PROGRAMLAMA ÖĞRETİMİNDE EĞİTSEL PROGRAMLAMA DİLİNİN FARKLI KULLANIMLARININ PROGRAMLAMA BAŞARISI VE KAYGISINA ETKİSİ

Faruk DEMİR

Doktora Tezi
Bilgisayar ve Öğretim Teknolojileri Eğitimi
Ana Bilim Dalı
Yrd. Doç. Dr. Engin KURŞUN
2015

T.C.
ATATÜRK ÜNİVERSİTESİ
EĞİTİM BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR VE ÖĞRETİM TEKNOLOJİLERİ EĞİTİMİ
ANA BİLİM DALI

PROGRAMLAMA ÖĞRETİMİNDE EĞİTSEL PROGRAMLAMA DİLİNİN FARKLI KULLANIMLARININ PROGRAMLAMA BAŞARISI VE KAYGISINA ETKİSİ

(The Effect of Different Usage of the Educational Programming Language in Programmıng Educatıon on the Programmıng Anxıety and Achıevement)

DOKTORA TEZİ

Faruk DEMİR

Danışman: Yrd. Doç. Dr. Engin KURŞUN

ERZURUM
Aralık, 2015

KABUL VE ONAY TUTANAĞI

Yrd. Doç. Dr. Engin KURŞUN danışmanlığında, Faruk DEMİR tarafından hazırlanan "Programlama Öğretiminde Eğitsel Programlama Dilinin Farklı Kullanımlarının Programlama Başarısı Ve Kaygısına Etkisi" başlıklı çalışma 28/12/2015 tarihinde yapılan savunma sınavı sonucunda başarılı bulunarak jürimiz tarafından Bilgisayar ve Öğretim Teknolojileri Eğitimi Ana Bilim Dalında Doktora Tezi olarak kabul edilmiştir.

Başkan: Doç. Dr. Hasan KARAL İmza: .....

Danışman: Yrd. Doç. Dr. Engin KURŞUN İmza: .....

Jüri Üyesi: Doç. Dr. Ercan TOP İmza: .....

Jüri Üyesi: Doç. Dr. Enver TATAR İmza: .....

Jüri Üyesi: Doç. Dr. Selçuk KARAMAN İmza: .....

Yukarıdaki imzaların adı geçen öğretim üyelerine ait olduğunu onaylarım.

12/02/2016

Prof.Dr.Kemal DOYMUŞ
Enstitü Müdürü

TEZ ETİK VE BİLDİRİM SAYFASI

Doktora Tezi olarak sunduğum "Programlama Öğretiminde Eğitsel Programlama Dilinin Farklı Kullanımlarının Programlama Başarısı ve Kaygısına Etkisi" başlıklı çalışmanın, tarafımdan, bilimsel ahlak ve geleneklere aykırı düşecek bir yardıma başvurmaksızın yazıldığını ve yararlandığım eserlerin kaynakçada gösterilenlerden olduğunu, bunlara atıf yapılarak yararlanılmış olduğunu belirtir ve onurumla doğrularım.

Tezimin kağıt ve elektronik kopyalarının Atatürk Üniversitesi Eğitim Bilimleri arşivlerinde aşağıda belirttiğim koşullarda saklanmasına izin verdiğimi onaylarım.

Lisansüstü Eğitim-Öğretim yönetmeliğinin ilgili maddeleri uyarınca gereğinin yapılmasını arz ederim.

☒ Tezimin tamamı her yerden erişime açılabilir.

☐ Tezim sadece Atatürk Üniversitesi yerleşkelerinden erişime açılabilir.

☐ Tezimin ..... yıl süreyle erişime açılmasını istiyorum. Bu sürenin sonunda uzatma için başvuruda bulunmadığım takdirde, tezimin tamamı her yerden erişime açılabilir.

28/12/2015

Faruk DEMİR

ÖZET

DOKTORA TEZİ

PROGRAMLAMA ÖĞRETİMİNDE EĞİTSEL PROGRAMLAMA DİLİNİN FARKLI KULLANIMLARININ PROGRAMLAMA BAŞARISI VE KAYGISINA ETKİSİ

Faruk DEMİR

2015, 211 sayfa

Programlama ile ilk karşılaşan acemi kullanıcılar programlamanın soyut yapısından, mantığının zorluğundan, olumsuz algılarından ve kaygısından dolayı zorluk çekmektedirler. Bu çalışmada, programlama öğretiminin önündeki zorlukların aşılmasına yönelik olarak, programlama öğretiminde eğitsel programlama dilinin derse entegrasyonunun, öğrencilerin akademik başarısına, performansına ve bilgisayar programlamaya yönelik kaygılarına olan etkisi araştırılmıştır. Deneysel yöntemlerden kontrol grupsuz ön-test son-test deney deseninin kullanıldığı çalışma, üç deney grubu ile yürütülmüştür. Birinci grupta eğitsel programlama dili, dersin uygulama kısmına, ikinci grupta teori kısmına, üçüncü grupta ise hem teori hem de uygulama kısmına entegre edilerek gruplar oluşturulmuştur. Rastgele örneklem yöntemi ile oluşturulan gruplar 61 erkek ve 26 kız olmak üzere 87 öğrenciden oluşmaktadır. Akademik başarının değerlendirilmesinde iki yöntem kullanılmıştır. İlki, ön bilgiyi ölçmeye yönelik olarak ön-test ve uygulama sonrası başarı değişimini saptamaya yönelik son-test, ikincisi ise, uygulama süreci içerisinde başarının değişimini belirlemeye yönelik yapılan beş adet performans testinden oluşmaktadır. Ayrıca grupların bilgisayar programlama kaygılarını ölçmeye yönelik Cheung vd. (1990) tarafından geliştirilen ölçek, araştırmacı tarafından geçerlilik ve güvenilirlik çalışması yapılarak Türkçeye uyarlanmış ve ön-test, son-test olarak kullanılmıştır. Öğrencilerin başarıları arasındaki ilişkiyi ve programlamaya yönelik kaygılarını belirlemek için varyans ve kovaryans analizlerinden, ders içi performanslarını belirlemek için ise Kruskal-Wallis test analizlerinin sonuçlarından yararlanılmıştır.

Çalışma sonucunda grupların akademik başarı puanları incelendiğinde, eğitsel programlama dilinin derse entegrasyon şeklinin, bilişsel ve üst düzey bilişsel beceri puanları açısından, teori entegre ve uygulama entegre grupları ile hem teori hem de

uygulama entegre grubu arasında anlamlı bir fark olduğu görülmektedir. Bu anlamlı farklılığın hem teori hem de uygulama entegre grup lehine oluştuğu tespit edilmiştir. Alt düzey bilişsel beceri puanları açısından ise gruplar arasında anlamlı bir farklılığın oluşmadığı görülmektedir. Performans açısından bakıldığında, ilk dört performans testinde eğitsel programlama dilinin teoriye entegre edildiği grup ile hem teori hem de uygulamaya entegre edildiği grup arasında anlamlı bir fark görülmektedir. Son performans testinde ise gruplar arasında anlamlı bir fark oluşmamıştır. Performans testlerinin bu sonuçları, akademik başarı testi ile gruplar bazında uyumluluk göstermekte olup, çalışmanın güvenirliğini olumlu yönde etkilemektedir. Programlamaya yönelik kaygı açısından grup içerisinde ön-test son-test sonuçlarına bakıldığında, eğitsel programlama dilinin uygulamaya entegre grup ile hem teoriye hem de uygulamaya entegre gruplarda anlamlı bir fark görülürken, teoriye entegre edilen grupta anlamlı bir fark oluşmamıştır. Bu bulgular eğitsel programlama dilinin dersin hem teori hem de uygulama kısmına entegre edilmesinin bilişsel ve üst düzey bilişsel beceriler açısından öğrencilerin programlamaya yönelik akademik başarılarını, performanslarını ve bilgisayar programlamaya yönelik kaygılarını olumlu yönde etkilediğini göstermektedir. Akademik başarı ve ders içi performansı artırmak ve bilgisayar programlamaya yönelik kaygıyı azaltmak için, eğitsel programlama dilleri, dersin hem teori hem de uygulamasına entegre edilerek kullanılabilir.

Anahtar Sözcükler: Programlama, Eğitsel Programlama Dili, Scratch, Başarı, Performans, Bilgisayar Programlama Kaygısı

ABSTRACT

Ph. D. Thesis

THE EFFECT OF DIFFERENT USAGE OF THE EDUCATIONAL PROGRAMMING LANGUAGE IN PROGRAMMING EDUCATION ON THE PROGRAMMING ANXIETY AND ACHIEVEMENT

Faruk DEMİR

2015, 211 page

Novice users first encounter with programming are having difficulty due to the abstract structure of programming, the difficulty of its logic and negative perceptions and anxieties. In this study, in order to overcome challenges of teaching programming, the impact of the educational programming language integration into the programming education on anxieties about student achievement, performance and programming was investigated. The study was conducted with a complete experimental method, pre-test post-test experimental design without a control group and it was carried out with three experimental groups. The educational programming language was integrated into practice in the first experimental group, in the second group, it was integrated only into the theory and in the third group and it was integrated into both the theory and practice. The group formed by the random sampling method consists of 87 students including 61 males and 26 females. Two methods were used to assess the achievement. The first method was a pre-test for measuring prior knowledge and a post-test to determine changes in the achievement after the implementation. The other method was five performance tests which have been conducted to determine the variation in achievement during the implementation. In addition, to measure computer programming anxiety scores of groups, a scale developed by Cheung et al. (1990) was used. The scale was adapted to Turkish by doing the validity and reliability study by the researcher and it was used as pre-test and post-test. Covariance and variance analysis were used to determine the relationship between achievements of students and anxieties about the programming and Kruskal-Wallis test was used to determine the classroom performance.

When academic achievement scores of groups were examined with respect to the form of the integration of educational programming language to the programming course, it appears to be a significant difference between the theory and practice integrated groups and both theory and practice integrated group in terms of cognitive skills and higher-level cognitive scores. This significant difference has been found to occur in favor of the group in which both the theory and practice integrated. There has not been a meaningful difference among groups in terms of lower-level cognitive skills scores. In terms of performance, while a significant difference was observed between the group in which the educational programming language was integrated only into theory part of the course and the group in which the educational programming language was integrated into theory and practice, significant differences did not occur in the final performance test between all three groups. These results of performance tests show compatibility with academic achievement tests and affect the reliability of the study positively. When results of pre-test and post-test were examined in terms of programming anxiety, a significant difference was observed within the group in which the educational programming language was integrated into both theory and practice and within the group in which the educational programming language was integrated only into practice. These findings show that the integration of the educational programming language into both theory and practice influences students' academic achievement and anxiety scores towards programming positively. Educational programming languages can be used by integrating into both theory and practice to improve performance and academic achievement and to reduce the anxiety towards computer programming.

Keywords: Programming, Educatıonal Programmıng Language, Scratch, Achievement, Performance, Computer Programming Anxiety,

ÖNSÖZ

Bu çalışmayı yürütürken her zaman bana yol gösteren, yardımlarını esirgemeyen ve daima çalışmalarıma destek olup ışık tutan Danışmanım Sayın Hocam Yrd. Doç. Dr. Engin KURŞUN'a gösterdiği ilgi, hoşgörü ve çabalarından ötürü teşekkürlerimi saygılarımla sunarım.

Tez izleme jürimde bulunan ve doktora eğitimimde eleştiri ve fikirleriyle bana her konuda yardımcı olan beni yönlendiren Sayın Hocam Doç. Dr. Selçuk KARAMAN'a ve Sayın Hocam Doç. Dr. Enver TATAR'a teşekkür ederim. Doktora eğitim sürecinde gösterdikleri ilgi, hoşgörü ve çabalarından dolayı Sayın Hocam, Doç. Dr. Yüksel GÖKTAŞ'a, Prof. Dr. Mustafa SÖZBİLİR'e ve Yrd. Doç. Dr. Türkan KARAKUŞ'a şükranlarımı sunuyorum.

Çalışmam boyunca hep yanımda olan ve desteğini esirgemeyen Yrd. Doç. Dr. Cemil ŞAHİN hocama, doktora eğitimi sırasında olumlu katkılarıyla destek veren Ömer BİLEN'e, Arş. Gör. Turgay DEMİREL'e, Gümüşhane Üniversitesi akademik personellerinden Öğr. Gör. Nihat ÖZDEMİR'e, Öğr. Gör. R. Özgür DOĞAN'a, Öğr. Gör. Oral DEMİR'e ve Yrd. Doç. Dr. Özcan OTKUN'a teşekkür ederim.

Bu çalışmayı yürütürken desteklerini her an yanımda hissettiğim Değerli annem ve babama, Sevgili eşim Hülya, oğlum Furkan ve kızım Esra'ya minnettarım.

Erzurum-2015 **Faruk DEMİR**

İÇİNDEKİLER

ÜÇÜNCÜ BÖLÜM

DÖRDÜNCÜ BÖLÜM

BEŞİNCİ BÖLÜM

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

BİRİNCİ BÖLÜM

1. GİRİŞ

1.1. Problem Durumu

Çağımızda bilgisayarların yaşam içindeki işlevi göz önüne alındığında, günlük hayatta karşılaşılan problemlere çözümler üretilmesini sağlayan programlara duyulan ihtiyacında boyutu daha iyi anlaşılmaktadır. Programlamaya duyulan bu ihtiyaç, programlarında önemini ortaya koymaktadır. Bu ihtiyaçların karşılanmasına yönelik pek çok program yazılmasının gerektiği, bunun içinde iyi bir programlama eğitimi ve öğretimine ihtiyaç duyulduğu ortadadır. Programların bu denli önemli olduğu günümüzde, programlama eğitimi ve öğretimi de o ölçüde önem arz etmekte olup, bunun içinde yaygın olarak programlama eğitimi verilmeye çalışılmaktadır (Perry, 2009).

Günümüz eğitim dünyasında bilgisayar programlama eğitimi ve öğretimi, dünyanın her yerinde yaygın olarak verilmektedir. Programlamaya yönelik verilen programlama temelleri dersleri, bilgisayar bilimleri alanında öğrenim görmeye başlayan öğrencilerin, neredeyse tamamının önünde zorlu bir engel teşkil ettiği ve akademik başarının, beklenilen düzeyde olmadığı bilinen bir gerçektir (Proulx, 2000; Lahtinen, Ala-Mutka ve Jarvinen, 2005; Fujiwara, Fushida, Tamada, Igaki ve Yoshida, 2012; Cornforth, 2014).

Programlama eğitimindeki akademik başarının, dünya genelinde beklenilen ve istenilen seviyede olmaması eğitim ve öğretimin her kademesinde olduğu gibi üniversite seviyesinde de, programlama eğitiminin ve öğretiminin kendine özgü zorlukları ve engelleri olduğunu göstermektedir. Bu engellerden bir tanesi programlama dersini alan üniversite öğrencilerinin profilleridir. Çalışmanın gerçekleştirildiği Meslek Yüksekokulları, üniversite içerisinde öğrenci olarak yarısının sınavla diğer yarısının ise sınavsız geçiş ile yerleştirildiği birimlerdir. Bu öğrencilerin sınavsız gelenleri belirli bir pratik bilgiye sahipken, sınavla gelenleri pratik bir bilgi birikimine sahip değillerdir. Pratik bilgi birikimine sahip olmayanların ise akademik olarak alt yapıları, pratik

bilgiye sahip olanlara göre daha iyi durumdadır. Ayrıca bugünkü meslek liseleri düşük başarılı öğrenciler, düşük kalitede eğitim, düşük statülü işler, üniversite yolu kapalı imajlarını içermektedir. Bu durum akademik seviyesi iyi olan öğrencilerin meslek liselerini tercih etmemelerine sebep olmaktadır. Meslek lisesine gelen bu öğrencilerin sınavsız geçişle Meslek Yüksekokullarına yerleşmeleri Melek Yüksekokullarının öğrenci profilini olumsuz etkilemektedir (Ceylan, 2009; Ceylan ve Elibol, 2009; Ceylan, Avan ve Elibol, 2009; Ünver, Yaylı ve Ceylan,2009; Ceylan ve Erdoğan,2009; Ergüzen ve Ceylan, 2009; Kurşat,2014; Şahin ve Fındık,2008; Şencan, 2008; Uçar ve Özerbaş, 2013; Kenar, 2015). Bu öğrenci profili diğer derslerde olduğu gibi programlama temelleri dersinde de akademik başarıyı olumsuz yönde etkilemektedir. Bennedsen ve Caspersen (2007), programlama dersini alan her üç öğrenciden birinin programlama derslerinden başarısız olduğunu belirtirken, Jenkins (2002) çalışmasında, üniversite seviyesinde ilk kez alınan programlama dersinin, konuları açısından öğrenilmesi en zor derslerden biri olduğunu belirtmektedir.

Araştırmacılar programlama öğrenimindeki zorlukların nedenlerini tespit edebilmek için (Michael, 1999; Nedzad ve Yasmeen, 2001; Michael ve John, 2001; Michael ve Desmond, 2002; Maria vd., 2003) birçok çalışma yapmışlardır (Sivasakthi ve Rajendran, 2011). Yapılan bu çalışmalarda bilgisayar programlama öğretimi, bilgisayar programlamaya yeni başlayan ve soyut kavramlardan oluşan programlama ortamıyla ilk kez karşılaşan öğrencilerin, neredeyse tamamının önünde büyük bir engel olarak durmaktadır (Cornforth, 2014; Fujiwara, Fushida, Tamada, Igaki ve Yoshida, 2012; Lahtinen, Ala-Mutka, Jarvinen, 2005; Proulx, 2000).

Alanyazında programlama öğretimi önündeki engeller ve zorluklar hakkında araştırmacıların birçok görüş bildirdikleri görülmektedir. Rountree, Robins ve Rountree (2003) yaptıkları çalışmada, programlama eğitiminin önündeki engellerin ve zorlukların; programlama öğreniminin karmaşık yapısı, bu yapıya özgü bilgileri öğrenme, bu bilgilerle ilgili yeni stratejiler geliştirme ve pratikte program yazabilme becerisine sahip olma gibi gereksinimlerden kaynaklı olduğunu belirtmektedirler. Jenkins (2002) yaptığı çalışmada programlamanın, öğrenilmesi en zor olan derslerden biri olduğunu belirtmektedir. Proulx (2000) göre ise başlangıç seviyesindeki temel programlama öğretimi, bilgisayar bilimleri alanı içerisinde öğrenim görmeye yeni başlayan öğrencilerin, neredeyse tamamına yakınının programlama dersinden

kalmalarına veya dersi bırakmalarına neden olmaktadır. Meisalo ve arkadaşları yaptıkları çalışmada benzer bulgulara ulaşmışlardır. Bu çalışmaya göre, programlama eğitimi alan öğrencilerinin %32'sinin, henüz eğitimin ilk yılında bilgisayar programlama dersini devamsızlıktan dolayı bıraktığını tespit etmişlerdir. Bu durumun sebebi araştırıldığında öğrenciler, programlama derslerinin teori ve uygulama yönlerinin çok zor olduğunu ve başarı kaygısı çektiklerini belirtmişlerdir (Meisalo, Suhonen, Torvinen ve Sutinen, 2002). Dünya genelindeki üniversitelerde verilen programlama temelleri dersi ile ilgili Kinnunen ve Malmi'nin (2008) yaptıkları araştırmada, bu derslere kayıt yaptıran öğrencilerin %20 - %40 oranında, ya aldıkları programlama dersini ya da bölümü bıraktıklarını tespit etmişlerdir. Dünya çapında çok uluslu ve çok kuruluşlu yürütülen bir çalışmada, üniversite öğrencilerinin programlama temelleri dersindeki genel akademik başarılarının düşük seviyede olduğu belirtmişlerdir (McCracken vd., 2001).

Programlama derslerindeki genel akademik başarının düşük olması, öğrencilerde motivasyon sorunlarına ve kaygıya sebep olmaktadır. Motivasyon sorunu yaşayan ve kaygı duyan bir öğrenci, öğrenmeyi gerçekleştirme de problem yaşamakta ve başarısız olabilmektedir (Jenkins, 2002). Programlama öğretiminde akademik başarıyı etkilediği düşünülen ve incelemeye alınan faktörlerden biri de, bilgisayar programlamaya yönelik kaygı düzeyidir. Bilgisayar programlama kaygı düzeyi yüksek öğrencilerin program yazma olasılıkları düşük olduğu, anlatılan konuları kavrayamadıkları ve yapılan etkinliklerde kendilerini geliştiremedikleri görülmektedir (Gürcan, 2005). Connolly, Murphy ve Moore (2007) yaptıkları çalışmada, öğrencilerin özellikle bilgisayar programlama temelleri dersindeki soyut ve karmaşık yapıları anlamakta ve bu soyut yapıları somutlaştırmakta güçlük çektikleri, stres ve kaygı yaşadıklarını belirtmişlerdir. Ancak, programlama öğrenimi kaygısı, farklı kişi ve durumlarda farklılık göstermektedir. McInerney (1990) yaptığı araştırmasında programlama dersini alan ve sorunsuz bir şekilde tamamlayan öğrencilerin kaygı düzeylerinin, diğer öğrencilere göre daha düşük olduğu görülmektedir. Dersi tamamlayan öğrencilerden erkek olanların kaygılarının kız olanlara göre, bilgisayarı sadece eğlence amaçlı kullananların kaygılarının bir ders için veya herhangi bir amaç için kullanmak zorunda kalanlara göre daha düşük olduğu bulunmuştur.

Programlamaya yönelik kaygı ve öğretim yaklaşımlarının yanı sıra, programlama eğitiminde başarı elde edebilmek için, inceleme altına alınan ve başarıyı etkilediği düşünülen faktörlerden bir diğeri ise, programlama eğitimi süresince öğrencilere sunulacak programlama dilidir (Jenkins, 2002). Bu bağlamda, programlama eğitimde kullanım için tercih edilecek programlama dilinin hangisi olması gerektiği sorusu, geçen zamana rağmen bilgisayar eğitimi ile ilgili alanyazınında önemini korumaktadır. Programlama eğitimde kullanılacak programlama dili kadar, hangi yöntem, teknik ve araçlara başvurulacağı da bu alanda çalışan araştırmacıların üzerinde durduğu konulardandır (Brusilovsky, Calabrese, Hvorecky, Kouchnirenko ve Miller, 1997).

Programlama eğitiminde tercih edilecek programlama dilleri için önerilen diğer bir husus ise programlama eğitiminde eğitsel programlama dillerinin kullanılabileceği ile ilgilidir (Navarro-Prieto ve Cañas, 2001). Bilhassa geçtiğimiz yıllar içerisinde, acemilere yönelik eğitsel programlama dili olan blok-temelli görsel programlama dillerinin uygun olacağı ileri sürülmüştür (Maloney, Resnick, Rusk, Silverman ve Eastmond, 2010). Zira blok-temelli görsel programlama dillerinin genel özelliği, metin tabanlı programlama dillerinin vazgeçilmez bir parçası olan, ancak dilden dile çeşitlilik gösteren sözdizimi ögelerinin yerini doğal dilde yazılmış ve sürükle-bırak yöntemiyle bir araya getirilen program bloklarının almasıdır. Sözdiziminin, acemilere zaman zaman karmaşık gelebilen yapısı, bu diller sayesinde doğal dile veya en azından "yalancı koda" olabildiğince yakın hale getirilmiştir. Bu sayede, öğrencilerin sözdizimi kurallarını ezberlemek zorunda kalmasının, sözdizimi yazımında hatalarla karşılaşmasının ve kod satırlarında boğulmasının önüne geçilmek istenmiştir (Klassen, 2006). Üstelik yalnızca doğru kombinasyonlarla bir araya getirilebilecek şekilde düzenlenmiş blok parçalarından oluşan yap-boz şeklindeki düzenek, öğrencinin program yapısı ve onu oluşturan elemanların görevi hakkında daha somut bir fikir elde edinebilmesini sağlamaktadır. Ancak bugüne kadar programlama öğretiminde kullanım için geliştirilen blok-temelli görsel programlama dillerinin genellikle yaşça küçük öğrenciler için hazırlanmış olduğu belirtilse de (Kelleher, Pausch ve Kiesler, 2007; Utting, Cooper, Kölling, Maloney, ve Resnick, 2010), her yaş grubundan öğrenci için programlama öğretiminde kullanılan bir araç olduğu dikkat çekmektedir.

Blok-temelli görsel programlama dillerinin, her yaş grubundaki öğrencilerin kullanabileceği özellikte olması ve Prensky'nin dijital yerliler olarak adlandırdığı günümüz öğrencilerinin de teknoloji ile iç içe olmaları, bu programlama dillerinin bulunduğu öğretim ortamlarına kolayca adapte olmalarını sağlamaktadır (Prensky, 2001). Dolayısıyla öğretim ortamlarına bilgi ve iletişim teknolojileri araçlarının entegre edilmesi öğrenciler için bir uyum problemi oluşturmayıp, tam tersine fırsatlar sunacağı düşünülmektedir. Öğretim ortamlarına, bilgi ve iletişim teknolojileri araçlarının entegrasyonu gerçekleştirilirken dijital yerli olmayan öğretmenlerin, öğrencilere oranla daha fazla uyum problemi yaşayacağı düşünülmektedir. Yapılan çalışmalar da (Becker, 2001; Ertmer, 2005) öğretmenlerin derslerine bilgi ve iletişim teknolojilerini entegre etme konusunda birçok problemle karşılaştıklarını göstermektedir. Öğretmenlerin bu uyum problemlerini aşabilmeleri için, bilgi ve iletişim teknolojilerine yönelik inanç, istek, değişim, tutum ve direnç gibi pedagoji ile ilgili iç engellerle başa çıkmaları gerekmektedir (Ertmer, 1999). Bahsedilen engellerin yanı sıra öğretmenlerin, öğrenme ve öğretme ortamına, bilgi ve iletişim teknolojilerini nasıl entegre edileceğine ilişkin net bir bilgiye sahip olmadıkları görülmektedir (Eby, 2001; Ertmer vd., 2003, Göktaş, Yıldırım ve Yıldırım, 2008).

Öğretmenlerin, teknoloji ile ilgili algılarının olumlu yönde gelişmesinin ve teknolojiyi kullanma isteği duymalarının, programlama derslerine bu teknolojileri entegre etmelerinde katkı sağlayacağı düşünülmektedir (Göktaş, Yıldırım ve Yıldırım, 2008). Bu noktadan hareketle programların eğitsel programlama dilleri ile oluşturulması ve kullanımlarındaki kolaylıklar ile ilgili yapılan çalışmalar, öğretmenlerde programlamaya karşı olumlu algılar geliştirmiştir. Bu durum öğrenme ortamların da eğitsel programlama dili teknolojilerinin entegrasyonu bakımından avantaj sağlamaktadır (Butler, 2007; Doğan ve Robin, 2008; Renda ve Sprouse, 2010; Cooper, 2003).

Programlama öğretiminde kullanılacak yöntemler konusunda yaptıkları çalışmada, bilgisayar bilimleri üzerine eğitim ve öğretim veren üniversite bölümlerinin birçoğunun, öğrencilerin ilk kez alacağı programlama derslerinin ve bu derslerdeki konuların öğrencilere nasıl aktarılacağı, hangi yöntemlerin kullanılacağı konusunda sıkıntılar yaşadıklarını ve çözüm üretme konusunda yetersiz kaldıklarını belirtmişlerdir (Allison, Orton ve Powell, 2002). Üniversitelerdeki eğitimciler çözüm üretme

noktasında farklı bakış açılarıyla sorunu gidermek için yeni yöntemleri ve öğretim araçlarını kullanarak daha etkili öğretim yöntemleri sunmaları beklenmektedir (Rogerson ve Scott, 2010). Dolayısıyla, bilgisayar bilimleri üzerine eğitim veren üniversite bölümlerinin çoğunda öğrencilerin programlama ile ilk kez tanışacağı derslerin nasıl verileceği sorununa cevap aranmalıdır (Allison vd., 2002).

Naps vd. (2003), programlama dersinde öğrenme ortamının düzenlenmesinin, nasıl yapılması gerektiği ile ilgili yaptığı araştırmada, öğrencilerin program görselleştirme araçlarını kullanarak verilen programlama örneklerini çözüme ulaştırabilmelerinin ve kendilerine özgü tasarımlarını da yapabilmelerinin önemli olduğunu belirtmektedir. Naps vd. (2003)'e göre programlama öğretimi; 1) Etkin öğretici materyallerin kullanımı, 2) Etkileşimli görselleştirme araçlarının kullanımı, 3) Öğretmen rehberliğinde laboratuvar çalışmalarının yapılması, 4) Kendi hızlarında öğrencilerin internet ortamında ilerleyebilmeleri, 5) Geçtiğimiz dört maddenin beraberce veya birbiri takip eder şekilde kullanılması gerektiğini belirttiği beş yöntemle yapılabileceğini belirtmektedir. Programlama öğretiminin bilinen ve en sık kullanılan yöntemlerinden biri de, konuların önce verilmesi ve sonrasında uygulamanın yapılmasıdır. Fakat yapılan araştırmalar bu yöntemin doğru olmadığını teori ve uygulamanın iç içe kullanılması gerektiğini göstermektedir (Crews ve Murphy, 2004; Ziegler ve Crews'in (Hu, 2004'de belirttiği üzere).

Günlük hayattaki problemlere bilgisayarlarla çözüm üretilirken ihtiyaç duyulan programlar, öğrenilmesi ve öğretilmesi zor olan yapılardır. Programlama ile ilk karşılaşanların programlamanın soyut, karmaşık ve zor olan yapısıyla başa çıkmaları gerekmektedir. Programlama öğretimi gerçekleştirenlerin ise bu zorlukları aşabilecek yeni programlama öğretimi stratejileri geliştirmeleri gerekmektedir. Bu stratejilerden bir taneside eğitsel programlama dillerinin programlama öğretiminde kullanılmasıdır. Eğitsel programlama dillerinin programlama öğretiminde kullanımı ile ilgili olarak programlama öğretimi gerçekleştirenlerin istekli oldukları görülmektedir. Fakat eğitsel programlama dillerini derslerinde nasıl kullanacaklarına yönelik olarak problemler yaşadıkları görülmektedir.

1.2. Çalışmanın Amacı

Bu çalışmanın amacı programlama öğretiminde eğitsel programlama dilinin farklı kullanımlarının programlama başarısı ve kaygısına etkisini araştırmaktır.

Bu amaca bağlı olarak şu araştırma sorularına cevap aranmaktadır;

1. Öğrenci başarısına yönelik olarak;
    a. Eğitsel programlama dilinin derse entegrasyon şekli, öğrencilerin ön-test son-test başarıları arasında anlamlı bir fark oluşturmakta mıdır?
    b. Eğitsel programlama dilinin derse entegrasyon şekli, öğrencilerin ön bilgileri kontrol altına alındığında son-test bilişsel beceri puanları arasında anlamlı bir fark oluşturmakta mıdır?
    c. Eğitsel programlama dilinin derse entegrasyon şekli, öğrencilerin ön bilgileri kontrol altına alındığında son-test alt düzey bilişsel beceri puanları arasında anlamlı bir fark oluşturmakta mıdır?
    d. Eğitsel programlama dilinin derse entegrasyon şekli, öğrencilerin ön bilgileri kontrol altına alındığında son-test üst düzey bilişsel beceri puanları arasında anlamlı bir fark oluşturmakta mıdır?
2. Ders içi performansa yönelik olarak;
    a. Eğitsel programlama dilinin derse entegrasyon şekli, ders içi performans testlerinin bilişsel beceri puanları arasında anlamlı bir fark oluşturmakta mıdır?
3. Bilgisayar programlama kaygı düzeylerine yönelik;
    a. Eğitsel programlama dilinin derse entegrasyon şekli, öğrencilerin bilgisayar programlamaya yönelik kaygıları arasında anlamlı bir fark oluşturmakta mıdır?
    b. Eğitsel programlama dilinin derse entegrasyon şekli, öğrencilerin bilgisayar programlamaya yönelik ön kaygıları kontrol altına alındığında, son-test kaygı düzeyleri arasında anlamlı bir fark oluşturmakta mıdır?

1.3. Araştırmanın Önemi

Dünyada ve ülkemizde eğitim teknolojileri ile ilgili yapılan araştırmalar ve tezlerdeki eğilimler incelendiğinde genellikle bilgisayar destekli öğretim, öğretim tasarımı, medya araştırmaları ve medya karşılaştırmaları çalışmalarının ağırlıklı olduğu görülmektedir (Klein, 1997; Caffarella, 1999; Ma, 2000; Masood, 2004; Hew, Kale ve Kim, 2004; Latchem, 2006; Hranstinski ve Keller, 2007; Costa, 2007; Mihalca ve Miclea, 2007; Ross, Morrison ve Lowther, 2010). Dünya genelinde eğitim teknolojileri araştırmalarının eğilimleri incelendiğinde, eğitim teknolojileri, bilişim teknolojileri, bilgisayar destekli öğretim, öğretim tasarımı ve medya araştırmaları gibi birçok eğilimin olduğu ve genellikle araştırmalarda deneysel çalışmaların tercih edildiği görülmektedir (Klein, 1997; Caffarella, 1999; Masood, 2004; Hew vd., 2007; Costa, 2007; Hranstinski ve Keller, 2007; Ross vd., 2010).

Türkiye'de ise eğitim teknolojileri araştırmalarındaki eğilimlerin belirlenmesine yönelik yapılan çalışmalar arasında; Şimşek vd. (2008) 64 tezin incelendiği çalışmalarında, çevrimiçi öğrenme, öğrenme-öğretme yaklaşımları ve çoklu ortam üzerinde yoğunlaştıklarını ve yöntem olarak ise genellikle nicel yöntemlerin tercih edildiğini; Alper ve Gülbahar (2009), yaptıkları araştırmada özellikle online eğitimin, e-öğrenme ve uzaktan öğrenmenin en çok çalışılan konulardan olduğunu ve yöntem olarak ise nicel yöntemlerin tercih edildiğini; Erdoğmuş ve Çağıltay (2009)'ın çalışmalarında ise, öğretim teknolojileri alanında en yaygın olan üç konunun öğrenci değişkenleri, medya ve medya karşılaştırmaları olduğunu belirtmişlerdir. Bu çalışma, araştırma yöntemi ve uygun medya ile metodun birlikte kullanılması ile önemlidir.

Öğretim teknolojileri alanının asıl amacı uygun medya ve yöntemleri bir arada kullanarak öğrenmeyi daha iyi hale getirmektir. Medya ve yöntem karşılaştırmalarına örnek verilebilecek en önemli çalışmalardan biri Clark ve Kozma'nın medya yöntem tartışmasıdır. Bu tartışmada Clark (1983), medyanın öğrenmeye etkisi olmadığını, Kozma (1994) ise medya veya gelişen teknolojilerin öğrenmeye etkisinin olabileceğini savunmaktadır. Bu tartışmanın bir boyutu da medya ve yöntem kendi alternatifleri ile yer değiştirebileceğidir. Clark (1994a) çalışmasında medyaların alternatifleri ile değiştirilebileceği söylemini, bir benzetme ile pekiştirmeye çalışmıştır. Clark'a göre medya bir ilaç gibidir. Bu ilacın fitil, şurup, hap ve enjeksiyon gibi farklı formları

olabilir. Kozma (1994a), medya özelliklerinin göz önüne alınarak birbiri ile değiştirilebileceğini fakat medya ile yöntemin birbirinden ayrıştırılmasının söz konusu olamayacağını savunmaktadır. Morrison (1994), Kozma ve Clark'ın fikirlerini irdelemiş ve sonuç olarak medya ve yöntemin birbirinden ayrılamayacak kadar iç içe olduklarının altını çizmiştir. Bir diğer araştırmacı Reiser (1994), tartışmaya öğretim tasarımı boyutuyla baktığının altını çizerek, Clark'ın ilaç benzetmesine farklı bir bakış geliştirmiştir. Reiser'e göre, menenjit gibi hastalıklarda, antibiyotiklerin damar içerisine enjekte edilmesi gerektiğini, antibiyotiğin diğer formlarda verilmesinin işe yaramayacağını örnek vererek, bazı durumlar içinde uygun olan medya kullanılmadığı zaman yöntemin etkisiz kalacağını belirtmiştir. Dolayısıyla medyanın alternatifleri ile değiştirilebilirliğinin genelleştirilemeyeceği ve yöntemle beraberce ele alınması gerektiği ve medyanın yönteme, yönteminde medyaya bağlı olabileceğini ileri sürmektedir. Bu çalışma program görselleştirme araçlarının derse entegrasyonunun Teori Entegre Grup (TEG), Uygulama Entegre Grup (UEG) ve Teori ve Uygulama Entegre Grup (TUEG) olarak üç yöntem ile gerçekleştirilmesi ve bu yöntemlerin medyanın kullanım şekline göre belirlenmesi (Reiser'in ilaç benzetmesi) medya ve yöntemin birbirine ne kadar bağlı olduğunun Yazıcı ve Kültür (2013) bir kanıtı olması yönüyle önemlidir.

Programlama öğretilmesi ve öğrenilmesi noktasında algoritma aşamasından program yazımına kadar olan çözümün tasarlanabilmesi, programlama becerisinin anlaşılması ve farklı yollarla kazandırılması sürecinde engel teşkil etmektedir (Moser, 1997; Pillay, 2003; Pillay ve Jugoo, 2005). Bu engelin üstesinden gelebilmek için farklı yolların ve yöntemlerin kullanıldığı birçok çalışma gerçekleştirilmiştir (Cañas vd., 1994; Wiedenbeck vd., 1999; Robins vd., 2003; Wiedenbeck vd., 2004; Bennedsen ve Caspersen, 2007; Esteves vd., 2011; Yadin, 2011;Cutts vd., 2014; Krpan vd., 2015; Michaelson, 2015; Bau vd., 2015). Bu çalışmalar sonucunda kullanım kolaylığı olan, görselleştirme olarak zengin ve etkileşimli birçok öğretim ortamı incelenmiştir. Alanyazında programlama ve algoritma mantığının kavratılmasını sağlayan farklı görselleştirme uygulamaları bulunmaktadır (Carlisle vd., 2005; Chen ve Morris, 2005; Hooshyar vd., 2015). Bu uygulamalardan bir tanesi de algoritmik düşünme mantığının öğretilmesi ve programlama temellerinin kavratılmasında kullanılan Massachusetts Teknolojisi Enstitüsün (MIT) de geliştirilen Scratch, blok-temelli görsel programlama

dili ortamıdır (Maloney vd., 2010; Resnick vd., 2009; Armoni vd., 2015). Yapılan araştırmada blok-temelli görsel programlama dilinin algoritma öğretiminden programlama yazımına geçişte kullanılmış olması ve geçişi kolaylaştırması yönüyle önemlidir.

Alanyazındaki farklı özellikleri olan programlama temellerini ve algoritma mantığını öğretmeye çalışan birçok araç bulunmaktadır. Programlamanın öğretilmesi noktasında öğretmenlerin bu araçları öğretim ortamlarına entegre etmede, teknolojiye yönelik inanç, tutum ve direnç gibi iç engellerle karşılaşmaktadırlar (Ertmer ve Hruskocy, 1999). Öğretmenlerin dersleri ile bu eğitim teknolojilerini entegre etme konusunda problemlerle karşılaştıklarını gösteren birçok çalışma alanyazında bulunmaktadır (Becker, 2001; Ertmer, 2005; Karal ve Berigel, 2006; Yükseltürk ve Top, 2013). Öğretmenlerin bu teknolojileri eğitim ortamlarına entegre etmek istedikleri fakat bu entegrasyonu nasıl gerçekleştireceklerini bilmediklerini görmekteyiz (Eby, 2001; Ertmer, vd., 2003, Göktaş, Yıldırım ve Yıldırım, 2008). Bu çalışmanın, görsel programlama dili için seçilen bir aracın, program öğretimi sürecinde derslere nasıl entegre edileceği konusunda rehberlik edeceği düşünülmektedir.

Programlama eğitiminin öneminin gün geçtikte arttığı çağımızda başta Tayvan ve Kore olmak üzere birçok ülkenin programlama derslerini ilkokul müfredatına dahil ettikleri görülmektedir (Lin ve Zhang, 2003). Türkiye'de bu konu ile ilgili Bilim, Sanayi ve Teknoloji Bakanlığının girişimlerinin olduğu görülmektedir. Bakanlık tarafından yapılan açıklamada, Milli Eğitim Bakanlığı ile çalışmalar yapıldığı ve programlama dersinin ilkokul seviyesinde öğretilmesi gerektiği belirtilmiştir (Işık, 2015). Bilim, Sanayi ve Teknoloji Bakanlığın yaptığı bu açıklamanın ülkemizde programlama öğretiminin geleceği için ümit verici bir adım olarak görülmektedir. Programlama öğretiminin öneminin anlaşılması, bu konuda yapılan çalışmanın önemini artırmaktadır.

Araştırma sonuçlarının uygulayıcılara ve araştırmacılara programlama öğretiminde eğitsel programlama dillerinin derse nasıl entegre edileceği konusunda yol göstereceği düşünülmektedir.

