Journal of Theoretical Educational Science, 14(4), 570-585, October 2021
Kuramsal Eğitimbilim Dergisi, 14(4), 570-585, Ekim 2021
[Online]: http://dergipark.org.tr/akukeg
DOI number: http://doi.org/10.30831/akukeg.892869

Algorithmic Thinking Skills without Computers for Prospective Computer Science Teachers

Bilgisayar Öğretmen Adayları için Bilgisayar Kullanmadan Algoritmik Düşünme Becerileri

Çetin GÜLER[*]

Received: 08 March 2021 **Research Article** **Accepted:** 11 May 2021

ABSTRACT: Algorithmic Thinking (AT) skills may be considered one of the necessary skills for everyone in the new century. Individuals and societies live in a time when they cannot avoid using information communication technologies (ICT). Hence, they need to be ICT literate. Teachers shall play an important role both for individuals and societies to address this need. However, are they ready to play such a role? This paper presents a suggestion for teacher training to help teachers how to play the role. The paper suggests including an elective course under the title AT into the computer science teacher training curricula and aims to present the course, progress and views of the students who took the course. A mixed-method approach was followed for the current study. The study was carried out with the participation of sixth-semester students of a Computer Education and Instructional Technology Department of an Education Faculty who may be considered prospective computer science teachers. Twenty-eight students were enrolled in the elective AT course offered for the period. Within the scope of the study, development of the course curriculum, instructional process of the course and evaluation of the course, in line with students' views and exam scores, are presented. Findings of the research suggest that the students find the Algorithmic Thinking course helpful for them to acquire some algorithmic thinking skills as well as some other academic and life-related thinking abilities. Also, the course may be considered as a necessary course particularly for the training process of computer science teachers. In addition, the students think that the offered course was effective and beneficial.

Keywords: Algorithmic thinking, computational thinking, computer science, course, prospective teacher.

ÖZ: Algoritmik düşünme becerileri bu yüzyılda yaşayan herkes için gerekli bir özellik olarak düşünülebilir. Günümüzde bireyler ve toplumlar bilgi iletişim teknolojilerini (BİT) kullanmak durumunda kalıyorlar. Bu durum onların BİT okuryazarı olmalarını gerektirmektedir. Bu gerekliliğin karşılanmasında öğretmenlere önemli görev düştüğü söylenebilir. Bu çalışmada, bilişim teknolojileri öğretmen adayları için Algoritmik Düşünme adıyla sunulan seçmeli bir ders ele alınmıştır. Çalışmada karma yöntem yaklaşımı izlenmiştir. Çalışma, bir Eğitim Fakültesi Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü altıncı yarıyılında öğrenim gören bilgisayar bilimleri öğretmen adayları ile gerçekleştirilmiştir. Söz konusu dönem için açılan seçmeli dersi 28 öğrenci almıştır. Çalışma kapsamında öğrencilerin görüşleri ve sınav puanları doğrultusunda dersin öğretim süreci ve dersin değerlendirilmesine yer verilmiştir. Bununla beraber ders programının gelişim sürecine de yer verilmiştir. Araştırmanın bulguları, öğrencilerin bazı algoritmik düşünme becerilerinin yanı sıra diğer akademik ve yaşamla ilgili düşünme becerilerini kazanmaları noktasında Algoritmik Düşünme dersini yararlı bulduklarını göstermektedir. Ayrıca ders, özellikle bilgisayar bilimleri öğretmenlerinin yetiştirilmesi açısından gerekli bir ders olarak değerlendirilebilir. Ek olarak öğrencilerin, sunulan dersin etkili ve faydalı bir ders olduğu yönünde görüşler belirttikleri de belirtilebilir.

Anahtar kelimeler: Algoritmik düşünme, bilgi-işlemsel düşünme, bilgisayar bilimleri, ders, öğretmen adayı.

[*] *Corresponding Author*: Assoc. Prof. Dr., Van Yüzüncü Yıl University, Van, Turkey, cetin@yyu.edu.tr, https://orcid.org/0000-0001-6118-9693

Citation Information
Güler, Ç. (2021). Algorithmic thinking skills without computers for prospective computer science teachers. *Kuramsal Eğitimbilim Dergisi [Journal of Theoretical Educational Science]*, 14(4), 570-585.

ISSN: 1308-1659

Algorithmic Thinking

Learning to code, developing programs and developing computational thinking (CT) skills are considered as qualifications that today's individuals should have in order to cope with the situations they face and may encounter (Kalogiannakis & Papadakis, 2017b). In the context of information communication technologies (ICT) and education, particularly in recent years, one of the centers of interest is computational thinking (CT). CT may mainly be considered to be related to AT and programming (Angeli & Valanides, 2020; Citta et al., 2019; Zhang & Nouri, 2019; Zhao & Shute, 2019). In fact, CT and AT are concepts that are interchangeable/equivalent and sometimes used instead of each other (Hromkovic et al., 2017; Pérez-Marín et al., 2020). The use of computational algorithmic thinking (CAT) concept (Thomas et al., 2017) is also considered the same. On the other hand, using a proper term that can encompass CT can be tricky for some cultures. For example, in Turkish, there are many terms that are similar but not the same for the term CT. Even though AT was presented as a factor under CT, AT can encompass all attributes of CT. Since some scholars use them interchangeably, AT can be a more familiar and widely accepted term, and AT will be used to encompass all of these terms for the purpose of this paper.

