Australian Educational Computing, *2018*, *33*(1)

Development of Students' Programming Abilities with the Means of Non-Programming Disciplines and Activities

Razakh Sakibayev, Spartak Sakibayev, and Bela Sakibayeva

Zhetysu State University, Kazakhstan

Abstract

The article contains the results of the study conducted in order to discover the most effective ways of developing students' programming abilities with the means of non-programming disciplines and activities. The authors observe the learning practices and habits employed by students with highly developed programming abilities and compare these practices and habits with those employed by students with poor academic performance at programming lessons. Then they use the results of the observation to formulate theoretical and educational principles that can be applied to programming instruction for effective development of programming abilities. The authors have discovered the factors that have the most significant impact on developing students' programming abilities. The article describes interdisciplinary connections existing between computer programming, writing scientific essays, reading scientific texts and solving chess problems that have not been fully covered so far in scientific journals. Also, it specifies which type of mathematical problems is especially useful in developing students' programming abilities.

Keywords

programming abilities; mathematical abilities; interdisciplinary connections of computer programming; impact of chess programming abilities; innovations in computer programming teaching; spaced learning

Introduction

Development of computer programming abilities in students plays an important role in their academic achievements. Not only these abilities can be vital for students' academic success in the specific field of computer programming. They are also the powerful means for preparing students for academic success in other subjects as well. Computer programming is an effective vehicle for developing mathematical thinking and abilities. Programming abilities can be easily transferred to performing mathematical tasks. According to Owolabi, Olanipekun, and Iwerima (2014), there is a strong correlation between mathematical and programming abilities. Programming also develops student's thinking abilities, problem-solving and complex cognitive skills.

Despite the importance of computer programming as an academic discipline, its methodology as currently employed by the majority of high schools and universities is still far from being optimal and effective. This situation is mainly caused by teachers paying more attention to providing concrete computer-related knowledge and facts rather than developing general algorithmic and programming abilities and skills. And even if they dedicate time to developing programming abilities in students, they try to accomplish this task remaining within the confines of the programming discipline itself. That is, they try to develop these abilities exclusively using programming. Besides, the majority of programming teachers still use old-fashioned teaching methods which are unable to fulfil the students' potential completely.

So far there has not been a sufficient number of articles and studies dedicated to the problems mentioned in the previous paragraph. Such a situation has served as the main motivational factor behind this article. The authors are convinced that in the educational context the questions of developing programming abilities must be given a priority and that these abilities are most effectively developed only through the means of non-programming disciplines and activities. Also, they believe that the introduction of new teaching methods, such as so-termed "spaced learning", is a necessity for the effective organization of the educational process of computer programming lessons.

In this article, the authors present the results of their study of the issues concerning the development of students' computer programming abilities with the means of non-programming disciplines and activities, such as mathematics, science, writing and chess. The authors argue that programming abilities are best developed namely through these disciplines and activities. The main goal of the study was to discover the factors which have the most significant impact on developing students' programming abilities and then formulate hypotheses to serve as the theoretical foundation for the discovered factors. The authors also mention the usage of a learning method termed "Spaced learning" as an effective optional instructional method which can also have a significant positive impact on students' abilities. The theoretical principles formulated by the authors are substantiated with statistical data obtained during the experimental verification of the formulated hypotheses.

Literature Review

The problem of effective development of analytical and logical skills and abilities, which lay the foundation for programming skills and abilities to be based on, has always been the focus of a vast amount of researches carried out by educators and computer scientists around the world. These researches cover both the questions of analytical thinking developed by specific

academic disciplines and the relationships existing between the forms of thinking developed by various academic disciplines and activities.

Thus, Balmes (2017) covers the existing connections between mathematical and programming thinking and the impact which one type of thinking has on another. White and Sivitanides (2015) cover the results of their practical experiments dedicated to estimating the level of correlation between mathematical and general programming abilities and skills. Duran (2016) verifies his hypothesis about the positive impact of mathematical background on students' academic performance in computer programming. All these authors argue that mathematical activities, in general, have a positive impact on students' programming abilities. In this sense, their conclusions coincide with those given in the present article. But they do not specify which branches of mathematics and types of mathematical problems are especially useful in developing programming abilities. Also, they do not mention the importance of recreational mathematics in developing algorithmic skills. The present article fills up these gaps in the aforementioned papers and additionally provide more details on why mathematics is useful for developing programming abilities.

