predmeta Informatika i računarstvo. Nije bilo neočekivanih događaja ili poteškoća s postupkom istraživanja. Slika 1 ilustrira model istraživanja ove studije. Sadržaj materijala i aktivnosti vezanih uz tečaj predstavljeni su u Tablici 1 (Erol i Kurt, 2017).

Slika 2.

Tablica 1.

Rezultati

Rezultati su dani u Tablici 2, Tablici 3 i Tablici 4.

Tablica 2.

Tablica 3.

Tablica 4.

Tablica 5.

Rasprava

H1: Iz Tablice 1 (stupac "Mean" za odjeljak "Scratch" i "Python") može se vidjeti da su učenici zadovoljavajuće ocijenili proučavanje obje programerske paradigme u odnosu na to koliko su interesantne. Detaljan uvid u dobivene rezultate pokazao je da je, za stupac „Scratch", u izjavama S1 (Tablica 5), 40 od 58 učenika (od kojih je 5 valjanih) odabralo jednu od vrijednosti od 3 do 5 na Likertovoj skali, dok je za programski jezik Python 38 od 58 učenika (51 valjano) odabralo isti raspon, tj. neki od odgovora označenih kao 3, 4 ili 5. Također, promatrajući istu tablicu i stupac S4, može se zaključiti da učenici uglavnom smatraju da ih učenje tekstualnih i vizualnih programskih jezika može dodatno motivirati za daljnje proučavanje programiranja. Sukladno tome, na temelju odgovora na tvrdnje S1 i S4, potvrđuje se hipoteza 1. Potvrda ove hipoteze također se može temeljiti na činjenici da je od 77 učenika petog razreda u školskoj 2016./17. godini 58 njih odabralo izborni predmet Informatika i računarstvo (unutar kojeg su se upoznali s osnovama informacijsko-komunikacijskih tehnologija i učili programski jezik Scratch). Ti su zaključci u skladu s rezultatima dobivenim u studijama prikazanim u brojnim radovima (Hromkovič, Kohn, Komm, i Serafini, 2016; Mladenović, Boljat, i Žanko 2018; Noone i Mooney, 2018). Svakako, na temelju proučene literature, mogao se i očekivati pozitivan stav učenika prema učenju vizualnih programskih jezika. S druge strane, rezultati dobiveni za odnos učenika prema proučavanju tekstualnoga programskog jezika mogu se smatrati posebno ohrabrujućima s obzirom na važnost učenja ove vrste jezika prema Lindstrom (2005), Ranum, Miller, Zelle i Guzdial (2010).

H2: Stupac „M" u Tablici 3 predstavlja srednju vrijednost odgovora vezanih za poznavanje pet tematskih područja (I/O upute, operator veze, ugrađene funkcije, algoritmi, kontrola protoka). Točni odgovori označeni su s 1, i obrnuto, netočni odgovori dobili su nultu vrijednost. Budući da je većina učenika točno odgovorila na pitanja vezana za područje algoritama u posttestu, može se zaključiti da studenti imaju pozitivan stav prema svladavanju jednostavnih algoritamskih zadataka tijekom učenja programskoga jezika. Stoga se hipoteza 2 potvrđuje. Dobiveni rezultat, iako se slaže s većinom gore navedene literature u uvodnom dijelu, može se smatrati donekle drugačijim od istraživanja danog, na primer, u radu Kalelioğlu i Gülbahar (2014), gdje nije pronađena veza između vještina rješavanja problema i programiranja, pomoću programskoga jezika Scratch.

H3: Analogno tome, rezultati koji se odnose na druga tematska područja, prikazana u Tablici 3, ukazuju da je hipoteza 3 također potvrđena. S druge strane, nešto lošiji rezultati dobiveni su za poznavanje iteracija (pomoću for petlje), što ukazuje na to da studenti nisu dovoljno svladali ovo područje pomoću tekstualnoga programskog jezika. To sugerira da je potrebno poboljšati metode učenja za učenje koncepta iteracija. Tablica tri također pokazuje da za sva tematska područja postoje statistički značajne razlike između prije i poslije testa. Zajedno s rezultatima predtesta, ovi rezultati podrazumijevaju da studenti bolje svladavaju pet navedenih tematskih područja kada koriste tekstualne programske jezike. Dobivena potvrda hipoteze, koja prema našim najboljim saznanjima nije istražena u suvremenoj literaturi, temelj je daljnjih istraživanja. To bi moglo uključivati definiranje i testiranje analogne hipoteze koja bi obuhvatila i druga važna tematska područja programiranja.