The term algorithm is named after al-Khwārizmī (Knuth, 1985). However, AT is an old form of thinking extending back to Pythagoras and the Jewish Kabbalah (Essl, 2007). For most people, the word algorithm can suggest something about computer science or programming. However, AT is not merely a way of thinking associated with computer science or programming (Hubalovsky & Korinek, 2015; Mayer, 1981). AT can be considered more of a form of thinking associated with problem-solving (Aho, 2012; Wing, 2006). Futschek (2006) defines AT as a pool of skills that can be used to understand and develop algorithms. It can be argued that lifelike experiences should be utilized in creating this skill pool (Hubalovsky & Korinek, 2015). AT can be described as high-level intellectual actions involving understanding problem situations, finding systematic and generalizable solutions to solve the problems, checking the accuracy and efficiency of the solutions and expressing these solutions step by step.

AT and Education

It was claimed that AT would affect every individual in every area of society (Wing, 2008). In this context, the search for inclusion of AT in educational programs continues (Dagiene et al., 2017; Grover et al., 2015). In many countries, including the USA and the UK, there are endeavors to include AT in education curriculums, particularly as an important part of the K-12 level curriculums (Dagiene et al., 2017; Papadakis & Kalogiannakis, 2019). When the place and the importance of ICT are considered today, being an ICT literate society is a necessity (Galezer et al., 1995; Gao et al., 2011). Teachers can be considered the most important actors in meeting this necessity by being effective role models (Gao et al., 2011). The first condition that teachers can fulfill this role is that teachers should be ICT literate themselves (Ayaz et al., 2015). Having AT skills can be considered as a prerequisite for ICT literacy (Galezer et al., 1995). Therefore, the teachers who will guide their students to gain AT skills should have AT skills. According to Cooper et al. (2000), many university students (including students in computer science) do not have AT skills. Despite the fact that it is twenty years old, Cooper's claim may remain valid for some students. When considering that the basic requirement for computer programming is AT (Bryant, 2017), it can be claimed that the basic requirement for acquiring ICT literacy is to acquire AT skills (Aho, 2012; Milkova & Sevcikova, 2017). Therefore, it is proposed to provide training for the development of AT skills before the training in the context of algorithms and program development (Ham, 2015; Hubálovský, 2013).

Including AT teaching into the curricula for all educational levels may be considered valuable (Kalogiannakis & Papadakis, 2017a, 2017b; Papadakis & Kalogiannakis, 2019). Most of the arrangements and updates within K-12 curriculums related to AT are associated with coding and computer program development in various countries (Dagiene et al., 2017). It is known that teaching programming is a challenging process (Grover et al., 2015). However, it is more difficult to teach the programming concept and structure than to teach a programming language in this process (Athanasiou et al., 2017). Brusilovsky et al. (1997) and Hubálovský et al. (2010) pointed out that a traditional approach is often used in teaching programming. In this conventional approach, problem situations are often used, which require the number and symbols to be processed through a general-purpose programming language. Even if the execution process of this approach involves algorithms, the main purpose is to teach a programming language (Galezer et al., 1995). Consequently, students do not acquire AT skills due to their intense effort to learn the programming language and inappropriate teaching practices (Grover et al., 2015; Hubálovský et al., 2010; Plerou et al., 2017). This approach is not effective particularly for younger students (Brusilovsky et al., 1997; Dagiene et al., 2017). Hence, students may find CS confusing and difficult (Papadakis, 2020) and grow some negative attitudes. Students who are expected to learn programming skills should first have AT skills (Hubálovský et al., 2010; Hunter, 1987) to address these difficulties. So, learning process of a programming language can be realized more effectively. From this point of view, it can also be argued that individuals with AT skills are more likely to be ICT literate.

As aforementioned, one of the most important factors for the students to gain AT skills is teachers. Teachers' beliefs, perceptions and attitudes related to AT influence both their practices and teaching approaches of AT (Kordaki, 2013). Therefore teachers, particularly computer science teachers should be trained to have AT skills and to have positive beliefs, perceptions and attitudes towards to the AT. Unfortunately, it can be claimed that the prospective computer science teachers graduate without having sufficient AT skills. Additionally, it is also stated that there are few studies on AT (Katai, 2015; Plerou et al., 2017) and no compromise on how AT skills can be gained (Grover et al., 2015; Milková, 2005; Milkova & Sevcikova, 2017; Silapachote & Srisuphab, 2017).

Purpose and the Significance of the Research

In the light of the aforementioned literature, a need to include AT in teacher training processes, particularly for prospective computer science teachers reveals itself. The need for the studies that are examining how to teach AT skills remains unsatisfied as well. In order to meet these needs, an elective course offered under the title of AT for prospective computer science teachers. The aim of the work presented in this paper is to examine this course. The work presents the curriculum development of the course, the instructional process of the course, and the evaluation of the course in the light of students' views and by exams' scores (the midterm and the final exams). Since there is a very limited number of studies subjecting an AT specific course and training process for prospective computer science teachers and considers the aforementioned issues, this work may add meaningful contributions to the field.