Maula (2015) discusses the correlation between students' reading and writing habits. Some of her results were used by the authors of the present article to formulate the hypothesis of their research. But Maula discusses the topic from the general point of view and does not cover the questions of the impact which reading and writing have on the development of logical thinking. On the contrary, the present article puts an emphasis on one specific kind of reading and writing, namely scientific reading and writing and analyzes the positive effect that they have on developing logical and, therefore, programming skills.

The works by Rosholm , Bjørnskov and Gumede (2017), Bart (2014), Sala and Gobet (2016), Trinchero and Sala (2016) and Gliga and Flesner (2013) are dedicated to the problems of estimating the level of relationship between chess studies and mathematical abilities. Also, their works deal with the impact which participation in chess has on the overall academic performance of students. However, they do not discuss the existing connections between chess and programming and the impact which chess thinking has on algorithmic abilities. These authors suggest that chess may be helpful in forming some particular educational traits, such as logical and analytical thinking and memory, but they have not discovered any direct way in which it affects the academic performance of students in mathematics. Unlike these works, the present article covers the specific questions of the relationship between chess and programming and argues that participation in chess directly and positively affects the development of mathematical and, therefore, programming abilities.

Seibel in (2009) presents interviews with prominent computer scientists and programmers. The book is not a scientific monograph but yet provides useful methodological insight into the internal intricacies of the profession, and advice on how to develop various skills and abilities necessary for success in computer programming. The present article contains attempts to build a formal theoretical foundation for some of the methodological advice given in (Ibidem).

The original study by Douglas Fields (2005) led to the formation of the "spaced learning" teaching method recommended by the present article for use in programming classes. It covers the general pedagogical and psychological factors which make the method an effective educational tool. However, Fields (Ibidem) does not discuss the specifics of applying this method to any particular academic discipline. In the present article, the authors demonstrate

how "spaced learning" can be applied to programming lessons in particular and discuss the benefits it can bring to programming education.

Materials and Methods

The research whose results are presented in this article was organized to have three major stages. At the first stage, the authors selected a target group of the study to be served as the source from which to collect experimental data for the research. The second stage was dedicated to collecting the experimental data from the target group. Finally, at the last third stage, the authors were engaged in building the theoretical foundation for the collected data and formulating the hypothesis of the research.

The target group of the study consisted of students from a local college studying in the same group and, therefore, having the same lesson schedule. All students from the target group spent an equal amount of time on both educational and leisure activities and came from the families with an equal level of income. The study's target group included both students with a high level of development of programming abilities and students, which had previously demonstrated a poor or average academic performance at computer programming lessons.

The initial phase at the stage of collecting the experimental data started with the authors accumulating all necessary and relevant information about the learning practices, habits and extracurricular interests and activities of the students with highly developed programming abilities and skills. The main method of collecting data at this initial phase was conducting interviews with academically successful students, their classmates and teachers and members of their families. After that, the authors applied the same procedure to obtain similar information concerning students with poor academic performance in programming lessons.

The next phase at the stage of collecting the experimental data was dedicated to making the students with poor academic performance in programming to employ the learning practices and habits used by the students with highly-developed programming abilities. At the same time, programming teachers were asked to apply the elements of the "spaced learning" method in their lessons. The authors observed the students' academic progress under the new learning strategy by keeping track of their scores.

At the third stage, the present study's authors formulated a series of hypotheses to cover the obtained experimental data. The hypotheses were to serve as methodological and pedagogical principles targeted at transformation and improvement of existing computer programming learning process in educational institutions and effective development of students' cognitive abilities in general, and programming abilities in particular. Since colleges around the world, on the whole, follow the same educational standards and instruction strategies the results and observations obtained from the local study group could be extrapolated to colleges in other regions as well.

Results

During the study, the authors tested their own methodological views on the development of programming abilities as well as analyzed existing learning practices and habits employed by students from the study's target group demonstrating a high level of these abilities. Also, the authors studied statistical data (Table 2) on the academic performance of the students with poor

or average academic performance in programming from the target group. Analysis of the obtained data allowed the authors to formulate five hypotheses which are given below.