1.4. Varsayımlar

1) Katılımcıların tamamı çalışmaya istekli ve gönüllüdür.
2) Denetim altına alınamayan değişiklikler, her üç grubu da aynı şekilde etkilemiştir.
3) Araştırmaya katılan öğrenciler, veri toplama araçlarını samimiyetle yanıtlamışlardır.

1.5. Sınırlılıklar

1) Programlama dersi "temel kavramlar, değişken yaklaşımı, giriş çıkış deyimleri, döngü yapısı, matematiksel ve mantıksal operatörler ve koşul yapısı" konuları ile sınırlıdır.
2) Çalışma görsel programlama dillerinden blok-temelli görsel programlama dili (Scratch) ile sınırlıdır.
3) Yapılan çalışmanın süresi altı hafta ile sınırlıdır.
4) Grupların rastgele örneklem yöntemi ile oluşturulmasında kız öğrencilerin gruplardaki dağılımı ile sınırlıdır.
5) Çalışma üç deney grubu ile sınırlıdır.
6) Eğitsel programlama dili ortamında değerlendirme yapılmamış olması ile sınırlıdır.
7) Araştırmacının üç grup içinde dersi anlatan kişi olması yönüyle sınırlıdır.

İKİNCİ BÖLÜM

2. KURAMSAL ÇERCEVE VE İLGİLİ ARAŞTIRMALAR

2.1. Kuramsal Çerçeve

2.1.1. Programlama nedir?

Bilgisayarlar gerçekleştirecekleri işlemleri anlayabilecekleri komut dizileri halinde getirirler. Bilgisayarda gerçekleştirilecek bir işlem veya bir problem çözümü için gerekli olan bilgiler, bilgisayar dilinde komutlara çevrilir. Bu komutların bir araya getirilip derlenmesi ve çalıştırılması sonucunda meydana gelen işlem algoritmasına, programlama denir (Kesici ve Kocabaş, 2007).

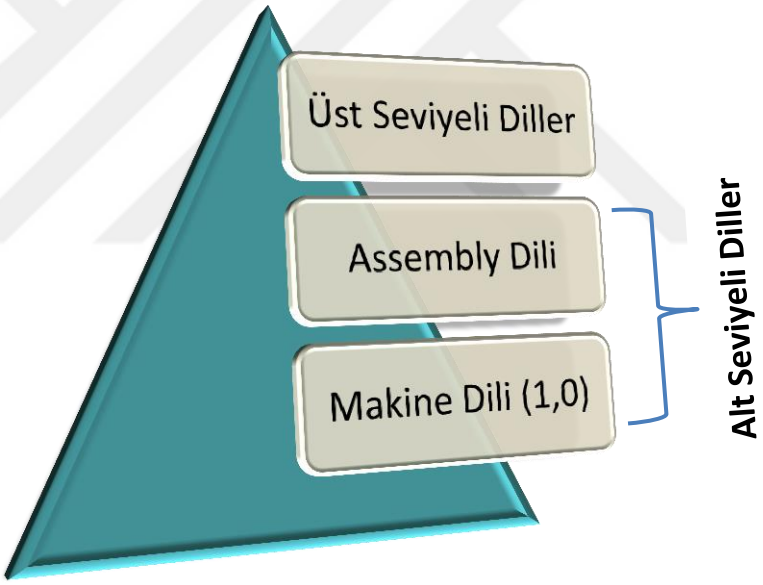
Aslında programlama gerçek yaşamdaki olayların bilgisayarda modellenmesidir. Bilgisayar programlama ise bu modellerin bilgisayarın anlayabileceği şekle sokulması işlemidir. Programcılar genellikle soyut yapıda olan programlamayı analojiler ve metaforlarla gerçek hayata yaklaştırarak öğrencilerin anlamasına yardımcı olurlar. Dolayısıyla program yaparken öncelikle problemi analiz etmek ve komutları düşünmeden probleme, gerçek hayatta çözüm üretmek ve sonrasında sadece araç olarak düşünülmesi gereken komutları kullanarak programlama yapısını oluşturmak gerekmektedir (Çölkesen, 2002).

Kesici ve Kocabaş (2007)'a göre günlük hayatta karşılaşılan bir problemin, bilgisayarda nasıl çözüme ulaştırılacağını belirleyen ve bilgisayarın işlemesini o yönde dizayn eden komut dizinine bilgisayar programı denir. Bu programların bilgisayarlar tarafından kullanılması ise yazılım (software) olarak adlandırılmaktadır.

Çölkesen (2002), program ve yazılım ayrımını şu şekilde yapmaktadır. Programlar bir görevi yerine getirmek için kendi içinde bir bütünlük arz eden algoritmik yapılardır. Yazılım ise bu programların veya program parçacıklarının bir araya getirilmesi ile oluşan ve günlük hayattaki problemlere çözüm üreten bütünlüklerdir. İlk zamanlardaki (1955'li yıllarda) programlamada iletken fişler yardımı ile bilgisayarlar

programlanırken, İngilizce olarak programlama ile ilgili kelimeler kısaltılarak komutlar oluşturulmuş ve bu komutların yardımı ile ilk olarak yüksek seviyeli diller kullanılmıştır.

Bu dillerin kullanılması ve programlamanın bu denli kolay olması birçok kişinin programlamaya ilgi duymasına yol açmış ve programcı sayısında ciddi bir artış gözlenmiştir. İlginin yoğun olması, beraberinde birçok komut yapısının ve kod yazım editörünün de oluşturulmasını sağlamıştır. Bundan dolayı birçok bilgisayar programlama dili ortaya çıkmıştır. Ortaya çıkan bilgisayar programlama dillerinin çokluğu karışıklığa yol açmıştır. Bu karmakşılığı çözmek için, bilgisayar programlama dilleri yapısına göre alt seviyeli diller ve üst seviyeli diller olmak üzere iki şekilde sınıflandırılmıştır (Kesici ve Kocabaş, 2007).



Şekil 2.1. Programlama dillerinin seviye bakımından birbirleri ile olan durumları

1. Alt Seviyeli Programlama Dilleri: Kodlanmasında ikili sayı sisteminin kullanıldığı bu diller, işlemcilerine göre, Makine dili (1,0) ve Assembly dili olarak isimlendirilmektedir.

2. Üst Seviyeli Programlama Dilleri: Program yazımında günlük hayattaki konuşmalardan esinlenilerek belirlenen komutlar kullanılarak yazımı gerçekleştirilen dillerdir. Bu dillere örnek olarak Basic, Fortran, Cobol, Pascal, Delphi, Java, Visual C,

C#, VisualBasic, VB.NET, Acces, Foxpro, Python gösterilebilir. Bu dillerin seviye bakımından birbirleri ile olan durumları Şekil 2.1'de görülmektedir.

Programlama dilleri seviye olarak birbirinden farklı olsa da sonuçta bilgisayarın anlayabileceği (1,0) şekle dönüştürülmeleri gerekmektedir. Herhangi bir programlama dili kullanılarak yazılan programların kaynak kodları mevcuttur. Özellikle yüksek seviyeli dillerin bilgisayarın anlayabileceği yapıya (1,0) dönüştürülmesi gerekmektedir. Bu dönüşüm program derleyici tarafından gerçekleştirilmektedir. Yazılan program satır, satır ilerletilerek derlenir, derleme sırasında programda hata varsa hata yapılan yerde programı, derleyiciler durdurarak kullanıcılara hatalarını düzeltmede fırsat verirler. Birçok yüksek seviyeli programlama dilinin derleyicisi mevcuttur (Kesici ve Kocabaş, 2007).

2.1.2. Programlama öğretimi

Bu kadar karmaşık ve soyut yapıya sahip olan bilgisayar programlarının öğreniminde ve öğretiminde zorluklar yaşanması çok doğaldır. Garner (2003) yaptığı çalışmada, programlama öğrenimi ve öğretiminin, geçen bunca yıllara rağmen çok fazla değişmediğini belirtmiştir. Ona göre öğretmenler, programlama yapılarını (değişkenler, operatörler, kontrol yapıları vb.) tahtada anlatırlar, öğrencilere birkaç örnek gösterirler ve sonucunda öğrencilerin programlama ile ilgili problemlere çözüm üretmelerini beklerler. Öğrencilerin ise öğretmenin anlatımı sırasında anladıklarını, fakat sonrasında kendilerinin problemlere çözüm üretemedikleri belirtmektedirler.

Programlama öğretimi gerçekleştirilirken, öğretmenler öğrencilerine programlama dersinde, günlük hayatta karşılaşılan bir probleme nasıl çözüm üretmeleri gerektiğini anlatırken, bilindik genel anlatımı değiştirerek, dersin planını oluşturmalıdırlar. Ders planı oluşturulurken, bilgisayar programlama hazırlık aşamaları göz önüne alınmalıdır. Bilgisayar programlama derslerinde, bir bilgisayar programının hazırlanması, beş temel aşamadan meydana gelir: Problemin Tanımı, Çözüm Yolunun Belirlenmesi (Programın Algoritmasının Hazırlanması), Programın Kodlanması, Programın Kodlarının Yorumlanması ve Derlenmesi, Programdaki Kod Hataların Belirlenmesi ve Giderilmesi (Kesici ve Kocabaş, 2007).

1. Problemin Tanımı: Probleme doğru çözüm üretebilmek için çok iyi tanımlamak ve anlamak gerekir. Problem hakkında yeterince bilgi sahibi değilseniz o konuda derinlemesine bir bilgi edinilmelidir. İyi tanımlanmış ve anlaşılmış bir problemin %50 'si çözülmüş demektir.

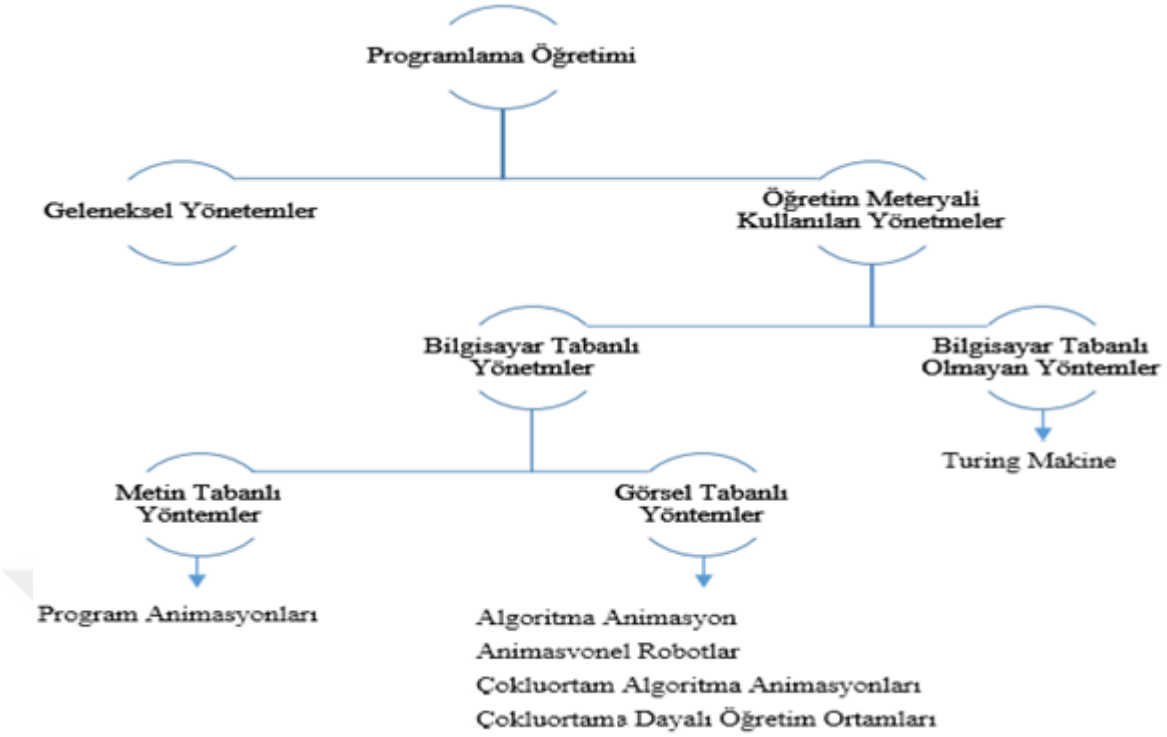
2. Çözüm Yolunun Belirlenmesi (Algoritmanın Hazırlanması): Problemin tanımlanması sorunsuz gerçekleştirilmiş ise çözümü için yapılacak işlemler, basamak basamak maddelendirilir. Algoritması oluşturulduktan sonra, sembollerle mantıklı bir şekilde, ayrıntılı olarak akış şeması oluşturulur. Çözüm için kullanılacak yöntemler kullanıcıya özgü olabilir.

3. Programın Kodlanması: Oluşturulan algoritma ve algoritmanın akış şeması sonrası, kullanılacak programlama dilinin sözdizimi kurallarına göre program yazılır.

4. Programın Yorumlanması ve Derlenmesi: İlgili programlama dilinde kodlanması gerçekleştirilen programın, derlenerek çalıştırılmaya hazır hale getirilmesidir.

5. Programdaki Hataların Belirlenmesi ve Giderilmesi: Bu aşamada derlemesi yapılmış program üzerindeki yazım ve mantık hataları belirlenerek giderilir. Yazım ve mantık hatalarının programlamadan anlayan, dışardan bir göz tarafından değerlendirilmesi, hataların görülmesini kolaylaştıracaktır. Bu aşamadan sonra program tekrar derlenerek çalıştırılır.

Bilgisayar programlama hazırlık aşamalarına göre programların hazırlanması, geleneksel programlama derslerine oranla daha iyi bir programlama öğretim ortamı sağlamaktadır. Programlama gibi soyut olan bir yapının oluşturulan bu öğretim ortamlarında daha iyi kavratılabilmesi ve somutlaştırılabilmesi için, birçok programlama dili öğretim materyali kullanılmaktadır. Gültekin (2006)'in yaptığı çalışmada bu öğretim materyallerinin sınıflandırılmasından ve alanyazındaki kullanımlarından bahsetmiş ve üst düzey bilişsel beceriler gerektiren programlama dillerinin öğretimi ve öğreniminin soyut yapısı, teknik kavramları ve bilişsel yüke etkisi yönüyle zor bir süreç olduğunu belirtmiştir. Programlama dillerinin zorluğu, aynı zamanda öğrenme içeriğinin geniş bir çerçevede Şekil 2.2'de görüldüğü gibi ele alınmasını gerekli kılar.



Şekil 2.2. Programlama dili öğretim materyallerinin sınıflandırılması (Kaynak: Gültekin (2006)'den uyarlanmıştır)

Geleneksel öğretim yöntemlerinin olumsuz yönlerini aşabilmek ve daha etkili bir bilgisayar programlama öğretimi gerçekleştirebilmek için bilgisayar programlama öğretimine yönelik çok çeşitli materyaller geliştirilmiştir. Alanyazına bakıldığında programlama öğretiminde bu öğretim materyallerinden en çok çokluortama dayalı öğretim ortamları üzerine yapılan çalışmalara rastlanmaktadır. Bundan dolayı bu ortamlara yönelik birçok araç geliştirilmiştir. Bu araçlardan bilgisayar programlama öğretiminde en sık kullanılanlarının algoritma ve programlama görselleştirme araçlar olduğu görülmektedir (Gültekin,2006). Bu çalışmaya paralel olan başka bir çalışmada çokluortama dayalı öğretim yazılımlarının programlama öğreniminde öğrencilerin akademik başarılarını nasıl etkilediği araştırılmıştır. Araştırmada ön-test son-test kontrol gruplu bir araştırma deseni kullanılmıştır. Araştırma süreci sonunda son-test uygulamasının üzerinden 20 günlük bir süre geçince kalıcılık test uygulanmıştır. Araştırma sonucunda çokluortama dayalı öğretim yazılımlarının geleneksel öğretime kıyasla öğrencilerin akademik başarıları üzerinde anlamlı bir fark oluşturmuştur. Araştırmanın diğer bir bulgusu ise acemi olan öğrencilerin, önceden programlama

eğitimi almış bireylerle ön-test verileri arasında anlamlı bir fark oluşurken son-test verileri arasında anlamlı bir fark oluşmamıştır (Yurdagül ve Gültekin, 2009).

2.1.3. Görselleştirme (araçları) ve görselleştirme özellikleri

Program görselleştirme araçları çeşitli metaforlar kullanarak programlama yapmaya olanak sağlayan veri yapılarının ve gerçek kodların görselleştirildiği yapılardır (Baecker ve Price, 1998). Bu araçlara Balsa, Jhave, Jeliot, Mindstorms, Small Basic, Alice ve Scratch örnek olarak verilebilir. Olsen, Pedersen, Harnes, ve Tosse (1990)' e göre programlamada kullanılan görsel nesneler; modüller, komutlar, formlar, programlar ve alt programlar olarak tanımlanmaktadırlar. Hirakawa, Tanaka ve Ichikawa (1990) ve Baldwin ve Kuljis (2000)'a göre ise bu nesneler, veri yapılarının ve algoritmaların simgeler halinde görselleştirilmesi ve buna benzer akış grafikleri olarak isimlendirilmesidir.

Bergin ve Martinez (1996)'e göre program görselleştirme araçlarının, öğretmenlere yeni öğretim yöntemi sunması, öğrencilerin geleneksel ders ortamından daha çok ilgisini çekmesi ve programlama öğretiminde bilgisayar öğretmenlerine daha kolay örnek oluşturma imkânı sunması yönüyle hem öğretmenlerin hem de öğrencilerin ilgisini çekmektedir. Bu açıdan aşağıda sunulan önermelerin önemli olduğu (Bergin ve Martinez, 1996)düşünülmektedir. Bunlar;

- Programlamanın karmaşık yapısının anlamlandırılması ve sadeleştirilmesi için resimler ve görsel materyaller kullanılmalıdır.
- Görselleştirme araçları öğretmenlere daha kısa zamanda etkili ve içerikli materyaller geliştirme imkânı sunmalıdır.
- Program görselleştirme araçları ile iyi tasarlanmış benzetimler öğrencilerin anlama ve buna bağlı olarak başarı düzeylerini yükseltir.

Konuların içeriğine bağlı olarak, kullanılacak olan görselleştirmenin tipide resim ve animasyon gibi farklılık göstermektedir. Animasyon kullanılması için, sunulacak konunun içeriğinin oldukça büyük olması, hareketlendirme gerektirmesi ve kavramların birbiri ile olan ilişkilerinin sunulması gerekmektedir. Görselleştirme araçları laboratuvar uygulamaları, sınıf sunuları ve öğrenci ödevlerinde kullanılabilir. Görselleştirme aracının asıl gücü soyut kavramları somutlaştırmasıdır. Bu nedenle bu araçlar

kullanıcılara konu hakkında bir tartışma zemini sağlamalıdır. Öğrencilere bu tartışma ortamları farklı bakış açıları kazandırmalı ve soyut olan programlama kavramlarını anlamlandırabilmelerine olanak sağlamalıdır (Bergin ve Martinez, 1996).

Asıl gücünü soyut kavramları somutlaştırmadan alan görselleştirme araçları yazılımın daha iyi anlaşılmasında, karmaşık olan yazılımların çözümlenmesinde, yazılım geliştirme hızının artırılmasında, yazılım hatalarının bulunmasında ve yazılım kalitesinin artırılmasında kullanılabileceğinden dolayı yazılım görselleştirmenin yararlı olabileceğini rapor edilmektedir (Bassil ve Keller, 2001).

2.1.4. Yazılım görselleştirme

Yazılım ile ilgili sorunların anlaşılması ve çözümlenmesi için, yazılım görselleştirme (YG), ortaya konulan çözüm önerilerinden biridir. Yazılımı anlamadaki zorlukların üstesinden gelmek için birçok yaklaşım önerilmiştir. Bu yaklaşımlardan en çok ilgi göreni, gelecekteki yazılımları da anlamada önemli bir araç olabileceği ileri sürülen, yazılım görselleştirme teknikleridir (Lemieux ve Salois, 2006).

Yazılım görselliğinin programlama dillerinin öğretimindeki önemi, yazılım görselleştirme ile kendisini göstermektedir. Yazılım görselleştirme ile ilgili literatür, bilgisayar yazılımlarının anlaşılabilmesi ve daha iyi kullanılabilmesi için insan ve makine ilişkisinin grafik ara yüzleri, animasyon ortamları ve görsel sanatlar yardımı ile gerçekleştirilebilmesi, yazılım görselleştirme olarak tanımlanmaktadır (Price, Baecker ve Small, 1993). Yazılım görselleştirme, Bilgisayar Grafikleri, Yazılım Mühendisliği, Bilgi Görselleştirme, Programlama Dilleri, Veri Madenciliği, İnsan Bilgisayar İlişkileri ve Bilgi Görselleştirme gibi birçok bilim dalını içeren geniş bir alanda kullanılan teknikleri birleştirmektedir (Diehl, 2005).

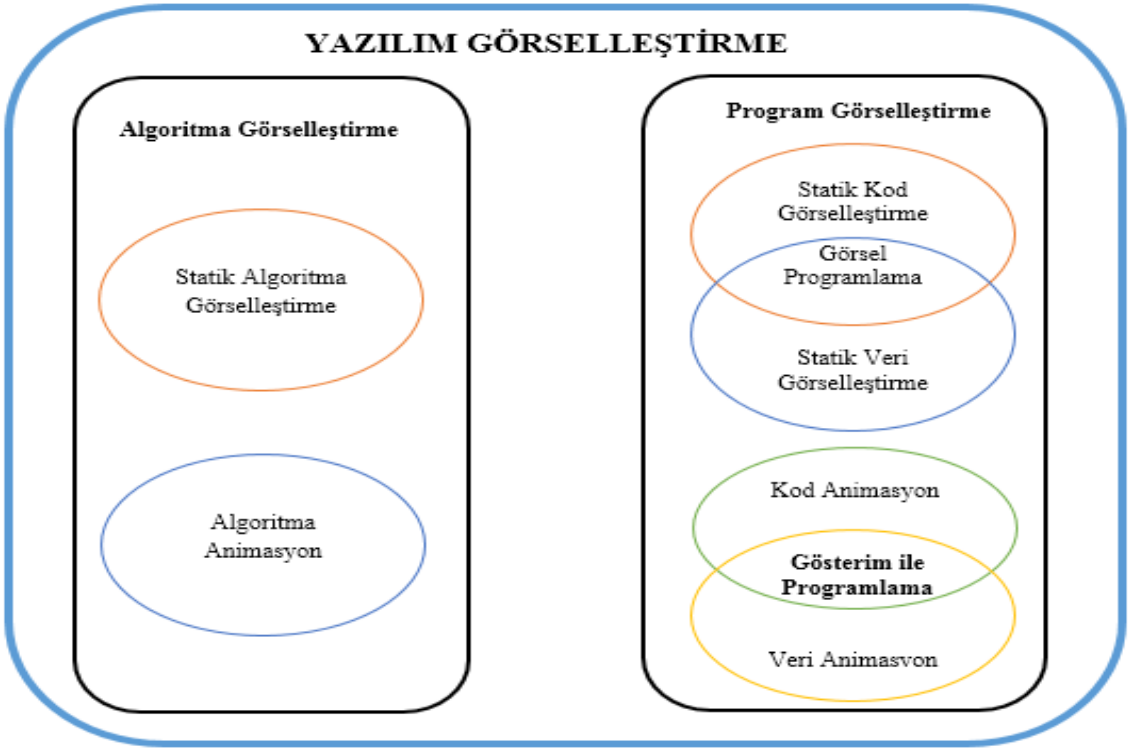
Bu bilim dallarındaki tekniklerin bir birleşimi olan yazılım görselleştirme, programlama öğretiminde öğrencilerin öğrenmesi önündeki zorlukları aşmak için kullanılan en önemli araçlardan biridir (Stasko, Kehoe ve Taylor, 2001; Almeida-Martinez, Urquiza-Fuentes ve Valezquez-Iturbide, 2009). Yazılım görselleştirme iki alt yapının bir araya gelmesiyle oluşmaktadır. Bunlar algoritma görselleştirme ve program görselleştirmedir. Algoritma görselleştirmede yazılım yapısı ile ilgili yüksek seviyeli

soyutlamalar görselleştirilirken, Program görselleştirmede sözdiziminde kullanılan veri yapıları ve program kodları görselleştirilir (Baecker ve Price, 1998).

Yazılım görselleştirme ile ilgili alanyazında yapılmış olan ilk çalışmalardan biri, Eick, Steffen ve Sumner, (1992)'in yaptığı çalışmadır. Bu çalışmada araştırmacılar, kod bloklarını renkli çizgilerle sembolize etmişlerdir. Bu sayede yazılımın soyut olan genel şeklinin anlaşılması ve programlama yapanların bilişsel düzeyine katkı sağlanması amaçlanmıştır. Laakso, Rajala, Kaila ve Salakoski (2008), geleneksel kuramsal yöntemler ile program görselleştirme araçlarının kullanıldığı öğretim yöntemlerini karşılaştırdıkları çalışmalarında, program görselleştirme yöntemlerinin öğrencilere programlama kavramlarını daha iyi öğrettiğini, öğrencilerin başarılarını olumlu yönde etkilediğini ve programlamaya yeni başlayan öğrenciler üzerinde daha etkili olduğunu belirtmişlerdir.

Moreno, Myller, Sutinen ve Ben-Ari (2004)'nin yaptıkları araştırma, görsel programlama dili aracı kullanan öğrencilerin, kullanmayan öğrencilere göre daha başarılı olduklarını belirlemişlerdir. Öğrenciler görsel programlama dili aracı ile ilgili olarak döngü, koşul ifadeleri ve nesneleri anlamayı basitleştirdiği, fakat yavaş olan animasyon ilerleme drumunun programlama dersleri için uygun olmadığını belirtmişlerdir.

Bilgisayar programlama öğretimi, yazılım görselleştirmenin kullanıldığı önemli bir alandır (Nevalainen ve Sajaniemi, 2006). Araştırmacılar, iyi tasarlanmış bir yazılım görselleştirmenin programlama öğretiminde yararlı olacağını düşünmektedirler. Baecker ve Price (1998) yaptıkları çalışmada görselleştirme araçlarından hangisinin daha faydalı olacağını sorgulamakta ve bu sorgulama sonucunda programlama için görselleştirme önerilerinde bulunmaktadırlar. Lemieux ve Salois (2006), yaklaşık 140 çalışmadan oluşan bir alanyazın araştırması gerçekleştirmişler ve araştırma sonucunda yazılım görselleştirme konusunun üzerinde çalışılabilecek aktif bir alan olduğu sonucuna ulaşmışlardır. Price, Baecker vd. (1998) yazılım görselleştirme yöntemleri ve bu yöntemlerin birbiri ile ilişkisini kuramsal bir çerçevede Şekil 2.3. 'deki gibi tasarlamışlardır.



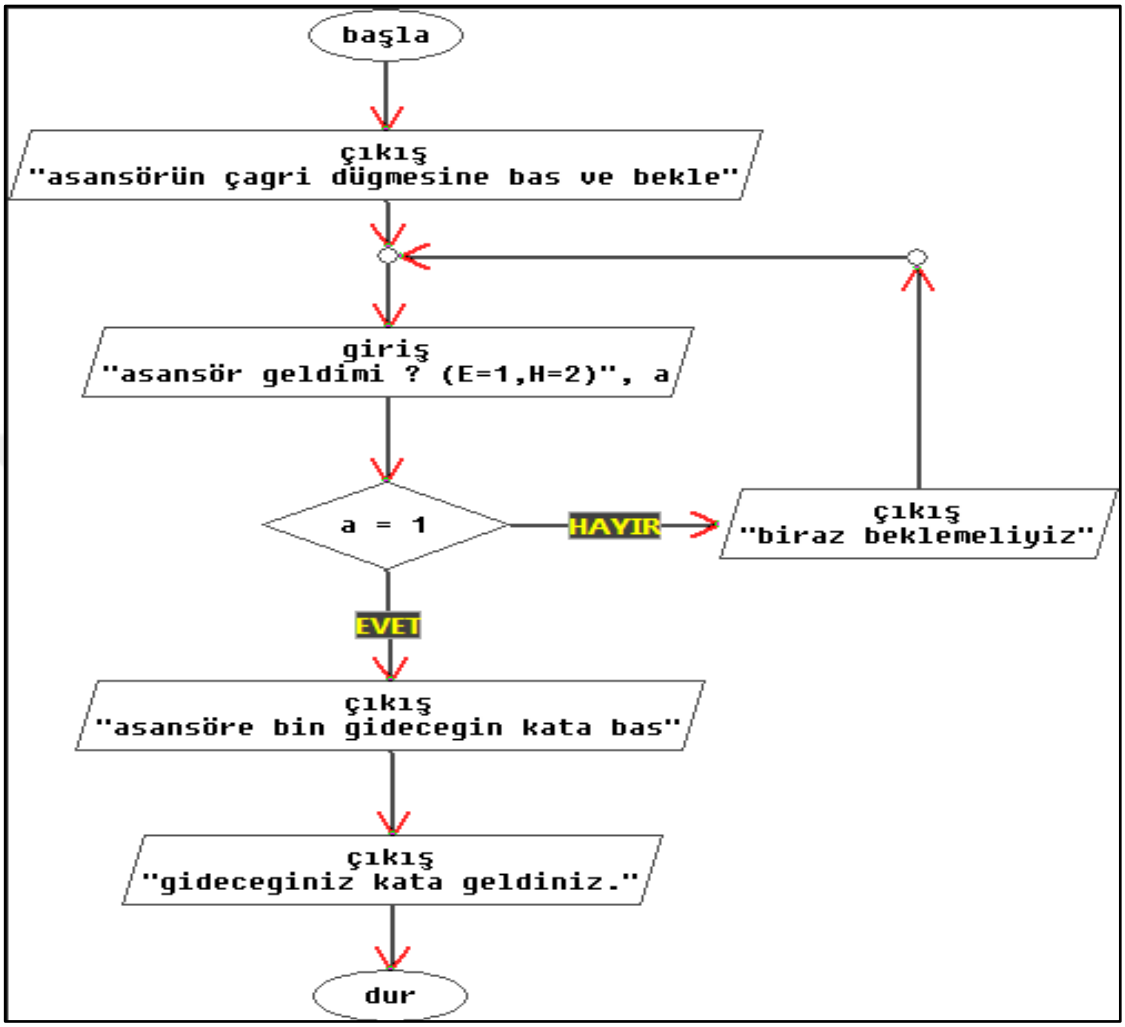
Şekil 2.3. Yazılım görselleştirme yöntemleri ve birbirleri ile ilişkisi (Kaynak: Price, Baecker vd. (1993)' den uyarlanmıştır).

Bu çalışma sonucunda yapılan yazılım görselleştirme tekniklerinin tanımlanmasında Algoritma Görselleştirme ve Program Görselleştirme diye iki temel alan üzerine odaklanılmıştır.

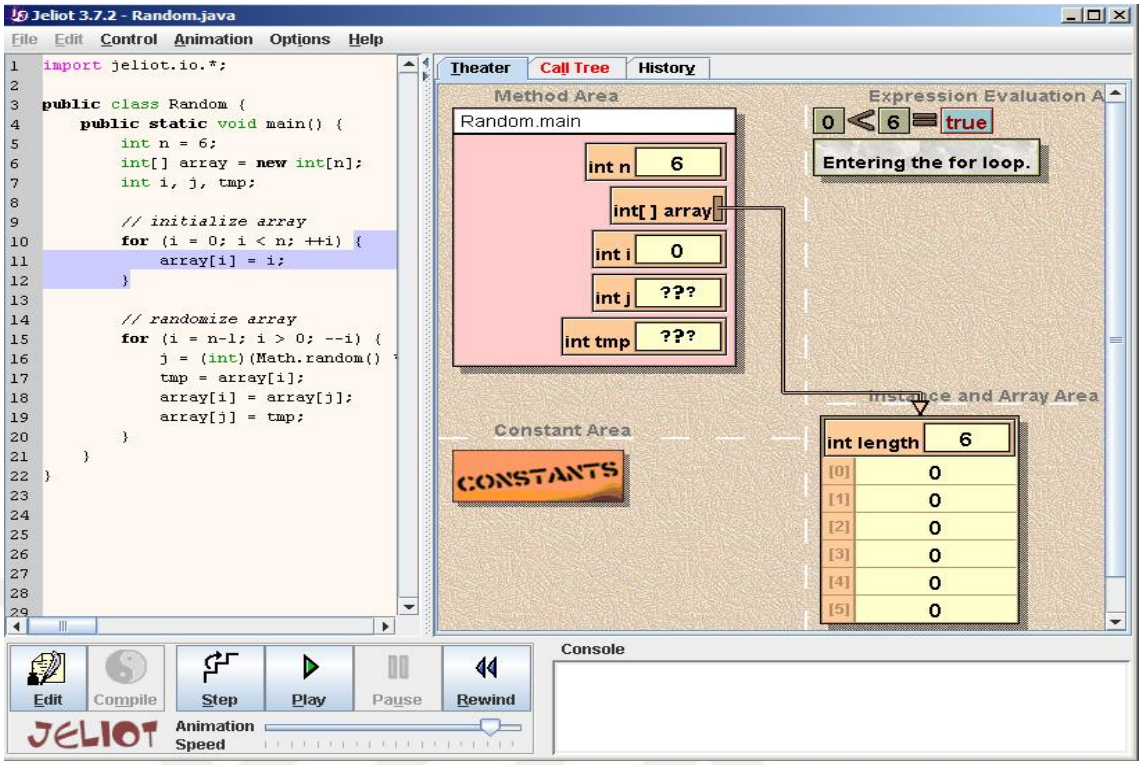
2.1.4.1. Algoritma görselleştirme

Bilgisayar programlamanın ilk ve en önemli adımı algoritma oluşturma sürecidir. Bu süreçte problemin çözümünde uygulanacak işlemler ve alınacak kararların sıralandığı akışın belirlenmesi gerçekleştirilmektedir (Aydın vd., 2004). Bu süreç sonunda oluşturulan algoritma yapılarının öğretiminde, algoritma görselleştirme araçları kullanılmaktadır. Algoritma görselleştirme araçları, yazılım görselleştirmeyi oluşturan iki araçtan biridir ve programlamanın çalışma şekli ile ilgilenen bir tekniktir. Bu teknik üst düzey programlama akışı ile ilgilenmekte olup, alt düzey programlama olan kod ve verilerle ilgilenmemektedir. Bu tekniğin iki yönü vardır, ilki yazılım akışının durağan yapısı ile ilgilenen durağan algoritma görselleştirme (Static algorithm visualization), ikinci yönü ise yazılım akışının dinamik yapısı ile ilgilenen ve bu dinamik yapının

oluşturulmasında animasyonların kullanıldığı algoritma animasyon (Algorithm animation) tekniğidir (Lemieux ve Salois, 2006).



Şekil 2.4. Static Algorithm Visualization



Şekil 2.5. Algorithm animation

2.1.4.2. Program görselleştirme

Program görselleştirme aşamaları Şekil 2.3'de görüldüğü gibi statik veri görselleştirme, statik kod görselleştirme, kod animasyon, veri animasyon, görsel programlama ve gösterim ile programlama aşamalarından oluşmaktadır.

Statik veri görselleştirmesi (Static Data Visualization), programlamada önemli bir yer edinen veri yapılarının gösteriminin kutular ve bu kutuların birbirine bağlanmasını sağlayan liste yapıları ile gösteriminin sağlanmasıdır. Yazılım görselleştirmenin program görselleştirme kısmında olan bu yöntemde, bilişsel aktiviteyi üst düzeyde tutmak için metaforlardan yararlanılmıştır.

Statik kod görselleştirmesi (Static Code Visualization), yazılım içerisindeki yapıların, komutların, kuralların ve kontrol akışının grafiksel bir şekilde gösterilmesidir. Bilişsel aktivitenin üst düzeyde tutulmasının amaçlandığı statik kod görselleştirme yönetminde de metaforlardan yararlanılmıştır. Bu metaforların en çok kullananlarından biri, şehirlerdeki binalar ve blokların kullanıldığı metafordur (Wettel ve Lanza, 2008).

Kod animasyonu (Code Animation), veri animasyonda olduğu gibi programın çalıştırılmasında, programlama kodlarının renklendirme gibi yöntemlerle dinamik bir şekle sokulmasıdır. Bu teknikte kod satırlarının programın akışına göre sırayla ilerlerken, farklı şekillerde renklendirilerek, kod satırlarını takip eden öğrencilerin yazılımın mantığını kavraması sağlanmaya çalışılmaktadır.

Veri Animasyonu (Data Animation), statik veri görselleştirme yöntemiyle sembolize edilen programlama yapılarında renklendirme ve hareketlendirme yapılarak dinamik bir şekle sokulması tekniğidir (Price, Baecker ve Small, 1993).

Görsel Programlama, programı anlamlandırmak amacı ile grafik ve animasyonların kullanıldığı bir tekniktir. Görsel programlama bileşenlerinin belirli bir yönteme göre birbirleriyle ilişkilendirilmesi sonucu ortaya çıkan program geliştirme yöntemi, görsel programlama olarak adlandırılır.

Gösterim ile programlama (Programming by demonstration), bu teknik yazılım görselleştirmenin, program görselleştirme ile olan kısmında bulunan ve programlama ile alakalı bir konudur. Bu programlama yöntemi, alt yapısı zayıf olan programcıların programlama yeteneklerini geliştirmek için örnek gösterimlerle oluşturulan bir tekniktir.

2.1.5. Program görselleştirmenin gelişimi

Program görselleştirmenin gelişimine bakıldığında, Logo isimli firma, 1960'lı yıllarda programlama öğretiminde geliştirdiği görselleştirmelerle, bu alanın öncülerinden biri olmuştur. Logo ortamına bakıldığında yazılan metinsel ve çevre komutlarına göre ekrandaki kuplumbağanın konumlandığı bir alan görülmektedir. Logo'nun çevre tasarımı, öğrencileri aktif olarak çevreyi keşfetmek, program kodunu yazarken ortaya çıkan sorunları gidermek, bilgi edinmek gibi yapılandırmacı yaklaşımları kullandığını ortaya koymaktadır. Logo ortamı temel fonksiyonel programlama öğretmek için kullanılabilmesine rağmen, matematiksel ilkelerin öğretimi için daha uygundur (Petre, 2007).

1967 yılında, MIT Media Laboratuvarlarında eğitim amaçlı kullanım için Logo alt yapısı ile tasarlanacak bir görsel programlama dili üzerinde çalışmışlardır (Papert, 1980). O yıllarda çok sınırlı olan dijital araçlarının yerine, kaplumbağa görünümünde ve bu adla isimlendirilen mekanik bir robot kontrolü sağlayan programlama ortamı

kullanılmıştır. Bu ortam bir kalem ile tek bir yüzey üzerinde LEFT, RIGHT, DRAW ve FORWARD komutlarının kullanıldığı basit görüntüler çizebilen, sensörlü bir yapıya sahiptir. Bu sayede planlanan bilgisayar ve kaplumbağalar arasındaki iletişim ve veri akışı gerçekleştirilmiştir. 1970 yıllarda birçok araştırmacının ilgisini çeken bu Logo ortamı, genellikle kaplumbağa ile geometrik şekiller ve türevlerinin oluşturulmasında kullanılmıştır (Papert 1971). Logo programlama faaliyetleri geometri, cebir, aritmetik gibi konuların öğretiminde kullanılmış ve araştırmalar sonucunda olumlu sonuçlara (Clements ve Battista, 1989; Clements ve Sarama, 1997; Edwards, 1991; Harel ve Papert, 1990; Papert, 1980, 1984, 1993) ulaşılmıştır. Ayrıca, çocukların mekânsal düşünme ve üst düzey düşünme becerileri ile yaratıcılıkları üzerindeki olumlu etkileri olduğu alanyazındaki çalışmalarla (Clements ve Sarama, 1997)desteklenmiştir.

Logo ortamı programlamada tek seçenek olmasa da matematiksel işlemler için kullanılan en yaygın yöntemdir (Harel, 1991). Logo gibi araçları kullanarak, öğrenciler çeşitli modeller geliştirebilir, defalarca bir mucit gibi programın çalışıp çalışmadığını test edebilir ve sonuçlarını görebilirler (Edwards, 1991; Papert, 1980, 1993). Birçok araştırmacı bilgisayar oyunu tasarımında, programlama faaliyetlerinde ve programlama kültürünü teşvik etmek amacıyla Logo programlama araçlarını kullanmışlardır (Harel ve Papert 1990). Ayrıca öğrencilerin yaratıcılıklarını, özgüvenlerini ve programlama bilgilerini oluşturmak amacı ile yine Logo programlama araçları kullanılmıştır (Hoyles ve Noss, 1992; Kafai, Franke, Ching ve Shih, 1998; McCue, 2010; Resnick, 1996). Başlangıçta, Logonun programlama ortamındaki kaplumbağası fiziksel olarak bilgisayara bağlı iken (Papert ve Süleyman, 1971; Papert, 1980), bu kaplumbağadaki fiziksel bağ 1970'lerde (Resnick, 1997) yerini kişisel bilgisayarlarında gelişmesi ile ekrandaki kaplumbağaya bırakmıştır.