Research Questions

"What are the views of the students about the AT course?" is the research question of this study. Sub questions of the study are as follows: (i) Was there a difference in exams' scores averages of the students? (ii) What are the views of the students about the AT course objectives? (iii) What are the views of the students about the AT course content? (iv) What are the views of the students about the AT course implementation? (v) What are the views of the students about the AT course evaluation? (vi) What are the views of the students about the AT course effect?

Method

A mixed method approach was followed and sequential explanatory design was conducted for the current study. The qualitative data were collected after the implementation of the course to both contributing quantitative findings and evaluation of the course. Quantitative part was designed as one group quasi experimental study. The details of the process of the research are given in the sections Participants and Context, AT Course Curriculum Development and Procedure.

Participants and Context

This study was carried out with the participation of sixth semester students of Computer Education and Instructional Technology Department of an Education Faculty. The participants may be considered as prospective computer science teachers. Twenty-eight students were enrolled in the selective AT course opened for this period. Ten of these students are female (35%), 18 are male (65%). However, the number of students who indicated their views on the course was 20. Some of the lessons that these students have to take are Programming Language I and II, Internet Based Programming, Database Management Systems. Also, there is a compulsory lesson (Algorithm Design and Development) that directly includes teaching algorithms or AT in their new curricula. Although Programming Languages I and II courses require some algorithm-related skills, teaching algorithms or AT do not sufficiently occur in these courses neither. Besides, the new course of new curricula seems to be the Programming Language I course in a different name, in practice. As aforementioned (see Introduction) studies suggested that such conventional approaches may not be effective enough to gain AT skills. Therefore, a course subjected to AT was deemed necessary and beneficial for these students.

AT Course Curriculum Development

Curriculum development model known as Taba-Tyler or rational planning was used for the development of the AT course curriculum. Accordingly, a curriculum has four basic components. These components are the objectives, content, learning experiences and evaluation. The objectives of the course were determined and assessed with the help of five academicians and relevant literature. The main objective of the AT course is to contribute to the development of AT skills of prospective computer science teachers. Students who take this course are expected to acquire the skills to produce sequential, logical solutions to problems they may encounter in different subjects areas and in all areas of life (as suggested by: Papadakis, 2020; Papadakis & Kalogiannakis, 2019). However, in the light of the relevant literature, it has been evaluated that the use of any programming language during the development of these skills will adversely affect the process. Therefore, only the pseudo-codes (Grover et al., 2015) and flow charts have been decided to be used within the learning-teaching process. Since puzzles are supposed to contribute logical evolution and imagination (Milková & Hùlková, 2013) and AT skills (Hsu & Wang, 2018), selecting these questions from puzzles is deemed appropriate. For this purpose, puzzles from Algorithmic puzzles book written by Levitin and Levitin (2011) was decided to be used.

Procedure

The AT course is offered as a four-hour course (two lectures and two labs) each week for a term of 14 weeks. During the first week, general information about the course and explanations were given. Later on, students' opinions about developing algorithms were asked to determine readiness of the students. In the light of these opinions, it has been evaluated that some students have basic algorithm development knowledge and skills, but most of them do not have. Therefore, first of all, basic theoretical information about algorithm development, and how the developed algorithms can be expressed with pseudo codes and flow charts were subjected. During this step, basic steps of algorithm development, which are identifying and understanding a problem/situation, finding a solution, checking the solution for accuracy and efficiency and expressing solution step by step with pseudo-codes and flow charts, are given with simple and clear examples (e.g., finding the sum of two numbers, sum of ten numbers, ordering three numbers). Details on how to write pseudo-codes and draw flow charts were given as well. In the following weeks, the solution of the puzzles in the direction of the pre-determined curriculum and the development of the algorithms have been provided with both pseudo-codes and flow charts. The development of the algorithms was carried out with the participation of the students, providing an interactive environment within the classroom. The steps of expressing the developed algorithms with the pseudo-codes and flow charts were carried out individually by each student during the lab hours.

Except for the first two weeks, the implementation of the AT course was usually as follows. (i) Asking students a puzzle verbally, (ii) Recognizing the time required for students to keep notes on the puzzle, repeating points not understood at this stage, (iii) Asking students to express what they are asked by their own words, (iv) Waiting for two or three students to express puzzle correctly, (v) Asking all students whether the puzzle is being understood correctly, and clarifying the points that are not understood, (vi) Asking students to define what is given/input and what is asked/output, (vii) Asking students to offer solution/s for the puzzle, (viii) Ensuring that the solution/s offered by the students are evaluated by other students, (ix) Waiting until all the students agree on the solution/s, allowing them to discuss during, (x) Ensuring that the solution/s being agreed are understood by all, clarifying the points that are not understood, (xi) Having students to draw flow charts and write pseudo-codes for a solution of each puzzle individually during the lab hours.