Hypothesis 1 (**H1):** Solving mathematical problems is the most effective way of developing programming abilities. By solving a problem, the authors mean solving them in a traditional way without using a computer. The discipline of computer programming has mathematical foundations. It has originated as part of applied mathematics the process of writing source code is very similar to the process of proving a theorem. Operators and expressions in source code are mathematical formulae used to implement a mathematical algorithm, and the implementation of this mathematical algorithm requires mathematical ability, skills and knowledge. The most effective way of developing these traits in students is through solving mathematical problems and performing mathematical manipulations. It is necessary to highlight that the most important result of solving mathematical problems for programming students is not some new mathematical knowledge gained during the solution process but the process itself which develops mathematical and, therefore, programming thinking.

However, not all types of mathematical problems are suitable for the development of programming abilities. One example of such problems is geometrical problems which require and develop spatial imagination and synthetic reasoning. Though these traits are very useful in the overall formation of students' mathematical thinking and discipline, their specifics do not match the specifics of programming reasoning. From the point of view of the specifics of programming thinking the most effective problems for developing programming abilities are those taken from the fields of combinatorial mathematics, algebra and number theory. The problems from these mathematical fields have a "discrete" nature and imply following some discrete formal algorithms which make them an optimal choice for developing programming abilities. For example, programming students could be asked to use Horner's method to solve without writing a program an algebraic system of linear equations or try to simplify some complex and lengthy algebraic expression. Another good mathematical problem is to test a number for primality using Fermat's methods available in the course of elementary number theory.

Taking part in various mathematical recreational tasks as an addition to solving mathematical problems is also an effective method of developing programming abilities in students. For example, a programming teacher can dedicate five minutes of a lesson to students' mathematical "warm-up" in the form of performing simple mental binary arithmetical calculations. This activity develops memory and teaches students to emulate the "machine thinking". Another useful example is to introduce numerical games where a student is given a task is to encode or decode some text-based message using the means of binary arithmetic.

It is almost impossible to develop students' programming abilities without simultaneously developing their mathematical abilities as well. In this sense, mathematics presents the most important interdisciplinary prerequisite for computer programming. Computer programming skills cannot be further developed if a student does not demonstrate even the most essential mathematical reasoning and problem-solving skills. It becomes very difficult to have academically successful programming classrooms in a school or university where the level of efficiency and organization of mathematics instruction does not meet any plausible standard. So-termed math anxiety also serves as an obstacle for students in developing their programming abilities. On the contrary, the majority of students with a high level of programming abilities demonstrates an interest and inclination towards mathematical activities, for example performing complex mathematical manipulations and solving various

mathematical problems. Moreover, they tend to view computer programming as a natural continuation of their math lessons and consequently dedicate a significant amount of time to both disciplines. It becomes possible to suggest that programming abilities can be developed even more effectively using means of mathematics rather than programming itself. Butler Lampson said in [8] that "...physics and mathematics, like other respectable disciplines, require that you think clearly to succeed in them. That is why many successful computer people come from these fields."

Hypothesis 2 (H2): interest in science is another prerequisite for the effective development of programming abilities in students. Reading science textbooks and doing problems from them require logical and analytic reasoning, insight into the essence and structure of things and understanding the cause and effect relationships. All these traits have a mathematical nature and therefore, as is shown in H1, have a direct impact on developing programming abilities. Scientific textbooks provide students with details and knowledge concerning the technical and physical underpinnings, and principles of computer science and this knowledge enable them to make optimal choices in the process of writing source code of their computer programs. Reading scientific textbooks allows to get acquainted with the procedure and psychology of scientific discoveries and, therefore, the basics of the scientific method in general. Students learn about the effective and working approaches and optimal strategies and methodologies employed by scientists in dealing with various scientific and technical problems. Then they learn how to effectively apply these approaches and strategies and methodologies in their studies in order to improve their academic performance in various disciplines, including computer programming. Bill Gates, the founder of Microsoft software company admitted in [9] that in his youth he was influenced by popular scientific lectures delivered by Richard Feynman, an American theoretical physicist and Nobel Prize winner.

Students with high scores in computer programming lessons from the study's target group demonstrated a significant level of interest in watching popular science video and reading popular science articles taken both from printed books and the Internet.

Hypothesis 3 (**H3)**: Reading material on the history of computer technologies is important for developing programming abilities. Interest in the historical and evolutional aspects of computer technologies demonstrates student's inclination towards extending and furthering his current level of knowledge on the subject. In the process of reading students get acquainted with the problems standing before computer scientists and technologists at some period of time, methods and approaches used for solving these problems, and motivation and reasons behind technological decisions made. This new knowledge is helpful in developing student's programming abilities.