1980'li yılların başından itibaren bilgisayar programlama eğitimine ve öğretimine yardımcı olarak görselleştirme araçlarının girmeye başladığı görülmektedir (Adelmann, Bischoff, ve Lauer, 2007; Baecker, 1998; Brown, 1988, 1991; Haajanen vd., 1997; Ramadhan, 2000; Rössling ve Freisleben, 2002). Programlama görselleştirmede kullanılan ilk araçlardan biri Baecker (1998)'in çubukların kullanıldığı, kısadan uzuna doğru çeşitli sıralama algoritmalarını gösterdiği, kısa film çalışmasıdır. Bu çalışmanın sonrasında, bilgisayar teknolojisi uzmanları bilgisayarın artan işlevsel fonksiyonlarıyla beraber daha etkileyici grafikler kullanarak, hassas araçlar yapmayı

düşünmüşlerdir. Bu düşünce, program görselleştirme araçlarının gelişimini hızlandırmıştır.

Program görselleştirme araçlarının gelişimi ile beraber öğrencilerin temel programlama yapılarını öğrenmelerine yardımcı olacağı iddia edilen bu araçların, somut temsillere dayalı olması, soyut olan programlama kavramlarının anlaşılmasını kolaylaştırmıştır. Tam tersi durumda, soyut programlama kavramlarının ve yapılarının anlaşılmasının zor olacağı savunulmuştur. Bu iddiaya karşı, fikir belirtenler olsa da, deneysel kanıtlardaki tutarlılık, bu görüşün güçlenmesini sağlamaktadır. Bu görüşler ışığında Stasko vd. (1993)'nin yaptıkları deneysel çalışmada, animasyonların programlama öğretimine yardımcı olduğunu ortaya koymuştur.

Alana yönelik olarak yapılan meta-analiz çalışmaları, öğrenciler üzerinde gerçekleştirilen görselleştirme aracı etkinliğini belirlemeye yönelik çalışmalar ve bu çalışmaları yapan araştırmacı sayısında bir artmış olduğunu göstermektedir. Bu bilgilere ışığında alanda Naps vd. (2002), tarafından önemli ve merkezi bir çalışma gerçekleştirilmiştir. Bu çalışmada görselleştirme teknolojileri üzerine alanyazındaki çalışmalar derinlemesine gözden geçirilmiş ve görselleştirme etkinliğinin içinde bulunduğu deneysel çalışmalar için bir çerçeve belirlenmiştir. Bu çerçeve ile program görselleştirme teknolojisi ile öğrenen katılımının da içinde bulunduğu bir sınıflandırma yapılması önerilmiştir. Tablo 2.1.'de sunulan taksonomi, öğrenen katılımı ile gerçekleştirilen faaliyetlerin niteliğine göre öğrenenler, altı seviyeye ayrılmıştır.

Tablo 2.1.

Engagement Taxonomy (Meşguliyet Taksonomisi)

Seviye	Açıklama
Görüntüsüz	Hiçbir görselleştirme teknolojisi kullanılmaktadır.
Görüntüleme	Öğrenci görselleri izleye bilir fakat aktif katılımı yoktur.
Yanıtlama	Soruları yanıtlarken Öğrenciler görselleştirme kullanır.
Değişimi	Öğrenciler girişi değiştirerek görselleştirmeyi değişebilirler.
Oluşturulması	Öğrenciler kendi görsellerini oluşturabilirler.
Sunulması	Öğrenciler geribildirim ve tartışma için hedef kitleye görselleştirme

Kaynak: (Naps vd., 2002).

Tablo 2.1'deki "Engagement Taxonomy"sini Myller, Bednarik, Sutinen ve Ben-Ari (2009)'da program görselleştirme ile ilişkili maddeler ekleyerek; (i) görsel olmadan, (ii) sadece görerek, (iii) kontrollü izleyerek, (iv) değer girerek, (v) yanıtlayarak, (vi) değiştirerek, (vii) modifiye ederek, (viii) oluşturarak, (ix) sunarak ve (x) değerlendirerek maddelerini kapsayan yazılım görselleştirme araçları taksonomosini oluşturmuşlardır.

Bu taksonomiler programlama öğretiminde görselleştirme araçlarının öğrencilere programlamayı nasıl daha iyi öğretebiliriz etkinliğinin şekillenmesini sağlamıştır. Scott Grissom (2000) program görselleştirme araçlarının etkinliğini belirlemede fakülte bazında görüşler ölçmek için bir anket geliştirmiş, anket sonuçlarına göre sınıfta program görselleştirme araçlarının kullanılmasının sağlayacağı faydalar şu şekilde ortaya konulmuştur. Öğretmenlerin yüzde (% 90)'nı daha keyifli bir öğretim olduğunu, yüzde (% 86)'sı artan bir öğrenci katılımının olduğunu, yüzde (% 83)'ü öğrencilerin sınıfı daha eğlenceli bulduğunu, yüzde (76)'sı öğrenci motivasyonunun artığını, (% 76)'sı algoritma ve temel programlama yapılarının öğretilmesine yardımcı olduğunu, yüzde (% 76)'sı teknolojinin anlamlı bir şekilde kullanılmasını sağladığını, yüzde (% 72)'si öğrenci başarısını artırdığını ve (% 62)'si de öğrencilerin programlamayı anlamalarını geliştirdiğine inandıklarını belirtmişlerdir. Yine bu çalışmaya göre, fakülte gözlemleri, öğrenci geri bildirimleri ve anketlere dayanarak bu tepkilerin çoğunun kişisel olduğunu, (% 52) oranında ölçülebilir olan öğrenci başarısında bir artış olduğunu belirtmişlerdir. Anketi cevaplayanların yüzde (%93)'nün görselleştirme kullanmanın öğrencilerin programlama kavramlarını bilgisayarda öğrenmelerine yardımcı olabileceği konusunda fikir birliğinde olduğunu ortaya koymuştur (Naps vd., 2006).

2.1.6. Program görselleştirme araç türleri

Bilgisayar programcılığı, teknolojik gelişmelere bağlı olarak hızlı bir şekilde gelişme göstermiş ve bu hızlı gelişim beraberinde birçok program görselleştirme aracının ortaya çıkmasına neden olmuştur. Çok farklı yapıları olan bu araçların sayısının çok fazla artması bir takım karışıklıklarıda beraberinde getirmiştir. SIGCSE (2006) konferansında bir grup bilgisayar bilimi panelisti program görselleştirme araçlarını;

- Anlatı araçları,
- Görsel programlama araçları,
- Akış şeması modeli araçları,
- Özel çıkış gerçekleştiren araçlar,
- Eğitsel programlama dili araçları,

olarak kategorilere ayırmışlardır. Anlatı araçları, bir film şeridi şekilde programlama mantığını tanıtırlar. Görsel programlama araçları kod bölümleri yanlışlarını ve grafiksel programlama kavramlarını görselleştirmek için bir ortam sağlarlar. Görsel programlama araçlarının kullandıkları kod editörü ortamı sayesinde sözdizimi hatalarını en aza indirirler. Akış-model araçları programların geliştirilmesi ve karmaşık programlama algoritmalarının görselleştirilmesi için yararlıdır. Özel çıkış gerçekleştiren program görselleştirme araçları, robotik kullanımı kolaylaştırır. Katmanlı dil araçları ise deneyim seviyesine göre ayarlanabilir ve esnek bir kullanıcı arayüzü ile kullanıcıya kolaylıklar sunabilir (Powers vd., 2006; Daly, 2009).

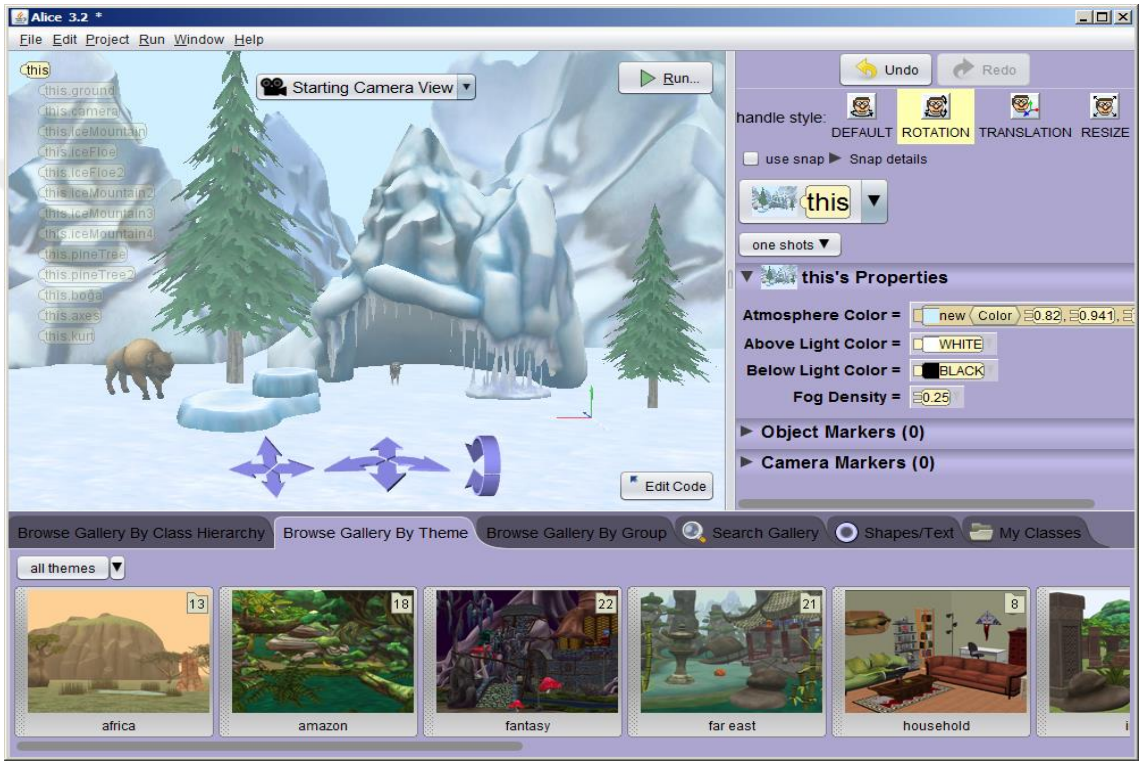
İnternet ortamında yazılım görselleştirme tekniklerini içerek birçok görselleştirme aracına ulaşma ve indirip kullanma imkânı bulunmaktadır. Bazı yazılım görselleştirme araçlarına, internet sayfaları üzerinden ulaşmak ve indirmeden online olarak kullanmakta mümkündür (Shaffer vd., 2010).

2.1.6.1. Anlatı araçları

2.1.6.1.1. Alice aracı

Randy Pausch'ın öncülüğünü yaptığı Carnegie Mellon Üniversitesinde bulunan bir ekip tarafından geliştirilen Alice isimli araç, kullanıcılarına programlamayı kavratmak için üç boyutlu programlama geliştirme ortamı sunmaktadır. Üç boyutlu ortamıyla çok ilgi çeken bu araç, özellikle programlamaya yeni başlayanlar için, eğlenceli bir programlama ortamı oluşturmaktadır (Dann, Cooper ve Pausch, 2000). Saint Joseph Üniversitesinde programlama ortamı olarak Alice'in kullanıldığı bir araştırmada, 21 kişiden oluşan öğrenci grubunda, Alice ile dersi alan öğrenci grubunun başarısı ile geleneksel yöntemlerle dersi alan öğrencilerin başarıları arasında anlamlı bir farklılık bulunmuştur (Cooper vd., 2003).

Nesne tabanlı programlama kalıplarını öğretmek için Alice-2 görselleştirme aracının kullanıldığı başka bir çalışmada, 21 öğrenci seçilmiş ve 7 haftada toplam 14 saat süreli bir kurs düzenlenmiştir. Nesne tabanlı programlama C++ dilinin kullanıldığı programlama dersinde öğrenciler iki gruba ayrıldı ve grupların birine Alice-2 ile eğitim verildi. Kursta bütün öğrencilere ön-test yapıldı ve kurs sonrasında son-test yapılarak çalışma tamamlandı. Çalışma sonucunda Alice-2 ile eğitim alan grubun başarısında anlamlı düzeyde bir fark olduğu görülmüştür (Al-Linjawi ve Al-Nuaim, 2010).



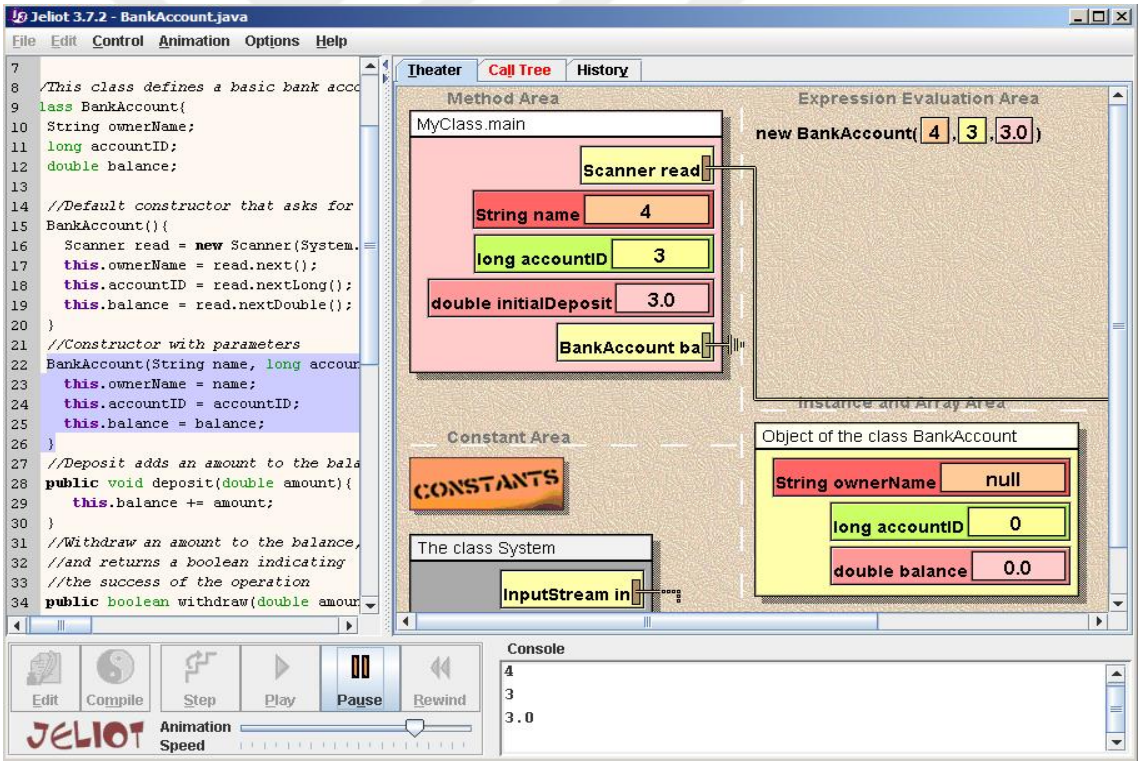
Şekil 2.6. Alice-2 programı ekran görüntüsü

2.1.6.2. Görsel programlama araçları

2.1.6.2.1. Jeliot aracı

Program görselleştirme için kullanılan araçlardan biri olan Jeliot-3 programı, temel kavramların öğrenilmesinde ve programlama genel bilgisinin geliştirilmesinde kullanılan etkili bir yazılım görselleştirme aracıdır. Özellikle programlamaya yeni başlayan acemi öğreciler tarafından çok tercih edilen bir yazılım görselleştirme aracıdır (Levy, Ben-Ari ve Uronen, 2003; Van Haaster ve Hagan 2004; Ragonis ve Ben-Ari

2005). Şekil 2.7 'de görüldüğü üzere, Jeliot-3 yazılım görselleştirme aracının iki bölmeli bir arayüzü vardır. Soldaki ilk bölme, programlama söz diziminim yapıldığı kaynak kod yazma editörü, ikinci bölmesi olan sağ taraftaki bölme ise editöre yazılan kodların görselleştirildiği kısımdır. Yapılan programlama ve ona göre oluşturulan programlamanın kontrol edilmesi için kullanılan Video Cassette Recorder (VCR) gibi düğmeler, arayüzün sol tarafında programın yazıldığı kod editörünün alt kısmıında bulunur. Dinamik bir yapıya sahip olan Jeliot-3 veri ve kontrol akışında nesnel yapıların, metotların ve kullanılan ifadelerin değerlendirmesini sağlar ve bu değerlendirme sonucunu, animasyonel bir şekilde görüntü olarak ifade eder. Kullanıcıların Jeliot-3 de yapması gereken, sol taraftaki kontrol tuşlarını kullanmaktır. Çok basit olan bu kullanımı öğrendikten sonra, kaynak kodun animasyonu kolayca oluşturulur.

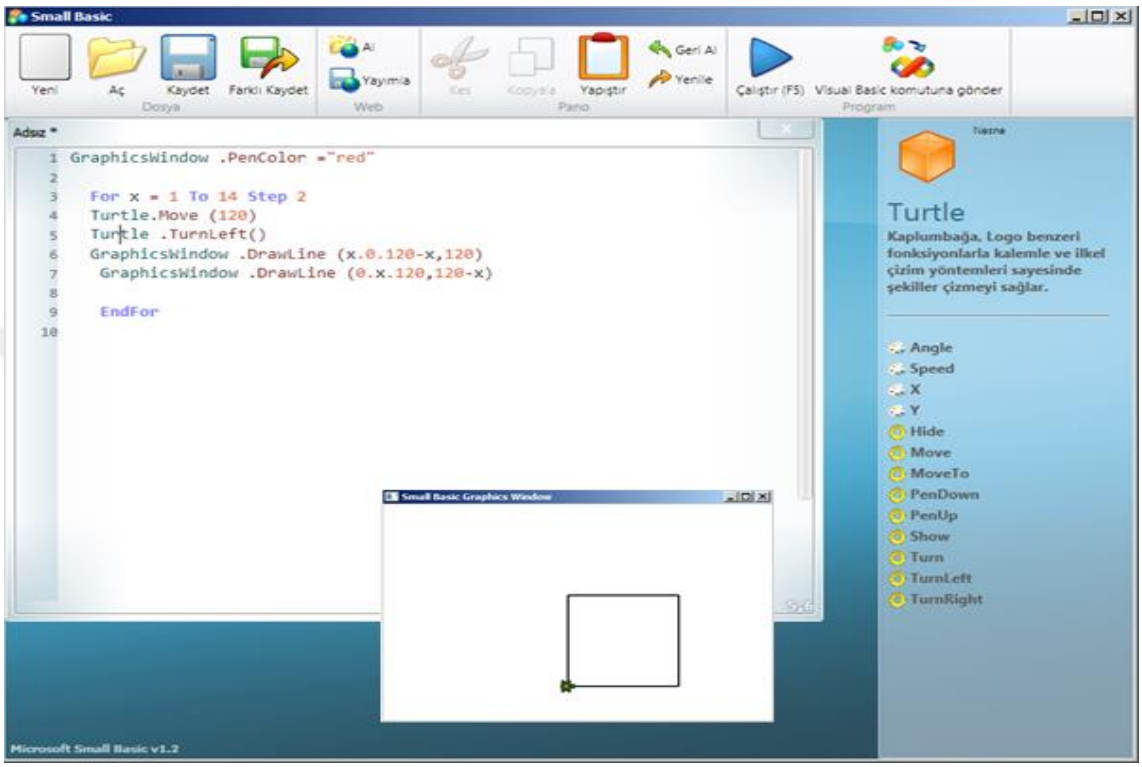


Şekil 2.7. Jeliot-3 program görselleştirme aracı ekran görüntüsü

2.1.6.2.2. Small basic

Yaş grubu olarak özellikle çocuk yaştaki kullanıcılar için tasarlanmış olan Small Basic, acemi düzeyde olan programlama ile yeni tanışanlar için eğlenceli, basit ve kolay

anlaşılabilen bir programlama aracıdır. Aslında Small Basic' in temel amacı, programlamanın temel zorluklarının üstesinden gelinerek, ileriki zamanlarda üst düzey programlama dillerinin öğretimi için bir hazırlık ve programlama bilgisi oluşturmaktır (Ousterhout, 1998).



Şekil 2.8. Small Basic programı ekran görüntüsü

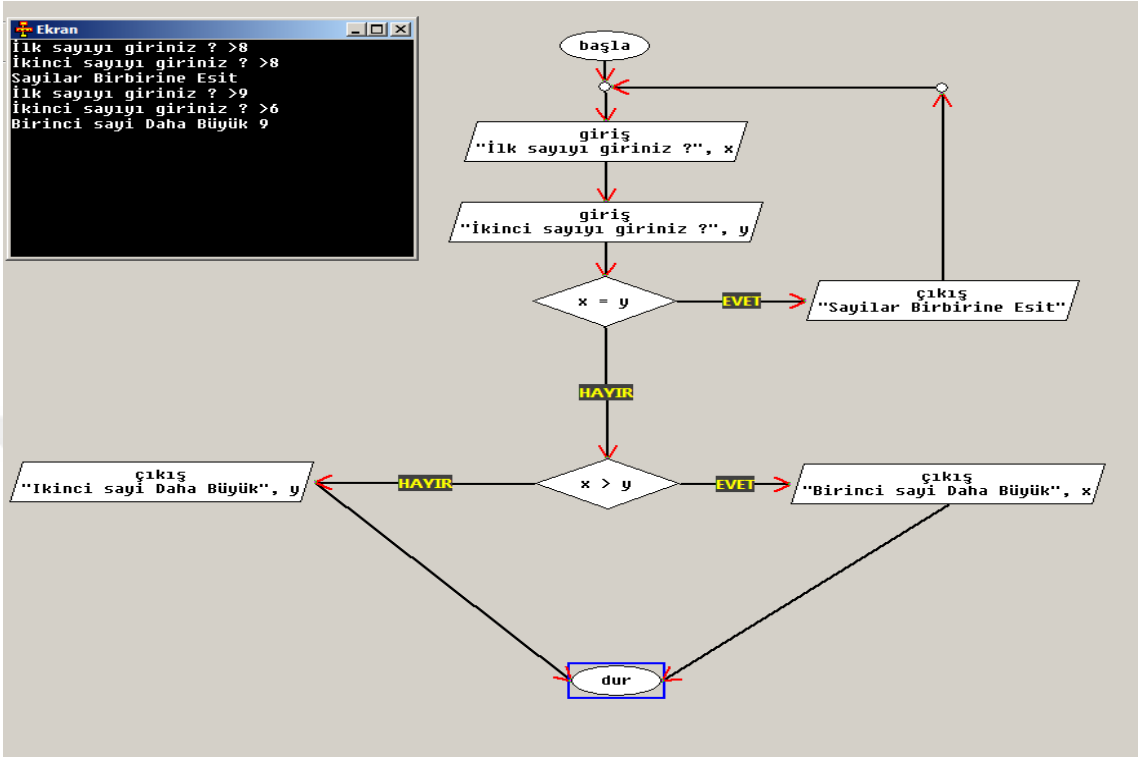
2.1.6.3. Akış şeması modeli araçlar

Program görselleştirme araçlarının derslerde kulanımı ile ilgili (Bravo, Marcelino, Games, Esteves ve Mendes, 2005)'in yaptıkları araştırmada akış şeması modeli araçların programlama dersini ilk kez alan öğrencilerin algoritma yapılandırma kabiliyetlerini geliştireceğini belirtmişlerdir.

2.1.6.3.1. Akış şeması (Flow chart)

Bir takım işlem ve kararların, sözcük ve sembollerin yardımı ile grafik haline getirilerek sunulması algoritmanın öğretimini kolaylaştırır. Akış şemalarında hiyerarşik bir veri yapısı ve böl ve yönet stratejisi kullanılmaktadır. Flow Chart programı, J. Ross Quinlan tarafından ilk olarak geliştirilmiş ve kullanılmıştır. Şekil 2.9 'da görülen flow

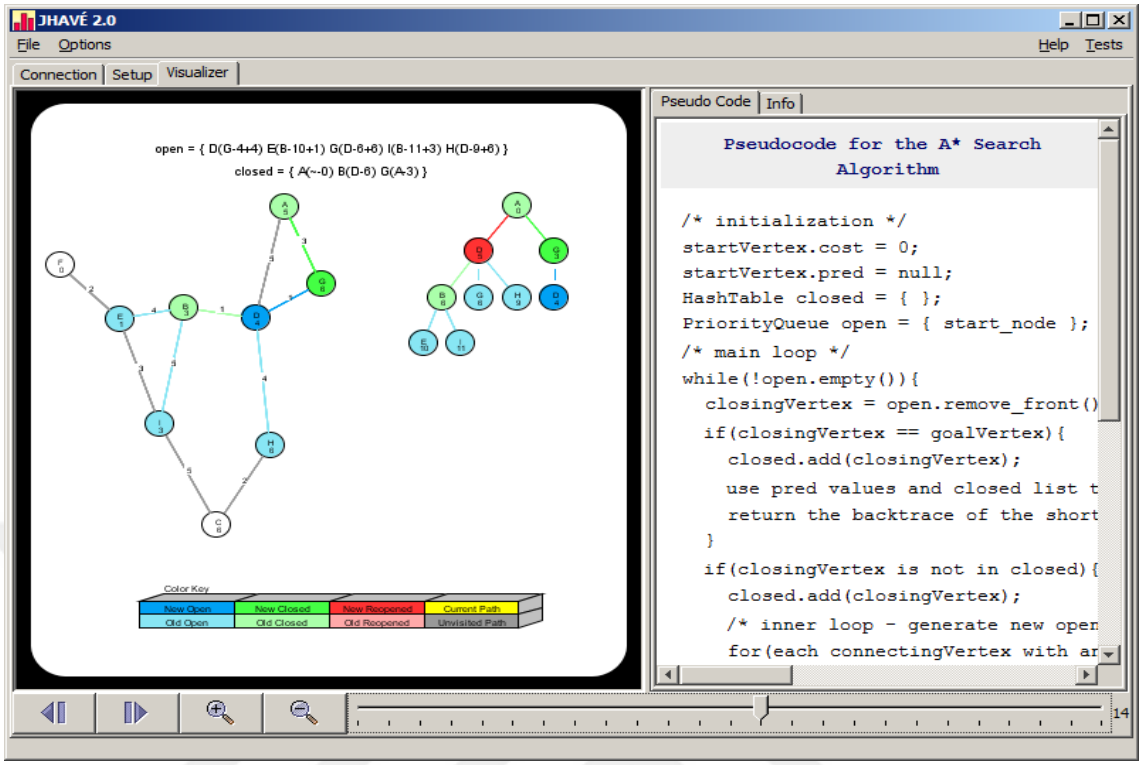
chart programı, ilk düğümü kök düğüm olan ve diğer düğümlerle ağaç şeklinde dalları olan bir akış şeması yapısına sahiptir (Quinlan,1986).



Şekil 2.9. Flow Chart Algoritma görselleştirme aracı ekran görüntüsü

2.1.6.3.2. Jhave aracı

Dershem ve Brummund (1998)'in tasarımını yaptığı "The Sort Animator" aracı algoritma görselleştirmede kullanılan başka bir görşelleştirme aracıdır. Birçok görselleştirme araç gibi Jhave'de kod ve animasyon kısmından oluşan ve algoritma kodu ile kodun animasyonunu beraberce gösteren iki pencereli bir kod yazım sayfasına sahiptir. Programın üstten aşağı doğru olan işleyiş şekline göre, sırası gelen komutu veya programlama yapısını renklendiren bu araç, kullanıcının programı takip etmesini ve programlama akışını anlayabilmesini sağlamaktadır. Naps vd. (2000) ise "The Sort Animator" aracını geliştirip bu araca Jhave isimi vermişlerdir. İstemci-sunucu mimarisine göre çalışan bu "algoritma animasyon" aracı, yapılandırmacı yaklaşıma da uymayan yapısı ve işleyiş nedeni ile günümüz şartlarında programlamaya yeni başlayanlar için, uygun görülmemekte ve fazla kullanılmamaktadır (Gültekin, 2006).

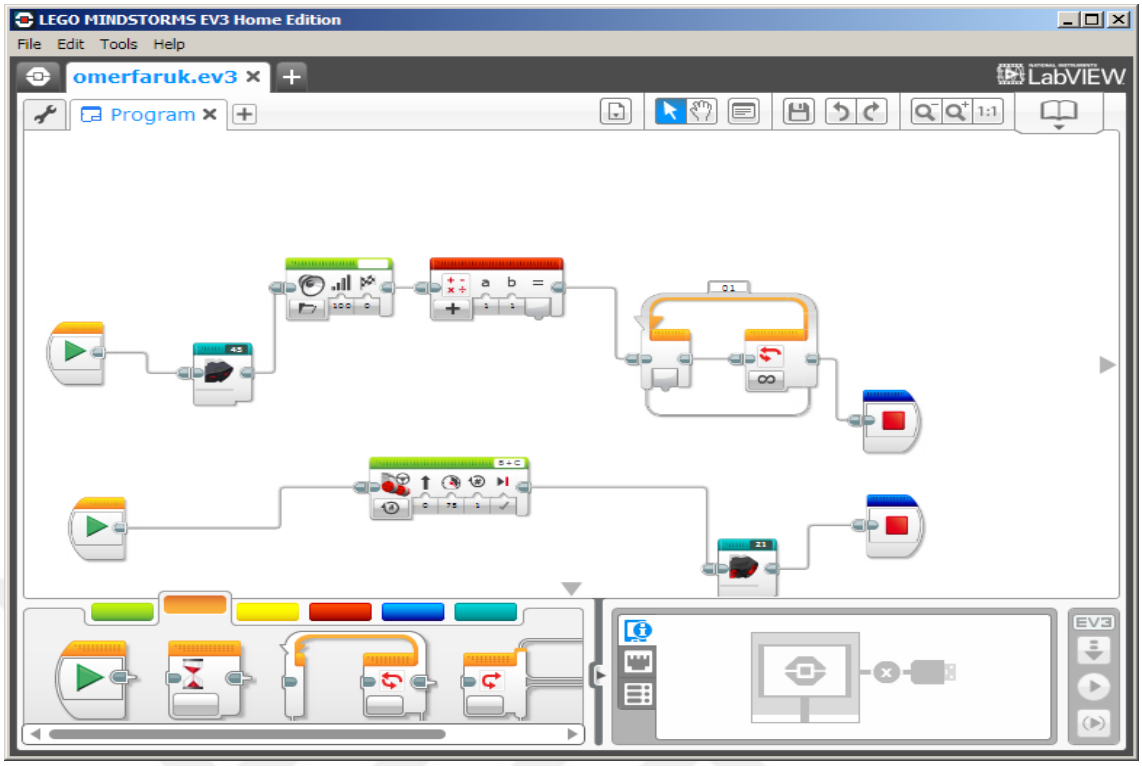


Şekil 2.10. JHAVE görselleştirme aracın ekran görüntüsü

2.1.6.4. Özel çıkış gerçekleştiren araçlar

2.1.6.4.1. Lego mindstorms

Program görselleştirme öncülerinden olan Lego, geliştirmiş olduğu oyuncakların üzerine programlanabilir kumanda bileşenleri ekleyerek, Lego Mindstorms adı verilen program görselleştirme ortamını oluşturmuştur. Ayrıca ticari ve ticari olmayan çeşitli amaçlar için birçok Lego araçları geliştirilmiştir. Örneğin Lego Mindstorms, ticari bir ürün olarak üretilen Lego tabanlı bir araçtır. Bu program görselleştirme aracı içerisinde klasik olarak motorlar, sensörler ve işlemciler gibi Lego blokları ve özel olarak ise birçok bloğu barındırmaktadır. Bu program görselleştirme ortamı kendi programlama yapıları ile program makinelerinin oluşturmasına ve sonrasında bu Lego programları üzerinde kontrol komutlarını geliştirmesine izin vermektedir (Resnick ve Ocko, 1990; Resnick, 1997).



Şekil 2.11. Lego Mindstorms ekran görüntüsü

Günümüzde artık birçok ülkede, ilköğretim çağında öğrencilere programlamanın temel kavramlarının öğretilmesine yönelik olarak, müfredatlarında değişikliklere gittiklerini görmekteyiz. Lego Mindstorms ortamı, bazı ülkelerde son yıllarda ilköğretimde programlamayı kavratmaya yönelik olarak kullanılmaktadır. Bu ülkelerin başında Tayvan ve Kore gelmektedir (Lin, Yen, Yang ve Chen, 2005). Tayvan'da ilköğretime, Kore'de ise 2010'da ortaokullara ve 2011'de liselere ders olarak konulmuştur. Yapılan çalışmalar, ilköğretim düzeyinde bilgisayar işletmenliğine ağırlık vermek yerine, programlamaya yönelik temel kavramların öğretilmesine ve programlama yapılarının kavratılmasına ihtiyaç olduğu belirlenmiştir (Choi vd., 2008).

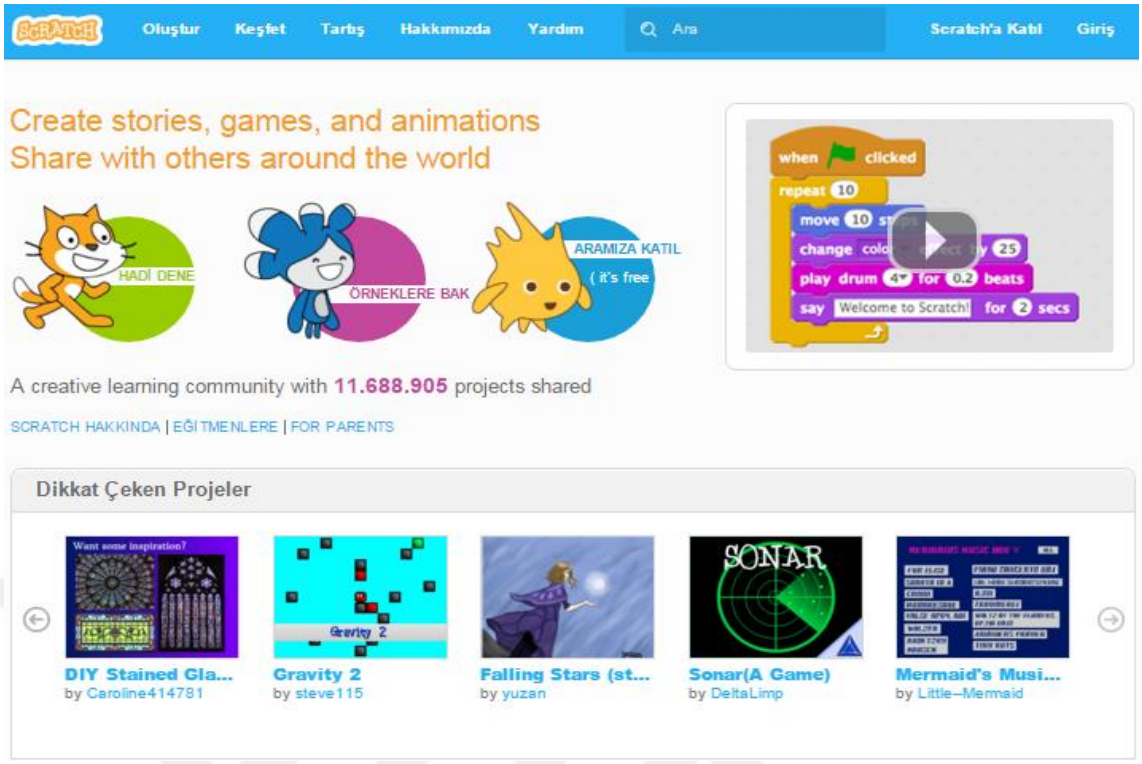
2.1.6.5. Eğitsel programlama dili araçları

2.1.6.5.1. Scratch aracı

Bravo, Marcelino, Games, Esteves ve Mendes (2005)'in yaptıkları araştırmada programlamayı sürükle-bırak yöntemine dayalı katmanlı ve blok-temelli eğitsel programlama dili araçları ile gerçekleştiren öğrencilerin, özellikle nesne yönelimli

programlama dillerini daha iyi anlamalarını ve analiz etmelerini sağlayacağını belirtmişlerdir. Blok-temelli yaklaşımını benimsemiş programlama dillerine, Papert'ın uzun bir dönem üzerinde çalıştığı MIT Media Lab. tarafından geliştirilmiş, son yıllarda önem kazanmış Scratch görsel blok-temelli programlama dili örnek olarak verilebilir (Maloney vd., 2004; Maloney vd., 2010). Scratch görsel blok-temelli programlama dilinin hedef kitlesini genellikle 8-16 yaş grubundaki öğrenciler ve programlamaya ilk başlayanlar oluşturmaktadır. Özünde Scratch, programlamaya temel oluşturmak amacı ile geliştirilmiş bir görsel blok-temelli ücretsiz olarak temin edilebilen bir eğitsel programlama dilidir (MIT, 2015). Öğrenciler bu dili kullanarak kendi hikayelerini, animasyonlarını ve oyunlarını oluşturabilmekte ve bu yolla temel programlama becerisi kazanabilmektedirler (Resnick vd., 2009).

Scratch görsel blok-temelli programlama dili, hem interaktif medya projeleri programlama için bir ortam hem de bu projeleri paylaşmak için bir platform yapısı sunan, zengin içerikli bir görsel programlama dil ortamıdır. Bu ortam, başlangıçta teknoloji firması Logo (Bolt, Beranek ve Newman) ile işbirliği içinde, MIT'de Seymour Papert ve arkadaşları tarafından 1960'ların sonlarında geliştirilen bir programlama dilidir. Bu özelliği ile Logo firması Scratch'ın önemli bir selefi ve entelektüel ilham kaynağıdır. Scratch oyun oluşturma, hikaye canlandırma ve simülasyon gibi etkileşimli medyayı oluşturmak için öğrencilere sunulmuş olan bir görsel blok-temelli programlama dilidir. Programlamada scratch tipik metin tabanlı (örneğin, C, Java, Logo) programlama dillerinde olduğu gibi görerek yazılan ve söz dizimi hataları olan bir yapısı bulunmamaktadır. Programlama yapılarının bloklar şeklinde sürükle-bırak yöntemi ile yazılan ve programlama söz dizimi hatalarını en aza indiren bir programlama dili olması yönüyle de metin tabanlı dillerden ayrılmaktadır.

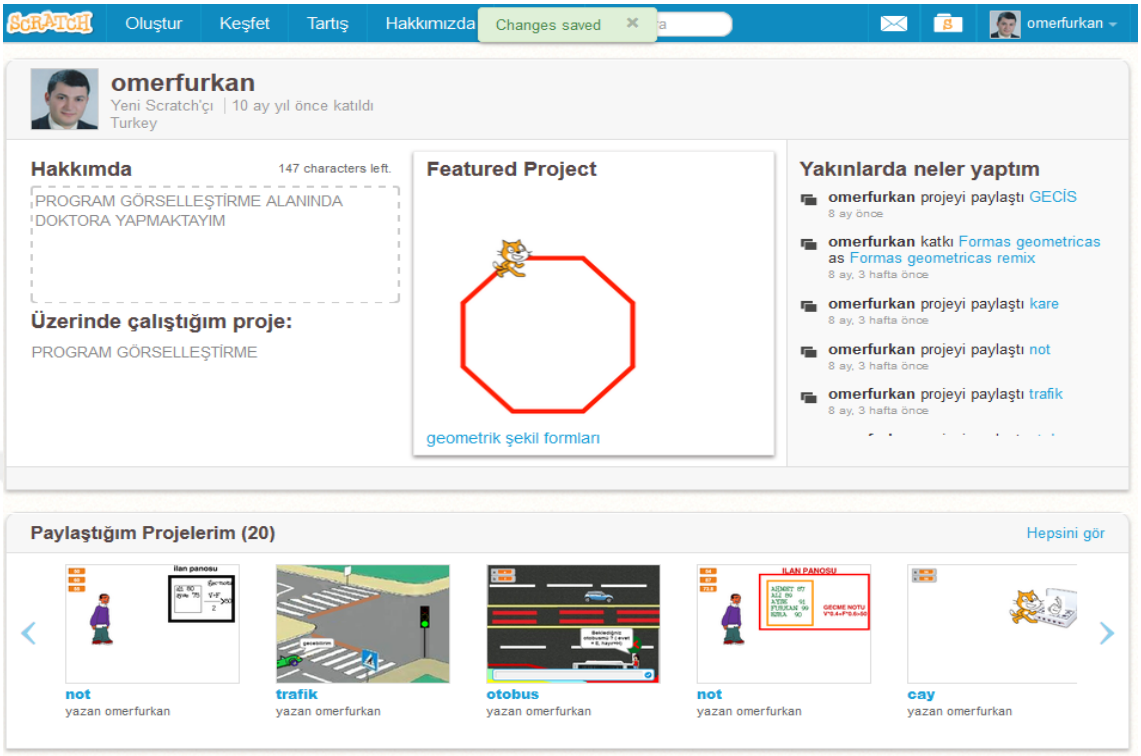


Şekil 2.12. Çevrimiçi Scratch web sitesi ekran görüntüsü (https://scratch.mit.edu/, 2015).

