EĞİTİM TEKNOLOJİSİ Kuram ve Uygulama
Cilt:10 Sayı:1 Yıl:2020

Makale Geçmişi / Article History
Alındı/Received: 21.11.2019
Düzeltme Alındı/Received in revised form: 25.12.2019
Kabul edildi/Accepted: 26.12.2019

BLOK TABANLI PROGRAMLAMA ETKİNLİKLERİNİN ÖĞRETMEN ADAYLARININ PROGRAMLAMAYA İLİŞKİN ÖZ YETERLİLİK ALGILARINA VE HESAPLAMALI DÜŞÜNME BECERİLERİNE ETKİSİ

Şeyhmus Aydoğdu[1]

Öz

21. yy becerileri incelendiğinde programlama öğretimi günümüzde oldukça önemlidir. Programlama öğretiminde, hesaplamalı düşünme becerilerinin ve öz yeterlilik algısının geliştirilmesi bu becerinin kazandırılmasında anahtar roldedir. Bu araştırmanın amacı, blok tabanlı programlama etkinliklerinin öğretmen adaylarının programlamaya ilişkin öz yeterlilik algıları ve hesaplamalı düşünme becerileri üzerindeki etkisinin incelenmesidir. Tek grup ön test-son test olarak yürütülen bu araştırmada öğretmen adaylarına 4 hafta boyunca toplamda 16 saat blok tabanlı programlama etkinlikleri yaptırılmıştır. Araştırma sonucunda, blok tabanlı programlama etkinliklerinin öğretmen adaylarının programlamaya ilişkin öz yeterlilik algıları üzerinde olumlu etkisi olmuştur. Buna karşın, yapılan etkinliklerin öğretmen adaylarının hesaplamalı düşünme becerileri üzerinde etkisi olmadığı görülmüştür.

Anahtar Kelimeler: blok tabanlı programlama; scratch; hesaplamalı düşünme becerisi; programlama öz yeterliliği

THE EFFECT OF BLOCK-BASED PROGRAMMING ACTIVITIES ON PRE-SERVICE TEACHERS' COMPUTER PROGRAMMING SELF-EFFICACY AND COMPUTATIONAL THINKING SKILLS

Abstract

Nowadays programming teaching is crucial in terms of 21st century skills. In programming teaching, the improvement of computational thinking skills and self-efficacy perception are key to gaining this skill. The purpose of this study is to investigate the effect of block-based programming activities on pre-service teachers' self-efficacy perceptions and computational

[1] Dr. Öğr. Üyesi, Nevşehir Hacı Bektaş Veli Üniversitesi, aydogduseyhmus@gmail.com, orcid.org/0000-0002-9075-8055

thinking skills. In this study, which was conducted as a single group pretest-posttest, pre-service teachers received 16 hours of block-based programming activities for 4 weeks. As a result of the research, block-based programming activities had a positive effect on pre-service teachers' programming self-efficacy perceptions. However, it was observed that the activities did not have any effect on the computational thinking skills of the pre-service teachers. In the teaching of programming, it is recommended to use block based programming activities in students with low self-efficacy.

Keywords: block-based programming; scratch; computational thinking skills; programming self efficacy

Summary

Nowadays it is clear that programming teaching is a necessity in schools (Akpınar & Altun, 2014; Dohn, 2019). Indeed, considering 21st century skills, programming teaching plays a key role in gaining these skills (Yukselturk & Altiok, 2018). In this case, it is very important that information technology teachers, as well as teachers in other fields to have basic programming skills for improving learning.

Learning programming is particularly difficult for students who have no previous programming experience (Robins et al., 2003). Because in the programming process, it is necessary to analyze the problem, convert the problem solution into steps, develop the program and test the solution (Janpla & Piriyasurawong, 2018). In addition, syntactic , conceptual and strategic information should be used together and synchronously in programming teaching (Bayman & Mayer, 1988; McGill & Volet, 1997). Therefore, in teaching programming, mini-programming languages (Brusilovsky et al., 1997), block-based tools (Basogain et al., 2018), educational robots (Sáez-López et al., 2019) and the use of web-based programming (Horvátha, 2018) environments are proposed, as well as studies to create a teaching model for programming teaching (Erümit et al., 2018). These tools and methods attract novice students in programming and provide a fun environment for students (Yükseltürk & Altıok, 2015).

Bandura (1997) defines self-efficacy as the judgment of one's ability to organize and conduct a particular type of performance. Low self-efficacy of the individual will lead to failure of the individual (Askar & Davenport, 2009; Schunk, 1991). In programming teaching, programming self-efficacy is the key variable for students' success (Yildiz Durak et al., 2019). In particular, the development of programming self-efficacy of novice pre-service teachers is important to link the course contents and algorithmic thinking in the future. In many studies, it is emphasized that self-efficacy belief has an important role in programming teaching (Tsai, 2019). It is thought that the use of the block-based programming tools for pre-service teachers will improve pre-service teachers' self-efficacy perceptions. Therefore, the effect of block-based programming activities on self-efficacy perceptions of pre-service teachers was investigated in this study.

The increase of interest in computational thinking began under Wing (2006)’s leadership (Denning, 2017). The concept of computational thinking is generally used as equal to the use of computer technologies (Yadav et al., 2017), but Wing (2006) defines computational thinking as a process of thinking that involves solving problems, designing systems and understanding human behavior based on concepts of computer science. Computational thinking skills can be

considered as thinking processes related to formulating problems, hence the phases of the problem solution and algorithms may be represented by these processes (Aho, 2012). These skills include the thinking and role-taking phases that can be applied to a large number of real-world problems that extend beyond programming (Faber et al., 2017). Computational thinking helps students divide a given real problem into pieces that can be controlled, abstraction to overcome complexities, recognize patterns, and create scalable algorithms (Yevseyeva & Towhidnejad, 2012). When studies on the development of computational thinking skills are examined, programming practices are the most preferred applications (Hsu et al., 2018).

Nowadays, computer courses are being conducted in many countries in order to improve students' computational thinking skills (Heintz et al., 2016). At this point, the training of teachers in this field is the key to the development of computational thinking skills of students (Hsu et al., 2018; Orvalho, 2017). Given the contribution of programming instruction to the development of computational thinking skills, it is of great importance to provide this training to pre-service teachers. From this point of view, the effect of block-based programming activities on computational thinking skills of students was investigated.

The purpose of this study is to examine the effect of block-based programming activities on pre-service teachers' computational thinking skills and self-efficacy perceptions of programming. For the purpose stated, research questions are as follows:

1. Is there a significant difference in pre-service teachers' self-efficacy perceptions before and after block-based programming activities?
2. Is there a significant difference in pre-service teachers' computational thinking skills before and after block-based programming activities?

This research was carried out as a single group pre-test and post-test. In this study, “Computer Programming Self-Efficacy Scale”, which was developed by Ramalingam and Wiedenbeck (1998) and adopted Turkish by Altun and Mazman (2012), was used to measure students' self-efficacy perceptions about programming. In order to measure students' computational thinking skills, “Computational Thinking Skill Scale”, which was developed by Korkmaz et al. (2017) for university students, was used.