Students with highly developed programming abilities from the study's target group dedicated a significant amount of the extracurricular time to reading articles and watching documentaries on various topics from the history of computer technologies.

Hypothesis 4 (**H4)**: Solving chess problems and studies as an extracurricular activity helps developing students' programming abilities. Chess provides an optimal platform for testing various mathematical, data search, data analysis and decision-making algorithms. The game requires and develops essential combinatorial, analytic and manipulative skills which relate it to mathematics, and therefore, as is suggested by H1, computer programming. Solving a chess problem has a lot in common with constructing an algorithm for implementing a computer program. Both activities require a detailed analysis of available source data, a clear

understanding of the end solution, seeing a relationship between source data and the solution and skills in constructing a concise algorithm of the solution. Chess problems teach programming students to evaluate available material, and this is an essential programming technique used in solving many programming problems. Also, chess problems develop other traits necessary for accomplishing programming tasks, such as concentration and mental discipline. Rosholm, Bjørnskov and Mikkelsen (2017) suggest the domains of chess and mathematics are contextually close to each other. Bart (2014) argues that to understand and evaluate chess positions, you must take into account the different mobility patterns of the pieces, requiring fluid intelligence and concentration capacity. Sala and Gobet (2016) argue that chess-related abilities and skills can be effectively used to develop abilities and skills in academic disciplines as well. Trinchero and Sala (2016) demonstrates that chess can be used as effective tools for developing mathematical problem-solving skills. Also, Gliga and Flesner (2013) demonstrate the positive impact that chess training has on school performance, memory, sustained attention and creativity.

From the point of view of developing programming abilities, the most educational value comes from solving those chess problems which require selecting the most optimal and effective solution from a large number of existing ones. A student's programming ability will be developed effectively by their attempts to construct a compact and efficient chess plan for analysis of a big number of variations, testing their consequences, evaluating their effectiveness and in the end selecting the best move.

Majority of students with highly developed programming abilities from the study's target group participated in chess clubs in their extracurricular activities.

Hypothesis 5 (**H5)**: Writing small essays on scientific topics is useful for developing programming abilities. On the whole, writing prose is a process which by its structure and logic resembles the process of writing source code for a computer program. Both writing and computer programming are based on pattern recognition and involve strategic planning. Both activities imply the search for the most optimal and compact arrangement of given expressions with the purpose of describing some idea in the clearest and understandable form. In this sense, scientific writing is the type of writing which bears the closest resemblance to the process of computer programming. Writing an essay on a scientific topic requires an accurate description of some complicated scientific phenomena and presentation of material in the most compact form. And this is very similar to trying to implement a complex algorithm using as few lines of source code as possible. Jamie Zawinski states that there is an overlap between programming and writing prose (Seibel, 2009: 26).

Majority of students with highly developed programming abilities from the study's target group demonstrated skills in writing academic essays on scientific topics.

In addition to these five hypotheses, which, in the authors' opinion, cover the main factors affecting the development of programming abilities in students, the study has also discovered an optional condition which can have a positive impact on academic achievements in the field of computer programming. The condition is to employ the method of so-termed "spaced learning" at computer programming lessons. According to this method, learning content is to be repeated three times with two short breaks between repetitions. During these short breaks, students are made to perform some other activities called distractor activities (Fields, 2005). Spaced learning allows teachers to organize the presentation of programming material in the most effective way of using long-term memory creations and thus, affects the development of

programming abilities using the principles of neuroscience. Specifically to a computer programming lesson this learning method can be applied the following way. A teacher starts the lesson by presenting a learning content in a textual form, without using specialized programming and mathematical symbols. Then after the first short break, he presents the same learning content using some non-textual means, for example, visual means in the form of sequence diagrams. After the next short break, the teacher starts presenting the same learning content using the concrete means of a specific programming language. This methodology protects students from experiencing fatigue and loss of concentration. What is more important is that this method of "spaced learning" allows for making computer programming lessons more attractive to students.

Practical experiments in computer programming lessons performed at the stage of the verification of the formulated principles demonstrate the general correctness of the obtained hypothesis.

Though it is necessary to notice that the first positive educational results appeared approximately one month since the authors introduced their hypotheses with experimental purposes to programming instruction of students from the study's target group. The experimental verification of the hypotheses was held in a local college which follows the standardized credit system with a 15-week semester and uses the grading system shown in Table 1.