Yaşam Boyu Kindergarten araştırma grubu, Scratch yazma ortamına eşlik edecek bir web sitesi tasarlamış ve bu sitede, *Scratch çevrimiçi topluluk* adında yüzlerce üyesi olan ve her geçen gün üye sayısı artan çevrimiçi bir topluluk oluşturulmuştur. Bu topluluk içindeki Scratch tasarımcıları, tasarım çalışmalarını paylaşmak, birbirlerinin programlama öğrenmelerine destek olmak ve birbirleri ile web sitesi üzerinden iletişime geçmek için bu çevrimiçi plartform bulunmaktadır. Mayıs 2007'de başlatılan Scratch çevrimiçi topluluk, yüzlerce Scratch projesini paylaşan binlerce üyesi (Resnick vd., 2009) ile, önemli ölçüde büyümüştür. Her gün üyeler, (çoğunlukla 8 ile 16 yaşları arasında) 2000'den fazla yeni Scratch projesini internet ortamına yüklemişler ve ilk 5 yılda 2,8 milyondan fazla proje paylaşılmıştır. Bu topluluk, Kasım 2015 tarihi itibarıyla paylaşılmakta olan 11.688.905 proje ile yaratıcı bir öğrenme topluluğu olmuştur (MIT, 2015).

Genellikle gençlerin kullandığı çevrimiçi bu site, kullanıcıların birbirleriyle etkileşim içinde (Kafai, Fields, Roque, Burke ve Monroy-Hernandez, 2012) oldukları sosyal bir ağ yapısı oluştururken, sosyal ağların bileşenlerini de içermektedir. Bu siteye

üye olan kişiler, kendi projelerini ve arkadaşlarının projelerini görüntüleyebildikleri ve bu projeler üzerinde inceleme yapabildikleri bir platforma sahiptirler.



Şekil 2.13. Scratch kullanıcı profil sayfası (https://scratch.mit.edu/users/ omerfurkan/, 2015)

Lawrence, Badre ve Stasko (1994), yaptıkları çalışmada programlama öğretiminde öğrencilerin bir yazılım yardımı ile kendi programlarını oluşturabilmeleri ve bu oluşturdukları programlara sınıf ortamı dışından da erişebilmeleri gerektiğini vurgulamışlardır. Scratch görsel blok-temelli programlama dili ortamında üyeler, birbirlerinin projelerine bakabilir, projeleri hakkında yorumlar yapabilir, projelerin içeriğini görebilir ve projeler üzerinde ekleme çıkarma yaparak problemi çözüme ulaştırmada etkin bir öğretim sağlayabilirler. Bakılan projelerin nasıl yapıldığı, Scratch ortamında incelenebilir veya bilgisayara kaydedilerek Scratch programı ile düzenleme yapılabilir.

Nelson ve Rice (2000), yaptıkları araştırmada, öğrencilerin programlama öğretiminde, programlama dilinden bağımsız olarak programlarını oluşturabilmelerinin önemine değinmiş ve öğrencilerin yazmış oldukları program kodları ile ilgili dönüt almalarının ne kadar önemli olduğunu vurgulamışlardır. Scratch topluluğuna yeni

katılan öğrencilere yazdıkları programlarla ilgili olarak, site içinde daha deneyimli olan kullanıcılar tarafından yardım edilebilmekte ve rehberlik sağlanabilmektedir. Yeni gelen kullanıcıların site tartışma platformu aracılığı ile sorularına yanıt bulmaları sağlanmaktadır. Bunların yanı sıra yeni kullanıcıların, site içerisinde sorularına yanıt bulunabilecekleri Scratch ortamında, programlamaya temel olabilecek şekilde hazırlanmış çeşitli bilgi sayfaları, video ve baskı kartları bulunabilmektedir.

Dinamik bir programlama platformuna sahip olan Scratch, programlama mantığı olarak görsel ve bileşen temelli iki yapının bir araya gelmesiyle oluşur. Geleneksel programlama mantığındaki söz dizimi yerini, burada hazır bir halde bulunan görsel bloklara bırakmaktadır. Programlamadaki ifadeleri sürükle-bırak mantığına göre bir araya getiren bu yapı, sözdizimindeki hata yapma riskini, hazır blokların birbirine uymayanlarının birleştirilememesi sayesinde en aza indirir (Ismail, Ngah ve Umar, 2010).

Bilgisayar programlama öğretmindeki hata yapma riskini en aza indiren bu yaklaşımın en önemli amacı, program yazan kişilerin daha kolay, rahat ve pratik bir şekilde yazılım gerçekleştirmesine olanak sağlamasıdır (Mohamad vd., 2011). Blok-temelli programlama yaklaşımının eğitim ve öğretim aracı olarak kullanılabileceğini gösteren araştırmalar alanyazında bulunmaktadır. Araştırmalarda özellikle blok programlama yaklaşımının programlamaya giriş aşamasında ve küçük yaş gruplarında programlama öğretimine yönelik ilgiyi ve akademik başarıyı olumlu etkilediği sonucuna ulaşılmıştır (Horn ve Jacob, 2006; McNerney, 2004). Akademik başarıyı olumlu etkilediği sonucuna ulaşılsada, bu alanda nicel ve deneysel çalışmalara ihtiyaç olduğu görülmektedir (Horn ve Jacob, 2007; Wang, Zhang, ve Wang, 2011).

2.1.7. Niçin Scratch aracı

Eğitsel programlama dili olarak belirlenen Scratch'ın programlama ortamıyla ilk kez karşılaşan acemi kullanıcılar için, soyut olan programlama yapısını somutlaştırmaya yönelik olarak arayüzü listeleri, kontrol yapıları, değişken ve operatör gibi programlama kavramları kullanıcının anlayabileceği bir şekle dönüştürerek kendi programlama ortamında kullanıcıya sunmaktadır. Bu yapıların somut bir şekilde gösterimi programlamaya yeni başlayanları programlama ortamına daha çok yaklaştırmaktadır.

Bu eğitsel programlama dili ortamı (örneğin bilgisayar bilimlerin, yazılım ve bilgisayar mühendisliği) alanlarında Wolz, Maloney ve Pulimood (2008), veya gelişmiş programlama konularına giriş sağlayacak temel programlama konularında Sivilotti, ve Laugel, (2008) kullanılmaktadır. Bunların yanısıra bilgisayar kulüplerinde gibi sınıf ortamı dışında Maloney, vd. (2008); Peppler, ve Kafai, (2007) yapılandırılmamış öğrenme faaliyetleri ile okul ortamında programlama öğretiminde kullanılmaktadır. Programlama dersini ilk kez alan veya programlama ile ilk kez karşılaşan acemi kullanıcıların, programlama yapabilmesini sağlamak için Fesakis ve Dimitrakopoulou , (2006); Guzdial, (2004); Maloney vd. (2008) tasarlanmış çok fazla sayıda eğitsel programlama dili ortamı mevcuttur. Scratch özellikleri ile bu ortamların birçoğunun yaptığı işlemleri gerçekleştirmesinin yanı sıra, kullanıcıların birbirleriyle iletişim kurmasını sağlaması, yaptıkları programları beraberce değerlendirmelerini ve program üzerinde değişiklik yapmalarını sağlayan birçok özelliği ile diğerlerinden ayrılan eğitsel programlama dili ortamıdır.

Farklı özellikleri ile diğer program görselleştirme ortamlarından ayrılan Scratch ile ilgili yapılan çalışmalara bakıldığında, bilgisayar programlama dersine girişte, Scratch'ın öğrencilerin (Romeike, 2007) keşfetme ve oluşturma düzeylerini geliştirdiği görülmüştür. Romeike'ye (2007) göre programlama ile gerçek hayat arasındaki bir bağın veya modelin kurulması gerekmektedir. Scratch ortamı programlama yapıları ve teknikleri bilinmeden de, öğrencilere kendi program modellerini oluşturma fırsatı vermektedir. Scratch ortamı eğitsel programlama dili ile programlamayı keşfetmeyi ve programlama ile gerçek hayat arasında bağlantı kurmayı sağlayarak, öğrencilere programlama konusunda ihtiyaç duydukları desteği sağlamaktadır.

Bu destek öğrencilerin temel kavramları öğrenmelerinde kullanılan Scratch ortamını ileri seviye programlama dillerine geçiş için bir basamak yapmaktadır (Lewis, 2010). Programlama bilgisi kısıtlı olan öğrencilerin Scratch ile programlama temelleri dersini alması ve değişken, döngü, şart ve operatör gibi yapıları tanıması öğrencileri heyecanlandırmaktadır (Malan ve Leitner, 2007). Scratch, multimedya ortamlarını da kullanarak, programlamayı (Maloney vd., 2008) kolaylaştırmaktadır. Ayrıca Scratch ortamı, öğrencileri motive edecek medya yönüyle, zengin bir programlama ortamı sağlamaktadır (Resnick vd., 2003). Scratch, programları oluştururken sunduğu renkli programlama blokları ve bu blokların sürükle-bırak mantığı ile program

oluşturmaktadır. Ayrıca program diziliminde yanlış kombinasyonlara izin vermemesi yönüylede kullanıcıların hata yapma olasılığını en aza indirmektedir (Wolz vd., 2008). Programlamayı kolaylaştırdığı bu yönleri ile birçok araştırmacı geleneksel laboratuvar ortamındaki programlama derslerine alternatif olan scratch ortamlarının, öğrenciyi teşvik ettiğini belirtmişlerdir (Margolis ve Fisher, 2002; Schuck, DeHaan ve McCray, 2003; Malcom vd., 2005).

Öğrenciler programları oluştururken sözdizimi yapısı ile karşı karşıya kalırlar ve soyut olan bu yapı programlamaya yönelik olarak konsantrasyon bozukluklarına sebep olabilir ve beraberinde programlama hatalarını getirebilmektedir. Hata yapılması başarılı olamayacağım hissini ve programlamaya yönelik kaygıyı artırabilmektedir (Smith, 2009). Bilgisayar programlamada bu tip sözdizimi hataları, programlama öğretimi için kullanılan eğitsel programlama dili ortamları ile en aza indirilmeye çalışılmıştır. Ayrıca öğrencilerin bu ortamlar sayesinde, programlama kavramlarını daha kolay anlamaları sağlanmaya çalışılmıştır (Smith, 2009).

Scholfield (2012) yaptığı çalışmada, programlama kavramlarının daha kolay anlaşıldığı scratch ortamının yenilikçi bir programlama dili ve etkileşimli bir yapısının olduğunu belirtmektedir. Öğrenciler bu ortamın programlamayı kolaylaştıran ve yaparak yaşayarak öğrenmeyi destekleyen eğlenceli bir ortam olduğunu söylemektedirler. Aynı çalışmada (Scholfield, 2012) bu yapının programlama eğitimi verenleri geliştirdiğini, öğrencilerin kendilerini rahat hissettikleri interaktif bir ortam sunduğunu, temel bilgisayar yapılarını anlamak için yararlı olduğunu, mühendislik alanında bile algoritma kavramlarının anlaşılmasında kolaylık sağladığını ve acemi kullanıcılara yönelik programlama ortamı sunduğunu belirtmektedir.

Programlama ve uygulama bilgisi olmayan acemi kullanıcıların, programlama kavramlarını detaylı bir şekilde öğrenmebilmelerinin bir yolu eğitsel programlama dili ortamlarıdır. Bu eğitsel programlama dilllerinden bir tanesi olan Scratch, programlama kavramlarını bloklar haline getirerek ve sözdizimi hatalarını azaltmak için sürükle-bırak metodu ile programlamayı gerçekleştiren etkileşimli bir ortam sunmaktadır (Adams, Webster ve Kolej, 2012). Bu etkileşimli ortamın hedef kitlesi genellikle ortaokul seviyesindeki öğrenciler (Rivzi vd., 2011) olsa da, yükseköğretim'de Harvard, Berkley ve University of California gibi bazı üniversiteler de bu ortamı, programlama temelleri

ve programlamaya giriş dersinde kullandıkları bilinmektedir (Resnick, vd., 2009). İçerik olarak scratch animasyonlu hikayeler, müzik videoları, tebrik kartları, fen projeleri, simülasyonlar ve sensör odaklı (Maloney vd., 2010) birçok yapının geliştirilmesinde kullanılmaktadır.

2.1.8. Programlama öğretiminin önündeki zorluklar

2.1.8.1. Programlamanın soyut yapısı

Programlama öğretimi gören öğrencilerin, programlama temelleri dersindeki akademik başarılarının genellikle düşük olduğu görülmektedir. Araştırmacılar bu sorunun nedenlerini tespit edebilmek için, bu alanda birçok çalışma (Michael, 1999; Nedzad ve Yasmeen, 2001; Michael ve John, 2001; Michael ve Desmond, 2002; Maria vd,, 2003; Sivasakthi ve Rajendran, 2011) yapmışlardır. Araştırmacılar bilgisayar programlama öğretiminin, bilgisayar programlamaya yeni başlayan ve soyut kavramlardan oluşan programlama yapısı ile yeni karşılaşan öğrencilerin, neredeyse tamamının önünde büyük bir engel oluşturduğunu belirtmişlerdir (Proulx, 2000; Lahtinen, Ala-Mutka, Jarvinen, 2005; Fujiwara, Fushida, Tamada, Igaki ve Yoshida, 2012; Cornforth, 2014). Bu alanda araştırma yapan bilim adamlarının bulguları, bilgisayar programlama öğretimi gören ve programlama dersi alan her üç öğrenciden birinin, programlama derslerinde başarı sağlayamadığıdır (Bennedsen ve Caspersen, 2007). Eğitimin her kademesinde, özelliklede yükseköğretim seviyesinde, içeriği, konuları ve yapısı açısından öğrencilerin öğrenmekte en çok zorlandıkları derslerden birinin, bilgisayar programlama dersi olduğu araştırmacılar tarafından belirtilmektedir (Jenkins, 2002).

Alanyazında programlama öğretiminde karşılaşılan zorluklara yönelik yapılan başka bir çalışmada (Proulx, 2000; Dunican, 2002; Jenkins, 2002) programlama öğretimine yeni başlayan öğrencilerin, programlamanın kendine özgü soyut yapıları, kavramları ve programlama mantığını çözmede ve programlama bilgisini anlamada zorluk çektiklerini tespit etmişlerdir. Esteves ve Mendes (2004), yaptıkları çalışmada programlama derslerinde nesne tabanlı programlama dilleri kullanımının, günümüzde en yaygın kullanıma sahip olan programlama dilleri olduğunu belirtmişlerdir. Bu programlama dillerinin anlatımında programlama yapıları ile ilgili bazı zorluklar

yaşanmaktadır. İlk karşılaşılan zorluk, programlama dillerinin dinamik yapıda olması, fakat ders anlatımı ve ders materyallerinin (metin kitaplar, düz anlatım vb.) statik formatta olmasıdır. Dolayısıyla dinamik bir dersin statik bir materyalle anlatılması oldukça güçtür. İkinci bir zorluk ise programlamanın soyut bir yapıya sahip olması ve öğrencinin bu yapının nasıl işlediğini ve sonuçta çözüme nasıl ulaştığını zihinlerinde canlandıramamalarıdır ki, bu da dersin anlaşılmasını ve kavranabilmesini zorlaştırmaktadır. Programlama dersleri diğer derslere kıyasla pratik yaklaşımı olan, sadece teorik değil aynı zamanda uygulanan bilgiye bağımlı bir derstir. Üçüncü olarak ise, programlama dillerindeki statik olan yapıyı dinamik, teorik olan yapıyı, uygulama yapısına yaklaştıracak öğrencilerin bu zorlukları aşabileceği simülasyonlar ve görselleştirmeler geliştirilmeli ve derslerde kullanılmalıdır. Bu yapılar sayesinde öğrencilerin programlama dersini anlamaları ve program geliştirmeleri sağlanabilecektir.

Alanyazında belirtilen programlama önündeki engellerden birinin de öğrencilerin yazılan programın soyut olan kodlarını takip etmedeki yetersizlikleridir (Thomas, Ratcliffe ve Thomasson, 2004). Basit bir programlama kodu için harcanan çaba ve ayrılan zaman ne kadar önemli ise o programda yapılan hataların bulunması da o ölçüde önemlidir. Ayrıca programlama yüksek düzeyde soyutlama ve talepleri uygulamada yoğun bir çaba gerektirir (Eckerdal, McCartney, Moström, Ratcliffe ve Zander 2006). Kinnunen vd. (2007)'ne göre bilgisayarlar zalimdir, onlara tam olarak ne yapmaları gerektiğini anlatmadığınızda, oldukça kesin olmamakla beraber size geribildirim vermeyeceklerdir.

2.1.8.2. Programlamanın imajı ve öğrenci özellikleri

Programlama öğretimi verilirken kullanılan geleneksel yöntemlerin, öğrencilerin programlama bilgisini ve mantığını anlamalarına, programlamaya yönelik zorlukları aşmalarına ve programlama sıkıntılarını gidermelerine katkı sağlamadığını belirtmişlerdir. Ancak programlama öğretiminde programlamaya yönelik becerilerin öğrenciye kazandırılmasında deneyimlerin rolünün yüksek olduğu görülmektedir (Traynor ve Gibson, 2004). Programlamada öğrencilerinin, çalışmaları sırasında aşmaları gereken ilk engellerden biride, programcılıkta tam bir usta nasıl olabilirim düşüncesidir (Eckerdal, Thuné ve Berglund, 2005; Robins vd., 2003). Winslow

(1996:18)'un, "Acemi bir programcının uzman olabilmesi yaklaşık on yıl sürer" iddiası eğer doğru ise, programlama bilgisi ve becerisinin kazanılmasının kolay bir süreç olmadığı açıktır. Benzer bir çalışmada Jenkins (2002: 53-58) programcının, "bilgisayar başından kalkmayan" ve "zamanının çoğunu internet başında geçiren" kişi olarak bilinen imajı, programcı olmak isteyen birçok öğrenci için bir engel teşkil etmektedir. Bilgisayar eğitimi önündeki büyük engellerden bir diğeri de, bilgisayarın olumlu bir imajının olmamasıdır (McGettrick vd., 2005).

Gomes ve Mendes, (2007b), öğrencilerin kendilerine özel kişiselleştirilmiş eğitim ortamlarının olmamasının bilgisayar öğrenme ile ilgili bir sorun olduğunu vurgularken, bu görüşe katılmayan Jenkins (2002) yükseköğretime gidecek birçok öğrencinin bu kişiselleştirilmiş eğitim ortamı imkânlarına sahip olduğunu, fakat temel sorunun bu olmadığını sorunun programlama becerisi olduğunu belirtmiştir. Jenkins(2002) ile aynı görüşte olan Simon vd. (2006), asıl sorunun öğrencilerin motivasyonunu ve bilişsel yeteneklerinin etkili olduğu konusunda hemfikir olduklarını belirtmişlerdir. Ayrıca yapılan başka bir çalışmada öğrencilerin kendi çalışma alışkanlıkları ve programlamaya ayırdıkları zamanın programlama öğrenmeye yetmeyeceğini belirtmişlerdir (Jenkins, 2002; Lahtinen, Ala-Mutka, ve Järvinen, 2005; Gomes ve Mendes, 2007b).

2.1.8.3. Programlama konularının özellikleri

Jenkins (2002) programlama öğretiminde karşılaşılan zorluklar hakkında yaptığı çalışmada; çok az öğrencinin programlamayı kolay buluyorum dediğini belirtmektedir. Bu çalışmada programlamaya yönelik becerileri elde etmenin neden bu denli zor olduğu araştırılmış, bazı programlama konularının doğasıyla, bazı konuların ise çalışma biçimiyle ilgisi olduğu tespit edilmiştir. Bulguların bir diğeri ise; öğretmen ve öğrencilerin tutum ve beklentilerinin, derinden birbirine bağlı olduğudur. Öğretmenler, tüm öğrencilere hızlı ve iyi programlama öğretecek, öğrencilerin karşılaştıkları zorlukları ve karmaşıklığı giderecek bir ortam geliştirmenin ne kadar önemli olduğunu vurgulamışlardır. Ayrıca yükseköğretim kurumlarında zamanın dar olması ve sınıf mevcutlarının çok olması nedeniyle, öğrenciler geribildirim almadıklarını ve öğretmenlerin ise bireysel açıklama yapamadıklarını belirlemiştir (Jenkins, 2002). Bu

yönde yapılan araştırmalar, dikkat eksikliği ve öğrencilerin farklı öğrenme sitillerinin de işin içine girmesinin ayrı bir sorun olduğunu göstermektedir (D'Souza vd., 2008).

Yükseköğretimde programlama öğretimine yönelik Monash Üniversitesinde yapılan bir çalışmada, üniversite 1. sınıf öğrencilerine programlama temellerinin öğretilmesine yönelik verilen bilgisayar programlama temelleri dersi kapsamında, dersi alan öğrencilerin büyük bir çoğunluğunun dersten başarısız oldukları görülmüştür. Bu başarısızlığın nereden kaynaklandığının tespitine yönelik yaptıkları çalışmayı derinleştirdiklerinde, başarısızlığın programlama temelleri dersi konuların yapısından, seçiminden ve sıralanışından kaynaklandığını tespit etmişlerdir. Bu soruna çözüm olarak ise konuların yapısının, seçiminin önemli olduğu ve sıralanışının basitten karmaşığa doğru olması gerektiğini, konu seçimi ve sıralanışında bu kriterlere dikkat edilmesinin başarıyı artıracağını belirtmişlerdir (Hagan, Sheard ve Macdonald, 1997).

Programlamaya yönelik zorlukların analiz edildiği diğer bir çalışmada ise (Samurçay'ın (Pane ve Myers, 1996'da belirttiği üzere) programlama öğretiminde programlama konuları ile ilk karşılaşan öğrencilerin en çok zorlandıkları konuların, değişken yaklaşımı ve bir değişkene değer atamanın nasıl gerçekleştirildiği, döngü kavramı ve döngü içerisinde değişken kullanımının nasıl gerçekleştirildiğinin anlaşılamadığı olmuştur. Ayrıca bu çalışmaya benzer bir çalışmada (Naser, 2008), programlamada algoritma basamakları takip edilirken değişkenlerin değiştirilmesi konusunda öğrencilerin zorluk çekildiğini belirtmiştir.

Programlama öğretimini gerçekleştirenler, programlamanın önündeki bu zorlukları giderebilmek için, teknolojik gelişmelerin elverdiği ölçüde yeni yöntemler denemektedirler (Matthiasdottir, 2006; Sajaniemi ve Hu,2006; Kölling ve Rosenberg, 2001; Cooper, Dann ve Pausch 2000). Fakat bu denenen yöntemler, çok çeşitlilik gösterdiğinden, bu alanda çalışanların çoğunluğunun üzerinde fikir birliğine vardığı etkin bir yöntem yoktur.

2.1.8.4. Programlama kaygısı

Scovel (1991), kaygı ve başarı arasındaki ilişkiyi ortaya koymak için alanyazında birçok araştırmayı incelemiş ve kaygıyı, kolaylaştırıcı ve engelleyici kaygı diye ikiye ayırmıştır. Yapılan bu iki ayrımdan ilki olan kolaylaştırıcı kaygı, öğrenme

işini gerçekleştiren kişiyi yeni öğrenme ortamıyla mücadeleye ve yaklaşma davranışını benimsemeya sevk etmekte ve ikincisi olan engelleyici kaygı ise, öğrenen kişinin yeni öğrenme ortamından uzaklaşmasına ve kaçınma davranışını benimsemesine sebep olmaktadır (Batumlu ve Erden, 2007).

Kaygı ile ilgili alanyazın incelendiğinde, kaygıyı farklı açıdan ele alarak Scovel'den farklı şekilde çeşitlere ayıran çalışmalar mevcuttur. Batumlu vd. ile Kapıkıran kaygıyı, durumluluk kaygı ve sürekli kaygı olmak üzere ikiye ayırmaktadırlar (Batumlu ve Erden, 2007; Kapıkıran, 2002). Sürekli olan kaygı, kişinin kendi içsel özelliklerinden kaynaklanırken; durumluluk kaygı ise kişinin bir durum, bir olay karşısında hissettiği olumsuz sonuç beklentisidir (Kapıkıran, 2002). Alanyazında meslek kaygısı, sınav kaygısı, matematik kaygısı ve bilgisayar kaygısı, durumluluk kaygıya verilebilecek örneklerdir (Gülümbay, 2006; Cambre ve Cook'in (Myers, 2006'da belirttiği üzere).

Bazı çalışmalar (Loyd ve Gressard, 1984) matematik kaygısının, bilgisayarlara karşı olumsuz tutum geliştirdiğinden bahsetmektedir. Bu da bilgisayar programlama başarısını büyük ölçüde etkilemektedir. Programlama ile ilgili faaliyetlerde rahat olan öğrenciler, programlama derslerinde daha iyi performans sergilemekte ve programlama kaygıları o ölçüde minimum olmaktadır. Böylece (bilgisayar kaygısı, programlama kaygısı ya da matematik kaygısı) olmaması veya programlama başarısı elde etmek için olması gerektiği kadar kaygı olması, programlamaya yönelik başarıyı olumlu yönde etkilemektedir (Lindbeck ve Dambrot, 1986).

Alanyazında bilgisayar kaygısının çeşitli tanımları mevcuttur (Gürcan-Namlu ve Ceyhan, 2003; Ceyhan, 2006). Marcoulides (1989) bilgisayar kaygısını, kişinin bilgisayar kullanma esnasında veya bilgisayarı kullandığında doğacak sonuçları düşündüğünde, beliren peşin hüküm veya korku olarak tanımlamıştır. Bir başka tanımda bilgisayar kaygısı; bilgisayar kullanıcılarının derste veya bilgisayar kullanmayı düşündüklerinde korku hissetmeleri olarak adlandırılmıştır (Chua, Chen ve Wong, 1999). Çırakoğlu (2004:15-18) da bilgisayar kaygısını; "bireyin bilgisayar teknolojisini kullanıyor olduğunu düşündüğünde veya gerçekten bilgisayar kullandığında yaşadığı endişe ve korku" olarak tanımlamaktadır.

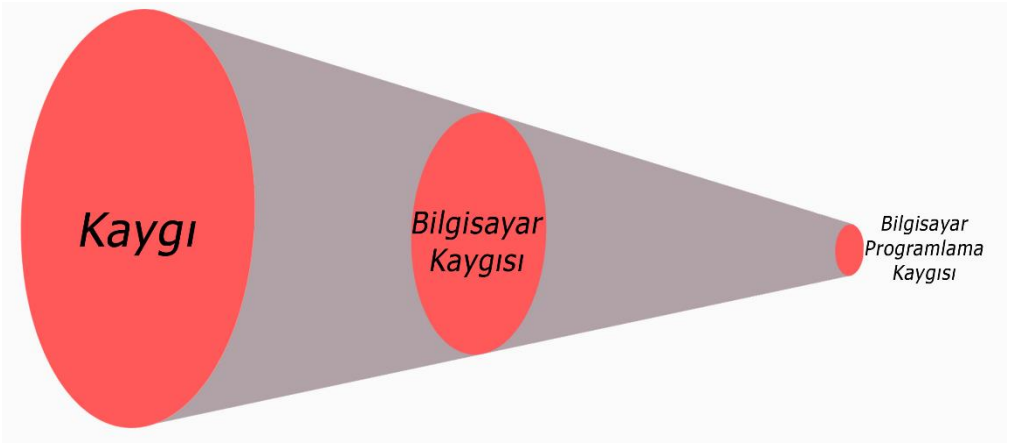
Bu tanımlar incelendiğinde, iki durumun öne çıktığı görülmektedir. Bunlardan ilki kişinin bilgisayar kullanırken ki durumu; bir diğeri ise bilgisayar kullanma ihtimalinin oluştuğunda ki bireyin korku ve endişesi durumudur (Gürcan ve Ceyhan, 2003). Bilgisayar kaygısı, bilgisayarın yüzeysel olarak kullanımına ve bilgisayarın içerisinde bulunduğu bir performans uygulanmasına etki etmektedir (Doyle, Stamouli ve Huggard, 2005). Bu kaygının, bireyin bilgisayar kullanımını veya bilgisayar kullanılarak gerçekleştirilen bir performansa olan etkisinin olumsuz olduğunu açık bir şekilde göstermektedir. Bilgisayar kaygısına sahip bireylerin yanı sıra bu kaygıyı taşıyan öğretmenlerin de bilgisayar teknolojilerini kendi eğitim ortamlarına entegre etmede çekingen davrandıkları bilinmektedir (Ceyhan, 2006). Gürcan-Namlu (2003) alanyazındaki çalışmaları incelediğinde, bilgisayar kaygısının, bilgisayarın öğrenilmesini engelleyen ve ona etki eden bir faktör olduğunu ve bu durumu araştırmaların ortaya koyduğunu belirtmektedir. Bilgisayar kaygı düzeyi yüksek olan kişiler bilgisayarla ilgili konularda ve etkinliklerde kendilerini geliştiremedikleri görülmektedir (Gürcan, 2005).

Programlama temelleri dersini alan kişiler üzerinde bu dersin bir takım duygusal ve sosyal değişimler oluşturduğu görülmektedir. Bu değişimler sırasında kaygı yerini korku, yeni bir ortama girme, kafa karışıklığı ve anlaşılması zor bir duruma bırakmaktadır. Bu duruma yönelik olarak "aklım o zaman nerede oluyor bilmiyorum, fakat onlar sınıfta olduğunda, ben gerçekten onları çok iyi anlayamıyorum" programlama temellerinin ilk kavramlarının üzerine diğer programlama bilgilerin inşa edileceği göz önüne alınırsa, güçlü bir temel oluşturulmasının büyük önem arz ettiği anlaşılmaktadır (Jenkins, 2002:53-58). Kaygı faktörü, öğrencilerin kendilerini rahat hissetmemelerine, tartışmalarda soru sormamalarına ve programlamaya başlayamamalarına neden olmaktadır (Bergin ve Reilly, 2005b). Wilson ve Shrock (2001) göre, rahat hissetme durumunun değerlendirilmesi, öğrencilerin kaygı seviyeleri ile ölçüldüğünü ve rahat hissetme seviyesinin başarının en güvenilir belirleyicisi olduğunu belirtmişlerdir. Programlama öğrenmede, yanlış anlama, karışıklık duygusu, odaklanamama, temel kavramların anlaşılamaması gibi durumlar kişinin rahat hissedememesi, kendisini yetersiz görmesine ve programlamaya karşı bir kaygı ve korku hissetmesine yol açmaktadır. Bunun sonucu olarak kısır bir döngüye girilerek programlamadan uzaklaşıldığı görülmektedir.

Bilgisayar kaygısının, bilgisayar becerileri üzerinde ne kadar önemli bir etkiye sahip olduğu ve öğrenci başarısı üzerinde ne derece belirleyici bir rol oynadığı görülmektedir. Ayrıca önceki bilgisayar deneyimleri, bilgisayar kaygısı üzerinde etkili olduğu, zayıf bir olasılık olsa da bilgisayar kaygısı ile bilgisayar deneyimi arasında negatif bir ekisi olduğu görülmektedir (Marcoulides, 1988).Yükseköğretime başlamadan önce programlama bilgisi ve deneyimi olmayan öğrencilerin, programlama dersiyle karşı karşıya kaldıklarında kendilerini rahat hissetmedikleri görülmektedir. Programlama ile karşılaşan öğrenciler, öğretim yöntemleri ve öğretim ortamlarında değişiklikler olsa da, bu kötü deneyimi üzerlerinden atamamaktadırlar. Bu durum öğrencilerin programlama öğrenme konusunda engel teşkil etmektedir. Sonuç olarak; öğrencilerin zihinlerinde programlamanın zor olduğu yerleşmekte ve ben bu işi beceremem, yeterli değilim, bu çok zor, asla anlayamayacağım ve elde edemeyeceğim kaygısı, programlama öğrenimini olumsuz etkilemektedir (Tan vd., 2009). Ayrıca Reed ve Palumbo (1988) çalışmalarında, bilgisayar kaygısı ile deneyim arasında negatif bir ilişkinin, aynı şekilde Liu ve Reed, (1992) geçmiş tecrübelerle bilgisayar kaygısı arasında negatif bir ilişkinin olduğunu bulmuşlardır. Buna ek olarak, McInerney ve Marsh Avustralyalı bir grup öğrenci üzerinde yaptığı bir araştırmada bilgisayar kaygısının öğrenmeyi kolaylaştırabilir bir parametre olduğunu bulmuşlardır (McInerney'in (Choi, Ligon ve Ward, 2002' belirttiği üzere).

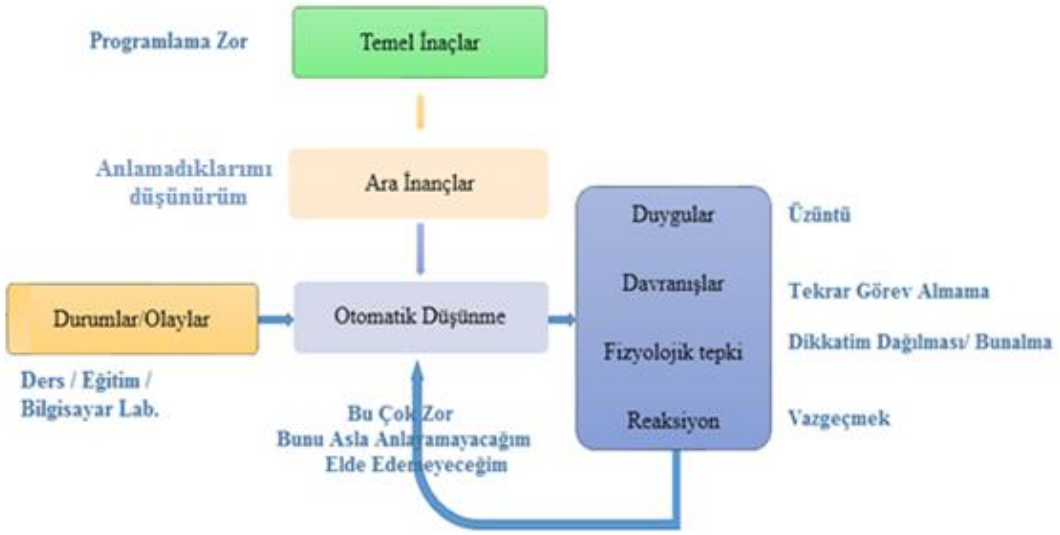
Programlama kaygısı konusunda eğitmenlerin rolü, alanyazında çok yer almamaktadır. Eğitmen olarak, önceki mezunların öğrencilere kapsamlı bir danışmanlık sağlamaları D'Souza vd. (2008)'ne göre daha iyi bir öğrenme sağlayacaktır. McCartney vd. (2007) yaptıkları çalışmada, D'Souza vd. (2008) ile paralel olarak, arkadaş yardımı alınmasının, öğrencilerin doğrudan sosyal etkileşim ile programlama konusunda programlama kabiliyetlerini ve akademik başarılarını artıracağını ve programlamaya yönelik kaygıyı azaltacağını belirtmektedir.

Kaygı, bilgisayar kaygısı ve programlama kaygısı yapılarının şekilsel gösterimi Şekil 2.15'te görüldüğü gibidir. Yapılan bu araştırmada kaygı, programlama için "psikolojik bir durumdur ve öğrenci ne zaman programlama ile karşılaşsa veya programlama ile mücadele etse kaygılanıp başarabilirim hissini kaybedebilir" (Connolly, Murphy ve Moore, 2007:12).



Şekil 2.14. Kaygı, Bilgisayar kaygısı ve programlama kaygısı arasındaki ilişki (Kaynak: Connolly, Murphy ve Moore, 2007'den uyarlanmıştır).

Connolly, Murphy ve Moore, (2007) çalışmalarında, programlama ile ilgili oluşturdukları bilişsel model Şekil 2.16.' te görülmektedir. Kaygı bir durumun yanlış veya işlevsiz değerlendirilmesine yol açmaktadır. Bundan dolayı programlama kaygısıda, programlama öğrenmekte olan öğrencilerin, yeteneklerini yanlış değerlendirmelerine ve kullanamamalarına yol açmaktadır. Kaygının sebep olduğu bu durum, öğrencilerin programlamaya karşı aktif olamamalarına, zihinsel şemaları oluşturamamalarına ve problemlere çözüm üretememelerine neden olmaktadır. Programlama kaygısı bilişsel modelinde, insanların şema oluşturamama, programlamaya karşı aktif olamama ve problemlere çözüm üretememe gibi olaylarla karşı karşıya kaldıklarında zihinsel olarak otomatik düşünceleri devreye girer, temel ve ara inançlar otomatik düşünceleri etkileyerek aktive eder ve sonrasında ise otomatik düşünceler tepkileri tetikler. Bu durumda programlama öğrenme korkusu duyan bir öğrencinin kendi temel inançları, bilgisayar programlama becerisinin önüne geçirdiği görülmektedir.



Şekil 2.15. Programlama Kaygı Bilişsel modeli (Beck, A. T., Emery, G., ve Greenberg, R. L. 2005)'den uyarlanmıştır

İnsanın iç ve dış inançları arasındaki uyumsuzluk, talepler ve kaynaklar, genellikle kaygının kavramsallaştırılmasının merkezindedir. Yükseköğretim ortamlarında programlama kaygısının tanımlanması önemlidir. İnsanların bilgisayar programlamaya karşı olan kaygısını azaltabilmek için onları bilgisayarla karşı karşıya getirmek gerektiği, bu yolla kaygının azaltılabileceği fakat karşı karşıya getirmenin şeklinin de kaygının olumlu yönde etkilenmesinde önemli bir rol oynadığı vurgulanmıştır. Bu karşı karşıya gelme eğitim ortamlarında sadece bilgisayar eğitimi ile ve iyi tasarlanmış eğitim ortamlarıyla gerçekleştirilmelidir (Gürcan, Namlu ve Ceyhan, 2003).

2.1.9. Programlama öğretimini kolaylaştırma yöntemleri

Programlama öğrenmeyi kolaylaştırmak, programlama başarısını artırmak ve programlamayı herkes tarafından öğrenilebilir hale getirmek için, birçok programlama dili ve bu programlama dillerinin daha kolay öğretebilecek öğrenme ortamları geliştirilmiştir (Conway, 1997; De Bonte, 1998; Resnick, vd., 2009; Maloney, vd., 2010). Bununla birlikte programlama ve algoritma öğretimine yönelik alanyazın incelendiğinde, geleneksel öğrenme ortamları haricinde kullanılan programlama öğretimine yönelik ortamlarla yapılan uygulamaların, öğrencilerin başarılarına ve performanslarına olumlu yönde etki yaptığı araştırmacılar tarafından bulunmuştur.

(Crescenzi vd., 2012; Futschek ve Moschitz, 2010; Futschek ve Moschitz, 2011; Gültekin, 2006; Monroy-Hernández ve Resnick, 2008; Peppler ve Kafai, 2005; Resnick vd., 2009; Vobornik, 2011). Benzer bir çalışmada Sanwar (2005) anlatarak öğretim, tebeşir ve konuşma ya da geleneksel yöntemlerle bilgisayar öğretimi yapmanın genellikle verimsiz olduğunu belirtmektedir. Araştırmacı odak nokta olarak programlama öğretme için geleneksel olmayan yöntemlerin kullanılması gerektiğini savunmaktadır.

Programlama öğretimi yapanlar kendilerine etkili bir destek sağlayabilecek (Robins vd., 2003) bir görselleştirme programı aracılığıyla, kendi programlama yeteneklerini ve öğrencilerin programlama yapma güvenlerini geri kazanmalarına yardımcı olabileceklerini bilmelidirler (D'Souza vd., 2008). Öğrencilerin programlamaya yönelik kendi güvenlerini geri kazanma ve kendilerine yaptıkları öneriler konusunda; McCartney vd. (2007) tarafından yapılan çalışmada öğrencilerin, programlamada uygulama, devamlılık, sosyal ağlar ve adım adım talimatlar konularının önemli olduğu yönünde önerileri olduğunu belirtmişlerdir.

Alanyazında benimsenmesi gereken yeni bir yaklaşım ise model tabanlı öğretim yöntemleridir (Caspersen ve Bennedsen, 2007; Muller, Haberman ve Averbuch, 2004). Bu modellerin, kendi programlarının geliştirilmesi için yapı taşları olarak öğrenciler tarafından kullanılan ve (Muller vd., 2004: 102-106) "yinelenen tasarım ya da programlama problemine bir uzman çözüm" üretirmesidir. Önerilen başka bir öğretim yöntemi ise öğrencilere programlamada temel oluşturmak için kullanabilecekleri bir yapı sağlamak (Caspersen ve Bennedsen, 2007; Mead vd., 2006; Thomas vd., 2004) yönünde olmalıdır. Bu yöntemleri kullanarak, düşük bilgi seviyesindeki öğrencilere geri bildirim ve değerlendirme sunarken, görevlerini tamamlamaları için amaç belirtilmeli ve net talimatlar verilmelidir (Mead vd., 2006). Bir konuda genel bir çerçeve çizilmeye çalışılmasına rağmen zayıf olan öğrencilerin hala faydalanamadıkları görülmektedir. Çünkü programlamaya yönelik sorunlar ve algıların, bu sorunun altında yatan en büyük neden olduğu görülmektedir (Thomas vd., 2004).