During the application process, firstly, block based programming activities are designed. In the design and sequencing of course contents, depending on the difficulties encountered in programming instruction, the algorithms for solving a problem, the use of conditions and loops are discussed. Therefore, in this research, it was emphasized that “Scratch is a development tools” and the concepts of input, process and output were focused within the scope of a problem in the course contents. The subjects discussed during the research process were processed for 4 weeks during the 4-hour course.

As a result of the research, block-based programming activities had a positive effect on the development of students' self-efficacy. This finding is consistent with the findings of Mazman and Altun (2013). As the participants in this study did not have previous programming experience, no comparison was made according to the preliminary experience level. Similarly , the effect of robotic coding activities on self-efficacy perceptions of students by Kasalak (2017) supports this finding of the research. When the sub-dimensions of simple and complex programming tasks of self-efficacy perception regarding programming are examined, it is seen that there is a positive development in these dimensions as well. Therefore, in the teaching process, block-based programming activities can be utilized to improve the self-efficacy of students who are novice programmers.

In this study, it was seen that block based programming activities had no effect on computational thinking skills of students. When the sub-dimensions of computational thinking skills are taken into consideration, it is understood that there is no significant difference between the pretest and posttest scores in these dimensions . This is thought to be due to the fact that the average computational thinking skill scores of the students were higher than the midpoint of the computational thinking skill scale before the application. Therefore, the effectiveness of these activities can be examined in subsequent studies by performing block-based programming activities on individuals with low computational thinking skills.

Giriş

Günümüzde okullarda programlama öğretiminin bir gereksinim olduğu açıkça ortadadır (Akpınar ve Altun, 2014; Dohn, 2019). Nitekim, 21. yy becerileri dikkate alındığında programlama öğretimi bu becerilerin kazandırılmasında anahtar rolündedir (Yükseltürk ve Altıok, 2018). Bu durumda, bilişim teknolojileri alan öğretmenlerinin yanı sıra diğer alanlardaki öğretmenlerin de temel programlama becerilerine sahip olmaları öğrenmenin geliştirilmesi açısından oldukça önemlidir. Öğretmenler için International Society for Technology in Education (ISTE) standartları incelendiğinde bu standartların öğrenen, lider, vatandaş, işbirlikçi, tasarımcı, kolaylaştırıcı ve analist başlıkları altında toplandığı görülmektedir (ISTE, 2019b). Özellikle liderlik ve tasarımcı başlıklarına dayalı olarak günümüzde her öğretmen adayının programlama ile ilgili temel becerilere sahip olması gerektiği düşünülmektedir.

Programlamayı öğrenme özellikle daha önce programlama ile ilgili deneyimi olmayan öğrenciler için oldukça zordur (Robins, Rountree ve Rountree, 2003). Çünkü programlama sürecinde problemin analizi, problem çözümünün adımlara dönüştürülmesi, programın geliştirilmesi ve çözümün test edilmesi süreçlerinin yürütülmesi gerekmektedir (Janpla ve Piriyasurawong, 2018). Bunun yanı sıra programlama öğretiminde söz dizimsel, kavramsal ve stratejik bilgilerin bir arada ve eş zamanlı bir şekilde kullanılması gerekmektedir (Bayman ve Mayer, 1988; McGill ve Volet, 1997). Bundan dolayı, programlamanın öğretiminde mini programlama dillerinin (Brusilovsky, Calabrese, Hvorecky, Kouchnirenko ve Miller, 1997), blok tabanlı araçların (Basogain, Olabe, Olabe ve Rico, 2018), eğitsel robotların (Sáez-López, Sevillano-García ve Vazquez-Cano, 2019) ve web tabanlı programlama ortamlarının (Horvátha, 2018) kullanımının önerilmesinin yanı sıra programlama öğretimine yönelik öğretim modeli oluşturma (Erümit ve diğerleri, 2018) çalışmaları bulunmaktadır. Bu araçlar ve yöntemler, programlama konusunda acemi öğrencilerin ilgisini çekmekte ve öğrencilere eğlenceli bir ortam sunmaktadır (Yükseltürk ve Altıok, 2015).

Programlama öğretiminde kullanılan araç ve yöntemlerin yanı sıra öğrencilerin konu hakkında başarılı olmalarına inançları da gerçekleştirilen öğretimin etkililiğini belirlemektedir. Programlama öğretiminde de, programlamaya ilişkin öz yeterlilik öğrencilerin başarılı olmaları için anahtar değişken rolündedir (Yildiz Durak, Yilmaz ve Yilmaz, 2019). Öğrencilerin, belirli bir konuda başarılı olmalarında o konudaki başarılı olma inançları oldukça önemli bir yere sahiptir. Bu inanç, öz yeterlilik inancı olarak adlandırılmaktadır. Bandura (1997), öz yeterlilik inancını kişinin belirli bir performans türünün düzenlenmesine ve yürütülmesine yönelik yeteneğinin yargısı olarak tanımlamaktadır. Bireyin öz yeterliliğin düşük olması bireyin başarısız olmasını beraberinde getirecektir (Askar ve Davenport, 2009; Schunk, 1991). Birçok çalışmada öz yeterlilik inancının programlama öğretiminde önemli bir yere sahip olduğu vurgulanmaktadır

(Tsai, 2019). Özellikle bilgisayar kullanma becerisi düşük olan öğrencilerin programlama öz yeterliliklerin geliştirilmesi gelecekte ders içerikleri ile algoritmik düşünme arasında bağ kurmaları açısından önemlidir. Öğretmen adayları için yukarıda belirtilen blok tabanlı programlama araçlarının kullanımının bu adayların öz yeterlilik algılarını geliştireceği düşünülmektedir. Bundan dolayı bu araştırmada blok tabanlı programlama etkinliklerinin öğretmen adaylarının öz yeterlilik algıları üzerindeki etkisi incelenmiştir.