Table 1. Grading system used in a local college from the target group

Letters	Range	Percentage	Description of the grade
A	4.0	95–100	Excellent
A−	3.67	90–94	Excellent
B+	3.33	85–89	Good
B	3.0	80–84	Good
B−	2.67	75–79	Good
C+	2.33	70–74	Satisfactory
C	2.0	65–69	Satisfactory
C−	1.67	60–64	Satisfactory
D+	1.33	55–59	Satisfactory
D	1.0	50–54	Satisfactory
F	0	0–49	Unsatisfactory

The experimental group consisted of 8 students with a previous average academic performance at computer programming lessons graded as "C-" (Satisfactory). The stage of the experimental verification of the hypothesis in programming lessons started at week 6 of an academic year and finished at week 14 of the same academic year. The authors studied the progress in the academic performance of the students during the experiment by tracking their grades which were given according to the system shown in Table 1. The results of the students' academic performance during the experiment are given in Table 2.

Table2. The students' academic performance during the experiment

Week No	Student 1	Student 2	Student 3	Student 4	Student 5	Student 6	Student 7	Student 8
6	C-	C-	C-	C-	C-	C-	C-	C-
7	C-	C	C	C+	C+	C-	C	C+

8	C-	C	C	C	C	C-	C	C+
9	C-	C	C+	C+	C+	C	C	C
10	B	B-	B-	B	C+	C	B+	B
11	B+	B	B-	B	C+	C+	B+	B-
12	B+	B+	B+	B	B-	B-	B	C+
13	B+	A-	B+	B-	B	B	B-	B
14	A-	A-	B+	B	A-	A	B	B+

As is shown in Table 2 approximately one month after the experiment started the students' academic performance in computer programming lessons showed the first signs of improvement. Starting from week 10 the academic performance of the target group increased significantly, and at the end of the semester, half of the students were given the "A" or "A-" (Excellent) grades. Moreover, in tandem with the improvement of academic performance at computer programming lessons, the students from the target group also saw a similar increase in academic performance in other disciplines as well, such as mathematical and physical sciences. Additionally, the students from the study's target group reported experiencing the increased level of interest and motivation towards computer programming lessons in particular and the whole academic process in general.

"Spaced learning" when applied specifically to computer programing education is an effective method for increasing the educational value of lessons. Repetition of a learning content several times using a different form of presentation on each repetition enables students to absorb it using the minimal amount of effort.

Discussion

Finding effective educational means for the development of students' programming abilities has been one of the central points of research on computer science educators. The majority of the research so far has been focused on developing programming abilities with the content of the programming lessons themselves. The authors of this article, however, argue that the most effective way of developing such abilities is through the means of non-programming disciplines and activities, such as mathematics, science, chess and writing.

It has been the opinion of many educators and scientists that the study of mathematics directly influences a student's interest and academic performance in computer programming. However, their works only state the existence of a general correlation between mathematics and computer programming without specifying which particular areas of mathematical science and types of mathematical problems are useful for developing programming abilities. The present article seeks to address these shortcomings and provide the necessary details about these particular areas and problem types.

The authors do not agree with the results of existing research which state that there is no direct relationship between participation in chess and mathematical and, therefore, computer programming academic performance. On the contrary, the article concludes that chess can be used as an effective educational tool for enhancing programming abilities in students.

Another topic considered in the present article and not yet enough covered in scientific literature is relationships existing between computer programming and such non-programming activities as reading popular scientific articles and writing scientific essays. The authors argue

that these activities bear a lot in common with programming and can effectively stimulate the development of students' programming skills. Also, the authors discuss another rare topic of modern researches – the application of the "spaced learning" method specifically for computer programming lessons.
The majority of modern research dedicated to computer programming education tend to focus more on the problems of introducing the latest technological achievements (such as interactive smart boards, smartphones and tablet PCs) to the educational process rather than the questions of effective development of students' algorithmic thinking and skills. The authors of the present article, however, put an emphasis on algorithmic thinking and skills as the main target of computer science education.