Data Collection and Analysis

Exams' papers, the students' views, the students' works of lab hours are the collected data. The two exams, which were confirmed by two experts in CS and teaching programming, were consisted of two questions where each one was relatively harder than the previous one. Students were asked to develop an algorithm for each question and express the algorithm in both pseudo-codes and flow charts, where each one of them assessed over 25 points (100 for the total). Students' views were gathered by seven online open-ended questions after the final exam. The questions were developed according to thematic structure presented by Zhao et al. (2017). The questions were designed to collect students' views on the course in general, outcomes, content, level, course implementation process, evaluation and effect. The questionnaire did not contain any questions about the identity of the students. Only gender and age information of the students were asked. Since the identities of the responders were unknown, the students had access to the questionnaire after their evaluation finished, and the voluntary nature of the participation have been considered as proof that the students reflected their sincere thoughts.

A Wilcoxon Signed Ranks test was conducted to compare exams' scores. For the descriptive analysis of the students' views, the thematic structure of what should be in a course curriculum presented by Zhao et al. (2017) was used. There are five themes in the related thematic structure and 16 code categories under these themes. The categories used are revised according to the nature of this study. As a result, nine categories were obtained under five themes (see Table 1). In the presentation of the findings, the students are coded as S1-S20.

Ethical Procedures

The study was designed in accordance with ethical principles and rules. The ethical committee approval number of the approval document is 5152763-604.01.02-E.36805.

Results

The first sub-question of the research is about the progress of the students according to exams' scores. In order to determine students' progress during the course process two exams were applied. These exams and their scorings were validated by the participation of three experts (all have the degree of Ph.D., one is in the field of measurement and evaluation, one is in computer science education and one is in computer engineering). According to the experts, both exams can be considered as equivalents and they are suitable for measuring students' progress in AT course. Therefore, even though the two exams consisted of different questions, based on the experts' views, their scores were treated as repeated measures. The exams' scores were examined and found not to be normally distributed. Therefore, a Wilcoxon Signed Ranks test was conducted to compare exams' scores. The test indicated that the final exam scores ($\bar{X}$=76.29, *SD*=10.08) were statistically significantly higher than midterm scores ($\bar{X}$=55.93, *SD*=5.28) *Z*=-4.51, *p*<.00. The results may suggest that the AT skills of the students showed some significant progress during the course.

The other sub-questions of the study are addressed by analyzing the responses that the students gave the questions. Themes and categories which reflects results of these questions with the code counts of the categories are presented in Table 1.

Table 1

Descriptive Analysis Themes, Categories Count and Percentages

Theme	Category	f	%
Curriculum Objective	Course related objectives	55	28
	Subject area related objectives	35	18
	Life related objectives	11	6
Curriculum Content	The level/grade of the course	29	15
	Scope of the course	17	8
Curriculum Implementation	Teaching-learning process	9	4
Learning Evaluation	Exams and students' work	5	3
Curriculum Effect	Satisfaction to curriculum	23	12
	Arouse students' interest in the curriculum	12	6
Total		196	100

The themes given in Table 1 are presented in an order as they asked in sub-questions.

Course Related Objectives

In the students' views, emphasizing was mostly found on the intellectual processes required for AT course. According to this, students have developed systematic, analytical and multifaceted thinking skills within the scope of this course and have put into practice the intellectual processes of reasoning. A student stated that "the puzzles that are asked in the lessons lead to systematic thinking and that we have to perform logical thinking action at a high level in the solution process." (S7), which may imply that they provide systematic thinking and were forced to be in more intellectual reasoning processes. Another student stated that "allows a person to think systematically and in such a way that computers can understand" (S15), indicating that, in addition to systematic thinking, students thought they could express the solutions in a computational way.

Subject Area Related Objectives

In this context, the students have mostly expressed their achievements in computer program development. A student stated, "I became aware of my programming knowledge. Along with that, it provided me with information on what to do and how to improve myself. I found out what my missing information about programming is "(S5), which indicates that the students who have taken courses related to computer programming have made more sense of information they already learned with AT course. Another student stated, "It was a very beneficial lesson that allows me to approach with different perspectives when writing a program." (S8), which demonstrates that AT course can enrich computer program development processes.

Life Related Objectives

Students' expressions in this context indicate that students have achieved some gains not only in situations facing palpable solutions, but also in seeing and being open to different aspects of events. A student stated, "You think differently, and there is not a single way to arrive at the end, not being on the same path does not really show that one of them is wrong." (S18), which can be regarded as a sign of a gain that can be useful even in conflicts of opinions that can be encountered in every aspect of life. The statement of another student, "I learned that there is a solution for every problem in my life" (S2) could be regarded as a positive achievement in order to demonstrate an effort to avoid giving up in the face of problems and to seek different solutions. Another student stated, "It helps people have a thinking ability. We face many problems in our daily lives. At the very least, it helps us to find solutions to these problems." (S5), which can be considered that the AT course can be useful in solving problems that may be encountered in life.

The Level/Grade of the Course

In order to find out the students' views on what level/grade would make AT course more beneficial, they were asked the questions, "If you have opportunity to choose the level/grade that you will take AT course, what level/grade would it be? And why?". The count of students and their preferred levels/grades are as follows; four students at each stage of their learning life, two students in primary school, two students in middle school, nine students in high school, and 12 students in university. A student, pointing to the achievements of this course, "I think it would have been more beneficial if we took this course before high school, in secondary education or even primary education. We would have gained some intellectual skills in younger ages." (S10). As a matter of fact, some students also stated, "I think the course should be offered in high school or at the beginning of university education. It would be helpful not only in programming but also in different aspects of life." (S8), "It should be offered in every level of every school regardless the education they offer related to computer science or not." (S12). The other statements of the students on this topic can be interpreted as that this course is more essential for computer programming or coding and that this course should be taken before programming courses.