Günümüzde öğretmen adaylarına kazandırılması önemli olan becerilerden biri hesaplamalı düşünme becerisidir. Hesaplamalı düşünme (computational thinking), öğrencilerin verilen gerçek bir problemi kontrol edilebilen parçalara ayırmasına, karmaşıklıkların üstesinden gelmek için soyutlama yapmalarına, örüntüleri tanımalarına ve ölçeklenebilir algoritmalar oluşturmalarına yardımcı olmaktadır (Yevseyeva ve Towhidnejad, 2012). Hesaplamalı düşünmeye yönelik ilgi artışı Wing (2006)'in önderliğinde başlamıştır (Denning, 2017). Hesaplamalı düşünme kavramı genellikle bilgisayar teknolojilerini kullanmakla eşitmiş gibi kullanılmaktadır (Yadav, Stephenson ve Hong, 2017) fakat Wing (2006) hesaplamalı düşünmeyi bilgisayar bilimi temelindeki kavramlara dayalı olarak problemleri çözmeyi, sistemleri tasarlamayı ve insan davranışını anlamayı içeren bir düşünme süreci olarak tanımlamaktadır. Hesaplamalı düşünme becerileri, problemleri formüle etme ile ilgili düşünme süreçleri olarak düşünülebilir dolayısıyla problem çözümünün aşamaları ve algoritmalar ile bu süreçler temsil edilebilir (Aho, 2012). Bu beceriler, programlamanın ötesine uzanan çok sayıda gerçek dünya problemlerine uygulanabilecek düşünme ve rol alma aşamalarını içermektedir (Faber, Wierdsma, Doornbos, van der Ven ve de Vette, 2017). Hesaplamalı düşünme becerilerinin geliştirilmesine yönelik çalışmalar incelendiğinde programlama uygulamalarının en çok tercih edilen uygulamalar olduğu görülmektedir (Hsu, Chang ve Hung, 2018).

Hesaplamalı düşünmenin bileşenleri olarak bazı topluluklar tarafından farklı kavramlar tanımlanmıştır. (CAS, 2015; ISTE, 2019a; ISTE ve CSTA, 2011) CAS (2015) hesaplamalı düşünmeyi bir mantık yürütme süreci olarak ele almış ve sınıf içerisinde gözlenebilir davranışlar olarak algoritmik düşünme, ayrıştırma, genelleme, soyutlama ve değerlendirme boyutlarında gruplandırmıştır. Farklı araştırmalarda ise hesaplamalı düşünme için bu adımlar araştırmalarda farklılık göstermektedir. Selby ve Woollard (2014), 39 araştırmada hesaplamalı düşünmeye yönelik yapılan tanımlamalardaki fikir birliği olunan boyutları aşağıdaki gibi belirtmiştir:

1. Düşünme süreci: Bu süreç, gerçek hayatta verilen bir problemin çözümü için bilgisayar bilimlerindeki zihinsel araçların ve tekniklerin kullanıldığı süreçtir (Furber, 2012).
2. Soyutlama: Soyutlama, hesaplamalı düşünmenin en temel kavramlarından biridir (Wing, 2008). Soyutlama, kişinin söz konusu sorunun veya durumun karmaşık yönlerini gizleyerek bir sorunun veya durumun belirli yönlerine bakmasına olanak tanır (Faber ve diğerleri, 2017).
3. Ayrıştırma: Ayrıştırma, büyük problemlerin, karmaşık sistemlerin veya görevlerin çözümünde gereklidir (Selby ve Woollard, 2014). Ayrıştırma ile karmaşık olan bir problem yönetilebilir küçük parçalara ayrıştırılır (ISTE, 2019a). Bu ayrıştırmada oluşturulan her parça anlaşılır, çözülebilir, geliştirilebilir ve değerlendirilebilir olmalıdır (CAS, 2015).
4. Algoritmik tasarım: Algoritmik tasarım, algoritmalar için temel oluşturan kesin talimatlar veya diziler geliştirme becerisidir (ISTE, 2019a). Bu, öğrencilerin kendi

bilgisayar programlarını yazmayı öğrendikçe geliştirdikleri temel bir beceridir (CAS, 2015).

5. Değerlendirme: Değerlendirme, oluşturulan çözümün doğrulanmasıdır. Değerlendirme, doğruluk, hız, kaynakların kullanımı, kullanım kolaylığı gibi açılardan çözümün incelenmesini ele alır (CAS, 2015).
6. Genelleme: Genelleme, oluşturulan problem çözüm sürecinin farklı bağlamlarda uygulanması olarak tanımlanmaktadır (ISTE ve CSTA, 2011). Genellemede, çözülen bir probleme benzer bir problem verilerek aradaki farklılıklar sorgulanır ve oluşturulan algoritmik tasarımın buna uyarlaması istenir (CAS, 2015).
7. Otomasyon: Oluşturulan çözümün bir bilgi işleyici tarafından işletilmesidir. Bu, karmaşık bir hesaplamayı çözen ya da monoton bir görevi tekrar eden bir bilgisayar ve oluşturulan algoritmayı işlemek için robotik kullanımı olabilir (Faber ve diğerleri, 2017).

Günümüzde öğrencilerin hesaplamalı düşünme becerilerinin geliştirilmesi için birçok ülkede bilgisayar kurslarında çalışmalar yapılmaktadır (Heintz, Mannila ve Färnqvist, 2016). Bu noktada öğrencilerin hesaplamalı düşünme becerilerinin geliştirilmesinde öğretmenlerin bu alanda eğitilmesi anahtar rolündedir (Hsu ve diğerleri, 2018; Orvalho, 2017). Programlama öğretiminin hesaplamalı düşünme becerisini geliştirmeye katkısı (Rodríguez-Martínez, González-Calero ve Sáez-López, 2019; Zhang ve Nouri, 2019) düşünüldüğünde öğretmen adaylarına bu eğitimin verilmesi oldukça büyük öneme sahiptir. Örneğin, Erol and Kurt (2017) tarafından bilişim teknolojileri öğretmen adaylarına scratch kullanılarak programlamaya yönelik motivasyon ve başarılarının incelendiği çalışma sonucunda scratch uygulaması kullanan öğretmen adaylarının motivasyonlarının ve başarılarının arttığı görülmüştür. Bu noktadan hareketle bu çalışma kapsamında geliştirilen blok tabanlı programlama etkinliklerinin öğrencilerin hesaplamalı düşünme becerileri üzerindeki etkisi incelenmiştir. Bu çalışma, bilişim teknolojileri öğretmen adaylarının yerine daha önce programlama ile ilgili herhangi bir çalışması olmayan fen bilgisi öğretmen adayları ile yürütülmüştür.

Bu çalışmanın amacı, blok tabanlı programlama etkinliklerinin öğretmen adaylarının hesaplamalı düşünme becerileri ve programlamaya ilişkin öz yeterlik algıları üzerindeki etkisinin incelenmesidir. Belirtilen amaç doğrultusunda aşağıdaki araştırma sorularına cevap aranmıştır:

1. Öğretmen adaylarının, programlamaya ilişkin öz yeterlik algılarında blok tabanlı programlama etkinlikleri öncesinde ve sonrasında anlamlı bir farklılık var mıdır?
2. Öğretmen adaylarının hesaplamalı düşünme becerileri düzeylerinde blok tabanlı programlama etkinlikleri öncesinde ve sonrasında anlamlı bir farklılık var mıdır?

Yöntem

Araştırma Modeli

Bu araştırma, deneysel desenlerden tek grup ön test-son test desen olarak yürütülmüştür. Araştırmanın simgesel görünümü Tablo 1 de verilmiştir.