Programlama başarısını artırabilmek amacı ile programlama öğretimi konusunda yapılan çalışmalarda, program görselleştirmenin bir çözüm olacağı ve görselleştirme araçları ile soyut olan programlama yapısının daha somut bir yapıya kavuşturulacağı

belirtilmiştir (Navarro-Prieto ve Cañas, 2001). Hatta bu görselleştirme ortamlarından veya görsel programlama dillerinden, blok-temelli görsel programlama dillerinin kullanılması önerilmiştir (Maloney, Resnick, Rusk, Silverman, ve Eastmond, 2010). Bunun nedeni olarak ise blok-temelli dillerin metin tabanlı söz dizimi yapısına sahip dillerdeki değişken, döngü, şart, operatör vb. yapıları içerisinde barındırması ve söz diziminde yapılabilecek hataların önüne geçecek bir sürükle-bırak mantığına göre yazılan bir yapıya sahip olmasını gösterilmiştir. Blok-temelli programlama dilleri, bloklar kullanılarak oluşturulan yapılarıyla, program yapmayı sağladığından acemi olan ve ilk olarak programlama ile karşılaşan öğrencileri programlamaya yaklaştırmakta, kaygılarını ve akademik başarılarını olumlu yönde etkilemektedir. Bu sayede programları ezberlemek zorunda kalmaktan, nerede hata yaparım kaygısından ve kod satırları arasında boğulmaktan öğrenciyi kurtarmaya çalışmışlardır (Klassen, 2006). Yap-boz tarzındaki blok-temelli dillerin, sürükle-bırak mantığına göre oluşturulan yapısı ve bu yapıdaki programlama komutlarının yanlış kombinasyonlarla dizilimine izin vermemesi, programlamaya yeni başlayanların hata yapma olasılıklarını en aza indirmektedir. Ayrıca programlama yapısındaki komutlar hakkında programlamaya yeni başlayan acemi kullanıcılara somut bilgiler vermektedir (Kelleher, Pausch ve Kiesler, 2007; Utting, Cooper, Kölling, Maloney, vd., 2010).

Robins, Rountree ve Rountree (2003) yaptıkları çalışmada, acemi kullanıcıların programlamayı öğrenmelerini kolaylaştırmak için programlama konusunda üç şey önermektedirler. Bunlar bilgi, strateji ve programlama modellerinin kullanılasıdır. Çalışmayı yapan araştırmacılar, programlama eğitimini veren eğitimcilerin bu üç yön hakkında bilgi sahibi olmaları gerektiğini ve söz dizimine geçmede, sadece bilgi değil daha somutlaştırılmış durumlara ve ortamlara odaklanılması gerektiğini söylemektedirler.

Programlamaya yönelik derin bir yaklaşım benimsemek yerine, daha yüzeysel bir yaklaşım benimsemek iyi olacaktır (Simon vd., 2006). Derin yaklaşım, kişisel deneyimlerle öğrenme gibi farklı konular ile ilgilidir; yüzeysel yaklaşım ise herhangi bir sorgulama ve gerçekleştirme anlayışı olmadan sınıf ortamını kapsar; yüzeysel öğrenmede programlama ortamında bulunma, derinlemesine öğrenme programın gerçekleşmesi halinde yeterlilik elde edilir (Jenkins, 2002; Schwartzman, 2006; Scott, 2008; Sewchurran, 2008; Simon vd., 2006).

2.2. Araştırmayla İlgili Belli Başlı Çalışmalar

Kinnunen ve Malmi (2008)'de yaptıkları çalışmada, dünya genelindeki üniversitelerde programlama temelleri ve programlamaya giriş dersine kayıt olan öğrencilerin %20 - %40 oranında hatta bu oranların üzerinde öğrencinin dersi veya bölümü daha ilk yılda bıraktıkları görülmektedir. Mccracken vd. (2001)'nin yürüttükleri uluslararası bir araştırmada ise paralel sonuçlara ulaşılmıştır. Araştırmaya göre; öğrencilerin %21'nin programlama derslerinde başarılı olabildikleri görülmüştür.

2.2.1. Akademik başarı ve performansla ilgili araştırmalar

Hansen, Schrimpsher ve Narayanan (1998) yaptıkları araştırmada, programlama öğretiminde, çoklu ortam kullanılarak oluşturulan görselleştirme araçlarının derslerde kullanılması, geleneksel metin tabanlı öğretim tekniklerinin derslerde kullanılmasından daha fazla akademik başarı üzerinde etki gösterdiği sonucuna ulaşılmıştır. Bu çalışmayla paralellik gösteren Klassen (2006)'in yaptığı çalışmada ise, programlama ders konularının geleneksel teorik yöntemlerle verilmesindense, görseller veya animasyonlarla oluşturulan görselleştirme ortamlarında verilmesi gerektiği söylemektedir. Görselleştirme ortamlarının kullanılarak anlatılan programlama derslerinin, öğrencilerin akademik başarılarını olumlu yönde etkilediği belirtilmiştir.

Arabacıoğlu, Bülbül ve Filiz (2007) programlama öğretmek için kendilerinin tasarımını yaptıkları bir uygulama dilinin, akademik başarı üzerindeki etkiyi araştırdıkları çalışmalarında, öğrencilerin, geleneksel programlama öğretim ortamlarında bir başkasının desteğine ihtiyaç duyduklarını belirtmişlerdir. Araştırmacılar etkili bir programlama öğretim için bu yöntemin uygun olmadığını belirtmişlerdir. Araştırma sonucunda öğrencilerin başarısının artırılabilmesi için görselliğin önemli olduğunu ve bu görselliği sağlayacak yardımcı görselleştirme araçlarının kullanılmasının programlama öğretiminde faydalı olacağını vurgulamışlardır.

Kaucic ve Asic (2011) Slovenya'da yaptıkları çalışmada programlamanın, soyut olan karmaşıklık ve zorluğu ile bilindiğini belirtmişlerdir. Araştırmada 4., 7. ve 9. sınıftan aldıkları toplam 32 öğrenciyle 5 ay boyunca görsel blok-temelli programlama dili ile araştırma yapmışlardır. Araştırma sonucunda programlama dili başlangıç

eğitiminin bu şekilde basite indirgenmiş bir ortamda verilmesinin, programlama öğrenimine olumlu yönde katkı sağladığını belirlemişlerdir. Görsel blok-temelli programlama dilinin, programlama kavramlarını öğrettiğini vurgulamışlardır. Başarı ve motivasyonu olumlu yönde etkileyen görsel blok-temelli programlama dilinin, ilerdeki çalışmalarda daha yoğun kullanılması gerektiği önermişlerdir.

Gültekin (2006) üniversite öğrencileri ile yaptığı bir araştırmada, çoklu ortama dayalı bir görsel programlam aracının, programlama öğretiminde kullanımının, öğrencilerin akademik başarıları üzerindeki etkisini araştırmıştır. Yaptığı araştırmanın sonucunda çoklu ortama dayalı görsel programlama aracının kullanımının, öğrencilerin akademik başarıları üzerinde istatistiksel olarak anlamlı bir etkisinin olduğunu tespit etmiştir. Görsel programlama aracının başarıyı olumlu yönde etkilediğinin belirtildiği başka bir araştırmada Cooper, Dann ve Pausch (2003) Ithaca College ve Saint Joseph Üniversitesi'nde görsel programlama araçlarının, öğrencilerin başarı düzeyine olan etkisini incelemişler ve Bilgisayar Bilimleri 1 dersinde görsel programlama aracının kullanıldığı deney grubunun, kullanmayan kontrol grubuna göre daha başarılı olduğu ve başarılı olan bu gruptaki öğrencilerin, Bilgisayar Bilimleri 2 dersine devam edenlerin oranının %91 olduğunu belirtmişlerdir.

Maheshwari (1997) de yaptığı bir araştırmada, programlama derslerinde geribildirim ve uygulama olanaklarını artırmak için, sistematik olarak tasarlanmış zengin öğretim faaliyetlerinden yararlanmak gerektiğinden bahsetmektedir. Maheshwari programlamada planlama, basitleştirme ve modelleme gibi uygulamayı destekleyici stratejiler kullanılması gerektiği görüşündedir. Bu durumun programlama öğretimine olan ilgiyi ve programlama başarısını artıracağı ve hızlı bir sözdizimi temeli sağlayacağı düşünülmektedir.

Peppler ve Kafai (2007) çalışmalarında katılımcılara görsel blok-temelli programlama dili yardımıyla bilgisayarda video oyunu programlamayı öğretmişlerdir. Katılımcılar, görsel blok-temelli programlama dili ile 19 çeşit oyun tasarlamışlardır. Araştırmanın ikinci yılında aynı katılımcılar ile oyun tasarlama çalışmaları devam ettiğinde, katılımcıların ses ve kostüm kullanarak daha karmaşık oyun programlayabildikleri gözlenmiştir. Araştırma sonuçları, ilk ve ikinci yılda geliştirilen projeler karşılaştırıldığında, öğrencilerin ilk yıla göre video oyunu tasarlama konusunda

daha fazla bilgi sahibi olduklarını göstermektedir. Araştırmada, görsel blok-temelli programlama dili kullanılarak video oyunları hazırlamanın, öğrenci başarısını artırdığı ve programlama öğrenmeye yardımcı olduğu belirtilmektedir. Robins, Rountree ve Rountree (2003) programlama konusundaki tespitleri, görsel programlama ve animasyon tabanlı araçlarla, grafiksel programlama dilleri, yansıtıcı yazma kullanımı ve problem kullanımı dâhil programlamanın pedagojisinin diğer alanlarında da kullanılması çok başarılı olmuştur. Bu araştırma sonuçlarına göre, "tek beden herkese uymaz" sözünden hareketle, eğitimcilerin programlama sürecinde sadece bilgiye yönelik öğretim çabalarına odaklanmaktan ziyade, başarılı görselleştirme gibi öğretim yöntemlerini de kullanmaya çalışmalıdırlar.

Grissom, McNally ve Naps (2003) yaptıkları araştırmada, birçok üniversitede görsel programlama dilleri ile yapılan çalışmaların sonuçlarını bir araya getirerek bu çalışmaların sonuçlarını analiz edilmiştir. Analizlerde görsel programlama dili ile etkileşimin öğrenme üzerindeki etkisi araştırılmıştır. Araştırma sonucunda, görsel programlam dili ile öğrenci ne kadar fazla etkileşime girerse o oranda öğrenmenin ve başarının olumlu yönde etkilendiği görülmektedir. Yükseköğretimde yapılan diğer bir araştırma da Malan ve Leitner (2007)'in yaptıkları çalışmadır. Malan ve Leitner (2007), yaptıkları çalışmada görsel blok-temelli programlama dili, yükseköğretimde Harvard Yaz Okulu'nun Computer Science dersinde kullanmışlardır. Araştırma kapsamında dönem başında öğrencilere, önceki programlama bilgileri üzerine anket yapılmıştır. Anket sonuçlarına göre, 25 katılımcı üzerinden % 52'sinin herhangi bir ön bilgisi yok iken, %32'sinin az bir bilgiye sahip olduğu görülmüştür. % 16'sının ise programlama ile ilgili güçlü bir alt yapısının olduğu bilgisi edinilmiştir. Dönem sonunda görsel blok-temelli programlama dilinin, önceden programlama deneyimi olan öğrencilerin başarısını nasıl etkilediği araştırılmıştır. 25 katılımcı içerisinden 19'u (% 76) pozitif etkisi olduğunu, 2'si (% 8) negatif etkisi olduğunu, 4'ü (%16) ise görsel blok-temelli programlama dilinin herhangi bir etkisi olmadığı görülmüştür.

Ramadhan (2000) ise, diğer araştırmacılardan farklı olarak, görsel programlama dilleri ile ilgili deney ve kontrol gruplu bir çalışma gerçekleştirmiştir. Gruplara ön-test yaptıktan sonra, görselleştirme kısmı olan bir görsel programlama dilinin deney grubunda, görselleştirme özelliği olmayan bir dili ise kontrol grubunda kullanmıştır. Uygulama sonucunda görselleştirme özelliği olan dilin kullanıldığı deney grubunda son-

test sorularının tamamında olmasa da döngü ve koşul içerikli sorularda, kontrol grubundan daha başarılı olmuşlardır. Araştırma sonunda; programlama derslerinde görselleştirme dillerinden dersin işlenişinde destek alınmasının öğrencilerin akademik başarılarına pozitif yönde etki ettiği görülmüştür.

Tekerek ve Altan (2014) yaptığı çalışmada, görsel blok-temelli programlama dilinin öğrenci başarısına etkisini incelemişlerdir. İlköğretim 6. sınıf'ta öğrenim gören 60 öğrenci üzerinde yapılan çalışmada, deneysel yöntemlerden, kontrol gruplu ön-test son-test deseni kullanılmıştır. Görsel blok-temelli programlama dilinin öğrenci başarısı üzerindeki etkisini belirlemek için başarı testi kullanılmıştır. Deney grubunda görsel blok-temelli programlama dili kontrol grubunda ise geleneksel yöntemlerden düz anlatım yöntemi ile ders işlenmiş, önbilgileri kontrol altına alındığında deney ve kontrol gruplarının son-test puanları arasında anlamlı bir fark oluşmazken, grupların kendi içlerinde başarıları karşılaştırıldığında anlamlı bir farklılık oluşmuştur.

Hundhausen, Douglas ve Stasko (2002) yaptıkları çalışmada, Tekerek ve Altan, (2014)'ın çalışmalarında görüldüğü gibi, görsel blok-temelli programlama dilinin öğrenme üzerinde pozitif bir etkisi olduğu ve öğrencilerin akademik başarılarını olumlu yönde etkilediği görülmüştür. Ayrıca program görselleştirmenin tek başına yeterli olmadığını önemli olanın görselleştirmenin yanında, görsel blok-temelli programlama dili ile öğrenciyi derse ve konuya dâhil etmek olduğu belirtilmiştir. Bu çalışma ile paralellik gösteren diğer bir çalışma olan Naps vd. (2003) çalışmasına göre, görsel blok-temelli programlama dili öğrencileri aktif olarak öğrenmeye katmalıdır. Araştırmacılar ayrıca programlama derslerinde öğrencilere yaptırılacak etkinliklerin Bloomun taksonomisine dayandırılması gerektiğini belirtmişlerdir.

Rogerson ve Scott (2010) çalışmalarında, fenomenolojik yaklaşım kullanarak son sınıf lisans öğrencilerinin anlattıkları kişisel deneyimleri ve zengin açıklamalarını ortaya çıkarmak için, programlama öğrenmede öğrencilerin deneyimlerinden, korku duygularının nasıl etkilendiğini incelemişlerdir. İncelemede üç odak alanının; birincisi, programlama öğrenmede öğrenci başarısı; ikincisi, programlama önündeki engeller ve üçüncüsü ise, eğitim araçları ve öğrenme metodolojileri üzerine durulmaktadır. Bu alanda fenomenolojik araştırmaya özellikle vurgu yapılmıştır. Kavramsal bir çerçeve olarak beceri ediniminde Cockburn kavramları ve yetişkin öğrenmede Dreyfus'un

seviyeleri tartışılmış ve öğrenci gelişimini incelemek için kullanılmıştır. Bu çalışmanın temel amacı, öğrencilerin yaşam deneyimlerini anlatmak ve onların bakış açısıyla programlamanın özünü ve anlamını keşfetmek olduğu için, bir yorumlayıcı duruş araştırmanın başarısını ve fenomenolojik yöntemin ideal olduğunu ortaya çıkarmıştır. Çalışmanın analizinden, altı tema ortaya çıkarılmıştır. Bu temalar; programlama ile ilişkili kaygı ya da korkular, ortaya çıkan olumsuz algılar, bu duyguların yol açan programlamanın doğası, iç faktörler, bu duyguları etkileyen dış faktörler, nihayetin de bu duyguları etkileyen ve bu duyguların öğrencilerin başarısını ya da beceri edinimine etkisidir. Bir model, olarak bu altı tema arsındaki ilişki ve etkileri önerilmiştir. Bu önerilere uyulması halinde, programlama öğrenen öğrenciler tarafından yaşanan zorlukların, eğitimciler tarafından anlaşılmaya çalışılacağı ve kendi eğitim ortamlarında etkin destek sunulacağı belirtilmiştir.

Fesakis ve Serafeim (2009) yaptıkları araştırmada, öğrencilerin görsel blok-temelli programlama dili ile ilgili olarak, bilgisayar bilimine giriş ve eğitim uygulamalarında, etkili uygulamaların benimsenmesi olanaklarını artırdığını, bilgisayar programlama ve bilişim eğitiminde pozitif bir etkiye sahip olduğu düşünülmektedir. Araştırma sonuçları, gelecekte öğretmenler için bilgisayar programlama derslerinde görsel blok-temelli programlama dili kullanılarak eğitim yapılabileceğini göstermektedir.

Lahtinen, Ahoniemi ve Salo (2007) yaptıkları çalışmada, görsel programlama araçlarının programlama derslerinde nasıl kullanılabileceği üzerinde durulmuştur. Bu çalışmada kullanılan görselleştirme aracı ile öğrencilere programlamadaki döngü, dizi, koşul ve fonksiyon kavramları anlatılmıştır. Araştırma sonucunda öğrencilerin koşul cümlelerini çözümlemesinde başarılı oldukları görülmesine rağmen fonksiyonlarda başarı sağlayamamışlardır. Araştırma sonucunda; programlar karmaşıklaştıkça görsel programlama araçlarına duyulan ihtiyacında arttığı ve karmaşıklığı giderecek olan görsel programlama araçlarının kullanımında öğrenci başarısını artırdığı görülmektedir.

Başer (2012) yaptığı çalışmada, programlama dili olarak Python dilini kullanmış ve programlama öğrenirken, öğrenenlerin hangi konularda ve genel program becerilerinde nerelerde zorlandıklarını ortaya çıkarmaya çalışmıştır. Ayrıca öğrenenlerin programlamayı en iyi nasıl öğrendiklerine odaklanılmıştır. Betimsel bir araştırma

modelinin kullanıldığı çalışmada, 22 maddelik 3'lü likert tipte programlama zorluklarını belirlemeye yönelik bir ölçek ve öğrenenlerin en iyi programlamayı nasıl öğrendiklerine yönelik, çoktan seçmeli bir maddenin kullanıldığı bir ölçek kullanılmıştır. Bilgisayar ve Öğretim Teknolojileri Eğitimi bölümü 2. sınıfta okuyan 96 öğrenci üzerinde gerçekleştirilen çalışmada, öğrenenlerin en çok nesne başlığı ve miras alma kavramlarında zorlandıkları ve bu konuların ötelenmesi gerektiği belirtilmektedir. Genel programlama becerilerinde ise öğrenenlerin başkalarının hatalarının bulunmasının çok zor olduğu belirtilmiştir. Son olarak ise, öğrenenlerin kendi başlarına ve akranlarıyla birlikte çalışınca en iyi öğrendiklerini vurgulamışlardır.

Velasquez vd. (2014) yaptıkları görsel blok-temelli programlama dili (Scratch) online sosyal ağ ortamındaki çalışmalarında, paylaşılan projeler hakkında rastgele seçilen 8000 yorum izlenerek veri madenciliği yapılmış ve analiz edilmiştir. Analizler sonucunda bu görsel blok-temelli programlama dilinin her dilde destek sunması, kullanıcısına bireysel program geliştirme imkânı vermesi, takım çalışması yaparak programlama hakkında kendilerini geliştirmesi ve bu görsel blok-temelli programlama dilinin, sosyal ağ ortamı ve bu ortamın etkileşimli yapısı programlama başarısını olumlu yönde etkilemektedir.

Programlama ile ilgili diğer bir çalışmada Yorulmaz (2008), tarafından internet kafelerde gençler ve çocukların yeni bir programlama ortamı olan Scratch'ı kullanmaları konusu incelenmiş ve daha faydalı kullanılabilmesini sağlayabilmek için öneriler sunulmuştur. Görsel blok-temelli programlama dili olan Scratch, kendi görselleştirme ortamıyla internet kafeye gelen çocuk ve gençlerin teknoloji kullanım becerilerini, oyun aktivitelerinden daha ileriye taşımayı hedeflemiştir. İnternet kafelerin son yıllarda yaygınlaşmış olması ve bu merkezlerde programlamayı öne çıkaran görsel blok-temelli programlama dili olan Scratch gibi dillerin kullanılmasının önemli olduğu görülmektedir.

Chang (2014) yaptığı çalışmasında, görsel blok-temelli programlama dilinin birçok çalışmada kullanıldığını ve çoğunluklada K-12 bilgisayar derslerinde, dersin içinde bütünleştirilerek olumlu sonuçlar alındığını belirtmiştir. Bu çalışmada görsel programlama dillerinden Alice ve Scratch'ın etkilerini göstermek içi özellikle programlamaya giriş dersinde düşük performanslı öğrencilere görsel programlama

dilleri sorulmuş ve yanıtlar karşılaştırılmıştır. Çalışmada öğrenme hedefleri, öğrenme kaygısı ve öğrenme arasındaki ilişki incelenmiştir. Sonuçlar düzeltici öğretim için uygun görsel programlama dili seçmede bilgisayar bilimleri eğitimcilerine sevk edilebilir. Yöntem olarak ise, Tayvan'ın güney kesiminde, sınırlı kaynaklardan dolayı, amaca uygun örneklem yöntemi ile iki grup oluşturulmuştur. Scratch grubunda 32 erkek ve 13 kızdan oluşan 45 öğrenci, Alice grubunda 35 öğrenci vardır. Scratch ve Alice gruplarına konuları tanıtılmış ve 10 hafta 3 saat olarak uygulama çalışması belirlenmiştir. 9. hafta ara sınav için 3 saatlik iyileştirme programına öğrenciler alınmış ve son-test olarak ara sınav yapılmıştır. Ayrıca öğrenme hedefleri, öğrenme kaygısı ve öğrenme arasındaki ilişki için anketler dağıtılmış ve puanlar verilmiştir. Sonuçlar Scratch için incelendiğinde Scratch ile düzeltici talimatlar almış öğrenciler, gerçekten C++ programlamada öğrenme kaygısını düşürdüğü görülmüştür. Alice, görsel programlamayı desteklemesine karşın anlamlı bir farklılık oluşturmamıştır. Alice, 3D olarak Scratch'den daha iyi görülmüştür. Scratch ve Alice'nin motivasyonu artırıcı ve aktif öğrenmeyi teşvik edici olduğu görülmüş ve Scratch'nin daha yaratıcı olduğu ve performans üzerinde etkili olduğu bulunmuştur.

Kendine özgü sözdizimi karmaşıllığı ve geleneksel programlama dilerinin yapısı, öğretmenlerin öğrencilerine bilgisayar programlamayı öğretirken zorluk yaşadıkları konuların başında gelmektedir. Scratch gibi görsel blok-temelli programlama dilleri bu zorlukların aşılmasında bir çözüm olarak görülmektedir. Bununla birlikte bu dillerin programlama ortamlarına uygunluğunu inceleyen çok az araştırma olduğu görülmektedir. Bu çalışma görsel blok-temelli programlama dili kullanılarak 6. Sınıf öğrencileri üzerinde yapılmıştır, çalışma sonucunda bu dilin programlama akışını kavramada öğrencilere yardımcı olduğu sonucuna ulaşılmıştır (Lai ve Guo, 2011).

(Chiu, 2015) yaptığı çalışmada, programlama derslerinin öğretiminde görsel blok-temelli programlama dillerinden App Inventor ve Scratch kullanılmıştır. Çalışmada ilk dönemde sekiz yüksek lisans öğrencisine Scratch ile temel programlama kavramları öğretilmiştir. Daha sonraki dönemde ise App Inventor, programlama kavramlarını öğretmek için kullanılmıştır. Öğrencilerde programlama başarısına yönelik kodlar, anketler ve mülakatlar yapılarak veriler toplanıp analiz edilmiştir. Analiz sonucunda programlama kavramlarının öğretilmesinde görsel blok-temelli programlama

dillerinin kullanımının olumlu bulunduğu görülmüştür. Ayrıca görsel blok-temelli programlama dilleri ile görsel ortamlarda öğrencilerin programlama deneyimlerini geliştiği görülmüştür. Fakat görsel blok-temelli programlama dillerinin mobil uygulamalarının kaygıyı azaltmadığı tespit edilmiştir. Bu durum App Inventor dilinin gelecekteki programlama öğretimlerinde kullanılabilmesi için tanıtılması gerektiği sonucuna ulaşılmıştır.

Yüksekokul öğrencileri üzerinde yapılan çalışmada, görsel blok-temelli programlama dili (Scratch)'ın kullanıldığı durumsal bir öğretim ortamı tasarlanmıştır. Bu ortamda öğrencilerin günlük hayattaki problemlere yönelik olarak görsel blok-temelli programlama dili ile çözümler tasarlamaları istenmiştir. Bu yolla programlama öğretiminin gerçekleştirilmiştir. Çalışmanın sonucunda görsel blok-temelli programlama dilinin öğrencilerin ilgisini uyandırdığını ve geleneksel programlama ortamlarına öğrenmelerini transfer edebildikleri sonucuna ulaşılmıştır. Bu durum programlamayı anlamayı, mantıksal düşünmeyi ve programlamaya yönelik pratik çözümler üretmeyi sağlamıştır (Wang ve Zhou. 2011). Bu çalışmada benzer bir çalışmada üniversite öğrencilerinin programlama temelleri dersinde görsel blok-temelli programlama dilinin kullanımının yüksek seviyeli dillere geçişte kolaylık sağladığı anlaşılmaktadır. Ayrıca görsel blok-temelli programlama dillerinin programlamanın zor olan bazı kavramlarının öğretilmesinde yardımcı bir araç olarak büyük ümit vaat ettiği görülmektedir (Garner, 2009).

Bilgisayar programlama, öğrencilere kod yazmanın yanında yaratıcı düşünme, problem çözme, işbirlikçi öğrenme ve kritik düşünme gibi üst düzey bilişsel becerileri kazandırmaktadır. Geleneksel programlama dillerinin kullanıcı dostu olmayan arayüzleri karmaşık kod dizimi öğrencilerin bilgisayar programlamayı öğrenmelerini imkânsız hale getirmektedir. Öğrenilmesi kolay, görsel özelliği olan ve kullanıcı dostu yeni görsel programlama dilleri programlama öğretiminde son yıllarda kullanılmaya başlamıştır. Bu dillerden bir tanesi görsel blok-temelli programlama dili (Scratch) dir. Scratch ile ilgili alanyazındaki 32 çalışma analiz edilmiş ve analiz sonucunda Scratch'nin programlamada etkili olduğu ve programlamayı zevkli hale getirerek anlaşılmasını kolaylaştırdığı ve programlama başarısını artırdığı görülmüştür. Scratch ortamı programlama bilgisi gerektirmeyen sürekle-bırak mantığıyla çalıştığı için

acemilerin programlamaya karşı motivasyonlarını artırarak, kaygılarını olumlu yönde etkilediği görülmektedir (Çatlak, Tekdal ve Baz, 2015).

2.2.2. Performans ile ilgili araştırmalar

Rizvi, Humphries, Major, Jones ve Lauzun, (2011) yaptıkları çalışmada, bilgisayar bölümlerinin son yıllarda öğrenci sayılarında artış olmasına rağmen, gelen öğrencinin zayıf bir matematik bilgisine sahip olduğu tespit edilmiştir. Bu çalışmada zayıf matematik bilgisi olan öğrencilere programlama derslerinin görsel blok-temelli programlama dilleri yardımıyla verilmesi yönüne gidilmiş ve öğrencileri, derste tutma, ders performansı ve programlama tutumlarının iyileştirilmesi gibi etkinlikler belirlenmeye çalışılmıştır. Çalışma sonucunda öğrencilerin öz-yeterliliklerinde, performanslarında ve başarılarında önceki yıllara göre artış gözlenmiştir. Üniversitedeki programlama dersi alan öğrenciler üzerinde yapılan benzer bir çalışmada Konvalina, Stephens ve Wileman (1983), üniversitedeki programlama dersindeki performansın; lisedeki matematik, genel ve bilgisayara yönelik derslerin not ortalaması ile ilişkili olduğunu savunmuşlardır. Çalışma sonucunda bilgisayar dersini alan 165 öğrenci üzerinde çalışma yapılmış ve bilgisayar dersi final sınavının %19'unu belirtilen bu faktörlerle açıklayabilmişlerdir. Bu etkinin ise ikinci dönem devam etmediği sonucuna ulaşmışlardır.

Naser (2008) A1-Azhar Üniversitesi'nde bir grup öğrenci üzerinde gerçekleştirdiği çalışmasında, görselleştirme araçlarının öğrencilerin performansına nasıl etki ettiğini araştırmıştır. Araştırmacı, çalışmasında geleneksel yöntem ile yeni ve eski versiyon görselleştirme araçları kullanılarak uygulanan farklı yöntemleri karşılaştırmıştır. Aslında yeni versiyon görselleştirme araçları eskilerine birkaç özellik eklenmesiyle oluşan görselleştirme araçlarıdır. Araştırmanın sonucunda şu bulgulara ulaşılmıştır; geleneksel yöntemlerdense, uygulanan yeni görselleştirme yöntemleri öğrenci performansını daha olumlu etkilemiştir. Ayrıca yeni versiyon görselleştirme araçları eski versiyon görselleştirme araçlarına oranla öğrencilerin başarısı üzerinde daha etkili olduğu görülmektedir. Benzer bir çalışmada Gülümbay (2006) yükseköğretimde bilgisayar web ortamında ders alan öğrenciler ile yüz yüze ders alan öğrencilerin bilgisayara yönelik kaygı durumları ile öğrenme stratejilerinin öğrenci başarısına etkisini incelemiştir. İki grubun bulunduğu çalışmada birinci grup yüz yüze

eğitim yapan 22 öğrenciden, ikinci grup ise bilgisayar web ortamında ders gören 31 öğrenciden oluşmaktadır. Çalışma sonunda başarıya ilişkin olarak yüz yüze eğitim yapan grubun lehine anlamlı bir fark oluşmuştur. Yüz yüze eğitim alanların son-test puanlarının bilgisayar web ortamında ders alanlara göre daha yüksek olduğu bulunmuştur.

Grant (2003) nesne tabanlı ve yapısal programlama öğretiminin öğrencilerin cinsiyetine, başarısına ve öğrenme stiline göre farklılık gösterip göstermediğini incelemiştir. Çalışmada iki grup üzerinde gerçekleştirilmiş yapısal dili 17, nesne tabanlı dili ise 24 kişi almıştır. Derslerden önce öğrencilerin öğrenme stilleri ve ön bilgileri belirlenmiştir. Dönem sonunda ise öğrencilerin programlama dersindeki notları alınmıştır. Çalışmanın sonucunda öğrencilerin programlama performansları ile öğrenme sitilleri ve cinsiyet arasında anlamlı bir ilişki bulunamamıştır. Benzer bir çalışmada Byrne ve Lyons (2001), programlama performansı ile bilgisayar ön deneyimleri, cinsiyet, akademik performans ve öğrenme stilleri arasında ilişki incelenmiştir. Bu çalışmada cinsiyet olarak erkek ve kadınlar arasında bir fark oluşmazken, programlama sonuçları arasında anlamlı bir fark oluştuğu görülmektedir. Çalışmanın bir diğer bulgusu ise öğrenme stilleri ile programlama performansı arasında anlamlı bir fark oluşmamasıdır. Bu çalışmalarla parelellik gösteren diğer bir çalışmada Goold ve Rimmer (2000) bilgisayar bölümünde okuyan 39 öğrenciyle 2 dönem süren çalışmalarında, bilgisayar derslerindeki performansın nelerden etkilendiğini belirlemeye çalışmışlardır. Öğrencilerle yapılan çalışmada öğrenme stilinin, öğrencilerin akademik başarısına ve performansına etkisinin olmadığı, motivasyona ise etkili olduğunu, ikinci dönemse bu etkilerin ortadan kalktığını tespit etmişlerdir.

Wilson ve Shrock (2001) toplamda 105 üniversite öğrencisi üzerinde yaptığı çalışmada programlamaya giriş ve programlama temelleri gibi ilk kez alınan programlama derslerinde hangi faktörlerin programlama başarısını etkilediğini incelemiş ve programlama performansı ile ilişkili değişkenleri tespit etmeye çalışmıştır. Çalışmanın sonucunda matematik başarı, matematik başarısızlık ve rahatlık seviyesi, programlama performansını etkileyen en önemli üç değişken olarak ortaya çıkarken, ön deneyim, cinsiyet ve çalışma stili gibi değişkenlerde anlamlı bir ilişki ortaya çıkmamıştır. Pillay ve Jugoo (2005) ilk olarak programlamaya başlayan öğrencilerin programlama konusunda zorluk yaşadıkları, yaşanan bu zorlukların programlama

performansını nasıl etkilediğini ortaya çıkarmak için yaptıkları çalışmada, iki ayrı üniversiteden toplam 97 öğrenci örneklem olarak alınmıştır. Çalışmada programlama performansını etkileyen faktörler, ön deneyimler, cinsiyet, problem çözme becerileri ve öğrenme stili olarak belirlenmiştir. Bu faktörlerden cinsiyet ve ön deneyimlerin programlama performansından bağımsız olduğu bulunurken, problem çözme becerisi ile anlamlı bir ilişki olduğu ve öğrenme stili ile ise üniversiteler arasında farklılıkların olduğu tespit edilmiştir. Bu çalışma ile paralellik gösteren Wiedenbeck (2005) çalışmasında zor olarak görülen programlamaya giriş derslerinin hangi faktörlerden dolayı zor olduğunu tespit etmek için bir model geliştirmiştir. Araştırmacı bu modelde bilgisayar programlama derslerinde başarıya etki eden faktörleri programlamaya yönelik ön deneyim, bilgi ve öz yeterlilik olarak tanımlamıştır. Çalışmanın örneklemini bilgisayar bölümünde okumayan 120 üniversite öğrencisi oluşturmaktadır. Çalışmada geçme notu programlama performansını yansıtmaktadır. Sonuç olarak ise öz yeterlilik algısı ve bilginin organizasyonu programlama performansını olumlu yönde etkilemiştir.

Milic (2009), gerçekleştirdiği çalışmasında programlamaya yönelik bilgiyi yüksek seviyede (programlama kodu üzerinde değişiklik yapabilme ve tekrar yazma) ve düşük seviye (sadece bir program kodunu okuyabilme) olarak iki seviyeye ayırmıştır. Bu iki seviyeyi göz önüne alarak bilgi seviyesi ile bireyin tutumu, bireyin zekâsı ve bireyin cinsiyeti arasındaki ilişkiyi incelemiştir. Çalışmaya katılan 79 öğrencinin programlama performansını belirlemek için düşük ve yüksek seviye programlama bilgisini ölçen iki ayrı test araştırmacı tarafından hazırlanmış ve kullanılmıştır. Çalışmanın en belirgin sonucu, yüksek seviyedeki programlama bilgisinin düşük seviyedeki programlama performansına bağlı olduğu bilgisidir. Bireyin bilgisayara karşı tutumu ve zekâsının yüksek seviyede programlama bilgisinin bir sonucu olduğu görülmektedir.Shute (1991) bireysel farklılıkların programlama performansına olan etkisini incelediği çalışmasında, çalışma belleği ile programlama performansı arasında anlamlı bir ilişki olduğunu tespit etmiştir. Sonuçlar, öğrenen davranışları, öğrenme ortamı ve öğrenme çıktılarına ilişkin bulguların birbiri ile etkileşim içinde olduğunu göstermiştir. Fincher vd. (2005), bilgisayar programlama derslerindeki programlama performansını belirlemeye yönelik yaptıkları çalışmada, 11 farklı kurumdan bilgisayar programlamaya giriş derslerindeki programlama performansını etkileyen faktörler

arasındaki ilişkileri incelemişlerdir. Araştırmacılar uzamsal görselleştirme bileşenleri ile programlama performansı arasında anlamlı bir ilişki olduğunu tespit etmişlerdir.

2.2.3. Bilgisayar programlama kaygısı ile ilgili araştırmalar

Rogerson ve Scott (2010:147-171) programlama kaygısına yönelik öğrenci görüşlerinin belirlenmesi için yaptıkları araştırmada, ilk ve en genel bulguları, öğrencilerin gerçekten programlama öğrenmeyle ilgili sorunlarının olduğudur. Görüşme esnasında katılımcıları en çok "*programlama*" kelimesini kullandıkları ve bu kelimeden rahatsızlık duydukları görülmekte, ayrıca yine katılımcıların görüşme esnasında kaygı, panik, kâbus ve stres gibi sözcükleri tekrar tekrar kullanmışlardır. Örneğin, *"Bu… bu tür beni çok vurguluyor cehennem gibi korkutucu"; "eğer ben programlama dersini alıyorsam benimle programlama arasında sorun var gibi hissediyorum".* Bazı katılımcılar programlama ile hayal kırıklığı yaşadıklarını anlatmaktadırlar. Örneğin, " *Ben, programlama ile benim özel bir problemim olduğunu düşünüyorum, beni çok kaygılandırıyor, bu da büyük bir sorun".* Hatta bazı katılımcılar programlama hakkındaki duygularını ve öğrenme deneyimlerini şu şekilde anlatmışlardır: " *… iyi biz gidip sadece atölyede zorla oturalım"* gibi anlatımlarda *"zorla "* kelimesini açıkça kullandıklarını, vurguladıkları ve isteksiz olduklarını belirten ifadeler kullandıklarına araştırma sonucunda ulaşılmıştır.

Fesakis ve Serafeim (2009) bilgisayar öğretiminde bilgi iletişim teknolojilerine karşı öğrencilerin tutumları ile ilgili olarak yürüttükleri araştırmalarında, öğrencilerin bilgisayar programlama öğretiminde görsel programlama ortamına alıştıkları görülmüştür. Bilgisayar öğretiminde öğrencilerin bilgi iletişim teknolojilerinden yararlanmalarını artırmanın ve uygulamaların bu ortamlarla yapılmasını sağlamanın stres ve kaygı yüzdesini azalttığı görülmektedir. Görsel programlam aracının kaygıya etkisinin araştırıldığı benzer bir çalışmada Bishop-Clark, Courte ve Howard (2007) yaptıkları çalışmada, programlama dersinde görsel programlama aracı kullanmışlardır. Araştırmaya katılan öğrenciler 2,5 hafta boyunca programlama dersinde görsel programlama aracı ile gerçekleştirilen programlama uygulamalarına katılmışlardır. Araştırma sonucunda görsel programlama aracı kullanan öğrencilerin programlama yapılarını ve kavramlarını daha iyi anladığını ve programlamaya olan güvenlerinin arttığını ve kaygılarının azaldığını belirtmişlerdir. Lisans düzeyinde yapılan bir

çalışmada öğrenciler, programlama aracı kullanılmadan ders işlenen grup ve programlama aracı kullanılarak ders işlenen grup olmak üzere iki gruba ayrılmışlardır. Bu gruplara bilgisayara yönelik kaygı testi ön-test ve son-test olarak uygulanmıştır. Sonuçlar programlama aracı kullanılan grubun kaygı düzeyinin, araç kullanılmayan gruba göre daha düşük olduğunu göstermiştir. Dolayısıyla programlama aracının derste kullanımı bilgisayar ve programlama kaygısını azaltmaktadır (Leso ve Peck, 1992).

Bilgisayar eğitiminin bilgisayara yönelik kaygıyı azalttığını belirten alanyazında birçok araştırma bulunmaktadır (Ayersman ve Reed, 1996; Bradley ve Russell, 1997; Häkkinen, 1994; Martocchio, 1992; Maurer ve Simonsen, 1993; Russell ve Bradley, 1996). Bu sonuçlarla birlikte bilgisayar eğitiminin tek başına bilgisayara yönelik kaygı düzeyini düşürmede yeterli olmadığı görülmektedir (McInerney, McInerney ve Sinclair, 1994). Gürcan, Namlu ve Ceyhan (2003) yaptıkları çalışmada, incelemiş oldukları alanyazında şu yönde bulgulara rastlamışlardır. Verilen bilgisayar eğitiminin bilgisayar kaygısını azaltıp azaltmayacağı, bilgisayar eğitiminin türüne göre değişiklikler göstermektedir. Bazı araştırmalar uygulanan yöntemlerin kaygıyı azalttığını gösterirken bazı araştırmalar bunun tam tersini belirtmektedir. Bilgisayar öğrenme stratejilerini geliştirmek için yapılan eğitim ve öğretimin, bilgisayara yönelik kaygıyı azaltmadaki etkisini ortaya koymak amacıyla Namlu, (2003) tarafından yapılan çalışmaya, bilgisayar programcılığı kursunu alan 37 öğrenci katılmış ve kaygıyı ortaya koymak için araştırmada "Bilgisayar Kaygı Ölçeği" kullanılmıştır. Deney ve kontrol gruplu çalışma sonucunda, öğretim stratejilerini geliştirmek için verilen eğitimin, deney grubunda bulunan öğrencilerin bilgisayara yönelik kaygı düzeylerinde azalma olduğu bulunmuştur.