Scope of the Course

The statement, "It was very difficult for us to work with flow charts, since most of us did not know anything about it" (S8) refers to the fact that most of the students who were in the second term of their third year, do not have basic knowledge about algorithm development. Some other statements, "At the beginning of the course all necessary information (Algorithm, Flow Chart, etc.) would increase the yield." (S12), "In high school years they taught the flow charts in the programming class, but that is not enough in my opinion." (S11), "the course should have been handled in a fundamental way, assuming that all students knew nothing." (S15), and "In order to be able to understand the more efficient part, simple questions can be answered in the first place, and flow charts and pseudo-code can be written for these questions" (S4) also indicated that the students lack basic knowledge and skills related to algorithm development, even though they already received some computer programming courses. However, S11 emphasized that "the subjects related to algorithm development were covered within a lesson in high school years, but this was not enough.".

Learning-Teaching Process

A student's statement, "We find very different answers by discussing the problems" (S2), summarizes how the learning-teaching process of the course was realized. Another student stated, "It would be better if each student is given a question and resolved beforehand and discussed with other friends in class" (S6), which indicates that he preferred exercises to be carried out outside the class in the form of homework. Another student stated, "I think that a circle is made and more interactive; for example, we should not be so individual" (S18), suggesting that classroom interaction can be enhanced by changing the classroom layout. The same student thinks that it is better to do group work during lab hours. Another student stated, "The course was completely dependent on the practice and the imagination of oneself" (S15), pointing to the role of mental processes in these practices.

Exams and Students' Work

Some statements of the students on this subject are "I think that it takes a lot of time to answer a question during exams, and this is a negative aspect of the exam being an hour or two." (S5), "I would like it more if the exams were project formed homework" (S3), "It is a beneficial course, but it would be better if there were no grades." (S17), "It would have been better if more time was given for the exams." (S8), and "I think it would be more beneficial if we were working with groups instead of individuals." (S13). These statements can be interpreted as the fact that students want more time for exams and perform project works in groups outside of class hours instead of individual classroom activities.

Satisfaction to Curriculum

There are 23 codes in this category, and only one can be considered a negative view on the curriculum. Some students' views in this category are "We were learning and we were enjoying it" (S16), "We learned a lot and having fun, and there is nothing that man cannot achieve." (S6), "It was a very fun course and we both learned and stressed during the lessons" (S8), "Entertaining teaching." (S12), "It is fun because it is based on logic." (S3). These views of the students imply that they continue to enjoy the course during the term. However, when expressing these views, they also stated that they learned a lot, stressed, had fun, and etc., which express their satisfaction with the course.

Arouse Students' Interest in the Curriculum

None of the students' views coded under this category is negative. A student stated, "We realize our dreams. I started without knowing anything, but now I want to sit and work for hours." (S18), which indicates that the course had a positive effect on the interests and motivations of the students. Another student stated, "Thanks to this course, I can do the practices I cannot do before, and it is a hope for the future." (S13) that may be considered to be another indicator of gains achieved by this positive effect. A Statement of another student, "A good start for programming in the future." (S12), suggests a positive reflection of the achievements of the course in other fields.

Discussions and Recommendations

Main findings of the research are discussed in this section. After the discussions, some recommendations are given with respect to the related literature and the findings of this study.

Although the students have taken the AT course in the sixth semester of the third year, they have stated that they do not have sufficient (or even no) knowledge and skills regarding AT before taking the course. However, they provide rich insights about what they thought they gained by taking the course. The findings of this study demonstrate that according to students who did not train in AT in their previous education processes (which supported by exams' scores), that AT skills are necessary for them (Hubálovský, 2013) and that AT skills can be beneficial in both professional, academic and real life situations (Papadakis & Kalogiannakis, 2017). To be able to perceive and understand various problem situations (Coufal et al., 2017; Dasso et al., 2005), execute mental processes to solve problems (Coufal et al., 2017; Saeli et al., 2011), express the solutions found in systematic and generalizable structures (Hromkovič, 2006; Papert, 1993), evaluate the effectiveness and efficiency of solutions, be open to different viewpoints, gaining skills for analytical thinking (Saeli et al., 2011), and etc. can be shown among the achievements that students thought they gained. Since these students are prospective teachers who are supposed to take an important role in the preparation of ICT literate individuals and society, the need for such lesson can be better understood by assessing the importance of teachers (Ayaz et al., 2015; Gao et al., 2011; Hubálovský et al., 2010; Hubalovsky & Musilek, 2013; Kalogiannakis & Papadakis, 2017b; Papadakis, 2020; Papadakis & Kalogiannakis, 2019) in this preparation. Considering potential benefits of such a course as AT, other teacher training programs may consider including the course in their curricula as well.