Tablo 1. Araştırmanın simgesel görünümü

Grup	Ön test	İşlem	Son test
G	O_1, O_2	X	O_3, O_4

G: Blok tabanlı programlama etkinlikleri gerçekleştirilen grup

O_1, O_3: Hesaplamalı düşünme becerileri ölçeği

O_2, O_4: Programlamaya ilişkin öz yeterlik ölçeği

X: Blok tabanlı programlama etkinlikleri

Tablo 1 de görüldüğü gibi araştırmada blok tabanlı programlama etkinliklerinin gerçekleştirildiği bir grup bulunmaktadır. Etkinlikler öncesinde ve sonrasında öğrencilerin hesaplamalı düşünme becerilerinin ve öz yeterlilik algılarının değişimlerinin incelenmesi amacıyla ön test ve son testler uygulanmıştır. Araştırma sürecinin detayları ve blok tabanlı programlama etkinliklerinin içerikleri Uygulama Süreci başlığı altında açıklanmıştır.

Çalışma Grubu

Araştırmanın çalışma grubu, bir üniversitenin 2018-2019 Bahar döneminde Bilgisayar I dersini alan Fen Bilgisi Öğretmenliği Bölümü 2,3 ve 4. sınıfında öğrenim gören 29 öğrenciden oluşmaktadır. Çalışma grubunda bulunan öğrencilerin programlama ile ilgili daha önce herhangi bir deneyimleri bulunmamaktadır. Bundan dolayı çalışmada araştırma grubu olarak bu grup tercih edilmiştir. Çalışma grubunun sınıf ve cinsiyete göre dağılımları Tablo 2 de verilmiştir.

Tablo 2. Çalışma grubundaki öğrencilerin sınıf ve cinsiyete göre dağılımları

Sınıf/Cinsiyet	Erkek	Kadın	Toplam
2	6	18	24
3	1	3	4
4	0	1	1
Toplam	7	22	29

Çalışma grubundaki katılımcılar ikinci sınıf veya sonrasında dersi alabildiklerinden grup içerisinde birinci sınıftan öğrenci bulunmamaktadır.

Veri Toplama Araçları

Bu bölümde araştırma kapsamında kullanılan veri toplama araçları açıklanmıştır. Araştırmanın amacı ve katılımcı özellikleri dikkate alınarak öğretmen adaylarının programlamaya yönelik öz yeterlilik algılarının incelenmesi amacıyla Altun ve Mazman (2012) tarafından Türkçe'ye uyarlanan "Programlamaya yönelik öz yeterlilik algısı ölçeği" ve öğretmen adaylarının hesaplamalı düşünme becerilerindeki değişimin incelenmesi amacıyla

Korkmaz, Çakır ve Özden (2017) tarafından geliştirilen hesaplamalı düşünme becerileri ölçeği kullanılmıştır.

Programlamaya yönelik öz yeterlilik algısı ölçeği

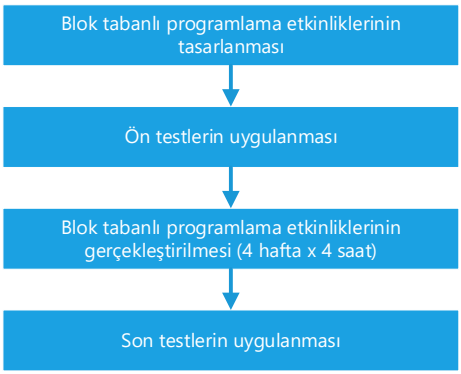
Çalışmada, öğrencilerin programlamaya yönelik öz yeterlilik algılarının ölçümü için Ramalingam ve Wiedenbeck (1998) tarafından geliştirilen Altun ve Mazman (2012) tarafından Türkçe'ye uyarlanan Programlamaya ilişkin öz yeterlilik algısı ölçeği kullanılmıştır. Ölçekteki maddeler 7'li likert tipindedir. Her bir madde için verilen cevaplar 1-"Hiç güvenmiyorum", 2-"Genellikle güvenmiyorum", 3-"Biraz güveniyorum", 4-"Kararsızım", 5-"Oldukça güveniyorum", 6-"Genellikle güveniyorum" ve 7-"Tamamen güveniyorum" şeklinde kodlanmıştır. Ölçekten alınan puanlar 9 ile 63 arasında değişmektedir. Programlamaya ilişkin öz yeterlilik algısı ölçeğinin "basit programlama görevleri" ve "karmaşık programlama görevleri" olmak üzere iki alt boyutu bulunmaktadır. Ölçeğin uyarlama çalışmasında ölçeğin tamamının Cronbach alpha güvenirlik katsayısı .928, "basit programlama görevleri" ve "karmaşık programlama görevleri" alt faktörleri için Cronbach alpha güvenirlik katsayısı sırasıyla .907 ve .943 olarak hesaplanmıştır. Bu araştırmada, ön test ve son test olarak kullanılan ölçeğin Cronbach Alpha güvenirlik katsayısı sırasıyla .941 ve .945 olarak hesaplanmıştır. Çalışmada "basit programlama görevleri" boyutunda ön test ve son testte hesaplanan güvenirlik katsayıları .901 ve .933'tür. "Karmaşık programlama görevleri" boyutu için Cronbach alpha güvenirlik katsayısı değerleri ön test ve son test için sırasıyla .925 ve .921 olarak hesaplanmıştır.

Hesaplamalı düşünme becerileri ölçeği

Alanyazında hesaplamalı düşünme becerisinin ölçülmesi için farklı yöntemler kullanıldığı görülmektedir. Bazı çalışmalarda hesaplamalı düşünme becerisinin bir ölçekle ölçümü yerine oyun geliştirme, nitel yöntemleri kullanma gibi derin yöntemler ihtiyaç olduğu vurgulanmıştır (Denner ve Werner, 2011; Denner, Werner ve Ortiz, 2012). Bunun yanı sıra González (2015) hesaplamalı düşünmenin ölçümü için ortaokul öğrencilerine yönelik çoktan seçmeli test geliştirmiştir. Bu çalışmada, öğrencilerin hesaplamalı düşünme becerilerinin ölçülmesi amacıyla Korkmaz, Çakır ve Özden (2017) tarafından üniversite öğrencilerine yönelik geliştirilen hesaplamalı düşünme becerisi ölçeği kullanılmıştır. Ölçekte, 5'li likert tipinde 29 madde bulunmaktadır ve ölçekteki her bir madde için öğrencilerin 1-"en olumsuz" ve 5-"en olumlu" değer aralığında puan vermesi istenmiştir. Ölçekten alınan toplam puan 29 ile 145 arasında değişmektedir. Hesaplamalı düşünme becerileri ölçeğinin; yaratıcılık, algoritmik düşünme, işbirliklilik, eleştirel düşünme ve problem çözme olmak üzere beş boyutu bulunmaktadır. Ölçeğin geliştirme çalışmasında ölçeğin tamamının Cronbach alpha güvenirlik katsayısı .822 olarak; yaratıcılık, algoritmik düşünme, işbirliklilik, eleştirel düşünme ve problem çözme boyutları için güvenirlik katsayıları sırasıyla .843, .869, .865, .784 ve .727 olarak hesaplamıştır. Bu çalışmada, ön test ve son test olarak uygulanan ölçeğin Cronbach alpha güvenirlik katsayıları ön test için .782 ve son test için .894'tür. Çalışmanın ön testinde toplanan verilere göre yaratıcılık, algoritmik düşünme, işbirliklilik, eleştirel düşünme ve problem çözme boyutları için Cronbach alpha güvenirlik katsayıları sırasıyla .55, .866, .777, .793 ve .743 olarak hesaplanmıştır. Son testte toplanan verilere göre yaratıcılık, algoritmik düşünme, işbirliklilik, eleştirel düşünme ve problem çözme boyutları için Cronbach alpha güvenirlik katsayıları sırasıyla .774, .825, .885, .822 ve .737'dir.