Chua, Chen ve Wong (1999), 1990-1996 yılları arasında, bilgisayara yönelik kaygı ile bilgisayar deneyimi, cinsiyet ve yaş arasındaki ilişkiyi inceleyen, yayınlanmış araştırmaları kapsayan bir meta-analiz çalışması yapmışlardır. Çalışma sonucunda cinsiyet olarak kızların erkeklere göre bilgisayara yönelik kaygılarının daha fazla olduğu, fakat kesin olmadığı, bilgisayar deneyimi ile bilgisayara yönelik kaygı arasında ise negatif yönlü bir ilişki olduğu tespit edilmiştir. Benzer bir çalışmada Bozionelos (2001)'un bilgisayara yönelik kaygı ile yaş değişkeni arasındaki ilişkiyi incelediği araştırmasında, genç olan bireylerin yaşlı olan bireylere oranla bilgisayara yönelik kaygıyı daha fazla yaşadıkları tespit edilmiştir. Bu bulguların yanı sıra, üniversite

öğrencilerinin %40'ından daha fazlasının bilgisayara yönelik olarak kaygı belirtisi gösterdiği belirtilmiştir. Tzavara ve Komis (2003) bilgisayar programlama kurslarının öğrencilerin kaygı ve stres seviyesini nasıl etkilediğine yönelik yaptıkları araştırmalarında, kursa katılan kız öğrencilerin erkek öğrencilere göre daha yüksek kaygı seviyesine ve stres düzeyine sahip oldukları ve ilgili kurslara katılımın, öğrencilerin stres seviyelerini aşağı çekmede yardımcı olacağını belirtmektedirler. Benzer bir araştırmada McInerney (1990), programlama dersini alan ve sorunsuz bir şekilde tamamlayan öğrencilerin kaygı düzeylerinin, diğer öğrencilere göre daha düşük olduğu görülmektedir. Ayrıca dersi tamamlayan öğrencilerinden erkek olanların kaygılarının kız olanlara göre daha düşük olduğu, bilgisayarı sadece eğlence amaçlı kullananların kaygılarının, bir ders için veya herhangi bir amaç için kullanmak zorunda kalanlardan daha düşük olduğu bulunmuştur.

Namlu ve Ceyhan (2002)'ın "Bilgisayar Kaygısı: Üniversite Öğrencileri Üzerinde Bir Çalışma" isimli çalışmada, bilgisayara yönelik kaygı ile cinsiyet, bilgisayar tecrübesi, bilgisayar dersi alma sayısı ve bilgisayarı olma durumu arasındaki ilişkinin incelenmiştir. Çalışmada kızların kaygısının erkeklere göre daha fazla olduğu, bilgisayar tecrübesi az olanların, bilgisayar dersi az alanların ve bilgisayarı olmayanların bilgisayara yönelik kaygılarının yüksek olduğu bulunmuştur. Namlu ve Ceyhan (2002)'ın çalışmasına paralellik gösteren bilgisayara yönelik kaygı ile ilgili bu çalışma, cinsiyet, bilgisayar kullanımı, durumsal ve sürekli kaygı değişkenleri incelenmiştir. Bilgisayara yönelik kaygı ile cinsiyet, durumsal kaygı ve bilgisayar kullanımı değişkenleri arasında anlamlı bir ilişkinin bulunmadığı, fakat sürekli kaygı ile bilgisayara yönelik kaygı arasında anlamlı bir ilişki tespit edilmiştir (Goldstein, Dudley, Erickson ve Richer, 2002).

Connolly, Murphy ve Moore (2007) yaptıkları çalışmada, öğrencilerin özellikle bilgisayar programcılığına giriş dersindeki soyut ve karmaşık yapıları anlamakta ve bu soyut yapıları somutlaştırmakta güçlük çektikleri, zorlandıkları ve stres ve kaygı yaşadıkları görülmektedir. Beckers, Wicherts ve Schmidt (2007)'in yaptıkları çalışmada, "bilgisayar kaygısı sabit bir karakteristik (kişilik özelliği) mi, yoksa değişebilir, geçici olabilen bir durum mu?" sorusuna cevap aramışlardır. Çalışmanın sonucunda bilgisayara yönelik kaygının durumsal (geçici, anlık) kaygıdan daha çok sürekli (bireyin kişisel özellikleri ile ilgili) kaygıyla ilişkili olduğu sunucuna

ulaşmışlardır. Benzer bir çalışmada Mahor, Henderson ve Deane (1997) yaptıkları araştırmada, bilgisayara yönelik kaygı ile durumsal kaygı arasında olumlu bir ilişki olduğunu, bilgisayar deneyimi ile bilgisayara yönelik kaygı arasında ise ters yönlü bir ilişki olduğunu tespit etmişlerdir.

Üniversite öğrencileri üzerinde yapılan bir araştırmada, öğrencilerin bilgisayar kullanım saatleri arttıkça buna bağlı olarak bilgisayar deneyimlerinin arttığı ve bunun sonucunda bilgisayara yönelik kaygının azaldığı görülmektedir (Necessary ve Parish, 1996). Bu çalışmaya paralel olarak öğretmen adaylarına yönelik yapılan başka bir çalışmada, bilgisayar kullanım sıklığının artması bilgisayara yönelik kaygıyı azalttığı görülmektedir (Ropp, 1999). Bilgisayar kaygısı ile performans arasındaki ilişkinin incelendiği benzer bir çalışmada, bilgisayara yönelik kaygının, bilgisayarla yoğun bir öğrenme ortamına maruz kalındığında değişikliğe uğradığını göstermektedir. Bilgisayar öğretim ortamında bireyin düşük kaygısı, yüksek kaygıya doğru giderse performansı gelişir, bireyin yüksek kaygısı iyice yükselirse performans bundan negatif yönde etkilenmekte ve düşmesine sebep olmaktadır (Buche, Davis ve Vician, 2007).

Yükseköğretim öğrencileri üzerinde yapılan bir çalışmada, öğrencilerin bilgisayara yönelik kaygıları ile bilgisayar okuryazarlığı arasındaki ilişki araştırılmıştır. Araştırma sonucunda bilgisayar okuryazarlığı ile bilgisayara yönelik kaygı arasında ters bir orantı olduğu görülmüştür. Bilgisayar kullanımı arttıkça bilgisayar okuryazarlığının da artacağı ve bilgisayara yönelik kaygınında böylelikle azalacağı belirtilmiştir (Akhu-Zaheya, Khater, Nasar ve Khraisat, 2012). Benzer bir çalışmada bilgisayar okuryazarlığı ile öğrencilerin öğrenme stratejileri incelenmiştir. Bilgisayar dersini alan 75 Tayvanlı öğrenci üzerinde gerçekleştirilen bu çalışmada, onların anlama, izleme gibi üst bilişsel becerileri ile öğrenme için yararlı kaynaklar kullanılarak başarıları ortaya çıkarılmıştır. Ayrıca iyi bir bilgisayar tutumları ve alt seviye kaygı düzeyleri, bu başarıda etken olmuştur (Tsai ve Tsai 2003).

Öğretmenler teknoloji eğitimi alma durumlarının değişken olarak alındığı bu çalışmada, teknoloji entegrasyonu eğitim almış öğretmenlerin bilgisayara yönelik kaygı düzeyleri, eğitim almamış öğretmenlere göre daha düşük olduğu görülmektedir. Öğrencilerin bilgisayara yönelik kaygılarının da bu durumdan olumlu etkilendiği görülmektedir. Teknoloji eğitimi almış öğretmenin, ders verdiği öğrencilerin bilgisayara

yönelik kaygılarının, teknoloji dersi almamış öğretmenin ders verdiği öğrencilerin kaygılarından daha düşük olduğu görülmektedir (Christensen, 2002).

ÜÇÜNCÜ BÖLÜM

3. YÖNTEM

Bu bölümde; öncelikli olarak araştırmanın tasarımı ve sonrasında araştırma sorularını cevaplandırmak için takip edilen adımlar detaylı olarak ele alınmıştır. Bu kapsamda, araştırmanın modeli, araştırma grubu, araştırma süreci, araştırmada kullanılan veri toplama araçları ve elde edilen verilerin analizinde kullanılan istatistiksel yöntem ve teknikler açıklanmıştır.

3.1. Araştırmanın Modeli

Bu çalışma bir devlet üniversitesinin bilgisayar programcılığı bölümü 1. sınıf öğrencileri üzerinde, programlama temelleri dersi kapsamında gerçekleştirilmiştir. Yapılan çalışmada eğitsel programlama dilinin dersin teori, uygulama ve hem teori hem de uygulama bölümlerine entegre edilmesinin başarıyı, performansı ve bilgisayar programlama kaygısını nasıl etkilediğinin belirlenmesi amaçlanmıştır. Bu amaç doğrultusunda çalışmada kontrol grupsuz ön-test son-test deneysel model kullanılmıştır (Büyüköztürk vd., 2008).

3.1.1. Araştırmanın deneysel modeli

Araştırmada nicel verilerin toplanmasında, değişkenler arasındaki neden-sonuç ilişkilerini ortaya çıkarmak için gözlenmek istenen verilerin üretildiği araştırma alanı olarak da adlandırılan deneysel desenden yararlanılmıştır (Büyüköztürk, 2001; Karasar, 2004). Üç deney grubunun bulunduğu çalışmada, deneysel desenlerden kontrol grupsuz ön-test son-test Tablo 3.1 modeli kullanılmıştır. Bu modelde deneysel işlemin etkisi tek bir grup üzerinde gerçekleştirilen çalışmayla test edilmektedir. Çalışmaya katılanların bağımlı değişkene bağlı ölçümleri, uygulama öncesinde ön-test, uygulama sonrasında son-test olarak aynı katılımcılar ve aynı ölçme araçları değiştirilmeden kullanılarak elde edilir. Seçkisizlik ve eşleştirme olmaması yönüyle desen tek faktörlü gruplariçi ya da tekrarlı ölçümler deseni olarak da adlandırılabilir. Deneysel desende tek gruba (G) ait

ön-test ve son-test değerleri arasındaki farkın (O1 - O2) anlamlılık boyutu test edilir (Büyüköztürk vd., 2008).

Tablo 3.1.

Araştırmanın Deneysel Desen Tasarımı

Grup	Ön-Test	İşlem	Son-Test
G_1	O_1	X_1	O_2
G_2	O_1	X_2	O_2
G_3	O_1	X_3	O_2

G, işlem yapılan grubu; O1, deney grubundan alınan ön ölçümü; X, deneysel işlemi; O2 ise deney grubundan alınan son ölçümü göstermektedir (Büyüköztürk vd., 2008).

Kontrol grupsuz ön-test son-test deney deseninin kullanıldığı araştırmada önce ön bilgi testi yapılarak, öğrencilerin programlama ile ilgili ön bilgileri ölçülmüştür. Uygulama yapıldıktan sonra ise çalışmada eğitsel programlama dilinin derse entegrasyon şeklinin öğrencilerin akademik başarılarına olan etkisini karşılaştırabilmek amacıyla son bilgi testi yapılarak başarılar arasında anlamlı bir farkın olup olmadığı tespit edilmiştir.

3.2. Araştırma Grupları

Çalışma grupları, bir devlet üniversitesinin bilgisayar programlama bölümünde öğrenim gören ve programlama temelleri dersini alan 61 erkek ve 26 kız olmak üzere 87 öğrenciden oluşmaktadır. Uygulama öncesinde kolay ulaşılabilir amaca uygun örnekleme yöntemi kullanılmış ve çalışma grubuplarında yer alan öğrenciler rastgele bir şekilde atanmıştır (Yıldırım ve Şimşek, 2006). Grupların oluşturulmasında programlama temelleri dersinin uygulama kısmına, eğitsel programlama dilinin entegre edilmesi ile oluşan gruba uygulama entegre grup (UEG), dersin teori kısmına entegre edilmesi ile oluşan gruba teori entegre grup (TEG) ve dersin hem teori hem de

uygulama kısmına entegre edilmesi ile oluşan gruba ise hem teori hem de uygulama entegre grup (TUEG) olarak adlandırılmıştır. Bu grupların demografik bilgileri Tablo 3.2'de verilmiştir.

Tablo 3.2.

Çalışma Grubundaki Öğrencilere İlişkin Demografik Bilgiler

Grup	Cinsiyet		Yaş				MYO'ya Girişi		Mezun Olduğu Lise			
	Erkek	Kız	18	19	20	21	Sınavlı	Sınavsız	Düz Lise	Meslek Lisesi	Anadolu Lisesi	Diğer
UEG	20	10	8	10	6	6	11	19	2	19	3	6
TEG	16	12	7	10	10	1	9	19		19	5	4
TUEG	25	4	5	9	9	6	10	19		19	1	9
Toplam	61	26	20	29	25	13	30	57	2	57	9	18

Çalışma grupları Tablo 3.3, Tablo 3.4 ve Tablo 3.5'de eğitsel programlama dilinin dersin teori veya uygulama kısımlarına entegrasyonuna göre isimlendirilmiştir.

Tablo 3.3.

Uygulama Entegre Grup (UEG)

Dersin Teori Bölümü (2 Saat)		Dersin Uygulama Bölümü (2 Saat)	
1 Saat	1 Saat	1 Saat	1 Saat
Teorik Anlatım	Teorik Anlatım	**Eğitsel Programlama Dili**	Editöre Yazma (Söz Dizimi)

Tablo 3.4.

Teori Entegre Grup (TEG)

Dersin Teori Bölümü (2 Saat)		Dersin Uygulama Bölümü (2 Saat)	
1 Saat	1 Saat	1 Saat	1 Saat
Teorik Anlatım	**Eğitsel Programlama Dili**	Editöre Yazma (Söz Dizimi)	Editöre Yazma (Söz Dizimi)

Tablo 3.5.

Teori ve Uygulama Entegre Grup

Dersin Teori Bölümü (2 Saat)		Dersin Uygulama Bölümü (2 Saat)	
1 Saat	1 Saat	1 Saat	1 Saat
Teorik Anlatım	**Eğitsel Programlama Dili**	**Eğitsel Programlama Dili**	Editöre Yazma (Söz Dizimi)

3.2.1. Araştırma gruplarının denkliği

Deneysel desenli çalışmalarda grupların denk olması çalışmadan daha sağlıklı veriler alınmasını sağlamaktadır. Özellikle de ön-test son-test olarak karşılaştırmalı desenlerde grupların denk olması daha da önem arz etmektedir. Eğer deneysel çalışmaya başlamadan önce grupların denklikleri kontrol edilmezse çalışmanın sonucunda ortaya çıkacak farklılıklar aslında deney öncesi farklılıkların sonucu olabilir. Bu durum araştırmacılar tarafından göz önüne alınarak değerlendirilmeli ve sonuçların kuşkulu olmasını önlemek adına deneysel işleme başlamadan önce grupların denklikleri araştırılmalıdır (Erden, 1998).

Bu nedenle deney gruplarındaki öğrencilerin bağımsız değişken hariç diğer değişkenler açısından denkleştirilmesi gerektiğinden araştırmadaki bağımlı değişkenlerin gruplarda kontrol altına alınması gerekmektedir. Değişkenleri kontrol altına almadaki amaç iç geçerliliği artırmak ve elde edilecek sonucun sadece denenen bağımsız değişkenden kaynaklanma ihtimalini artırmaktır (Karasar, 2005). Dolayısıyla deney gruplarına etki edebilecek diğer değişkenlerin kontrol altına alınması ile ortaya çıkacak sonucu sadece eğitsel programlama dilinin derse entegre edilme şekli etkileyecektir.

Denkleştirme bağımsız değişken dışındaki kontrol altına alınabilen diğer değişkenlerin birbiriyle denkleştirilmesidir. Bu yönde ilk olarak çalışmada öğrencilerin programlamaya giriş dersi arasınav notlarının denk olup olmadığı Tablo 3.6’ da incelenmiştir.

Tablo 3.6.

Öğrencilerin ProgramlamaTemelleri Dersi Ara sınav Puanlarına Göre Denklik Durumu

	N	Ortalama ($\overline{X}$)	Standart Sapma	df	F	Sig.
UEG	30	42.13	17.31	2	.495	.612
TEG	28	38.03	20.50			
TUEG	29	42.31	17.15			

p< .05

Tablo 3.6'da görüldüğü gibi UEG grubu öğrencilerinin programlamaya giriş dersi arasınav dönem ortalamaları ($\overline{X}$ =42.13), TEG grubu öğrencilerinin programlamaya giriş dersi arasınav dönem ortalamaları ($\overline{X}$ =38.03) ve TUEG grubu öğrencilerinin programlamaya giriş dersi arasınav dönem ortalamaları ($\overline{X}$ =42.31) olarak bulunmuştur. Ortalamalar arasında fark görülmesine rağmen, gruplar arasındaki bu farkın anlamlı olup olmadığına belirlemek amacı ile tek yönlü varyans analizi (ANOVA) yapılmış ve farkın olmadığı [$t_{(2)}$=.495, p>.05] görülmüştür. Bu sonuç gruplar arasında ara sınav not ortalaması açısından denk olduğunun göstergesidir.

Diğer bir inceleme ise programlama temelleri dersi kapsamında hazırlanan akademik başarı testinin gruplar arasında ön-test puanları açısından denklikleridir. Bu denklik Tablo 3.7'de gösterilmektedir.

Tablo 3.7.

Öğrencileri Programlama Temelleri Dersi Akademik Başarı Öntest Puanlarına Göre Denklik Durumu

	N	Ortalama ($\overline{X}$)	Standart Sapma	df	F	Sig.
UEG	30	51.80	10.23	2	.022	.979
TEG	28	52.36	11.74			
TUEG	29	52.13	11.27			

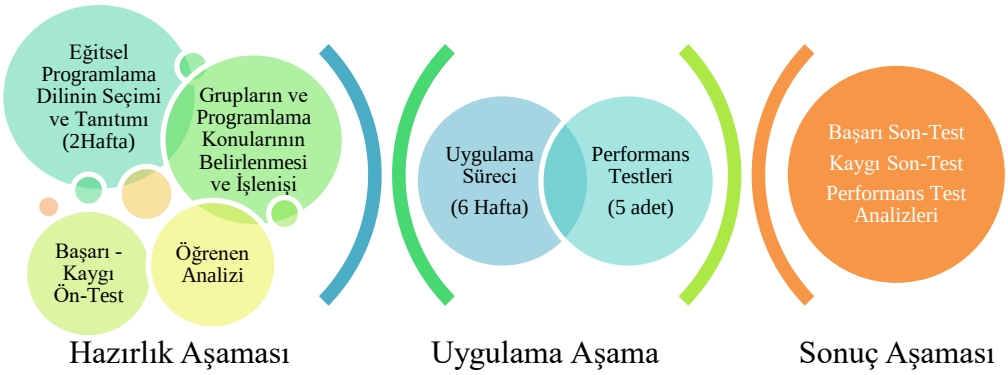
p< .05

UEG grubu öğrencilerinin programlamaya giriş dersi akademik başarı ön-test puan ortalamaları ($\overline{X}$ =51.80), TEG grubu öğrencilerinin programlamaya giriş dersi akademik başarı ön-test puan ortalamaları ($\overline{X}$ =52.36) ve TUEG grubu öğrencilerinin programlamaya giriş dersi akademik başarı ön-test puan ortalamaları ($\overline{X}$ =52.13) olduğu Tablo 3.7'de görülmektedir. Ortalamalar arasında fark görülmesine rağmen, gruplar arasındaki bu farkın anlamlı olup olmadığına belirlemek için Tek yönlü varyans analizi (ANOVA) yapılmış ve fark olmadığı ortaya çıkmıştır [$t_{(2)}$=.022, p>.05]. Bu sonuç gruplar arasında akademik başarı ön-test puan ortalaması açısından denk olduğunu belirtmektedir.

Grupların denkliğini kontrol etmek için yapılan akademik başarı ön-test puan ortalamaları ile ara sınav ortalamaları incelendiğinde grupların denk olduğu söylenebilir. Ayrıca yaş olarak aynı sınıfta olduklarından, mezuniyet olarak meslek lisesi çıkışlıların gruplarda ağırlıklı olmasından, meslek yüksekokuluna giriş şekli (sınavlı-sınavsız) ve derse giren eğitimcinin aynı olması bakımından deneysel işlem öncesi grupların bağımlı değişkenler açısından denk olduğu söylenebilir.

3.3. Araştırma Süreci

Arasınavlar sonrasında gerçekleştirilen araştırma süreci, ASSURE öğretim tasarım modeline göre tasarlanmıştır. Assure modelinin öğrenenlerin analizi, hedeflerin belirlenmesi, medya ve materyalin seçimi, medya ve materyalin kullanımı, öğrenen katılımında uygulamanın gerçekleştirilmesi ve elde edilen verilerin değerlendirilmesinin yapıldığı aşamalar, araştırma süreci içerisinde bulunan hazırlık, uygulama ve sonuç aşmaları içerisinde ele alınmıştır. Bu bağlamda öğrenenlerin analizi, dersin konu, hedef ve içeriğinin belirlenmesi ve eğitsel programlama dilinin seçimi hazırlık aşamasında gerçekleştirilmiştir. Eğitsel programlama dilinin kullanımı ve öğrencilerin katılımının sağlandığı uygulama sürecininde içinde bulunduğu aşama uygulama aşamasını oluşturmaktadır. Elde edilen sonuçların değerlendirilmesi ise sonuç aşamasında gerçekleştirilmiştir. Bu süreç Şekil 3.1'de şematik olarak gösterilmiştir.



Şekil 3.1. Araştırma süreç modeli

3.3.1. Hazırlık aşaması

Öğrenen analizinin yapılması, programlama konularının belirlenmesi, akademik başarı testinin oluşturulması ve çalışmaya başlanmadan önce ön-test olarak uygulanması, bilgisayar programlamaya yönelik kaygı ölçeğinin hazırlanması ve uygulama öncesi ön-test olarak uygulanması, eğitsel programlama dili aracının kullanıcılara tanıtımı, grupların oluşturulması ve öğretim amacının belirlenmesi ile eğitsel programlama dili aracının kullanımına kadar geçen süreç hazırlık aşaması olarak belirlenmiştir.

Hazırlık aşamasında eğitsel programlama dilinin tanıtımı ve öğrencilerin bilgilendirilmesi amacı ile 2 hafta, (haftada 4 ders saatinden toplam 8 ders saati) eğitsel programlama dili hakkında öğrencilere detaylı bir şekilde bilgi verilmiştir. Bu süre içerisinde öğrencilere programlama öğretimi için temel teşkil eden, öğrencilerin öğrenmekte en çok zorluk çektiği ve eğitsel programlama dili ile uygulanabilir olan konular, 4 bilgisayar uzmanının görüşü alınarak ve alanyazın taraması yapılarak belirlenmiştir. Bu konuların belirlenmesinden sonra ayrıca konuların programlama içerisindeki ağırlıkları, Bloom taksonomisi bilişsel düzey tasarım basamakları (Hatırlama, Anlama, Uygulama, Çözümleme, Değerlendirme, Yaratma) göz önüne alınarak, bilgisayar uzmanları ile beraber, öğrenci başarısını belirlemeye yönelik akademik başarı testi hazırlanmıştır (Ek 1). Hazırlık aşmasında yapılan diğer bir çalışma ise Cheung, vd. (1990)'nin çalışmasındaki Bilgisayar Programlama Kaygı Ölçeği (computer programming anxiety scale) ölçeği 116 deneğe uygulanmış ve elde edilen

veriler ile testin geçerlilik ve güvenilirlik çalışması yapılarak Türkçeye uyarlanması gerçekleştirilmiştir. Bu testin Türkçeye uyarlaması hakkında ayrıntılı bilgi, veri toplama araçları bölümünde verilmiş ve testin orjinali ise (Ek 2)'de yer almaktadır. Hazırlık aşamasında bu işlemlerin ardından grupların belirlenmesine yönelik olarak çalışmaya katılacak öğrencilerin tamamının isim listesi bilgisayarda yazılmıştır. Listeden isimler tek, tek kesilerek katlanıp bir kutuya konulmuştur. Belirlenecek üç grup için birer kutu hazırlanmış ve grupların ismi (A, B ve C) olarak kutuların üzerine yazılmıştır. Olasılıklı örneklem yöntemlerinden her bireyin örnekleme seçilme şansının eşit olduğu basit tesadüfi örnekleme (Arıkan, 2004) yöntemi ile rastgele çekilen isimler sırayla kutulara atılmıştır. Kutuların hangi gruba ait olduğunu belirlemek için ise yine rastgele örneklem yöntemi kullanılmıştır. Bu yöntem sonucunda A kutusu için eğitsel programlama dilinin uygulamaya entegre edildiği grup, B kutusu için eğitsel programlama dilinin teori kısmına entegre edildiği grup ve C kutusu için ise eğitsel programlama dilinin hem teori hem de uygulamaya entegre edildiği grup olarak kurada çıkmış ve bu yönde gruplar oluşturulmuştur.

3.3.1.1. Öğrenen analizi

Hazırlık aşamasında ilk olarak bigisayar programlama bölümündeki 30 öğrencinin oluşturduğu bir gruba programlama önbilgilerini, programlamaya bakış açılarını ve programlama konularının bilinirliklerini belirlemeye yönelik öğrenen analiz çalışması gerçekleştirilmiştir. Bu aşamada uygulamanın yapılacağı bölümde bulunan 4 bilgisayar uzmanı ile programlama genel konularını içeren programlama soruları oluşturulmuştur. Hazırlanan sorular bilgisayar başında belirlenen 30 öğrenciye basitten zora doğru bir sıralama ile sorulmuştur. Bu sorular şu şekilde sıralanmaktadır: Ekranda gördüğünüz satırlar size neyi anımsatmaktadır? Program içerisinde bir değişken tanımlaması gösterirmisiniz? Bu programda bir döngü ifadesi varmıdır, varsa bize nasıl işlediğini anlatırmısınız? Bu program içerisinde bir koşul ifadesi varmıdır, varsa nasıl bir işlem yaptığını anlatırmısınız? Gördüğünüz bu program kodu nasıl bir işlem yapıyor?, Bu programın ekran çıktısını bize söyleyebilirmisin?, Sorulara verilen cevaplarda 8 öğrencinin ilk soruda takıldığı ve bu yüzden programlama önbilgilerinin eksik olduğu görülmüştür. Öğrencilerin 11'i ise ikinci ve üçüncü sorulara kadar cevap verebildiği, değişkenin ve döngü yapılarının tanınabildiği fakat döngü ifadesinin nasıl

işlediğinin çözülemediği belirlenmiştir. Altı öğrencinin koşul ifadesine kadar gelebildiği ancak koşul ifadesinin program içerisinde nasıl işlem yaptığını belirleyemediği ve 5 öğrencinin ise program kodunun nasıl bir işlem yaptığını belirleyebildiği ve içlerinden 3 öğrencinin programın ekran çıktısını söyleyebildiğini, görülmektedir. Bu cevaplar soruları hazırlayan bilgisayar uzmanları tarafından analiz edildiğinde, programlama dersleri için verilen programlama öğretiminin hedefine tam manası ile ulaşmadığını, eksiklerinin bulunduğunu ve bu eksikliklerin giderilebilmesi içinde alternatif yöntemlere ihtiyaç olduğunu belirtmişlerdir (Arabacıoğlu, Bülbül ve Filiz, 2007).

3.3.1.2. Çalışma konularının belirlenmesi ve işlenişi

Uygulamaya başlamadan önce Bilgisayar Programlama bölümü 1. Sınıf öğrencilerinin programlama temelleri dersinde hangi konuların verilmesinin programlama kavramlarının öğrenilmesinde daha etkin olacağının belirlenmesine yönelik Milli Eğitim Bakanlığının bilgisayar programlama ders müfredatı 4 bilgisayar uzmanı tarafından incelenmiş ve müfredata uyumlu olarak çalışma konuları belirlenmştir. Hangi konuların programlama yapısını anlamada daha etkin olacağına yönelik alınan uzman görüşleri çerçevesinde temel kavramlar, değişken yaklaşımı, giriş çıkış deyimleri, döngü yapısı, matematiksel ve mantıksal operatörler ve koşul yapısı kavramlarının programlama yapısını öğrenmede bilinmesi gereken en öncelikli kavramlar olduğu üzerinde fikir birliğine varılmıştır. Bu konuların önceliklerinin belirlenmesi, öğrencilerin programlama konularını öğrenirken hangi konuda zorluk yaşadıkları ve bu zorukların ortaya konulduğu araştırmalar, alanyazında bulunmaktadır (Çetin, 2013). Bu araştırmalardan bir tanesi Pea (1986) yaptığı çalışmadır. Bu çalışmada programlama konularında döngüler konusunda öğrencilerin döngü yapısını kavrayamadığını ve döngü içinde kalan ifadeleri döngü dışında gördükleri gözlemlemiştir. DuBoulay (1986) yaptığı diğer bir çalışmada, programlamaya yeni başlayanların, değişken ve kontrol yapıları konuları ile kontrol değişkeninin döngünün her dönüşünde yenilenmesinde, nasıl arttığını anlamada zorluk çettiklerini belirtmektedir. Aynı şekilde Sleeman, Putnam, Baxter ve Kuspa (1986), öğrencilerin kontrol değişkeninin döngü yapısı içerisinde nasıl değer aldığını belirleyemediklerini ifade etmişlerdir.

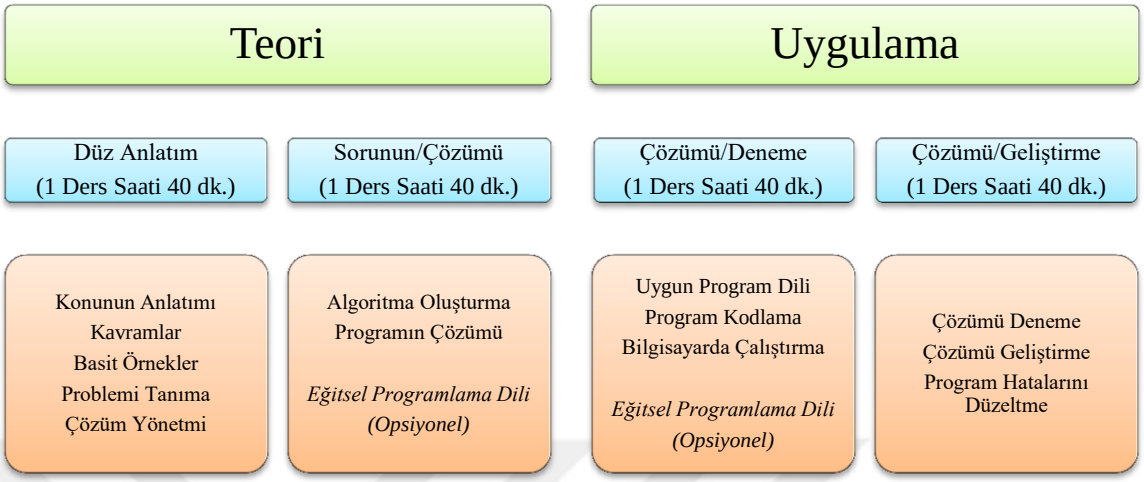
Bilgisayar uzmanı olan öğretim elemanlarının görüşleri ve alanyazındaki çalışmalar incelendiğinde, özellikle koşul ve döngü ifadeleri başta olmak üzere temel kavramlar, değişken yaklaşımı, giriş çıkış deyimleri ve operatörler konularının programlama temelleri dersi için öğrencilerin öğrenmede zorluk çektiği konuların başında geldiği görülmektedir. Ayrıca programlama için temel teşkil eden "temel kavramlar, değişken yaklaşımı, giriş çıkış deyimleri, döngü yapısı, matematiksel ve mantıksal operatörler ve koşul yapısı kavramları" konuları eğitsel programlama dilleri ve öğretim materyalleri ile tasarlanabilir olması konularının belirlenmesinde diğer bir belirleyici etken olmuştur.

Hazırlık aşamasında uygulama için programlama konularının içerisinden "temel kavramlar, değişken yaklaşımı, giriş çıkış deyimleri, döngü yapısı, matematiksel ve mantıksal operatörler ve koşul yapısı" başlıkları belirlenmiştir. Başlıklara karar verildikten sonra, konu içeriklerine uygun eğitsel programlama dilinin kullanıldığı, programlama aşamalarının göz önüne alındığı ders tasarımı gerçekleştirilmiştir. Ders tasarımında grupların uygulama ve teori saatlerine ve verilecek örneklerin her grup için aynı olması özen gösterilmiştir. Ayrıca ders tasarımındaki ders planı 6 haftalık ve haftada 2 saat teori ve 2 saat uygulama olmak üzere 4 ders saatini kapsamaktadır. Derslerin tasarlanmasında programlama yazımında ve problemlere programlar ile çözüm üretme konularında programcılar arasında farklı yaklaşımlar izlendiği görülse de hemen hemen bütün programcılar aşağıdaki programlama aşamalarını izlemektedirler (Gümüşay,2012).

1) Problemi tanımlama.
2) Çözüm yöntemi geliştirme.
3) Algoritma geliştirme.
4) Çözümü kâğıt üzerinde gösterme.
5) Uygun programlama dilinin seçimi.
6) Programın kodlanması.
7) Bilgisayarda programın çalıştırılıp denenmesi.
8) Programı geliştirme.

Çalışmada haftalık ders tasarımı oluşturulurken hemen hemen bütün programcıların uyguladıkları programlama aşamaları kullanılmıştır. Bu aşamalar ve alan

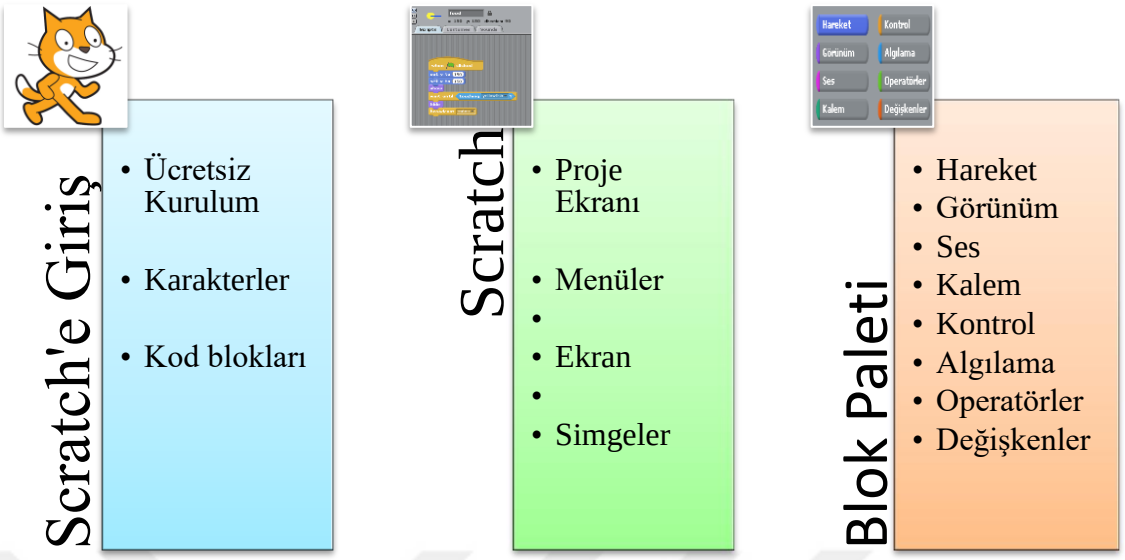
uzmanlarının görüşleri çercevesinde Şekil 3.2.'deki ders tasarımı oluşturulmuş ve haftalık bir konuyu kapsayan ders tasarımı ekler bölümünde (Ek 3) verilmiştir.



Şekil 3.2. Haftalık ders tasarımı

3.3.1.3. Eğitsel programlama dilinin seçimi ve tanıtımı

Belirlenen konuların eğitsel programlama dili ile tasarıma uygun olması, programlama yapılarını (değişken, kontrol, operatörler ve giriş çıkış deyimleri) içinde barındırması ve programlama hatalarına izin vermeyen bir yapıda olması belirleyici etken olmuştur. Ayrıca kullanılacak eğitsel programlama dilinin belirlenmesinde, programlama öğrenmeyi kolaylaştıracak ve programlamada her yaş grubuna hitap etmesi, başka bir belirleyici etken olmuştur. Etkileşime açık, ilgi çekici, yaratıcı düşünmeyi destekleyen, sistematik analiz yapılabilen, işbirliği sağlayan, etkili interaktif tasarımı olan, ücretsiz ve sürekli öğrenmeyi destekleyen özellikleri Scratch'ın eğitsel programlama dili olarak seçilmesinde diğer belirleyici etkenlerdir (Rusk, Resnick ve Maloney, 2012). Scratch programlama ortamı tasarım yoluyla program öğretmeyi hedef edinen yapısı ile öğrenmeyi kolaylaştıracağı düşünülmektedir. Eğitsel programlama dilinin tanıtımında ve kullanımında Şekil 3.3.'deki basamaklar kullanılmıştır.



Şekil 3.3. Eğitsel programlama dilinin tanıtımı

3.3.2. Uygulama aşaması

Uygulama aşamasının 6 haftalık sürecinde, ilk olarak haftada 4 saat olmak üzere toplam 24 ders saati olan ders programının uygulaması gerçekleştirilmiştir. Uygulamanın diğer bir etkinliği ise performans testlerinin uygulanmasıdır. Performans testlerinin hazırlanmasına yönelik olarak ise 6 haftalık uygulama için 5 adet performans testi hazırlanmıştır. İlk hafta konusu olan temel kavramlar, genel bir giriş oluşturduğu için bu haftada bir performans testi düşünülmemiştir. Performans testlerinin hazırlanmasında akademik başarı testi için yapılan ihtiyaç analizi sonucunda belirlenen konular referans alınarak performans testleri oluşturulmuş ve başarı testlerinin oluşturulmasında alınan uzman yardımı araştırmacı tarafından aynı uzmanlara başvurularak burada da alınmıştır. Performans testlerinin hazırlanması ile ilgili ayrıntılı bilgi veri toplama araçları bölümünde verilmiş ve bu testler ekler (EK 5) bölümüne konulmuştur.

3.3.3. Sonuç aşaması

Araştırmanın sonuç aşamasında ise araştırmacı ön-testlerini gerçekleştirdiği akademik başarı ve bilgisayar programlamaya yönelik kaygı testlerinin son-test'lerini gerçekleştirmiş ve öğrencilerin uygulama sonrası son-test sonuçlarını almıştır. Uygulama aşamasında yapılan performans testlerinin sonuçları alınmış ve bütün testlerin verilerinin analizleri gerçekleştirilmiştir.

3.4. Veri Toplama Araçları

Çalışmanın verileri nicel veri toplama araçlarıyla toplanmıştır. Nicel veri toplama aracı olarak; akademik başarı testi, bilgisayar programlamaya yönelik kaygı ölçeği ve performans testleri kullanılmıştır. Çalışmada araştırma soruları için kullanılan veri toplama araçları Tablo 3.8'de gösterilmiştir.

Tablo 3.8.

Araştırma Sorularına Göre Çalışmada Kullanılan Veri Toplama Araçları

Araştırma Soruları	Veri Toplama Araçları
1	Akademik Başarı Testi
2	Performans Testleri
3	Bilgisayar Programlama Kaygı Ölçeği

3.4.1. Akademik başarı testi

Araştırmada başarıya yönelik verilerin toplanması için programlama geliştirme becerilerini ölçen akademik başarı testi kullanılmıştır. Akademik başarı testi hazırlanırken ilgili öğretim programında yer alan kazanımlar ve soruların bilişsel düzeylerini gösteren belirtke tablosu Tablo 3.9'da sunulduğu gibi hazırlanmıştır. Başarı testini oluşturan soruların öğretim programının kazanımlarını karşılayacak düzeyde olmasına özen gösterilmiştir.

Tablo 3.9.

Akademik Başarı Testi Madde Belirtke Tablosu

Hedefler	Sorular
Temel kavramları ayırt eder.	1, 3, 4, 5, 7, 9, 13, 15, 17, 22, 26, 29,31
Değişken yaklaşımını çözümler.	2, 4, 5, 6, 11, 20, 22
Giriş çıkış deyimlerini kavrar.	2, 4, 5, 6, 10, 12, 23, 24, 25, 26
Döngü yapısını takip eder ve günlük hayattaki problemlerde kullanır.	8, 10, 12, 13, 15, 18, 22, 28
Matematiksel ve Mantıksal operatörleri ayırt eder.	2, 3, 4, 7, 9, 11, 12, 14, 16, 19, 21,
Koşul yapısını kavrayıp, günlük hayattaki problemlerin çözümünde kullanır.	2, 8, 9, 12, 14, 16, 22, 23, 27, 29, 30

Araştırmada kullanılan akademik başarı testinin kapsam geçerliliğini sağlamak üzere bir eğitim uzmanı ve dört ayrı konu alanı uzmanı mevcut soruların, bu hedef davranışları ölçüp ölçmediği konusunda değerlendirmeleri istenmiştir. Bir eğitim uzmanı ve dört ayrı konu alanı uzmanının değerlendirmelerinden sonra alınan geri bildirimler doğrultusunda akademik başarı testine son şekli verilmiştir.

Ayrıca eğitim ve alan uzmanlarından soruları yenilenmiş bloom taksonomisine göre "Hatırlama, Anlama, Uygulama, Çözümleme, Değerlendirme, Yaratma" basamakları ölçü alınarak gruplandırmaları ve hedef konuları göz önüne bulundurarak bu işlemi gerçekleştirmeleri istenmiştir. Alan ve eğitim uzmanların soruları analiz etmeleri Tablo 3.10.'de; bu analiz sonucunda oluşturulan madde belirtke tablosuda Tablo 3.11'de verilmiştir.