While determining the course content, it is assumed that the students have at least some AT skills. However, in the implementation of the course, most of the students were found to have almost no AT skills. Therefore, it would be more beneficial for such a course to be prepared at the most basic level of content, assuming that the students do not have any skills about AT. Beginning with the concept of algorithm, the development process of algorithms and dealing with the usage of pseudo-code and flow charts to express developed algorithms with many examples can contribute to the effect and efficiency of the course. Expressing solutions in this way before switching to programming or coding can provide meaningful contributions to AT development (Fidge & Teague, 2009; Fincher, 1999; Hromkovic et al., 2017; Mayer, 1981).

Algorithmic puzzles have been determined and used for the course. Most of the students found this course beneficial and fun. In the learning-teaching process of the course, it may be more useful to use puzzles/problems that are open-ended, blurred and closer to real-life conditions (Dagiene et al., 2017; Hubalovsky, 2012; Hubálovský, 2013; Hubalovsky & Korinek, 2015) than traditional programming instructions (Brusilovsky et al., 1997). However, by evaluating the views of some students and readiness of students, using simple, well-known/common algorithmic problems in the first weeks of the course may provide positive contributions to the effect of the course. Related literature (Brusilovsky et al., 1997; Ham, 2015; Hubálovský et al., 2010) and findings of this study suggest that no programming languages or environments be used in the process of acquiring AT skills. Programming languages and environments such as Scratch, Alice, Kodu, which are claimed to require no programming or coding skills, are also no exceptions. Even such programming languages and environments can limit the process of AT development of students, no matter they require mostly moving visual blocks rather than writing codes. Therefore, it may be more beneficial for the students in the AT class to realize algorithm development processes using only paper and pencils.

The students' views also suggest that this course should be given at least before giving any computer programming courses (Galezer et al., 1995) or early stages of learning life (Akpınar & Altun, 2014; Galezer et al., 1995; Grover et al., 2015; Hubalovsky & Šedivý, 2013; Katai, 2015). Considering the relationship between computer programming and AT (Coates, 2010; Hubalovsky, 2012; Hubálovský, 2013; Hubalovsky & Korinek, 2015), this course may be considered as one of the core courses in computer science education. After submitting this course (Hubálovský et al., 2010), a course in the form of AT II, including the development of computer programs, can be presented. The content of AT II course can include basic algorithms and data structures such as searching and sorting algorithms, list and linked list, queue and stack structures. In presenting these topics, the necessary theoretical information can be presented first and then the association with daily life can be made with proper examples, questions or puzzles. Solutions can be expressed in pseudo-codes and flow charts using the approach of the AT course and then these pseudo-codes and flow charts into any programming language. Such an AT II course can also provide effective gains for the use of AT skills in the software development process, as well as helping students to consolidate and improve AT skills. Alternatively, the content of the AT and AT II courses mentioned here can be intensified and presented under a single course. Such a course can be divided into AT for the first half of the period (first seven weeks) and AT II for the second half (the rest of the term).

Conclusion

According to the exams' scores and views of the students, the offered AT course may be considered an effective and useful course, despite some deficiencies and problems identified and reported. Consequently, such a course should be given in departments related to computer science. It might be useful to present courses and topics related to AT skills at different stages of education process. It may be more useful to give this course before computer programming courses. The use of no programming language but only the use of the pseudo-codes and flow charts may be considered an element in the positive outcomes of the course.

Limitations and Implications

Due to some official and administrative restrictions, the AT course could be opened in the sixth semester of the third year of the CEIT Department. Consequently, the students who took this course had already taken some computer programming-related courses in the previous terms. Whereas it makes more sense that AT course is given before such courses. Taking an AT course before or after these courses may influence students' feelings and thoughts about the course. Therefore, the term of the AT course given in, and therefore the characteristics of the course participants, may be regarded as a limitation for the research. A more complicated study with two similar participant groups can isolate this limitation. Offering an AT course before the computer programming courses for one group and after for the other group then make participants evaluate the course can provide more meaningful findings in this context.

The researcher of this study is also the instructor of the AT course. This may have affected data collection and data analysis phases for various reasons. The researcher has been involved with the students during class hours, but he could not keep observational notes in this process. This shortcoming has been tried to be solved by keeping the research journals after the class hours. The students' views about the course were intended to be determined by the interviews, but the researcher decided that interviewing would not be appropriate by evaluating the conflicts of interest that could originate from the researcher's teaching role. Instead, at the end of the term and after the student's assessments were done, students' views were received by an online form, with student identities being kept confidential. With the contribution of an additional researcher/s, research can be conducted by employing various data collection methods, which may improve the quality of the findings.

Conflicts of Interest

There are no conflicts of interest in this study.

References

Aho, A. V. (2012). Computation and computational thinking. *Computer Journal*, *55*(7), 832-835. https://doi.org/10.1093/comjnl/bxs074

Akpınar, Y., & Altun, A. (2014). Bilgi toplumu okullarinda programlama eğitimi gereksinimi [The need of teaching programming in knowledge society schools]. *İlköğretim Online*, *13*(1), 1-4.