Uygulama Süreci

Uygulama sürecinde izlenen aşamalar Şekil 1 de verilmiştir.



Şekil1. Uygulama sürecinde izlenen aşamalar

Uygulama sürecinde öncelikle ders sürecinde yapılacak olan blok tabanlı programlama etkinlikleri tasarlanmıştır. Tasarlanan etkinlikler, öğrencilerin ortak aldıkları ders sürecinde gerçekleştirilmiştir. Ders içeriklerinin tasarlanmasında ve sıralanmasında, programlama öğretiminde karşılaşılan zorluklara bağlı olarak bir problemin çözümüne yönelik algoritma oluşturma, koşulların ve döngülerin kullanımı konuları ele alınmıştır. Programlama öğretiminde sistematik bir yaklaşım önerilmediğinden (Erümit ve diğerleri, 2018) ve ülkelerdeki ve kültürlere bağlı olarak eğitim sistemlerinin farklılığından programlama öğretimi doğrudan sisteme entegre edilemediğinden dolayı (Hsu ve diğerleri, 2018) programlama içeriği animasyon hazırlama (Lee, 2011) olarak görülmesine ve okullarda bu dersin öneminin anlaşılamamasına neden olabilmektedir (Akpınar ve Altun, 2014). Bundan dolayı bu araştırmadaki konular ele alınırken Scratch uygulamasının sadece bir uygulama geliştirme aracı olduğu vurgulanmış ve ders içeriklerinde bir problem kapsamında girdi, süreç ve çıktı oluşturma kavramlarına odaklanılmıştır. Bu kapsamda Scratch uygulamasında ders içeriklerinde ele alınan konular ve konuların sıralaması aşağıdaki gibidir:

A. Algoritma, program, programcı, problem, girdi, çıktı, değişken vb. temel programlama kavramları
B. Scratch uygulamasının tanıtımı ve uygulamada bulunan bloklar
    a. Temel girdi (olaylar ve algılama bileşeni) ve çıktı (görünüm bileşeni) işlemleri
    b. Aritmetik ve mantıksal operatörler
C. Koşulların kullanımı (Kontrol bileşeni)
    a. Girilen bir sayının tek veya çift olduğunun belirlenmesi
D. Döngülerin kullanımı (Kontrol bileşeni)
    a. Belirli bir aralıktaki sayıları söyleyen uygulama
E. Scratch uygulamasında arayüz işlemleri
    a. Kukla, dekor, sahne kavramları
    b. Kukla hareket ettirme
    c. Dekor değiştirme
    d. Bir kuklayı buton olarak tanımlama
F. Problemlere yönelik algoritma oluşturma ve bunları uygulama

a. Değişkenler arasında değer aktarma
b. Girilen iki sayıyı karşılaştırma
c. Belirli bir aralıktaki sayıları toplayan uygulama
d. Girilen bir N değeri için 1'den N'e kadar olan sayıların toplamını hesaplayan uygulama
e. Girilen bir N değeri için 1'den N'e kadar olan tek sayıların toplamını hesaplayan uygulama
f. Girilen bir N değeri için 1'den N'e kadar olan çift sayıların toplamını hesaplayan uygulama
g. Girilen bir N değeri için N! Değerini hesaplayan uygulama
h. Girilen bir sayının mükemmel sayı olup olmadığını söyleyen uygulama.
i. Girilen a ve b değerleri için ab değerini hesaplama.
j. Girilen N tane değerden en küçük ve en büyük değerleri bulma
k. Bilgisayarın tuttuğu bir sayıyı bulma uygulaması
l. Bireyin tuttuğu bir sayıyı bilgisayarın bulması
m. Blok kullanarak fonksiyon oluşturma
n. abc üç basamaklı sayı olmak üzere abc = a3 + b3 + c3 eşitliğini sağlayan sayıların bulunması

Ders konuları listesinde her bir konu için Scratch uygulamasında kullanılan bileşen parantez içerisinde belirtilmiştir. Araştırma sürecinde ele alınan konular 4 hafta boyunca 4 saatlik ders boyunca işlenmiştir. Uygulamanın ilk haftasında öğrencilere programlama ile ilgili temel kavramlar, günümüzde programlamanın yeri ve önemi, algoritma geliştirme, programlama dillerinin yapısı gibi temel kavramlar ve blok tabanlı programlama aracı olarak kullanılacak Scratch uygulamasının arayüzü ve blokların yapısı anlatılmıştır. İkinci hafta, girilen bir metnin kukla tarafından söylenmesi uygulaması ile programlama etkinlikleri başlatılmıştır. Bu hafta bir problem karşısında girdi, süreç ve çıktı planlamasının nasıl yapılacağı ve metinsel olarak oluşturulan bir algoritmanın Scratch uygulamasında hangi bloklar kullanılarak yapılabileceği öğrencilere açıklanmıştır. Bu duruma örnek bir uygulama Şekil 2 de verilmiştir.

sayi = 1 den 10'a kadar 1 artarak tekrarla

Şekil2. Oluşturulan algoritmanın blok tabanlı geliştirme karşılığı örneği

Üçüncü ve dördüncü hafta boyunca koşul ve döngüler kullanarak çözülebilecek problem örnekleri işlenmiştir. Bu haftalarda ele alınan örnek bir problem olan "Girilen N tane sayıdan en büyüğünü söyleyen uygulamayı geliştiriniz." Probleminin çözümü Şekil 3 de verilmiştir.

BAŞLA

YAZ(“N değerini gir”)

OKU(N)

enbuyuk = 0

say = 1 den N’e kadar 1 artarak tekrarla

YAZ(say + “. Değeri gir”)

OKU(soylenensayi)

EĞER(enbuyuk < soylenensayi)

Enbuyuk = söylenen sayi

YAZ(“Girilen sayıların en büyüğü:”+ enbuyuk)

BİTİR

Şekil3. Girilen N tane sayıdan en büyüğünü söyleyen uygulama

Programlama etkinlikleri kapsamında yapılan uygulamaların tamamı Şekil 2 ve Şekil 3 de görüldüğü gibi öncelikle algoritma olarak yazılmış, ardından Scratch uygulamasında uygulanmıştır. Programlama etkinliklerinde dekor ve kuklaların değişiminden daha çok var olan bir problemin çözümünün Scratch uygulaması kullanılarak nasıl yapılabileceği üzerine odaklanmıştır.