Tablo 3.10.

Akademik Başarı Testi Sorularının Ayrımı

	S-1	S-2	S-3	S-4	S-5	S-6	S-7	S-8	S-9	S-10	S-11	S-12	S-13	S-14	S-15	S-16	S-17	S-18	S-19	S-20	S-21	S-22	S-23	S-24	S-25	S-26	S-27	S-28	S-29	S-30	S-31
N	A	U	U	U	U	U	U	U	U	U	U	Ç	U	U	U	U	H	U	Ç	U	Ç	H	Ç	U	U	U	U	U	U	U	A
İ	A	Y	Y	D	Y	Y	Y	D	D	Y	Y	D	D	Y	D	D	Y	Y	D	D	Y	Y	D	D	Y	Y	Y	Y	D	Y	D
R	A	Ç	Ç	Ç	U	U	A	A	Ç	Ç	Ç	U	U	U	A	Ç	H	U	Y	U	U	H	Ç	U	U	U	Ç	Ç	A	U	A
Ş	A	Ç	U	Ç	Ç	Ç	U	Ç	U	Ç	Ç	Ç	U	D	U	Ç	H	U	D	U	Ç	H	D	D	Ç	Ç	Ç	Ç	Ç	D	A
C	A	Ç	U	U	Ç	Ç	U	Ç	A	Ç	Ç	D	U	Y	A	Ç	Ç	U	Y	U	Ç	H	D	D	Ç	Ç	Ç	Ç	Ç	D	U
	A	Ç	U	Ç	Ç	Ç	U	Ç	U	Ç	Ç	D	U	Y	A	Ç	H	U	Y	U	Ç	H	D	D	Ç	Ç	Ç	Ç	Ç	D	A

Not: Hatırlama (H), Anlama (A), Uygulama (U), Çözümleme (Ç), Değerlendirme (D), Yaratma (Y) olarak adlandırılmıştır. Bir eğitim uzmanı (C) ve dört konu alanı uzmanı (N, I, R, Ş) uzmanların soruları değerlendirilmelerinde konsensüs sağlanma durumuna göre soru bilişsel süreç boyutuna ayrılmıştır. Konsensüs sağlanamadığı durumlarda ise (Örnek S_4) alt ve üst düzey bilişsel beceri boyutlarından hangisine soru ağırlığı olarak yakın ise o boyuta dahil edilmiştir.

Bir eğitim uzmanı ve dört ayrı konu alanı uzmanı soru analizlerindeki farklı görüşlerinin değerlendirilmesinde, alt bilişsel ve üst bilişsel süreç boyutları belirleyici olmuştur. Çoğunluğun görüşü hangi boyuta doğru ağırlık olarak kaymışsa soru belirtke tablosunda o boyuta yazılmıştır.

Tablo 3.11.

Akademik Başarı Testi Sorularının Belirtke Tablosu

BİLİŞSEL SÜREÇ BOYUTU											
Hatırlama		**Anlama**		**Uygulama**		**Çözümleme**		**Değerlendirme**		**Yaratma**	
17. ve 22. sorular		1. , 15. ve 31. Sorular		3. , 7. , 9. , 13. , 18. ve 20. Sorular		2. , 4. , 5. , 6. , 8. , 10. , 11. , 16. , 21. , 25. , 26. , 27. , 28. ve 29. sorular		12. , 23. , 24. ve 30. Sorular		14. ve 19. Sorular	
Soru sayısı	%	Soru sayısı	%	Soru sayısı	%	Soru sayısı	%	Soru sayısı	%	Soru sayısı	%
2	6.45	3	9.68	6	19.35	14	45.16	4	12.90	2	6.45
Alt Düzey Bilişsel Beceriler						**Üst Düzey Bilişsel Beceriler**					

Başarı testi bu çalışmanın araştırmacısı tarafından uzmanlarının görüşlerine başvurularak dersin amaçlarına uygun olarak tasarlanmıştır. Dersin amaçlarına göre, ders sonunda öğrenciler;

- Temel kavramları (sabitler, veri tipleri, tanımlar vb) ayırt eder,
- Değişken yaklaşımını çözümler,
- Giriş çıkış deyimlerini kavrar,
- Döngü yapısını takip eder ve günlük hayattaki problemlerde kullanır,
- Matematiksel ve Mantıksal operatörleri ayırt eder,
- Koşul yapısını (if, if-else, Switch-Case) kavrayıp günlük hayatta problem çözümünde kullanır, kazanımlarını edinirler.

Akademik başarı testi, bir eğitim uzmanı ve dört ayrı konu alanı uzmanı tarafından, kullanılan ifadelerin uygunluğu ve içeriğin dersin amaçlarına yansıtması açılarından incelenmiş ve içerik geçerliliği sağlanmıştır. Akademik başarı testinin amacı öğrencilerin programlama konularını yenilenmiş bloom taksonomisinin bilişsel süreç boyutlarına göre ölçmektir. Akademik başarı testi öğrenci gruplarına ön-test ve son-test olarak uygulanmıştır. Akademik başarı testi yenilenmiş bloom taksonomisi bilişsel düzey (Hatırlama, Anlama, Uygulama, Çözümleme, Değerlendirme, Yaratma) boyutundaki sorulardan oluşmaktadır.

3.4.2. Bilgisayar programlama kaygı ölçeği

Diğer bir veri toplama aracı olan bilgisayar programlama kaygı ölçeği araştırmacı tarafından Cheung vd. (1990)'nin Meaningful Measurement in the Classroom Using the Rasch Model: Some Exemplars. çalışmasındaki "computer programming anxiety scale" ölçeğinin geçerlilik ve güvenilirlik çalışmasının yapılarak Türkçeye uyarlanması gerçekleştirilmiştir.

3.4.2.1. Bilgisayar programlama kaygı ölçeğinin türkçeye uyarlanması

Türkçe'ye uyarlanması işleminde üçü ingiliz dili, üçü türk dili uzmanı öğretim elemanından oluşan altı kişilik bir grup tarafından, ayrı ayrı ölçeğin Türkçeye ve ingilizceye çevirisi yapılmıştır ve sonrasında bu 6 kişilik grup bir araya gelerek ölçeğe Türkçeye uyarlama konusunda son şeklini vermişlerdir. Çalışmada 19 maddeden oluşan testin Türkçe uyarlamasından sonra geçerlilik ve güvenilirlik testleri de yapılarak ön-test son-test olarak kullanılmıştır. Ölçeğin geçerlilik ve güvenirliliği için yapılan çalışmanın ayrıntıları aşağıda sıralanmıştır.

3.4.2.2. Bilgisayar programlama kaygı ölçeğinin güvenilirlik çalışması

Güvenirlik, ölçme aracının ölçmek istediği özelliği hatasız olarak ölçebilme derecesi olarak tanımlanabilir (Atılgan, Kan, ve Doğan, 2006). Diğer bir tanımla güvenirlik, ölçme aracının ölçmek istediği özelliği ne derece doğru ölçtüğü ile ilgilidir (Büyüköztürk, 2010). Bu tanımlardan da anlaşılacağı üzere güvenirlik, ölçme aracının tutarlı olarak benzer koşullarda benzer sonuçlar ortaya koyması durumudur.

Güvenirlik bir ölçme aracının geçerliliğini de etkiler. Bilinmesi gereken önemli bir özellik, geçerli bir ölçme aracının mutlaka güvenilir olmasının gerektiğidir (Alpar, 2010). Bu çerçevede ilk olarak veri toplama aracı elde etmek için hazırlanan ifadeler kendi içinde tutarlı olma, kararlı olma ve gözlenmek istenen davranışları uyarabilme gücü bakımından incelenmelidir. Bundan dolayı hazırlanmış olan ham haldeki ifadeler arasından madde seçme söz konusu olursa, genellikle madde veya ölçek puanları temel alınmaktadır. Ölçme aracının güvenirliğini belirlemede birçok yöntem kullanılabilir. Bir veri toplama aracındaki maddelerin ölçme gücünü belirlemek için 1-iç tutarlılık

ölçütüne dayalı t-testi, 2-korelasyona dayalı olmak üzere iki madde analizi türü önerilmektedir (Tezbaşaran'ın (Bozdoğan ve Öztürk, 2008'de belirttiği üzere).

Bilgisayar programlama kaygı ölçeğinin güvenirliği (iç tutarlılığı) madde analizinde, ölçeğe cevap verenlerin toplam sayısından alttan %27 ve üstten %27 alınarak grup ortalamaları farkına göre madde analizi yapılmış ve sonrasında madde-toplam korelasyonuna göre madde analizi gerçekleştirilmiştir. Ayrıca, güvenirlik katsayısı Cronbach Alpha ile de veri toplama araçlarının güvenirlik hesaplamaları yapılmıştır.

Veri toplama araçlarında yer alan maddelerin ayırt edicilik güçlerini belirlemeye yönelik her bir madde için alt grup ve üst grup bilgisayar programlama kaygı ölçeği puan ortalamaları arasındaki fark t- testiyle hesaplanmıştır. Hesaplama yapılırken bilgisayar programlama kaygı ölçeği toplam puanları düşükten yükseğe doğru sıralanmıştır. Uygulanan 116 veri toplama aracının alttan %27 ve üstten %27' nin meydana getirdiği grup 31 öğrenciden oluşmaktadır. Bu öğrenciler bir devlet üniversitesinin Meslek Yüksekokulu Elektrik ve Enerji bölümü ile Elektronik ve Otomasyon bölümünde bilgisayar programcılığı dersini alan öğrencilerden oluşmaktadır.

Yapılan analiz sonucunda bilgisayar programlama kaygı ölçeği madde-toplam korelasyonu dikkate alınarak, veri toplama aracının maddelerinin güvenirlik değerleri tespit edilmiştir. Madde - toplam korelasyon oranı ,30 ve daha yüksek oranda olan anket maddeleri bireyleri ayırt etmede iyi olduğu ,20 ile ,30 arasında orana sahip maddelerin zorunlu kalınması durumunda ankete alınabileceği ve ,20 den oran olarak daha düşük maddelerin ise ankete dahil edilmemesi gerektiği alan yazında belirtilmektedir (Büyüköztürk, 2010). Bu araştırmada bilgisayar programlama kaygı ölçeği veri toplama aracının t değerleri p=0.001 düzeyin de anlamlı olduğu, bilgisayar programlama kaygı ölçeği veri toplama aracı için tüm maddelerin madde-toplam korelasyonları ,48 - ,89 arasında değiştiği görülmüştür. Madde-toplam korelasyonu sonuçları doğrultusunda, programlama kaygı ölçeği veri toplama aracı içerisinde bulunan maddelerin güvenirlik düzeylerinin yüksek olduğu, bu anket maddelerinin benzer davranışları belirlemeye yönelik olduğu ve ayırt edici bir özelliğe sahip olduğu anlaşılmaktadır.

Tablo 3.12.

Kaygı Testi Madde - Toplam Korelasyon Oranı

Madde-Toplam İstatistikler					
	Öğe Silindi ise Ortalama Ölçek	Öğe Silindi ise Varyans Ölçeği	Düzeltilmiş Madde-Toplam Korelasyonu	Karelerin Çoklu Korelasyonu	Öğe Silindi ise Cronbach's Alpha
Kaygı Soru_1	30.5968	447.982	**.482**	.517	.963
Kaygı Soru_2	29.6935	440.052	**.526**	.490	.962
Kaygı Soru_3	29.5968	415.458	**.780**	.733	.959
Kaygı Soru_4	29.6129	433.225	**.596**	.580	.962
Kaygı Soru_5	29.9032	432.515	**.577**	.464	.962
Kaygı Soru_6	29.4032	434.310	**.699**	.725	.960
Kaygı Soru_7	29.8710	415.491	**.772**	.827	.959
Kaygı Soru_8	29.3871	420.208	**.787**	.774	.959
Kaygı Soru_9	29.4355	422.840	**.784**	.694	.959
Kaygı Soru_10	30.0000	420.426	**.787**	.835	.959
Kaygı Soru_11	29.9194	432.174	**.700**	.710	.960
Kaygı Soru_12	29.3710	421.254	**.818**	.826	.959
Kaygı Soru_13	29.2097	423.709	**.781**	.716	.959
Kaygı Soru_14	30.0000	424.328	**.727**	.724	.960
Kaygı Soru_15	29.6290	412.565	**.850**	.814	.958
Kaygı Soru_16	29.5645	413.397	**.855**	.837	.958
Kaygı Soru_17	29.6613	409.342	**.894**	.868	.958
Kaygı Soru_18	29.9677	417.278	**.821**	.772	.959
Kaygı Soru_19	30.1452	418.913	**.781**	.812	.959

Bilgisayar programlama kaygı ölçeğinin güvenirlik çalışması için ise örneklem haricinde 116 kişi ve 19 madde üzerinden belirlenen iç tutarlılık katsayısı olan cronbach alpha (α = ,93) olarak tespit edilmiştir. Yapılan ölçümler ve değerlendirmeler bilgisayar programlama kaygı ölçeğinin, veri toplama aracı olarak oldukça güvenilir olduğu göstermektedir.

3.3.2.3. Bilgisayar programlama kaygı ölçeğinin geçerlilik çalışması

"Geçerlik, ölçme araçlarının ölçmeyi amaçladığı özelliği, başka her hangi bir özellikle karıştırmadan doğru bir şekilde ölçebilme derecesi olarak tanımlanır." (Alpar,

2010 s.75). Kline (1994)'da yaptığı çalışmasında geçerlilik ve güvenilirlik çalışmlarının yapılabilmesi için örneklemin büyüklüğünün belirlenmesi; örneklem büyüklüğü belirlenirken, denek değişken oranının istenilen düzeyde olup olmadığının tespit edilmesi, gerektiğini belirtmektedir. Denek değişken (madde) oranının 10:1 oranında tutulmasını, fakat bu oranın koşullara göre aşağıya çekilebileceğini, en az 2:1 oranında olmasının gerektiğini bildirmektedir (Kline'in (Büyüköztürk, 2010'da belirttiği üzere). Bu araştırmadaki sonuçlar doğrultusunda, yapılan çalışmada denek değişken (madde) oranı göz önüne alınarak örneklem büyüklüğünün yeterli olup olmadığı tespit edilmiştir. 19 maddeden oluşan ölçek için yaklaşık olarak 6:1 oranı olan 116 kişilik bir örneklem büyüklüğü ile çalışılmıştır. Bu örneklem büyüklüğünün Kline (1994)'nin çalışmasındaki oranlarla karşılaştırılınca istenilen düzeyde bir orana karşılık geldiğini görmekteyiz.

Örneklem büyüklüğü oranı sağlandıktan sonra veri toplama aracının geçerliliği için veri toplama aracı bir devlet üniversitesinin Meslek Yüksekokulu Elektrik ve Enerji bölümü ile Elektronik ve Otomasyon bölümü bilgisayar programcılığı dersini alan öğrencilere uygulanmış, elde edilen verilerin yapılacak faktör analizi için uygun olup olmadığı Kaiser-Meyer-Olkin (KMO) katsayısı ve Bartlett'in Sphericity testi aracılığı ile incelenmiştir. KMO "0" ile"1"arasında değişiklik gösteren bir değerdir ve 1'e ne kadar yaklaşırsa değerin derecesi yükselir. Faktör analizinin iyi olması demek KMO katsayısının ,80'den daha büyük olması demektir. Bununla birlikte, KMO değerinin ,60'dan (çoğunlukla ,50'nin üzerinde) ise yüksek, Bartlett'in Sphericity testinin ise anlamlı çıkması, verilerin analiz edilebilmesi için uygun olduğunu göstermektedir (Alpar, 2010; Büyüköztürk, 2010). Bilgisayar programlama kaygı ölçeği veri toplama aracının KMO katsayısı ,911 ve Bartlett'in Sphericity testi (Approx Chi-Square=1095,848) $p<0.001$ düzeyinde anlamlı bulunmuşlardır. Türkçeye uyarlanması, geçerliliği ve güvenilirliği yapılmış olan bilgisayar programlama kaygı ölçeği olarak verilmiştir.

3.4.63. Performans testleri

Araştırmada öğrencilerin ders sonrası haftalık programlama ders başarılarını ölçen performans testleri kullanılmıştır. Uygulama boyunca 5 adet performans testi gruplara uygulanmıştır. Performans testleri haftalık ders süresi bittikten sonra

yapılmıştır. Testlerin hazırlık aşamasında akademik başarı testi için gerçekleştirilen ihtiyaç analizi sonucunda oluşan konular referans alınmış ve testler bu konular kapsamında dersin işleyişine paralel olarak hazırlanmıştır. Performans testleri sonucunda elde edilen veriler gruplar bazında akdemik başarı ile karşılaştırılmış ve aralarındaki ilişki analiz edilmiştir.

Bir eğitim uzmanı ve dört ayrı konu alanı uzmanından, performans testi sorularını yenilenmiş bloom taksonomisi "Hatırlama, Anlama, Uygulama, Çözümleme, Değerlendirme, Yaratma" basamakları ölçü alınarak gruplandırmaları ve hedef konuları göz önüne bulundurarak bu işlemi gerçekleştirmeleri istenmiştir. Uzmanların soruları analizleri, Tablo 3.13.'de; bu analiz sonucunda oluşturulan madde belirtke tablosuda Tablo 3.14'de verilmiştir.

Tablo 3.13.

Performans Testleri Sorularının Ayrımı

	P1			P2						P3								P4							P5								
	1	2	3	1	2	3	4	5	6	1	2	3	4	5	6	7	8	1	2	3	4	5	6	1	2	3	4	5	6	7	8	9
N	H	H	Ç	H	H	H	Ç	A	A	A	A	A	A	Ç	A	Y	A	U	U	A	A	A	A	U	U	Ç	A	A	A	A	A	A
İ	H	H	Ç	H	D	Y	D	D	A	D	D	Y	Y	Y	Y	D	A	D	D	Y	D	D	Y	D	U	D	D	D	D	Y	Y	D
R	H	H	Ç	H	H	H	Ç	A	A	A	Ç	A	Ç	Ç	Ç	A	A	U	U	H	U	Ç	Ç	U	U	H	Ç	Ç	Ç	Ç	Ç	Ç
Ş	A	A	A	H	H	H	D	Y	H	A	Ç	A	Ç	Ç	D	Y	A	U	U	Ç	U	Y	Y	Y	Y	D	D	D	D	Ç	Ç	D
C	H	H	Ç	H	H	A	Ç	Y	A	A	Ç	A	Ç	A	Ç	Y	A	U	U	Ç	U	Ç	Y	U	Ç	Ç	Ç	Ç	Ç	Ç	Ç	D
	H	H	Ç	H	H	H	Ç	Y	A	A	Ç	A	Ç	Ç	Ç	Y	A	U	U	Ç	U	Ç	Y	U	U	D	D	D	D	Ç	Ç	D

Not: Hatırlama (H), Anlama (A), Uygulama (U), Çözümleme (Ç), Değerlendirme (D), Yaratma (Y) olarak adlandırılmıştır. Bir eğitim uzmanı (C) ve dört konu alanı uzmanı (N, I, R, Ş) uzmanların soruları değerlendirilmelerinde konsensüs sağlanma durumuna göre soru bilişsel süreç boyutuna ayrılmıştır. Konsensüs sağlanamadığı durumlarda ise (Örnek S_4) alt ve üst düzey bilişsel beceri boyutlarından hangisine soru ağırlığı olarak yakın ise o boyuta dâhil edilmiştir.

Bir eğitim uzmanı ve dört ayrı konu alanı uzmanı soru analizlerindeki farklı görüşlerinin değerlendirilmesinde bilişsel süreç boyutu ve çoğunluğun görüşü belirleyici olmuştur.

Tablo 3.14.

Performans Testleri Sorularının Belirtke Tablosu

Bilişsel Düzey	Sorular	Toplam	Yüzde (%)
Hatırlama	P_1.1, P_1.2, P_2.1, P_2.2, P_2.3,	5	15,6
Anlatma	P_2.6, P_3.1, P_3.3,P_3.8,	4	12,5
Uygulama	P_4.1, P_4.2, P_4.4, P_5.1, P_5.2	5	15,6
Çözümleme	P_1.3, P_2.4, P_3.2, P_3.6, P_4.3, P_4.5, P_3.4, P_3.5, P_5.7, P_5.8,	10	31,3
Değerlendirme	P_5.3, P_5.4, P_5.5, P_5.6, P_5.9	5	15,6
Yaratma	P_2.5, P_3.7, P_4.6,	3	9,4
Toplam		32	100

3.5. Veri Analizi

Bu bölümde çalışmada nicel veri toplama araçlarıyla toplanan verilerin nasıl analiz edileceği anlatılacaktır. Nicel veriler akademik başarı testi, bilgisayar programlama kaygı ölçeği ve performans testleri ile elde edilmiştir.

3.5.1. Nicel verilerin analizi

Nicel verilerin elde edilmesi sırasında deneysel desenden yararlanılmıştır. Deneysel desen, değişkenler arasındaki neden sonuç ilişkilerini belirlemek için gözlenmek istenen verilerin üretildiği çalışma ve araştırma alanıdır (Büyüköztürk, 2001; Karasar, 2004). Nicel veriler SPSS 19 For Windows Desktop programında analiz edilmiştir. Öğrencilerin bilgisayar programlama kaygıları "bilgisayar programlama kaygı ölçeği" ile akademik başarıları "akademik başarı testi" ile uygulama öncesinde ve sonrasında ölçülmüştür. Uygulama öncesi ve sonrası yapılan ölçümlerde deneysel işlemin akademik başarı ve bilgisayar programlama kaygısına yönelik etkisinin test edilmesi amacıyla parametrik testlerden grup için alizlerde t-testi, gruplar arası anlizlerde; tek yönlü varyans analizi (ANOVA), eşleştirilmiş grupların analizi için eşleştirilmiş örneklem t-testi ve kovaryans analizi (ANCOVA) kullanılırken, normal

dağılım göstermeyen performans testlerinde non-parametrik testlerden Kuruskal Walles test kullanılmıştır. Ayrıca akademik başarı ve performans testlerinin zaman içinde dağılımını göstermek için excel grafikleri kullanılmıştır.

3.6. Araştırmacının Rolü

Araştırmacı uygulama sürecinin içinde ve derslerin anlatımında bulunmuştur. Programlama temelleri dersinde yapılan uygulama, 3 grup içinde aynı öğretim elemanının verdiği 1. Sınıfta okuyan 87 öğrenci üzerinde gerçekleştirilmiştir. Mesleki deneyim olarak araştırmayı yapan ve dersi anlatan öğretim elemanı doktora yapmakta olup 15 yıllık programlama yeterliliğine sahiptir. Araştırmacının her üç grup içinde dersi anlatan kişi olmasından dolayı gruplar arasında ders anlatımından kaynaklı bir farklılık olumadığı söylenebilir. Araştırmacı her grupta aynı şekilde ders anlatmaya azami ölçüde özen göstermiştir.

3.7. Çalışmanın Geçerlilik ve Güvenirliği

Eğitim araştırmalarının değerini belirleyen en önemli unsurlar araştırmanın geçerliği ve güvenirliğidir (Büyüköztürk, Çakmak, Akgün, Karadeniz ve Demirel, 2010). Bir araştırmanın sonuçlarının geçerli ve güvenilir olması için hem veri toplama araçlarının hem de çalışmanın süreciyle ilgili geçerlik ve güvenirlik önlemlerinin alınması gerekir. Veri toplama araçlarının geçerlik güvenirlik bilgileri ilgili başlıklarda açıklandığından burada ayrıca açıklanmayacaktır. Araştırmanın desenine göre alınması gereken geçerlik ve güvenirlik önlemleri de değişebilir. Nicel araştırmalarda geçerlilik, sonuçların hatasız olmasıyla, güvenirlik ise sonuçların genellenebilirliği ile ilişkilidir (McMillan & Schumacher, 2010). Bu bilgiler ışığında çalışmanın yürütülmesi esnasında alınan geçerlilik ve güvenirlik önlemleri Tablo 3.15'de özetlenmiştir.

Tablo 3.15.

Çalışmada Alınan Geçerlik Güvenirlik Önlemleri

	Alınan önlem	Nasıl yapıldığı
Geçerlilik	**Araştırmanın yöntemin detaylı açıklanması**	Yöntemin kullanılma gerekçesi ayrıntılı şekilde açıklanmıştır
		Veri toplama/analiz süreçleri ayrıntılı olarak açıklanmıştır
		Örneklem grubunun özellikleri ve örneklem seçim şekli ayrıntılı şekilde açıklanmıştır
		Kullanılan veri toplama araçlarıyla ilgili geçerlik ve güvenirlik önlemleri belirtilmiştir
		Uygulama süreci ayrıntılı olarak açıklanmıştır
	Varsayım ve sınırlılıklar açıklanması	Çalışmanın varsayım ve sınırlılıkları giriş bölümünde belirtilmiştir
	Dış değişkenin kontrol altına alınması	Deney gruplarında aynı öğretmen dersi yürütmüştür
	Araştırmacının rolü	Araştırmacı rolü ayrıntılı şekilde açıklanmıştır
	Gönüllülük	Uygulamalar gönüllülük esasına göre yapılmıştır
Güvenirlik	**Akran değerlendirmesi ve uzman görüşü**	Akademik başarı testi ve performans testlerinin geliştirilmesinde alan uzmanlarından görüş alınmıştır.
	Güvenirlik hesaplamalarının	Kullanılan veri toplama araçlarıyla ilgili güvenirlik hesaplamaları belirtilmiştir
	Uygulama süresinin uzun tutulması	Uygulama süresi uzun tutulmuştur (6 hafta)

DÖRDÜNCÜ BÖLÜM

4. BULGULAR

Normal dağılım gösteren verilerin analizinde parametrik testlerden tek grup (One-Sample t testi), tek yönlü varyans analizi ANOVA, tek yönlü kovaryans analizi ANCOVA ve eşleştirilmiş iki grup (Paired Samples t testi) kullanılmıştır. Normal dağılım göstermeyen verilerin analizinde ise parametric olmayan testlerden Kruskal-Wallis Testi kullanılmıştır.

4.1. Akademik Başarı Testi ANOVA Varsayımları

Tek yönlü varyans ANOVA testlerinin uygulanabilmesi için normallik/aykırı değerler ve varyans eşitliği değerleri Tablo 4.1. ve Tablo 4.2.'de verilmiştir.

Tablo 4.1.

Eğitsel Programlama Dilinin Son-Test Bilişsel Beceriler Akademik Başarı Testi Normalite Değerleri

		Son-Test Alt Düzey Bilişsel Beceriler	Son-Test Üst Düzey Bilişsel Beceriler	Son-Test Bilişsel Beceriler
N		87	87	87
Normal Parametreler a.b	Ortalama	29.64	1502.54	66.98
	Standart Sapma	4.74	747.01	12.9
Uç Farklılıklar	Mutlak	.11	.12	.13
	Pozitif	.06	.12	.06
	Negatif	-.11	-.11	-.13
Kolmogorov-Smirnov Z		1.08	1.17	1.27
Asymp. Sig. (2-tailed)		.191	.128	.079

p< .05

Eğitsel programlama dilinin Son-Test bilişsel beceriler açısından yapılan normalite testi sonuçları; Son-Test Alt Düzey Bilişsel Becerler anlamlılık düzeyi p = .191, Son-Test Üst Düzey Bilişsel Becerler anlamlılık düzeyi p = .128 ve Son-Test Bilişsel Becerler anlamlılık düzeyi ise p = .079 olarak bulunmuştur. Son-Test bilişsel

beceriler anlamlılık düzeyleri p > .05 olduğundan normal bir dağılım göstermektedirler. Bu sonuçlara göre parametrik testler kullanılmıştır.

Tablo 4.2.

Eğitsel Programlama Dilinin Son-Test Bilişsel Beceriler Akademik Başarı Testi Homojenlik Testi Değerleri

	Homojenlik Testi Değerleri			
	Levene Statistic	**df1**	**df2**	**Sig.**
Son-Test Alt Düzey Bilişsel Becerler	.55	2	84	.577
Son-Test Üst Düzey Bilişsel Becerler	1.07	2	84	.346
Son-Test Bilişsel Becerler	2.65	2	84	.077

p< .05

Eğitsel programlama dilinin Son-Test bilişsel beceriler açısından yapılan homojenlik testi sonuçları; Son-Test Alt Düzey Bilişsel Becerler anlamlılık düzeyi p = .577, Son-Test Üst Düzey Bilişsel Becerler anlamlılık düzeyi p = .346 ve Son-Test Bilişsel Becerler anlamlılık düzeyi ise p = .077 olarak bulunmuştur. Son-Test bilişsel beceriler anlamlılık düzeyleri p > .05 olduğundan homojen bir dağılım göstermektedirler. Bu sonuçlara göre parametrik testler kullanılmıştır.

4.2. Akademik Başarı Testi ANCOVA Varsayımları

Tek yönlü kovaryans ANCOVA testlerinin uygulanabilmesi için normallik/aykırı değerler ve varyans eşitliği değerleri Tablo 4.1. ve Tablo 4.2.'de verilmiştir. Tek yönlü kovaryans ANCOVA testlerinin uygulanabilmesi için diğer bir varsayımsa değişkenlerin bağımsız olmasıdır.

Tablo 4.3.

Eğitsel Programlama Dilinin Son-Test Bilişsel Beceriler Akademik Başarı Testi Bağımsız Değişkenler Test Değerleri

		Son-Test Alt Düzey Bilişsel Beceriler	Son-Test Üst Düzey Bilişsel Beceriler	Son-Test Bilişsel Beceriler
Son-Test Alt Düzey Bilişsel Beceriler	Pearson Correlation	1	.376**	.667**
	Sig. (2-tailed)		.000	.000
	N	87	87	87
Son-Test Üst Düzey Bilişsel Beceriler	Pearson Correlation	.376**	1	.941**
	Sig. (2-tailed)	.000		.000
	N	87	87	87
Son-Test Bilişsel Beceriler	Pearson Correlation	.667**	.941**	1
	Sig. (2-tailed)	.000	.000	
	N	87	87	87

**. p < .01

Son-Test Alt Düzey Bilişsel Beceriler ile Son-Test Üst Düzey Bilişsel Beceriler arasında 0.376, Son-Test Alt Düzey Bilişsel Beceriler ile Son-Test Bilişsel Beceriler arasında 0.667ve Son-Test Üst Düzey Bilişsel Beceriler ile Son-Test Bilişsel Beceriler arasında ise 0.941 yüksek derecede pozitif bir ilişki vardır. Anlamlılık değeri ise p< .05'den küçük olduğu için anlamlıdır.

4.3. Akademik Başarıya Yönelik Bulgular

4.3.1. Grupların Akademik Baş arıya Yönelik Ön-Test Son-Test Puan Ortalamaları.

Eğitsel programlama dili, dersin teori kısmına entegre edilmiş grup (TEG), dersin uygulama kısmına entegre edilmiş grup (UEG) ve dersin hem teori hem uygulama kısmına entegre edilmiş grup (TUEG)'ta uygulanmış ve öğrenci başarısı üzerindeki etkilerine yönelik sonuçlar Tablo 4.4.'de verilmiştir.

Tablo 4.4.

Çalışma Gruplarının Akademik Başarı Ön-Test ve Son-Test Puan Ortalamaları

Entegrasyon Şekli	Ön-Test Ortalama($\overline{X}$)	Son-Test Ortalama($\overline{X}$)
UEG	51.08	65.50
TEG	52.35	62.04
TUEG	52.27	73.31

Tablo 4.4' de UEG grubunun ön-test puan ortalaması $\overline{X}$ =51.08 son-test puan ortalamalaması $\overline{X}$ = 65.50 olduğu, TEG grubunun ön-test puan ortalaması $\overline{X}$ =52.35; son-test puan ortalamalaması $\overline{X}$ =62.04 olduğu ve TUEG grubunun ön-test puan ortalaması $\overline{X}$ =52.27 ve son test puan ortalamalamasının $\overline{X}$ =73.31 olduğu görülmektedir.

4.3.2. Eğitsel programlama dilinin derse entegrasyon şekli, öğrencilerin ön-test son-test başarıları arasında anlamlı bir fark oluşturmakta mıdır?

Tablo 4.5.

Eğitsel Programlama Dilinin Derse Entegrasyon Şeklinin Grup İçi Öğrenci Başarıları Arasındaki Farka Yönelik T-Testi Sonuçları

Entegrasyon Şekli	Testler	N	Ortalama ($\overline{X}$)	Standart Sapma	t	df	Sig.
UEG	Akademik Başarı Ön-test	30	51.8	10.23	-4.68	29	.000
	Akademik Başarı Son-test	30	65.5	14.49			
TEG	Akademik Başarı Ön-test	28	52.4	11.70	-4.39	27	.000
	Akademik Başarı Son-test	28	62.0	11.40			
TUEG	Akademik Başarı Ön-test	29	52.3	11.27	-9.57	28	.000
	Akademik Başarı Son-test	29	73.3	10.40			

**p< .05*

Tablo 4.5'te görüldüğü gibi UEG grubu ön-test puan ortalamasının $\overline{X}$ =51.8 ve son-test puan ortalamasının $\overline{X}$ =65.5 olduğu; TEG grubu ön-test puan ortalamasının $\overline{X}$ =52.4 ve son-test puan ortalamasının $\overline{X}$ =62.0 olduğu; TUEG grubunda ise ön-test puan ortalamasının $\overline{X}$ =52.3 ve son-test puan ortalamasının $\overline{X}$ =73.3 olduğu ve bağımlı örneklem t-testi verilerine göre grup içerisinde $p < .05$ düzeyinde anlamlı bir fark olduğu görülmektedir.

4.3.3. Eğitsel programlama dilinin derse entegrasyon şekli, öğrencilerin ön bilgileri kontrol altına alındığında, son-test bilişsel beceri puanları arasında anlamlı bir fark oluşturmakta mıdır?

ANCOVA testinin yapılabilmesi için gerekli şartların oluşturulmasına yönelik varsayımlar ele alınmış. UEG grupta son-teste katılan 30 kişiden bir kişinin son test puanı uç değer olduğu için silinmiştir. Üç grubun ön bilgilerinin etkilerini ortadan kaldırmak için, tek yönlü kovaryans analizi (ANCOVA) yapılmıştır. Tek yönlü Kovaryans analizi (ANCOVA) ön bilgiler kontrol altına alınarak, eğitsel programlama dilinin entegrasyon şekli ile son-test puanları arasındaki ilişkiyi değerlendirmek amacıyla yapılmıştır.

Tablo 4.6.

Öğrencilerin Son-Test Bilişsel Becerileri Puanları Açısından Öğrenci Başarılarını Gösteren Ancova Sonuçları

Kaynak	Karelerin Toplamı Tip III	df	Karelerin Ortalaması	F	Sig.	Kısmi eta S.
Düzeltilmiş Model	3421.50[a]	3	1140.50	8.52	.000	.235
Kesen	8028.68	1	8028.68	59.98	.000	.420
Son-Test Bilişsel Beceriler	1509.18	1	1509.18	11.27	.001	.120
Entegrasyon Şekli	1905.45	2	952.72	7.11	.001	.146
Hata	11109.48	83	133.84			
Toplam	404940.00	87				
Düzeltilmiş Toplam	14530.98	86				

p< .05

ANCOVA'nın bu değerlere bağlı olarak anlamlı [F(2-87)=7.118, p = .001] olduğunu için önbilgiler kontrol altına alındığında, grupların son-test ortalama puanları arasında anlamlı farkın oluştuğu Tablo 4.6'da görülmektedir.

Tablo 4.7.

Öğrencilerin Son-Test Bilişsel Becerileri Puanları Açısından Öğrenci Başarı Ortalamaları.

Entegre Şekli	Ortalama ($\overline{X}$)	Standart Sapma	N
UEG	65.50	14.49	30
TEG	62.04	11.38	28
TUEG	73.31	10.41	29
Toplam	66.99	13.00	87

Anlamlılığın hangi gruptan kaynaklandığını ortaya koymak için Post Hoc testi yapılmıştır. Test sonucunda (TEG)'in ortalaması ($\overline{X}$ = 62.04), (UEG)'nin ortalaması ($\overline{X}$ = 65.50) ve (TUEG) ortalaması ise ($\overline{X}$ = 73.31) olduğu Tablo 4.7'de görülmektedir.

Tablo 4.8.

Öğrencilerin Son-Test Bilişsel Becerileri Puanları Açısından Öğrenci Başarılarını Gösteren Ancova Post-Hoc LSD Testi Değerleri.

(I) 'in Uygulama	(J) Uygulama Türü	Ortalama Farkı (I-J)	Std . Hata	Sig .
UEG	**TEG**	3.67	3.04	.690
	TUEG	-7.62*	3.01	.040
TEG	**UEG**	-3.67	3.04	.690
	TUEG	-11.30*	3.06	.001
TUEG	**UEG**	7.62*	3.01	.040
	TEG	11.30*	3.06	.001

p < .05

Tablo 4.8'de görüldüğü üzere, Post-Hoc karşılaştırma analizi, gruplar arasındaki farkın hangi gruptan kaynaklandığını belirlemek için yapılmış olup, karşılaştırma

sonucunda TUEG grubunun TEG' den ve UEG' den farklı olduğunu bulunmuştur. Farkın TUEG gruptaki öğrencilerin son-test ortalamaları, TEG ve UEG gruplarından anlamlı olarak TUEG grubu lehinde olduğu görülmüştür.

4.3.4. Eğitsel programlama dilinin derse entegrasyon şekli, öğrencilerin ön bilgileri kontrol altına alındığında son-test alt düzey bilişsel beceri puanları arasında anlamlı bir fark oluşturmakta mıdır?

Çalışmada gruplar oluşturulurken öğrencilerin bireysel özellikleri ve bilgi seviyelerinin sonuçları etkileyebileceği göz önüne bulundurulmuş ve homojen bir dağılımın oluşmama ihtimaline karşılık öğrencilerin ön bilgileri kontrol altına alınarak son-test alt düzey bilişsel beceri puanları incelenmiştir. Elde edilen veriler Tablo 4.9.'da sunulmuştur.

Tablo 4.9.

Öğrencilerin Son-Test Alt Düzey Bilişsel Becerileri Puanları Açısından Öğrenci Başarılarını Gösteren Ancova Sonuçları

Kaynak	Karelerin Toplamı Tip	df	Karelerin Ortalaması	F	Sig.	Kısmi eta S.
Düzeltilmiş Model	88.32[a]	3	29.44	1.32	.273	.046
Kesen	3206.28	1	3206.28	143.87	.000	.634
SonTest Alt Düzey Bilissel Beceriler	23.75	1	23.75	1.06	.305	.013
Entegrasyon Şekli	58.05	2	29.02	1.30	.277	.030
Hata	1849.62	83	22.28			
Toplam	78389.00	87				
Düzeltilmiş	1937.95	86				

p< .05

Gruplar arasında alt düzey bilişsel becerilere yönelik anlamlı bir farkın olup olmadığını belirlemek amacı ile ANCOVA yapılmıştır. Tablo 4.9'da görüldüğü gibi [F(2-87)=1.303, p = .277] gruplar arasında alt düzey bilişsel beceri puanları arasında önbilgiler kontrol altına alındığında anlamlı fark olmadığı görülmektedir.

4.3.5. Eğitsel programlama dilinin derse entegrasyon şekli, öğrencilerin ön bilgileri kontrol altına alındığında son-test üst düzey bilişsel beceri puanları arasında anlamlı bir fark oluşturmakta mıdır?

Çalışmada gruplar oluşturulurken öğrencilerin bireysel özellikleri ve bilgi seviyelerinin sonuçları etkileyebileceği göz önüne bulundurulmuş ve homojen bir dağılımın oluşmama ihtimaline karşılık öğrencilerin ön bilgileri kontrol altına alınarak son-test üst düzey bilişsel beceri puanları incelenmiştir. Elde edilen veriler Tablo 4.10.'da sunulmuştur

Tablo 4.10.

Öğrencilerin Son-Test Üst Düzey Bilişsel Becerileri Puanları Açısından Öğrenci Başarılarını Gösteren Ancova Sonuçları

Kaynak	Karelerin Toplamı Tip	df	Karelerin Ortalaması	F	Sig.	Kısmi eta S.
Düzeltilmiş Model	2112.00[a]	3	704.00	8.03	.000	.225
Kesen	3086.38	1	3086.38	35.20	.000	.298
SonTest Üst Düzey Bilişsel Beceriler	804.99	1	804.99	9.18	.003	.100
Entegrasyon Şekli	1420.47	2	710.23	8.10	.001	.163
Hata	7275.64	83	87.65			
Toplam	130721.00	87				
Düzeltilmiş Toplam	9387.65	86				

p < .05

Gruplar arasında üst düzey bilişsel becerilere yönelik anlamlı bir farkın olup olmadığını belirlemek amacı ile ANCOVA yapılmıştır. Tablo 4.10'da görüldüğü gibi [F(2-87)=8.102, p = .001] gruplar arasında üst düzey bilişsel beceri puanları arasında önbilgiler kontrol altına alındığında anlamlı fark bulunduğu anlaşılmaktadır.