Conclusion

The research conducted by the authors confirm the opinion of many educators, computer scientists and psychologists that computer programming abilities can be effectively developed through the means of non-programming activities such as mathematics, science, chess problems and writing. Moreover, the authors argue that the former activities provide much more efficient means for the effective development of programming abilities than the discipline of programming itself. The theoretical principles proposed in the article, when applied to the instruction process, enable students to develop not only their programming skills and abilities, but also can improve their overall academic achievements, logical and analytic thinking, memory and cognitive abilities.

Computer programming lessons, and therefore, the process of development of students' programming abilities are most effective when they are held as part of a learning method termed "spaced learning". This methodology provides the most effective means for absorption of learning content and, thus, is an efficient mechanism for developing learner's memory and abilities.

The methodological strategies proposed by this article can be used for effective organization of computer programming lessons in high schools and initial courses of universities. Also, the methodology can be applied by teachers of mathematical and physical sciences as a general-purpose educational tool for increasing students' academic performance, skills and motivation.

The main limitation of the results obtained in the present article is that they are oriented exclusively towards developing analytical and logical sides of thinking. However, it is often the case when a computer programmer faces a problem whose solution requires spatial imagination and synthetic skills. The question of the development of such synthetic skills lies beyond the scope of this article.

Computer programming, as any other academic discipline, cannot evolve without close interdisciplinary connections. And often it is the theoretical and practical advances in other scientific disciplines that make it possible the progress in computer programming instruction methods.

Acknowledgement

The authors wish to express acknowledgement to Ing. Miloš Ulman, PhD, from the Prague Agrotechnical University for his seminars and lectures, which provided methodological ideas for the present article.

References

Ali, P, Ali, S., Farag, W. (2014) An instrument to measure math attitudes of computer science students. *International Journal of Information and Education Technology*, 4(5): 459-462.

Balmes, I. L. (2017) Correlation of mathematical ability and programming ability of the computer science students. *Asia Pacific Journal of Education, Arts and Sciences*, 4(3): 82-88.

Bart, W. (2014) On the effects of chess training on scholastic achievement. *Frontiers in Psychology*, 5: 762. doi: 10.3389/fpsyg.2014.00762.

Brookshire, R., Crews, T., Brown, III H. (2009) Student success in a university introductory networks and telecommunications course: contributing factors. *International Journal of Information and Communication Technology Education*, 5(1): 53-61.

Duran, I. L. (2016) The role of mathematics background in the performance of bscs students in computer programming subject. *International Journal of Multidisciplinary Research and Modern Education (IJMRME),* 2(1): 147-150.

Fields, R. D. (2005). *Making Memories Stick. Scientific American, 292(2): 58–63.*

Gliga, F., Flesner, P. I. (2013) **Cognitive benefits of chess training in novice children.** *Procedia - Social and Behavioral Sciences*, 116(21): 962-967.

Lammers, S. (Eds.). (1986) Programmers at work. Redmond, WA: Microsoft Press.

Markoff, J. (2009) Gates puts Feynman lectures online. Retrieved from https://tierneylab.blogs.nytimes.com/2009/07/15/gates-puts-feynman-lectures-online/

Maula, I. (2015) The correlation between students' reading habit and their ability of writing narrative text (a correlational study on the eleventh graders of SMAN 1 Kajen Pekalongan in the academic year of 2014/2015), doctoral dissertation, Universitas Negeri Semarang, Semarang, Indonesia.

Owolabi, J., Olanipekun, P., Iwerima, J. (2014) Mathematics ability and anxiety, computer and programming anxieties, age and gender as determinants of achievement in basic programming. *GSTF International Journal on Computing (JoC)*, 3(4):109-113.

Rosholm, M., Bjørnskov, M., Mikkelsen, K. (2017) Your move: the effect of chess on mathematics test scores. *PLoS ONE*, 12(5): e0177257. doi: 10.1371/journal.pone.0177257

Sala, G., Gobet, F. (2016) Do the benefits of chess instruction transfer to academic and cognitive skills? A meta-analysis. *Educational Research Review*, 18: 46–57.

Seibel, P. (2009) Coders at Work: Reflections on the Craft of Programming. New York, NY: Apress.

Trinchero, R., Sala, G. (2016) Chess Training and Mathematical Problem-Solving: The Role of Teaching Heuristics in Transfer of Learning. *Eurasia Journal of Mathematics*, *Science & Technology Education*, 12(3): 655–668.

White, G., Sivitanides, M. (2015) An empirical investigation of the relationship between success in mathematics and visual programming courses. *Journal of Information System Education*, 14(4): 409-416.