H4: Ova hipoteza potvrđuje se na osnovi danih odgovora na tvrdnje S2b (Tablica 5) iz završnoga testa. Rezultati iz Tablice 4 (redak S2) otkrivaju da učenici smatraju da znanje iz matematike utječe na bolje svladavanje njihova prvog tekstualnog programskog jezika. Međutim, analiza korištenjem Pearsonova Chi-Square testa pokazala je da postoji statistički značajna razlika između odgovora na ove dvije tvrdnje (Pearsonov koeficijent: 37,36, $p < 0{,}002$; vidjeti Tablicu 4). Na taj su način učenici zauzeli jedinstveni stav da su određene vještine iz matematike potrebne za svladavanje tekstualnoga programskog jezika (Python), te se može zaključiti da je hipoteza potvrđena poglavito za Python kao prvi programski jezik. Dobiveni rezultati mogu se smatrati zadovoljavajućima, posebno ako uzmemo u obzir da učenje programskih jezika podrazumijeva matematičko rješavanje problema (Popat i Starkey, 2019).

H5: Iako se brojni autori poput Bruce i sur. (2004), Fincher (1999), Robins, Rountree, i Rountree (2003) bave čimbenicima koji utječu na sklonost učenika prema programiranju, ovaj rad uvodi novi pristup analizirajući ovisnost interesa za učenje, sklonosti prema samostalnom učenju i poteškoća u svladavanju jezika u odnosu na dvije programerske paradigme. Stoga se postavlja sljedeća hipoteza: „Razlike u postignućima učenika i njihov interes za predmet utječu na učenje programskoga jezika“. Ova se hipoteza potvrđuje kroz analizu više aspekata. Prvo - promatranjem odgovora na tvrdnje S5a i S5b (tablica 5) iz završnoga testa prikazanoga u Tablici 4. Iako su učenici ocijenili ove izjave u odnosu na programski jezik Python nešto niže nego prethodne (S1 - S4, tablice 2 i 4), srednje vrijednosti mogu se smatrati zadovoljavajućima. Na temelju odgovora učenika i dobivenih rezultata može se vidjeti da je postojala statistička značajna razlika između ove dvije izjave (Pearsonov koeficijent: 50,46, $p < 0{,}001$; vidjeti Tablicu 4). Odnosno, učenici vjeruju da se programski jezik Scratch usvaja lakše od Pythona. Drugo - analizom odgovora na tvrdnje S3a i S3b (Tablica 5). Također, između ova dva pitanja postoji statistička značajna razlika (Pearsonov koeficijent: 40,04, $p < 0{,}001$; vidjeti Tablicu 4). Tako su učenici koji su učili Python samostalnije radili na praktičnim vježbama.

Zaključak

Pregled relevantne literature dane u ovom radu pokazuje da je vizualna paradigma najprikladnija za studente koji uče prvi programski jezik. Također, pokazalo se da je učenje prvog tekstualnoga programskog jezika brže i „opuštenije“ ako su čenici već učili vizualno programiranje. Budući da se Scratch i Python smatraju jednim od najpopularnijih i najjednostavnijih programskih jezika za vizualno i tekstualno programiranje, respektivno, i dio su kurikula u Republici Srbiji, istraživanje znanja i stavova učenika u ovom radu realizirano je u vezi s tim programskim jezicima. Detaljna analiza trenutačne literature ukazala je na druge važne aspekte povezane s učenjem programiranja (među programerima – „početnicima“) koje je trebalo istražiti (povezanost s algoritamskim razmišljanjem, matematičkim znanjem, utjecajem ostalih aspekata (poput neovisnog učenja, poteškoća, zainteresiranost itd.)).

Rezultati istraživanja, prikazanog u radu, pokazali su da je učenicima bilo lakše svladati vizualni programski jezik Scratch nego tekstualni programski jezik Python. Učenici su pokazali zadovoljavajući stupanj napretka u učenju programskoga jezika Phyton stjecanjem znanja iz: korištenja I/O instrukcija, petlji, algoritama... Primijećeno je da su učenici počeli razvijati sposobnost računalnoga razmišljanja kroz svoje aktivno sudjelovanje u tijek njihove nastave (provođenjem vježbi programskih jezika na računalu).

Rezultati ovoga istraživanja su zadovoljavajući s obzirom na to da učenici nisu imali prethodnih iskustava s tekstualnim programiranjem, kao i s obzirom na njihovu dob, motivaciju itd. Uvođenje Informatike i računastva kao obveznog predmeta i programiranja u osnove škole u Republici Srbiji od velike jevažnosti. Škole su dobile odgovarajuću nastavnu opremu jer je u većini škola bila zastarjela. Učitelji informatike i računalstva stječu važniju ulogu, tj. imaju priliku djecu dovesti u svijet informacijske tehnologije jer njihov predmet nema više izborni status.

Budući će rad biti usmjeren na proširivanje istraživanja na dva načina. Jedan je uključivanje učenika sedmih i osmih razreda. Drugi je proširivanje broja ispitanih učenika, tako da istraživanje obuhvaća više škola. Također, još jedan budući smjer istraživanja bit će istražiti stavove nastavnika informatike prema istraživačkim problemima predstavljenim u ovom radu.