Angeli, C., & Valanides, N. (2020). Developing young children's computational thinking with educational robotics: an interaction effect between gender and scaffolding strategy. *Computers in Human Behavior, 105, 105954*. https://doi.org/10.1016/j.chb.2019.03.018

Athanasiou L., Topali P., Mikropoulos T.A. (2017). The use of robotics in introductory programming for elementary students. In Alimisis D., Moro M., Menegatti E. (Eds.), *Educational robotics in the makers era. Edurobotics 2016. Advances in intelligent systems and computing*, vol 560. Springer, Cham.

Ayaz, M. F., Oral, B., & Söylemez, M. (2015). Evaluation of the post-graduate theses on teacher education in Turkey. *Elementary Education Online*, *14*(2), 787-802.

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., & Miller, P. (1997). Mini-languages: A way to learn programming principles. *Education and Information Technologies*, *2*(1), 65-83.

Bryant, E. A. (2017). An unnamed intersection: Where computing meets liberal arts. In *New Directions for Computing Education* (pp. 103-118). https://doi.org/10.1007/978-3-319-54226-3_7

Citta, G., Gentile, M., Allegra, M., Arrigo, M., Conti, D., Ottaviano, S., Reale, F., & Sciortino, M. (2019). The effects of mental rotation on computational thinking. *Computers & Education, 141,* 103613. https://doi.org/ARTN10361310.1016/j.compedu.2019.103613

Coates, P. (2010). *Programming architecture* (1st ed.). Routledge. https://doi.org/10.4324/9780203841488

Cooper, S., Dann, W., & Pausch, R. (2000, Nov. 9 - 12). Developing algorithmic thinking with Alice. The proceedings of ISECON 2000, Philadelphia, PA.

Coufal, P., Hornik, T., Hubalovsky, S., & Musilek, M. (2017). Simulation of the automatic parking assist system as a method of the algorithm development thinking. *International Journal of Education and Information Technologies, 11*, 37-43.

Dagiene, V., Sentance, S., & Stupuriene, G. (2017). Developing a two-dimensional categorization system for educational tasks in informatics. *Informatica*, *28*(1), 23-44. https://doi.org/10.15388/Informatica.2017.119

Dasso, A., Funes, A., Riesco, D. E., Montejano, G. A., Peralta, M., & Salgado, C. (2005, April 14-15). Teaching programming. I Jornadas de Educación en Informática y TICs en Argentina, Buenos Aires, República Argentina.

Essl, K. (2007). Algorithmic composition. In N. Collins & J. d'Escrivan (Eds.), *The Cambridge companion to electronic music* (pp. 107-125). https://doi.org/10.1017/CCOL9780521868617.008

Fidge, C., & Teague, D. (2009). Losing their marbles: syntax-free programming for assessing problem-solving skills. Proceedings of the Eleventh Australasian Conference on Computing Education-Volume 95, Wellington, New Zealand.

Fincher, S. (1999). What are we doing when we teach programming? Frontiers in Education Conference, 1999. FIE'99. 29th Annual Frontiers in Education Conference. Designing the Future of Science and Engineering Education, San Juan, Puerto Rico.

Futschek, G. (2006). Algorithmic thinking: the key for understanding computer science. In M. R.T. (Ed.), *Lecture notes in computer science* (Vol. 4226, pp. 159-168). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11915355_15

Galezer, J., Beeri, C., Harel, D., & Yehudai, A. (1995). A high-school program in computer-science. *Computer*, *28*(10), 73-80. https://doi.org/10.1109/2.467599

Gao, P., Chee, T. S., Wang, L. L., Wong, A., & Choy, D. (2011). Self reflection and preservice teachers' technological pedagogical knowledge: Promoting earlier adoption of student-centred pedagogies. *Australasian Journal of Educational Technology*, *27*(6), 997-1013.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, *25*(2), 199-237. https://doi.org/10.1080/08993408.2015.1033142

Ham, D. (2015). Playful calculation. In N. Lemon (Ed.), *Revolutionizing arts education in K-12 classrooms through technological integration* (pp. 125-144). https://doi.org/10.4018/978-1-4666-8271-9.ch006

Hromkovič, J. (2006, November 7-11). Contributing to general education by teaching informatics. International Conference on Informatics in Secondary Schools-Evolution and Perspectives, Vilnius, Lithuania.

Hromkovic, J., Kohn, T., Komm, D., & Serafini, G. (2017). Algorithmic thinking from the start. *Bulletin of the European Association for Theoretical Computer Science* (121), 132-139.

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

Hubalovsky, S. (2012). Modeling and computer simulation of real process - solution of Mastermind board game. *International Journal of Mathematics and Computers in Simulation*, *6*(1), 107-118.

Hubálovský, S. (2013). Modeling and simulation of real process - passing through the labyrinth as a method of development of algorithm thinking and programming skills. *International Journal of Mathematics and Computers in Simulation*, *7*(2), 125-133.

Hubalovsky, S., & Korinek, O. (2015). Evaluation of algorithmic thinking of students using control testing environment. *International Journal of Education and Information Technologies*, *9*, 205-208.

Hubálovský, S., Milková, E., & Pražák, P. (2010). Modeling of a real situation as a method of the algorithmic thinking development and recursively given sequences. *WSEAS Transactions on Information Science and Applications*, *7*(8), 1090-1100.