Verilerin Analizi

Çalışmada toplanan verileri SPSS 20 uygulaması kullanılarak analiz edilmiştir. Öğrencilerin programlamaya yönelik öz yeterlik algılarındaki ve hesaplamalı düşünme becerilerinde değişimin incelenmesi için, araştırmada toplanan veriler ilişkili örneklemler t-testi yöntemi ile analiz edilmiştir.

Bulgular

Bu bölümde araştırma sorularına bağlı olarak öğrencilerin programlamaya yönelik öz yeterlik algısına ve hesaplama düşünme becerisi düzeylerine ilişkin bulgulara yer verilmiştir. Araştırmada toplanan verilerin analiz işlemine başlamadan önce verilerin normal dağılım gösterip göstermediklerine ilişkin Kolmogorov-Smirnov testi sonuçları, q-q plot grafikleri ve basıklık-çarpıklık değerleri incelenmiş ve verilerin normalden ciddi bir sapma göstermediği belirlenmiştir.

Programlamaya Yönelik Öz Yeterlik Algısına İlişkin Bulgular

Araştırmada blok tabanlı programlama etkinlikleri öncesinde ve sonrasında öğrencilerin programlamaya ilişkin öz yeterlik algısı ölçeğinden aldıkları ortalama puanların t-testi sonuçları Tablo 3 de verilmiştir.

Tablo 3. Programlamaya Yönelik Öz Yeterlik Algısı Öntest ve Sontest Ortalama Puanların t-testi Sonuçları

Ölçüm (Öz Yeterlik)	N	$\overline{X}$	S	sd	t	p
Öntest	29	24.98	13.31	28	-6.18	.00
Sontest	29	38.91	12.56			

Öğrencilerin blok tabanlı programlama etkinlikleri sonrasında programlamaya yönelik öz yeterlik algılarında anlamlı bir artış olduğu bulunmuştur, t(28) = -6.18, p<.05. Öğrencilerin programlama etkinlikleri öncesinde öz yeterlik algısı puan ortalamaları $\overline{X}$ =24.98, programlama etkinlikleri sonrasında $\overline{X}$ =38.91'e yükselmiştir. Bu bulgu, blok tabanlı programlama etkinliklerinin, öğrencilerin programlamaya yönelik öz yeterlik algılarını artırmada etkiye sahip olduğunu göstermektedir.

Programlamaya ilişkin öz yeterlik algısı ölçeğinin basit ve karmaşık programlama görevleri olmak üzere iki alt boyutu bulunmaktadır. Öğrencilerin, araştırma öncesinde ve sonrasında alt boyutlardan almış oldukları ortalama puanların t-testi sonuçları Tablo 4 de verilmiştir.

Tablo 4. Programlamaya Yönelik Öz Yeterlik Algısı Boyutları Öntest ve Sontest Ortalama Puanların t-testi Sonuçları

Boyut	Min puan	Mak puan	Ölçüm	N	$\overline{X}$	S	sd	t	p
Basit programlama görevleri	3	21	Öntest	29	9.14	5.11	28	-6.49	.00
			Sontest	29	14.81	4.75			
Karmaşık programlama görevleri	6	42	Öntest	29	15.84	8.98	28	-5.05	.00
			Sontest	29	24.10	8.46			

Öğrencilerin basit programlama görevlerinde öntest ve sontest puanları arasında anlamlı bir farklılık bulunmuştur, t(28) = -6.49, p<.05. Öğrencilerin, basit programlama görevleri boyutundaki sontest ortalama puanı $\overline{X}$ =24.98, öntest ortalama puanına $\overline{X}$ =9.14 göre daha

yüksektir. Bu bulgu, blok tabanlı programlama etkinliklerinin öğrencilerin basit programlama görevlerine yönelik öz yeterlik algılarını artırmada etkili olduğunu göstermektedir.

Karmaşık programlama görevlerinde öğrencilerin öntest ve sontest puanları arasında anlamlı bir farklılık çıkmıştır, t(28) = -5.05, p<.05. Öğrencilerin karmaşık programlama görevlerine yönelik öz yeterlik ortalama puanları blok tabanlı programlama etkinlikleri öncesinde $\overline{X}$ =15.84 iken etkinlikler sonrasında $\overline{X}$ =24.10 olmuştur. Bu bulgu, blok tabanlı programlama etkinliklerinin öğrencilerin karmaşık programlama görevlerine yönelik öz yeterlik algılarını artırmada etkili olduğunu göstermektedir.

Hesaplamalı Düşünme Becerisi Düzeyine İlişkin Bulgular

Öğrencilerin, blok tabanlı programlama etkinlikleri öncesinde ve sonrasında hesaplamalı düşünme becerileri ölçeğinden aldıkları ortalama puanların t-testi sonuçları Tablo 5 de verilmiştir.

Tablo 5. Hesaplamalı Düşünme Becerisi Öntest ve Sontest Ortalama Puanların t-testi Sonuçları

Ölçüm (Hesaplamalı Düşünme Becerisi)	N	$\overline{X}$	S	sd	t	p
Öntest	29	112.74	9.50	28	.23	.823
Sontest	29	112.24	12.50			

Öğrencilerin, hesaplamalı düşünme becerileri testinden aldıkları öntest ($\overline{X}$ =112.74) ve sontest ($\overline{X}$ =112.24) puan ortalamaları arasında anlamlı bir farklılık olmadığı bulunmuştur, t(28)=.23, p>.05. Bu bulguya göre, blok tabanlı programlama etkinliklerinin öğrencilerin hesaplamalı düşünme becerileri üzerinde bir etkisi olmadığı söylenebilir. Hesaplamalı düşünme becerisi ölçeğinden alınan puan aralığı 29 ile 145 arasında değişmektedir. Alınabilecek puan aralığının orta noktası (87) dikkate alındığında öğretmen adaylarının öntest ve sontest puanlarının orta noktanın üstünde olduğu görülmektedir. Hesaplamalı düşünme becerisi ölçeğinin alt boyutlarına (yaratıcılık, algoritmik düşünme, işbirliklilik, eleştirel düşünme ve problem çözme) göre öğrencilerin öntest ve sontest puan ortalamalarının t-testi sonuçları Tablo 6 da verilmiştir.

Tablo 6. Hesaplamalı Düşünme Becerisi Boyutları Öntest ve Sontest Ortalama Puanların t-testi Sonuçları

Boyut	Min puan	Mak puan	Ölçüm	N	$\overline{X}$	S	sd	t	p
Yaratıcılık	8	40	Öntest	29	34.90	2.93	28	-.32	.75
			Sontest	29	35.13	3.50			
Algoritmik düşünme	6	30	Öntest	29	19.45	4.42	28	-1.51	.14
			Sontest	29	20.49	4.39			
İşbirliklilik	4	20	Öntest	29	17.45	2.11	28	1.35	.19
			Sontest	29	16.66	2.98			
Eleştirel düşünme	5	25	Öntest	29	18.23	3.54	28	-.69	.49
			Sontest	29	18.73	3.31			
Problem çözme	6	30	Öntest	29	22.72	3.88	28	1.88	.07
			Sontest	29	21.23	3.49			

Hesaplamalı düşünme becerisinin yaratıcılık boyutu incelendiğinde, öğrencilerin blok tabanlı kodlama etkinlikleri öncesinde bu boyuttan aldıkları ortalama öntest ortalama puanı ($\overline{X}$ =34.90) ile uygulama sonrasında aldıkları sontest ortalama puan ($\overline{X}$ =35.13) arasında anlamlı bir farklılık bulunmamaktadır, t(28)=-32, p>.05. Algoritmik düşünme boyutunda, öğrencilerin öntest puan ortalamaları ($\overline{X}$ =19.45) ile son test puan ortalamaları ($\overline{X}$ =20.49) arasında anlamlı bir farklılık bulunmamaktadır, t(28)=-1.51, p>.05. İşbirliklilik boyutu incelendiğinde, bu boyutta öğrencilerin uygulama öncesindeki öntest ortalama puanları ($\overline{X}$ =17.45) ile uygulama sonrasındaki sontest ortalama puanları ($\overline{X}$ =16.66) arasında anlamlı bir farklılık bulunmamaktadır, t(28)=1.35, p>.05. Eleştirel düşünme boyutunda öğrencilerin öntestten aldıkları ortalama puan ($\overline{X}$ =18.23) ile sontestten aldıkları ortalama puan ($\overline{X}$ =18.73) arasında anlamlı bir farklılık bulunmamaktadır, t(28)=-.69, p>.05. Son olarak, problem çözme boyutunda öğrencilerin uygulama öncesindeki öntest puan ortalamaları ($\overline{X}$ =22.72) ile uygulama sonrasındaki sontest puan ortalamaları ($\overline{X}$ =21.23) arasında anlamlı bir farklılık bulunmadığı görülmektedir, t(28)=1.88, p>.05. Bu bulgular, blok tabanlı programlama etkinliklerinin öğrencilerin hesaplamalı düşünme becerisinin alt boyutları üzerinde anlamlı bir etkisi olmadığını göstermektedir. Boyutların her birinin orta noktası incelendiğinde katılımcıların öntest ve sontest puanlarının orta noktanın üzerinde olduğu söylenebilir.

Sonuç ve Öneriler

Bu çalışma kapsamında, blok tabanlı programlama etkinliklerinin öğretmen adaylarının programlamaya ilişkin öz yeterlilik algıları ve hesaplamalı düşünme becerileri üzerindeki etkisi incelenmiştir. Çalışma tek grup ön test-son test deneysel desen olarak yürütülmüştür. Araştırmanın çalışma grubunu Fen Bilgisi Öğretmenliği bölümünde öğrenim gören 29 öğrenci oluşturmaktadır. Araştırmada blok tabanlı programlama aracı olarak Scratch uygulaması kullanılmıştır. 4 haftalık uygulamanın başlangıcında ve bitiminde öğrencilere programlamaya ilişkin öz yeterlilik algısı ve hesaplamalı düşünme becerileri ölçeği uygulanmıştır.

Araştırma sonucunda, blok tabanlı programlama etkinliklerinin öğrencilerin programlama öz yeterliliklerini geliştirme üzerinde olumlu yönde etkisi bulunmuştur. Bu bulgu, Mazman ve Altun (2013) tarafından yapılan çalışmada ortaya çıkan bulgular ile örtüşmektedir. Bu çalışmadaki katılımcıların geçmiş programlama deneyimleri olmadığından ön deneyim düzeyine göre karşılaştırma yapılmamıştır. Benzer şekilde Kasalak (2017) tarafından robotik kodlama etkinliklerinin öğrencilerin programlamaya ilişkin öz yeterlilik algıları üzerindeki etkisi de araştırmanın bu bulgusunu destekler niteliktedir. Programlamaya yönelik öz yeterlilik algısının basit ve karmaşık programlama görevleri alt boyutları incelendiğinde bu boyutlarda da olumlu yönde gelişme olduğu görülmektedir. Bundan dolayı özellikle programlamayı yeni öğrenen öğrencilerde veya programlamadan korkan öğrencilerin öz yeterliliklerinin geliştirilmesi için öğretim sürecinde blok tabanlı programlama etkinliklerinden yararlanılabilir.

Bu araştırmada blok tabanlı programlama etkinliklerinin öğrencilerin hesaplamalı düşünme becerileri ve hesaplama düşünme becerilerinin alt boyutları üzerinde bir etkisi olmadığı görülmüştür. Alanyazında programlama öğretiminin hesaplamalı düşünme becerisini geliştirdiğine yönelik araştırmalar bulunmaktadır (Alsancak Sırakaya, 2019; Rodríguez-Martínez ve diğerleri, 2019; Zhang ve Nouri, 2019). Bu durumun, uygulama öncesinde öğrencilerin ortalama hesaplamalı düşünme becerisi puanlarının, hesaplamalı düşünme becerisi ölçeğinin orta noktasının üstünde olmasından kaynaklandığı düşünülmektedir. Bundan dolayı, sonraki araştırmalarda hesaplamalı düşünme becerisi düşük bireyler üzerinde blok tabanlı programlama etkinlikleri gerçekleştirilerek bu etkinliklerin etkililiği incelenebilir.

Sınırlılıklar

Bu araştırmanın örneklem boyutu çalışmanın en temel sınırlılıklarından biridir. Bu durum, bulguların genellenebilirliğinin sınırlanmasını beraberinde getirmektedir. Bundan dolayı, daha büyük örneklem grupları ile deney ve kontrol grupları oluşturularak blok tabanlı programlama etkinliklerinin programlama öz yeterliliği ve hesaplamalı düşünme becerisi üzerindeki etkisi incelenebilir.

Kaynakça

Aho, A. V. (2012). Computation and Computational Thinking. *The Computer Journal, 55*(7), 832-835. doi:10.1093/comjnl/bxs074

Akpınar, Y., & Altun, A. (2014). Bilgi toplumu okullarında programlama eğitimi gereksinimi. *2014, 13*(1).

Alsancak Sırakaya, D. (2019). Programlama Öğretiminin Bilgi İşlemsel Düşünme Becerisine Etkisi. *Türkiye Sosyal Araştırmalar Dergisi, 23*(2), 575-590.

Altun, A., & Mazman, S. G. (2012). Programlamaya ilişkin Öz Yeterlilik Algısı Ölçeğinin Türkçe Formunun Güvenirlik ve Geçerlik Çalışması. *Eğitimde ve Psikolojide Ölçme ve Değerlendirme Dergisi, 3*(2), 297-308.

Askar, P., & Davenport, D. (2009). An investigation of factors related to self-efficacy for Java Programming among engineering students. *Online Submission, 8*(1).