Hubalovsky, S., & Musilek, M. (2013). Modeling, simulation and visualization of real processes in LOGO programming language as a method of development of algorithm thinking and programming skills. *International Journal of Mathematics and Computers in Simulation*, *7*(2), 144-152.

Hubalovsky, S., & Šedivý, J. (2013). Algorithm development and computer simulation of position order decoding of Mastermind board game. *Applied Mechanics and Materials*, *333-335*, 1353-1356. https://doi.org/10.4028/www.scientific.net/amm.333-335.1353

Hunter, B. (1987). What is fundamental in an information age? A focus on curriculum. *Education and Computing*, *3*(1-2), 63-73. https://doi.org/10.1016/s0167-9287(87)80513-7

Kalogiannakis, M., & Papadakis, S. (2017a, August 21-25). Pre-service kindergarten teachers acceptance of “ScratchJr” as a tool for learning and teaching computational thinking and Science education. Proceedings of the 12th Conference of the European Science Education Research Association (ESERA), Research, practice and collaboration in science education, Dublin, Ireland.

Kalogiannakis, M., & Papadakis, S. (2017b, August 21-25). A proposal for teaching ScratchJr programming environment in preservice kindergarten teachers. Proceedings of the 12th Conference of the European Science Education Research Association (ESERA), Dublin, Ireland.

Katai, 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

Knuth, D. E. (1985). Algorithmic thinking and mathematical thinking. *American Mathematical Monthly*, *92*(3), 170-181. *https://doi.org/10.2307/2322871*

Kordaki, M. (2013). High school computing teachers' beliefs and practices: A case study. *Computers & Education, 68*, 141-152. https://doi.org/10.1016/j.compedu.2013.04.020

Levitin, A., & Levitin, M. (2011). *Algorithmic puzzles*. OUP USA.

Mayer, R. E. (1981). The psychology of how novices learn computer-programming. *Computing Surveys*, *13*(1), 121-141.

Milková, E. (2005). Developing of algorithmic thinking: The base of programming. *International Journal of Continuing Engineering Education and Life-Long Learning*, *15*(3-6), 135-147.

Milková, E., & Hùlková, A. (2013). Algorithmic and logical thinking development: Base of programming skills. *WSEAS Transactions on Computers*, *12*(2), 41-51.

Milkova, E., & Sevcikova, A. (2017). Algorithmic thinking and mathematical competences supported via entertaining problems. *International Journal of Education and Information Technologies*, *11*, 80-86.

Papadakis, S. (2020). Evaluating a game-development approach to teach introductory programming concepts in secondary education. *International Journal of Technology Enhanced Learning*, *12*(2), 127-145.

Papadakis, S., & Kalogiannakis, M. (2017). Using gamification for supporting an introductory programming course. the case of classcraft in a secondary education classroom. In *Interactivity, game creation, design, learning, and innovation* (pp. 366-375). Springer.

Papadakis, S., & Kalogiannakis, M. (2019). Evaluating a course for teaching introductory programming with Scratch to pre-service kindergarten teachers. *International Journal of Technology Enhanced Learning*, *11*(3), 231-246. https://doi.org/10.1504/Ijtel.2019.100478

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. ERIC.

Pérez-Marín, D., Hijón-Neira, R., Bacelo, A., & Pizarro, C. (2020). Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children? *Computers in Human Behavior*, *105*, *105849*. https://doi.org/10.1016/j.chb.2018.12.027

Plerou A., Vlamos P., Triantafillidis C. (2017) *The Effectiveness of Neurofeedback Training in Algorithmic Thinking Skills Enhancement*. In Vlamos P. (Ed.), GeNeDis 2016. Advances in Experimental Medicine and Biology, vol 988. Springer, Cham. https://doi.org/10.1007/978-3-319-56246-9_14

Saeli, M., Perrenet, J., Jochems, W. M., & Zwaneveld, B. (2011). Teaching programming in secondary school: a pedagogical content knowledge perspective. *Informatics in Education*, *10*(1), 73-88.

Silapachote, P., & Srisuphab, A. (2017). Engineering courses on computational thinking through solving problems in artificial intelligence. *International Journal of Engineering Pedagogy*, *7*(3), 34-49. https://doi.org/10.3991/ijep.v7i3.6951

Thomas, J. O., Rankin, Y., Minor, R., & Sun, L. (2017). Exploring the difficulties african-american middle school girls face enacting computational algorithmic thinking over three years while designing games for social change. *Computer Supported Cooperative Work-the Journal of Collaborative Computing and Work Practices*, *26*(4-6), 389-421. https://doi.org/10.1007/s10606-017-9292-y

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

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences*, *366*(1881), 3717-3725.

Zhang, L. C., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education, 141,* 103607. https://doi.org/10.1016/j.compedu.2019.103607

Zhao, D. C., Ma, X. R., & Qiao, S. B. (2017). What aspects should be evaluated when evaluating graduate curriculum: Analysis based on student interview. *Studies in Educational Evaluation*, *54*, 50-57. https://doi.org/10.1016/j.stueduc.2016.11.003

Zhao, W., & Shute, V. J. (2019). Can playing a video game foster computational thinking skills? *Computers & Education, 141,* 103633. https://doi.org/10.1016/j.compedu.2019.103633