Bandura, A. (1997). *Self-efficacy: The exercise of control*: Macmillan.

Basogain, X., Olabe, M. Á., Olabe, J. C., & Rico, M. J. (2018). Computational Thinking in pre-university Blended Learning classrooms. *Computers in Human Behavior, 80*, 412-419.

Bayman, P., & Mayer, R. E. (1988). Using conceptual models to teach BASIC computer programming. *Journal of Educational Psychology, 80*(3), 291-298. doi:10.1037/0022-0663.80.3.291

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A., & Miller, P. (1997). Mini-languages: a way to learn programming principles. *Education Information Technologies, 2*(1), 65-83. doi:10.1023/a:1018636507883

CAS. (2015). Computational Thinking: A Guide for Teachers. Retrieved from http://community.computingatschool.org.uk/files/6695/original.pdf

Denner, J., & Werner, L. (2011). *Measuring computational thinking in middle school using game programming.* Paper presented at the Annual Meeting of the American Educational Research Association.

Denner, J., Werner, L., & Ortiz, E. (2012). Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education, 58*(1), 240-249.

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33-39.

Dohn, N. B. (2019). Students' interest in Scratch coding in lower secondary mathematics. *British Journal of Educational Technology*. doi:10.1111/bjet.12759

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. doi:https://doi.org/10.1016/j.chb.2017.08.017

Erümit, K. A., Karal, H., Şahin, G., Aksoy, D. A., Aksoy, A., & Benzer, A. İ. (2018). Programlama Öğretimi için Bir Model Önerisi: Yedi Adımda Programlama. *2018, 44*(197). doi:10.15390/eb.2018.7678 %j eGİtİm ve bİlİm

Faber, H. H., Wierdsma, M. D., Doornbos, R. P., van der Ven, J. S., & de Vette, K. (2017). Teaching computational thinking to primary school students via unplugged programming lessons. *Journal of the European Teacher Education Network, 12*, 13-24.

Furber, S. (2012). *Shut down or restart? The way forward for computing in UK schools*. In. Retrieved from https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

González, M. R. (2015). *Computational thinking test: Design guidelines and content validation.* Paper presented at the Proceedings of EDULEARN15 conference.

Heintz, F., Mannila, L., & Färnqvist, T. (2016, 12-15 Oct. 2016). *A review of models for introducing computational thinking, computer science and computing in K-12 education.* Paper presented at the 2016 IEEE Frontiers in Education Conference (FIE).

Horvátha, G. (2018). *A web-based programming environment for introductory programming courses in higher education.* Paper presented at the Annales Mathematicae et Informaticae.

Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education, 126*, 296-310. doi:https://doi.org/10.1016/j.compedu.2018.07.004

ISTE, & CSTA. (2011). Operational Definition of Computational Thinking. Retrieved from https://id.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf

ISTE. (2019a). ISTE Standards for Students. Retrieved from https://www.iste.org/standards/for-students

ISTE. (2019b). ISTE Standarts for Educators. Retrieved from https://www.iste.org/standards/for-educators

Janpla, S., & Piriyasurawong, P. (2018). The Development of Problem-Based Learning and Concept Mapping Using a Block-Based Programming Model to Enhance the Programming Competency of Undergraduate Students in Computer Science. *TEM Journal, 7*(4), 708.

Kasalak, İ. (2017). *Robotik Kodlama Etkinliklerinin Ortaokul Öğrencilerinin Kodlamaya İlişkin Özyeterlik Algılarına Etkisi Ve Etkinliklere İlişkin Öğrenci Yaşantıları.* Eğitim Bilimleri Enstitüsü,

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior, 72*, 558-569. doi:https://doi.org/10.1016/j.chb.2017.01.005

Lee, Y.-J. (2011). Empowering teachers to create educational software: A constructivist approach utilizing Etoys, pair programming and cognitive apprenticeship. *Computers & Education, 56*(2), 527-538. doi:https://doi.org/10.1016/j.compedu.2010.09.018

Mazman, S. G., & Altun, A. (2013). Programlama–I dersinin BÖTE bölümü öğrencilerinin programlamaya ilişkin öz yeterlilik algıları üzerine etkisi. *Öğretim Teknolojileri & Öğretmen Eğitimi Dergisi, 2*(3).

McGill, T. J., & Volet, S. E. (1997). A Conceptual Framework for Analyzing Students' Knowledge of Programming. *Journal of Research on Computing in Education, 29*(3), 276-297. doi:10.1080/08886504.1997.10782199

Orvalho, J. (2017). *Computational thinking for teacher education.* Paper presented at the Scratch2017BDX: Opening, inspiring, connecting.

Ramalingam, V., & Wiedenbeck, S. (1998). Development and Validation of Scores on a Computer Programming Self-Efficacy Scale and Group Analyses of Novice Programmer Self-Efficacy. *19*(4), 367-381. doi:10.2190/c670-y3c8-ltj1-ct3p

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer science education, 13*(2), 137-172.

Rodríguez-Martínez, J. A., González-Calero, J. A., & Sáez-López, J. M. (2019). Computational thinking and mathematics using Scratch: an experiment with sixth-grade students. *Interactive Learning Environments*, 1-12. doi:10.1080/10494820.2019.1612448

Sáez-López, J.-M., Sevillano-García, M.-L., & Vazquez-Cano, E. (2019). The effect of programming on primary school students' mathematical and scientific understanding: educational use of mBot. *Educational Technology Research and Development*, 1-21.

Schunk, D. H. (1991). Self-efficacy and academic motivation. *Educational psychologist, 26*(3-4), 207-231.

Selby, C., & Woollard, J. (2014). Refining an understanding of computational thinking.

Tsai, C.-Y. (2019). Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy. *Computers in Human Behavior, 95*, 224-232. doi:https://doi.org/10.1016/j.chb.2018.11.038

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

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717-3725.

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Commun. ACM, 60*(4), 55-62. doi:10.1145/2994591

Yevseyeva, K., & Towhidnejad, M. (2012). *Work in progress: Teaching computational thinking in middle and high school.* Paper presented at the 2012 Frontiers in Education Conference Proceedings.

Yildiz Durak, H., Yilmaz, F. G. K., & Yilmaz, R. (2019). Computational Thinking, Programming Self-Efficacy, Problem Solving and Experiences in the Programming Process Conducted with Robotic Activities. *Contemporary Educational Technology, 10*(2), 173-197.

Yükseltürk, E., & Altıok, S. (2015). Bilişim teknolojileri öğretmen adaylarının bilgisayar programlama öğretimine yönelik görüşleri. *Amasya Üniversitesi Eğitim Fakültesi Dergisi, 4*(1), 50-65.

Yükseltürk, E., & Altıok, S. (2018). Blok Tabanlı Programlama. In Y. Gülbahar (Ed.), *Bilgi İşlemsel Düşünmeden Programlamaya* (pp. 242-266). Ankara: Pegem Akademi.

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