Tablo 4.11.

Öğrencilerin Son-Test Üst Düzey Bilişsel Becerileri Puanları Açısından Öğrenci Başarıları Ortalamaları.

Uygulama Türü	Mean (X)	SD	N
UEG	36.47	10.45	30
TEG	33.00	9.59	28
TUEG	42.45	9.31	29
Toplam	37.34	10.44	87

ANCOVA testinde sonucu bulunan anlamlı farkın hangi grup ya da gruplardan kaynaklandığını saptamak amacıyla Post Hoc LSD test yapılmış olup bu testin sonuçları Tablo 4.12'de görülmektedir.

Tablo 4.12.

Öğrencilerin Son-Test Üst Düzey Bilişsel Becerileri Puanları Açısından Öğrenci Başarılarını Gösteren ANCOVA Post-Hoc LSD Değerleri

(I) 'in Uygulama Türü	(J) Uygulama	Ortalama Farkı (IJ)	Std . Hata	Sig .
UEG	TEG	3.81	2.46	.125
	TUEG	-6.08*	2.43	.015
TEG	UEG	-3.81	2.46	.125
	TUEG	-9.89*	2.48	.000
TUEG	UEG	6.08*	2.43	.015
	TEG	9.89*	2.48	.000

p< .05

Post Hoc LSD testi sonucuna göre Tablo 4.12'de görüldüğü üzere Post-Hoc karşılaştırma analizi, gruplar arasındaki farkın hangi gruptan kaynaklandığını belirlemek için yapılmış olup, karşılaştırma sonucu UEG' nin TUEG' den ve TEG' in de TUEG' den farklı olduğunu bulunmuştur. Farkın TUEG grupdaki öğrencilerin son-test ortalamaları, TEG ve UEG gruplarından anlamlı olarak TUEG grubun lehinde olduğu görülmüştür.

4.4. Ders İçi Performansa Yönelik Bulgular

4.4.1. Eğitsel programlama dilinin derse entegresyon şekli, ders içi performans testlerinin bilişsel beceri puanları arasında anlamlı bir fark oluşturmakta mıdır?

Eğitsel programlama dilinin derse entegrasyonun şeklinin öğrencilerin akademik başarıları üzerindeki etkisine ön-test son-test ile bakılmıştır. Öğrencilerin akademik başarı testlerini cevapladıkları zaman diliminde, fiziksel, bilişsel ve çevresel etkenlerin etkili olabileceğini göz önüne alınmış ve her ders sonunda akademik başarı testleri ile uyumlu kısa bir test yapılması gerektiğine karar verilmiştir. Bu amaçla süreç içerisinde de öğrencilerin bilgilerinin ölçülmesine yönelik olarak beş adet performans testi yapılmışıtr.

4.4.1.1. Eğitsel programlama dilinin derse entegresyon şekli, ders içi performans testi 1'in bilişsel beceri puanları arasında anlamlı bir fark oluşturmakta mıdır?

Süreç içerisindeki başarıyı belirlemeye yönelik yapılan testlerden Performans 1 testinin normallik/aykırı değerler sonuçları Tablo 4.13.'de verilmiştir. Performans 1 testi normal dağılım göstermediği için parametrik olmayan testlerden Kruskal-Wallis testi kullanılmıştır.

Tablo 4.13.

Performans 1 Testi Normalite Değerleri

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Entegrasyon Şekli	**Statistic**	**df**	**Sig.**	**Statistic**	**df**	**Sig.**
Performans testi 1 in Bilişsel Becerileri Not Ortalaması	**UEG**	.104	30	.200*	.963	30	.374
	TEG	.138	28	.185	.954	28	.248
	TUEG	.222	29	.001	.820	29	.000

p<.,05

Tablo 4.13'de görüldüğü gibi, analizlerin parametrik ve parametrik olmayan testlerden hangisi ile yapılacağına karar vermek için yapılan normalite testi sonuçları doğrultusunda performans 1 testi değerleri normal bir dağılım göstermediği görülmüştür. Bu sonuçlara göre parametric olmayan testlerden Kruskal-Wallis Testi kullanılmıştır.

Tablo 4.14.

Eğitsel Programlama Dilinin Derse Entegresyon Şekli, Ders İçi Performans Testi 1'in Bilişsel Beceri Ortalama Puanları Açısından Kruskal-Wallis Test Sonuçları

Performans Test Unsurları	**Entegrasyon Şekli**	**Sayı (N)**	**Mean Rank**	**Chi-Square**	**df**	**Asymp. Sig.**
Performans Testi 1 'in bilişsel beceri değerleri	**UEG**	30	45.35	6.27	2	.043
	TEG	28	34.88			
	TUEG	29	51.41			
	Toplam	87				

p< .05

Performans Testi 1'in bilişsel beceri puanları acısından anlamlı bir farkın olduğu Tablo 4.14'te görülmektedir. Bu farkın hangi grup ya da gruplardan kaynaklandığını belirlemek amacı ile Kruskal-Wallis Post-Hoc testi yapılmış olup ilgili değerler Tablo 4.15'de verilmiştir.

Tablo 4.15.

Eğitsel Programlama Dilinin Derse Entegresyon Şekli, Ders İçi Performans Testi 1'in Bilişsel Beceri Ortalama Puanları Açısından Kruskal-Wallispost-Hoc Testi Sonuçları

Performans Test Unsurları	**Örnek1-Örnek2**	**Test İstatistik**	**Standart Hata**	**Std. Test İstatistik**	**Sig.**	**Adj.Sig**
Performans Testi 1 'in bilişsel beceri değerleri	**TEG-UEG**	10.47	6.61	1.58	.113	.340
	TEG-TUEG	-16.53	6.67	-2.47	.013	.040
	UEG-TUEG	-6.06	6.55	-.92	.355	1.00

p< .05

Performans Testi 1'in bilişsel becerileri puanları yönünden anlamlı farklılığın TEG ve TUEG grupları arasında olduğu Tablo 4.15'de görülmektedir.

4.4.1.2. Eğitsel programlama dilinin derse entegresyon şekli, ders içi performans testi 2'nin bilişsel beceri puanları arasında anlamlı bir fark oluşturmakta mıdır?

Çalışma süresi içerisindeki başarıyı belirlemeye yönelik yapılan testlerden Performans 2 testinin normallik/aykırı değerler sonuçları Tablo 4.16.'da verilmiştir. Performans 2 testi normal dağılım göstermediği için parametrik olmayan testlerden Kruskal-Wallis testi kullanılmıştır.

Tablo 4.16.

Performans 2 Testi Normalite Değerleri

	Entegrasyon Şekli	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Performans testi 2'nin Bilişsel Becerileri Not Ortalamaları	**UEG**	.18	30	.009	.85	30	.001
	TEG	.18	28	.018	.96	28	.411
	TUEG	.12	29	.200*	.93	29	.091

p< .05

Tablo 4.16'da görüldüğü gibi, analizlerin hangi testlerle yapılması gerektiğine karar vermek için yapılan normalite testi sonuçları doğrultusunda, performans 2 testi değerleri normal bir dağılım göstermediği görülmüştür. Bu sonuçlara göre parametric olmayan testlerden Kruskal-Wallis Testi kullanılmıştır.

Tablo 4.17.

Eğitsel Programlama Dilinin Derse Entegresyon Şekli, Ders İçi Performans Testi2'in Bilişsel Beceri Ortalama Puanları Açısından Kruskal-Wallistestsonuçları

Performans Test Unsurları	Entegrasyon Şekli	Sayı (N)	Mean Rank	Chi-Square	df	Asymp. Sig.
Performans Testi 2 'nin bilişsel beceri değerleri	**UEG**	30	46.48	7.17	2	.028
	TEG	28	33.91			
	TUEG	29	51.17			
	Toplam	87				

p< .05

Tablo 4.17'de görüldüğü gibi, Performans Testi 2'nin bilişsel beceri puanları acısından anlamlı bir farkın olduğu anlaşılmaktadır. Bu farkın hangi grup ya da gruplardan kaynaklandığını belirlemek amacı ile Kruskal-Wallis Post-Hoc testi yapılmış olup ilgili değerler Tablo 4.18'de verilmiştir.

Tablo 4.18.

Eğitsel Programlama Dilinin Derse Entegresyon Şekli, Ders İçi Performans Testi2'nin Bilişsel Beceri Ortalama Puanları Açısından Kruskal-Wallis Post-Hoc Testi Sonuçları

Performans Test Unsurları	**Örnek1-Örnek2**	**Test İstatistik**	**Standart Hata**	**Std.Test İstatistik**	**Sig.**	**Adj.Sig**
Performans Testi 2 'in bilişsel beceri değerleri	TEG-UEG	12.57	6.60	1.90	.057	.170
	TEG-TUEG	-17.26	6.65	-2.59	.009	.028
	UEG-TUEG	-4.68	6.54	-.71	.473	1.00

p< .05

Performans Testi 2'nin bilişsel becerileri puanları yönünden anlamlı farklılığın TEG ve TUEG grupları arasında olduğu Tablo 4.18'den anlaşılmaktadır.

4.4.1.3. Eğitsel programlama dilinin derse entegresyon şekli, ders içi performans testi 3'ün bilişsel beceri puanları arasında anlamlı bir fark oluşturmakta mıdır?

Ön-test ve Son-test arasında yapılan ve ders içi başarıyı belirlemeye yönelik olan Performans 3 testinin normallik/aykırı değerler sonuçları Tablo 4.19'da verilmiştir. Performans 3 testi normal dağılım göstermediği için parametrik olmayan testlerden Kruskal-Wallis testi kullanılmıştır

Tablo 4.19.

Performans 3 Testi Normalite Değerleri

	Entegrasyon Şekli	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Performans testi 3'ün Bilişsel Becerileri Not Ortalamaları	**UEG**	.126	30	.200*	.947	30	.138
	TEG	.197	28	.007	.950	28	.202
	TUEG	.238	29	.000	.890	29	.006

p< .05

Performans 3 testi değerlerinin normal bir dağılım gösterip göstermediğini belirlemek amacı ile yapılan test sonuçlarına göre normal bir dağılım görülmemektedir. Bu sonuça göre parametrik olmayan testler kullanılmıştır.

Tablo 4.20.

Eğitsel Programlama Dilinin Derse Entegresyon Şekli, Ders İçi Performans Testi 3' Ün Bilişsel Beceri Ortalama Puanları Açısından Kruskal-Wallistest Sonuçları

Performans Test Unsurları	Entegrasyon Şekli	Sayı (N)	Mean Rank	Chi-Square	df	Asymp. Sig.
Performans Testi 3' ün bilişsel beceri değerleri	**UEG**	28	45.45	11.18	2	.004
	TEG	28	32.00			
	TUEG	29	54.09			
	Toplam	87				

p< .05

Performans Testi 3'ün bilişsel becerileri puanları açısından anlamlı bir farkın olduğu anlaşılmaktadır. Bu farkın hangi grup ya da gruplardan kaynaklandığını belirlemek amacı ile Post-Hoc testi yapılmış olup ilgili değerler Tablo 4.21'de verilmiştir.

Tablo 4.21.

Eğitsel Programlama Dilinin Derse Entegresyon Şekli Ders İçi Performans Testi3'ün Bilişsel Beceri Ortalama Puanları Açısından Kruskal-Wallis Post-Hoc Testi Sonuçları

Performans Test Unsurları	Örnek1-Örnek2	Test İstatistik	Standart Hata	Std.Test İstatistik	Sig.	Adj.Sig
Performans Testi 3'ün bilişsel beceri değerleri	TEG-UEG	13.45	6.59	2.03	.041	.124
	TEG-TUEG	-22.08	6.65	-3.32	.001	.003
	UEG-TUEG	-8.63	6.53	-1.32	.186	.559

p< .05

Performans Testi 3'ün bilişsel becerileri puanları yönünden anlamlı farklılığın TEG ve TUEG grupları arasında olduğu Tablo 4.21'den anlaşılmaktadır.

4.4.1.4. Eğitsel programlama dilinin derse entegresyon şekli, ders içi performans testi 4'ün bilişsel beceri puanları arasında anlamlı bir fark oluşturmakta mıdır?

Çalışmanın başlangıcından bitimine kadar olan süre içerisindeki başarıyı belirlemeye yönelik yapılan testlerden Performans 4 testinin normallik/aykırı değerler sonuçları Tablo 4.22.'de verilmiştir. Performans 4 testi normal dağılım göstermediği için parametrik olmayan testlerden Kruskal-Wallis testi kullanılmıştır

Tablo 4.22.

Performans 4 Testi Normalite Değerleri

	Entegrasyon Şekli	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Performans testi 4'ün Bilişsel Becerileri Not Ortalamaları	UEG	.186	30	.010	.917	30	.022
	TEG	.145	28	.136	.953	28	.239
	TUEG	.178	29	.020	.881	29	.004

p< .05

Normalite testi, analizlerin hangi testlerle yapılması gerektiğine karar vermek için yapılmış olup, bu test sonucunda performans 4 testi değerleri normal bir dağılım

göstermediği görülmüştür. Bulunan sonuca göre parametrik olmayan testler kullanılmıştır.

Tablo 4.23.

Eğitsel Programlama Dilinin Derse Entegresyon Şekli, Ders İçi Performans Testi 4'ün Bilişsel Beceri Ortalama Puanları Açısından Kruskal-Wallistest Sonuçları

Performans Test Unsurları	**Entegrasyon Şekli**	**Sayı (N)**	**Mean Rank**	**Chi-Square**	**df**	**Asymp. Sig.**
Performans Testi 4' ün bilişsel beceri değerleri	**UEG**	30	40.83	9.55	2	.008
	TEG	28	35.55			
	TUEG	29	55.43			
	Toplam	87				

p< .05

Tablo 4.23'te görüleceği üzere, Performans Testi 4'ün bilişsel becerileri puanları açısından anlamlı bir farkın olduğu anlaşılmaktadır. Bu farkın hangi grup ya da gruplardan kaynaklandığını belirlemek amacı ile Post-Hoc testi yapılmış olup test sonuçları Tablo 4.24 verilmiştir.

Tablo 4.24.

Eğitsel Programlama Dilinin Derse Entegresyon Şekli, Ders İçi Performans Testi 4'ün Bilişsel Beceri Ortalama Puanları Açısından Kruskal-Wallis Post-Hoc Testi Sonuçları

Performans Test	**Örnek1-Örnek2**	**Test İstatistik**	**Standart Hata**	**Std.Test İstatistik**	**Sig.**	**Adj.Sig**
Performans Testi 4'ün bilişsel beceri değerleri	**TEG-UEG**	5.28	6.63	.796	.426	1.00
	TEG-TUEG	-19.87	6.68	-2.97	.003	.009
	UEG-TUEG	-14.59	6.57	-2.22	.026	.079

p< .05

Performans Testi 4'ün bilişsel becerileri puanları yönünden anlamlı farklılığın TEG ve TUEG grupları arasında olduğu Tablo 4.24'de görülmektedir.

4.4.1.5. Program görselleştirme aracının derse entegresyon şekli, ders içi performans testi 5'in bilişsel beceri puanları arasında anlamlı bir fark oluşturmakta mıdır?

Çalışma zamanı içerisindeki başarıyı belirlemeye yönelik yapılan testlerden Performans 5 testinin normallik/aykırı değerler onuçları Tablo 4.25.'de verilmiştir. Performans 5 testi normal dağılım göstermediği için parametrik olmayan testlerden Kruskal-Wallis testi kullanılmıştır.

Tablo 4.25.

Performans 5 Testi Normalite Değerleri

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Entegrasyon Şekli	**Statistic**	**df**	**Sig.**	**Statistic**	**df**	**Sig.**
Performans testi 5'in Bilişsel Becerileri Not Ortalamaları	**UEG**	.161	30	.045	.876	30	.002
	TEG	.104	28	.200*	.977	28	.764
	TUEG	.124	29	.200*	.895	29	.007

p< .05

Tablo 4.25'e göre analizlerin hangi testlerle yapılması gerektiğine karar vermek için yapılan normalite testi sonuçları görülmektedir. Analiz sonucunda performans 5 testi değerlerinin normal bir dağılım göstermediği görülmüştür. Bu sonuçlara göre parametric olmayan testler kullanılmıştır.

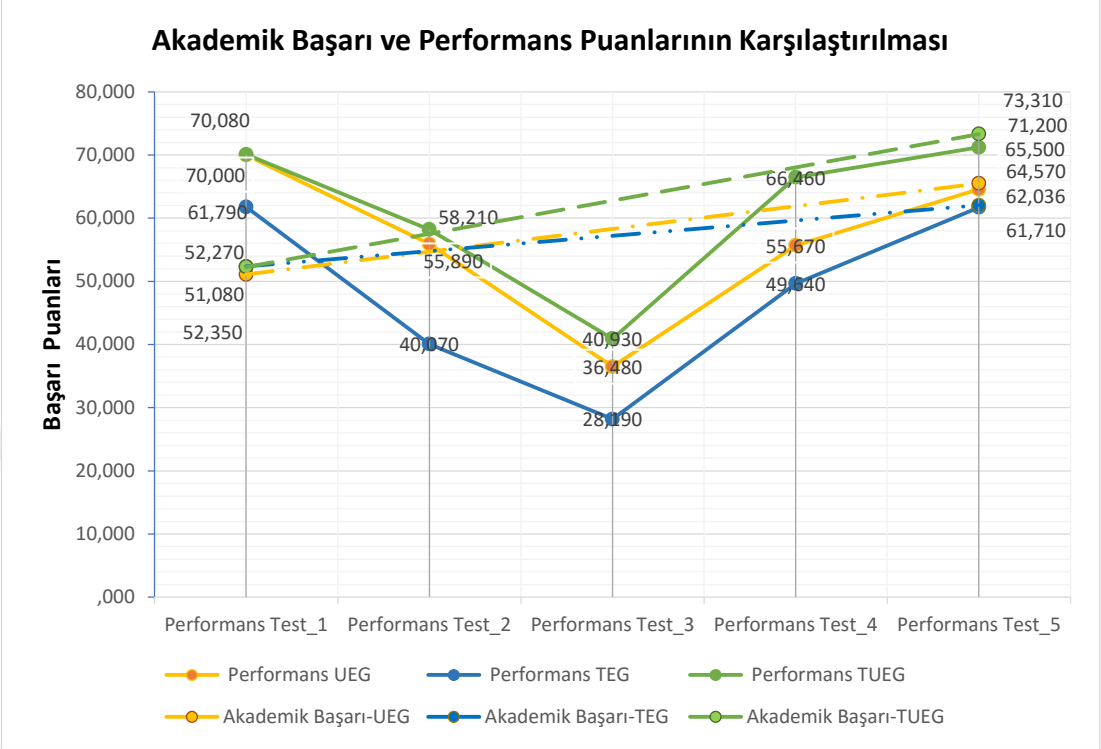
Tablo 4.26.

Eğitsel Programlama Dilinin Derse Entegresyon Şekli, Ders İçi Performans Testi 5'in Bilişsel Beceri Ortalama Puanları Açısından Kruskal-Wallis Test Sonuçları

Performans Test Unsurları	**Entegrasyon Şekli**	**Sayı (N)**	**Mean Rank**	**Chi-Square**	**df**	**Asymp. Sig.**
Performans Testi 5' in bilişsel beceri değerleri	**UEG**	30	41.92	2.37	2	.306
	TEG	28	40.21			
	TUEG	29	49.81			
	Toplam	87				

p< .05

Tablo 4.26'da görüleceği üzere gruplar arasında, Performans Testi 5' in bilişsel becerileri puanları açısından anlamlı bir farkın olmadığı anlaşılmaktadır.



Şekil 4.1. Akademik başarı ve performans testlerinin puan ortalamaları karşılaştırılması

Eğitsel programlama dilinin derse entegrasyon şekline göre öğrencilerin akademik başarı ve performans puan ortalamaları arasındaki ilişki Şekil 4.1'de grafiksel olarak gösterilmiştir. UEG grubun başarı ve performans testleri puan ortalamaları; ön-test puan ortalaması $\overline{X}$ = 51.08, performans 1 testi puan ortalaması $\overline{X}$ = 70.00, performans 2 testi puan ortalaması $\overline{X}$ = 55.89, performans 3 testi puan ortalaması $\overline{X}$ = 36.48, performans 4 testi puan ortalaması $\overline{X}$ = 55.67, performans 5 testi puan ortalaması $\overline{X}$ = 64.57 ve son-test puan ortalaması $\overline{X}$ = 65.50 olarak bulunmuştur. TEG grubun başarı ve performans testleri puan ortalamaları; ön-test puan ortalaması $\overline{X}$ = 52.35, performans 1 testi puan ortalaması $\overline{X}$ = 61.79, performans 2 testi puan ortalaması $\overline{X}$ = 40.07, performans 3 testi puan ortalaması $\overline{X}$ = 28.19, performans 4 testi puan ortalaması $\overline{X}$ = 49.64, performans 5 testi puan ortalaması $\overline{X}$ = 61.71 ve son-test puan ortalaması $\overline{X}$ = 62.04 olarak bulunmuştur. TUEG grubun başarı ve performans testleri puan

ortalamaları; ön-test puan ortalaması $\overline{X}$ = 52.27, performans 1 testi puan ortalaması $\overline{X}$ = 70.08, performans 2 testi puan ortalaması $\overline{X}$ = 58.21, performans 3 testi puan ortalaması $\overline{X}$ = 40.93, performans 4 testi puan ortalaması $\overline{X}$ = 66.46, performans 5 testi puan ortalaması $\overline{X}$ = 71.20 ve son-test puan ortalaması $\overline{X}$ = 73.31 olarak bulunmuştur.

4.4. Bilgisayar Programlama Kaygı Testi Paired Testi Varsayımları

Eşleştirilmiş iki grup (Paired Samples t) testi uygulanabilmesi için normallik/aykırı değerler ve varyans eşitliği varsayımları Tablo 4.27. ve Tablo 4.28.'de verilmiştir.

Tablo 4.27.

Bilgisayar Programlamaya Yönelik Kaygı Testi Normalite Test Değerleri

		Kaygı Son-Test	**Kaygı Ön-Test**
N		81	81
Normal Parametreler [a.b]	Ortalama	1.34	1.85
	Standart Sapma	.84	.91
	Mutlak Değer	.12	.12
Uç Farklılıklar	Pozitif	.12	.11
	Negatif	-.08	-.12
Kolmogorov-Smirnov Z		1.1	1.1
Asymp. Sig. (2-tailed)		.141	.169

p< .05

Bilgisayar programlamaya yönelik kaygı Ön-Test normalite testi anlamlılık düzeyi p = .169 ve bilgisayar programlamaya yönelik kaygı Son-Test normalite testi anlamlılık düzeyi p = .141 olarak bulunmuştur. Bilgisayar programlamaya yönelik kaygı testi anlamlılık düzeyleri p > .05 olduğundan normal bir dağılım göstermektedirler. Bu sonuçlara göre parametrik testler kullanılmıştır.

Tablo 4.28.

Bilgisayar Programlamaya Yönelik Kaygı Testi Homejenlik Test Değerleri

	Homojenlik Testi Değerleri			
	Levene Statistic	**df1**	**df2**	**Sig.**
Programlama Kaygısı Ön-Test	1.10	2	78	.336
Programlama Kaygısı Son-Test	2.81	2	78	.066

p< .05

Bilgisayar programlamaya yönelik kaygı düzeyi için yapılan homojenlik testi sonuçları bilgisayar programlamaya yönelik kaygı Ön-Test anlamlılık düzeyi p = .336 ve bilgisayar programlamaya yönelik kaygı Son-Test anlamlılık düzeyi p = .066 olarak bulunmuştur. Bilgisayar programlamaya yönelik kaygı anlamlılık düzeyleri p > .05 olduğundan homojen bir dağılım göstermektedirler. Bu sonuçlara göre parametrik testler kullanılmıştır.

4.5. Bilgisayar Programlamaya Kaygı Testi Ancova Varsayımları

Tek yönlü kovaryans analizi ANCOVA testi uygulanabilmesi için normallik/aykırı değerler ve varyans eşitliği değerleri Tablo 4.27. ve Tablo 4.28.'de verilmiştir. Tek yönlü kovaryans ANCOVA testlerinin uygulanabilmesi için diğer bir varsayımsa değişkenlerin birbirinden bağımsız olmasıdır.

Tablo 4.29.

Bilgisayar Programlamaya Yönelik Kaygı Testi Bağımsız Değişkenler Test Değerleri

		Kaygı Ön-Test	**Kaygı Son-Test**
Kaygı Ön-Test	Pearson Correlation	1	.693**
	Sig. (2-tailed)		.000
	N	87	87
Kaygı Son-Test	Pearson Correlation	.693**	1
	Sig. (2-tailed)	.000	
	N	87	87

**. p < .01

Bilgisayar programlamaya yönelik Kaygı Son-Test ile Kaygı Ön-Test arasında 0.693 yüksek derecede pozitif bir ilişki vardır. Anlamlılık değeri ise p< .05'den küçük olduğu için anlamlıdır.

4.5. Bilgisayar Programlama Kaygısına Yönelik Bulgular

4.5.1. Grupların bilgisayar programlama kaygısına yönelik ön-test son-test puan ortalamaları

Eğitsel programlama dilinin dersin teori kısmına entegre edilmiş grup (TEG), dersin uygulama kısmına entegre edilmiş grup (UEG) ve dersin hem teori hem uygulama kısmına entegre edilmiş (TUEG) gruplarına uygulanmış bilgisayar programlamaya yönelik kaygı testinin gruplar üzerindeki etkilerine yönelik sonuçları tablolarla ortaya konulmuştur.

Tablo 4.30.

Çalışma Gruplarının Bilgisayar Programlama Kaygısına Yönelik Ön-Test ve Son-Test Puan Ortalamaları

Entegrasyon Şekli	Ön-Test Ortalama($\overline{X}$)	Son-Test Ortalama($\overline{X}$)
UEG	2.03	1.19
TEG	1.37	1.49
TUEG	2.14	1.34

Tablo 4.30' da görüldüğü gibi, UEG ön test puan ortalamalarının $\overline{X}$ = 2,03 olurken, son-test puan ortalamalarının $\overline{X}$ =1,19 olduğu; TEG grubunun ön-test puan ortalamalarının $\overline{X}$ =1,37, son-test puan ortalamalarının ise $\overline{X}$ =1,49 olduğu; TUEG grubunun ön-test puan ortalamalarının $\overline{X}$ =2,14 ve son test puan ortalamalarının $\overline{X}$ =1,34 olduğu görülmektedir. Yani UEG ve TUEG gruplarında kaygı düzeyinde anlamlı bir fark görülürken TEG'de ise anlamlı bir fark görülmemektedir.

4.5.2. Eğitsel programlama dilinin derse entegrasyon şekli, öğrencilerin bilgisayar programlamaya yönelik kaygıları arasında anlamlı bir fark oluşturmakta mıdır?

Öğrencilerin bilgisayara yönelik kaygı düzeylerinin incelendiği eşleştirilmiş örneklem t-testi verileri Tablo 4.31.'de verilmiştir.

Tablo 4.31.

Eşleştirilmiş Örneklem T-Testi (Paired Samples t-testi) Ortalamaları

Entegrasyon Şekli			Ortalama	N	Standart Sapma	Std. Ortalama Hata
UEG	Pair 1	Kaygı Ön-Test	2.03	28	.923	.174
		Kaygı Son-Test	1.19	28	.765	.144
TEG	Pair 1	Kaygı Ön-Test	1.37	27	.773	.148
		Kaygı Son-Test	1.49	27	.765	.147
TUEG	Pair 1	Kaygı Ön-Test	2.14	26	.859	.168
		Kaygı Son-Test	1.34	26	1.00	.197

Tablo 4.32.

Eşleştirilmiş Örneklem t-Testi (Paired Samples t-Test)

		Ortalama	Standart Hata	Std. Ortalama Hata	95% Güven Aralığı Farkı		t	df	Sig. (2-tailed)
					Alt	Üst			
UEG	**Kaygı Ön-Test Kaygı Son-**	.84	1.09	.206	.41	1.26	4.09	27	.000
TEG	**Kaygı Ön-Test Kaygı Son-**	-.12	.98	.190	-.51	.26	-.64	26	.529
TUEG	**Kaygı Ön-Test Kaygı Son-**	.80	1.22	.239	.31	1.2	3.36	25	.002

p< .05

Tablo 4.32'ye göre gruplar arasında eşleştirilmiş örneklem t-testi sonuçlarından anlaşılacağı üzere, UEG ve TUEG gruplar içerisinde bilgisayar programlamaya yönelik kaygı düzeyi açısından anlamlı bir fark görülürken; TEG olan grupta anlamlı bir fark görülmemektedir.

5.4.3. Eğitsel programlama dilinin derse entegrasyon şekli, öğrencilerin bilgisayar programlamaya yönelik ön kaygıları kontrol altına alındığında, son-test kaygı düzeyleri arasında anlamlı bir fark oluşturmakta mıdır?

Üç grubun son-test bilgisayar programlamaya yönelik kaygı düzeyinin belirlemesi öncesi test sonuçlarının etkilerini ortadan kaldırmak için, kovaryans analizi tek ortak değişken ile (ANCOVA) yapılmıştır. Tek yönlü Kovaryans analizi (ANCOVA) bilgisayar programlamaya yönelik ön-test kaygı düzeyleri kontrol altına alınarak, eğitsel programlama dilinin derse entegrasyon şekli ile bilgisayar programlamaya yönelik son-test kaygı düzeyi puanları arasındaki ilişkiyi değerlendirmek amacıyla yapılmıştır.

Tablo 4.33.

Kaygı Son-Test Ortalamalar.

Entegrasyon Şekli	Ortalama ($\overline{X}$)	Standart Sapma	N
UEG	1.19	.766	28
TEG	1.49	.765	27
TUEG	1.34	1.01	26
Toplam	1.34	.85	81

Grupların bilgisayar programlamaya yönelik kaygı düzeyi ortalamalarının belirlenmesi ilişkin test sonucunda. UEG' nin ($\overline{X}$ = 1.19) , TUEG'nin ($\overline{X}$ = 1.34) ve TEG'nin ise ($\overline{X}$ =1.49) olduğu Tablo 4.33'de görülmektedir.

Tablo 4.34.

Öğrencilerin Bilgisayar Programlamaya Yönelik Ön Kaygı Düzeyleri Kontrol Altına Alındığında, Bilgisayar Programlamaya Yönelik Son-Test Öğrenci Kaygı Düzeylerini ANCOVA Sonuçları

Kaynak	Karelerin Toplamı Tip III	df	Karelerin Ortalaması	F	Sig.
Düzeltilmiş Model	2.88	3	.96	1.35	.264
Kesen	14.73	1	14.73	20.74	.000
Kaygı_On-Test	1.60	1	1.60	2.25	.137
Entegrasyon Şekli	2.18	2	1.09	1.53	.222
Hata	54.71	77	.71		
Toplam	203.39	81			
Düzeltilmiş Toplam	57,60	80			

p< .05

Verilerin analizinde tek yönlü kovaryans (ANCOVA) kullanılmıştır. Tek yönlü kovaryans analizi (ANCOVA) sonucları [F(2-81)= 1.536, p= .222] gruplar arasında bilgisayar programlamaya yönelik ön kaygılar kontrol altına alındığında son-test kaygı düzeyi ortalama puanları arasında istatistiki olarak anlamlı bir farkın olmadığı Tablo 4.34'te görülmektedir.

4.6. Bölüm Özeti

Çalışmada eğitsel programlama dilinin derse entegrasyon şeklinin akademik başarılar arasında anlamlı bir fark oluşturduğu görülmektedir. Bu başarının TUEG grubu ile TEG grup ve UEG drup arasında ve TUEG grup lehinde oluştuğunu görülmektedir. Süreç içerisinde akademik başarı testine uyumluluk gözteren ders içi performans testlerinde performans 1, 2, 3 ve 4 testlerinde TUEG grup ile TEG arasında, TUEG lehine anlamlı bir fark görülürken diğer gruplar arasında anlamlı bir fark bulunamamıştır. Ders içi performans 5 testinde ise gruplar arasında anlamlı bir fark bulunamamıştır. Eğitsel programlama dilinin derse entegrasyon şeklinin bilgisayar programlama kaygı düzeyleri arasında TUEG grup ve UEG grupta anlamlı bir fark oluştururken, TEG grupta anlamlı bir fark oluşturmamıştır.

BEŞİNCİ BÖLÜM

5. TARTIŞMA, SONUÇ VE ÖNERİLER

Bu çalışmada eğitsel programlama dilinin programlama derslerine entegrasyon şeklinin öğrencilerin akademik başarılarına, ders içi performanslarına ve bilgisayar programlama kaygı düzeylerine etkisinin araştırılması amaçlamıştır. Bu amaça bağlı alt amaçlar test edilmiş ve elde edilen bulgular bu bölümde tartışılmıştır.

5.1. Akademik Başarı

Bu çalışmada öğrencilerin başarılarını belirlemeye yönelik olarak hazırlanan başarı testi ön-test son-test olarak uygulanmış ve elde edilen bulgular bu bölümde tartışılmıştır.

5.1.1. Grupiçi akademik başarı

Eğitsel programlama dilinin derse entegrasyon şeklinin öğrenci başarıları arasındaki farkları tespite yönelik olarak yapılan t-testi bulgularına göre UEG, TEG ve TUEG gruplarının akademik başarı ön-test ile akademik başarı son-test puanları arasında anlamlı farklılık bulunmuştur. Bu farkın çalışmada kullanılan eğitsel programlama dilinden kaynaklandığı düşünülebileceği gibi, yapılan bu çalışma ve bütün deneysel çalışmalarda beklenen bir sonuç olması yönüyle değerlendirilebilir. Ayrıca farkın verilen eğitimden kaynaklandığı ve üç grupta da akademik başarıda artışın görülmesi, eğitimin üç grup için de benzer bir şekilde verildiğinin bir göstergesi olduğu söylenebilir.

5.1.2. Gruplar arası akademik başarı

Eğitsel programlama dilinin derse entegrasyon şeklinin öğrencilerin ön bilgileri kontrol altına alındığında son-test bilişsel beceri puanları arasındaki farka yönelik olarak bulgular incelendiğinde, bilişsel beceri ve üst düzey bilişsel becerileri puanları

acısından, eğitsel programlama dilinin programlama derslerinin teori kısmına veya uygulama kısımına entegre edilmesinin gruplar arasında anlamlı bir fark oluşturmadığı görülmektedir. Eğitsel programlama dilinin programlama derslerinin hem teori hem de uygulama kısmına entegre edilmesinin gruplar arasında anlamlı bir fark oluşturduğu görülmektdir. Alt düzey billişsel becerileri puanları acısından bakıldığında ise gruplar arasında anlamlı bir farklılığın oluşmadığı görülmektedir.

Bu bulgular doğrultunda; programlama dersinde öğrencilerin, bir programı anlayabilmesi, işlem satırlarını takip edebilmesi, değerlendirebilmesi ve sonuça oluştırabilmesi için analiz, sentez, çözümleme ve değerlendirme gibi üst düzey bilişsel becerilere sahip olması gerekmektedir (Çakıroğlu, Sarı ve Akkan, 2011). Bu becerilerin program yazma da etkili olması bilişsel ve üst düzey bilişsel beceriler acısından anlamlı bir farka yol açmış olabilir.

Traynor ve Gibson, (2004) programlamaya yönelik becerilerin öğrencilere kazandırılmasının önemli olduğunu; Jenkins (2002) programlamaya yönelik becerilerin elde edilmesinin ne kadar önemli olduğunu, bunun da programlama konularının doğasından kaynaklandığını; Başer (2012) öğrencilerin programlama konularından nesne yönelimli konularda ve programlama becerisi olarak ise programlama hatasını bulmada zorlakdıklarını; Wiedenbeck (2005) programlama derslerinde başarıya etki eden faktörleri programlamaya yönelik ön deneyim, bilgi ve öz yeterlilik olduğunu; Pillay ve Jugoo (2005) programlama konusunda acemilerin yaşadıkları zorlukların programlama performansını etkilediğini ve bu etkinin üst düzey bilişsel becerilerden problem çözme becerisi ile anlamlı bir ilişkisinin olduğu; Marcoulides (1988) bilgisayar becerileri ile bilgisayar kaygısı arasında pozitif bir ilişkinin olduğu; Tsai ve Tsai (2003) öğrencilerin anlama izleme ve çözümleme gibi bilişsel beceriler, programlamanın öğrenilmesinde etkili olduğunu tespit etmişlerdir. Bu araştırmalar programlamaya yönelik üst düzey bilişsel becerilerin öğrencilere kazandırılmasının önemli olduğunu bu becerilerin programlama performansını ve progragramlamaya yönelik kaygıyı etkilediğini göstermesi yönüyle çalışmanın sonuçlarını desteklediği görülmektedir.

Bu sonuçlar eğitsel programlama dillerinin programlama öğretimde dersin sadece teori kısmına entegre edilmesinin akademik başarıya etki etmediğini, uygulama kısmına entegre edilmesinin ise bir fark oluşturduğunu, fakat bu farkın anlamlı

olmadığını göstermektedir. Bu etkinin tam kendini gösterebilmesi için programlama derslerinde eğitsel programlama dilinin dersin hem teori hem de uygulama kısmına entegre edilmesi gerekmektedir. Beklenenin aksine, yapılan çalışmada teori entegre grup ve uygulama entegre grup arasında anlamlı bir farklılığın oluşmaması ve uygulama entegre grup ile hem teori hem de uygulama entegre grup arasında anlamlı bir farklılığın oluşması bu araştırmanın en göze çarpan sonucudur. Bu sonuçta üçüncü grup olarak hem teori hem de uygulama entegre grubun oluşturulmasının, yapılan çalışmada gruplar arasında anlamlı bir akademik başarı farklılığının oluşmasında ne denli önemli olduğunu ortaya koymuştur.

Programlama öğretiminde eğitsel programlama dilinin dersin hem teori hem de uygulama kısmında kullanılması, teoride alınan bilginin bilişsel olarak kavranmasına ve uygulamada kullanılmasına yardımcı olmuş olabilir. Wang ve Zhou (2011), yaptıkları çalışmada eğitsel programlama dilinin öğrencilerin ilgisini uyandırdığını; Garner, (2009), üniversite öğrencilerinin programlama temelleri dersinde eğitsel programlama dilinin kullanımının yüksek seviyeli dillere geçişte kolaylık sağladığını; Wang ve Zhou (2011), eğitsel programlama dillerinin kullanımı öğrencilerin, program yazımı söz dizimine geçişlerini kolaylaştırdığını belirtmişlerdir.

Eğitsel programlama dilinin programlama öğretiminde dersin hem teori hem de uygulama kısmında kullanılması öğrencilerin daha fazla eğitsel programlama dili ile meşgul olmasına, programlama öğrenimi üzerinde bir katalizör etkisi yaparak bilişsel yükün azalmasına ve öğrenmelerinin olumlu yönde etkilenmesine neden olmuş olabilir. Grissom, McNally ve Naps (2003), yaptıkları meta analiz çalışmasında, eğitsel programlama dilleri ile öğrencilerin meşguliyetinin artması öğrenmeyi etkilediği; Chiu, (2015), eğitsel programlama dillerinin öğrencilerin ilgisini çektiğini, bu ilgi sonucunda eğitsel programlama dil ortamlarında daha fazla zaman geçirdikleri ve bu durumun öğrencilerin programlama deneyimlerini geliştirdiğini belirtmiştir.

Fesakis ve Serafeim (2009), öğrencilerin eğitsel programlama dili ile ilgili olarak, bilgisayar bilimine giriş ve programlama uygulamalarında, etkili uygulamaların benimsenmesi olanaklarını artırdığını ve bilgisayar programlama eğitiminde pozitif bir etkiye sahip olduğunu; Klassen (2006), programlama ders konularının geleneksel teorik yöntemler yerine, görseller veya animasyonlarla oluşturulan eğitsel programlama dilleri

ile verilmesinin öğrencilerin akademik başarılarını olumlu yönde etkilediğini; Hansen, Schrimpsher ve Narayanan (1998) eğitsel programlama dillerinin programlama derslerinde kullanılması, metin tabanlı öğretim tekniklerinin derslerde kullanılmasından daha fazla akademik başarı üzerinde etki gösterdiği; Arabacıoğlu, Bülbül ve Filiz (2007), öğrencilerin başarısının artırılabilmesi için eğitsel programlama dillerinin önemli olduğunu ve bu dillerinin kullanılmasının programlama öğretiminde etkili olacağını; Kaucic ve Asic (2011), eğitsel programlama dili gibi basite indirgenmiş bir ortamın programlama öğetiminde kullanılmasının, programlama öğrenimini olumlu yönde etkilediğini; Peppler ve Kafai, (2007) iki yıl süren bir çalışmada katılımcılara eğitsel programlama dili kullanılarak bilgisayarda video oyunu tasarlamanın, programlama öğrenmede etkili olduğunu; Cooper, Dann, ve Pausch (2003)' in Ithaca College ve Saint Joseph Üniversitesi'nde eğitsel programlama dillerinin öğrencilerin başarı düzeyine olumlu etkisinin olduğunu; Hundhausen, Douglas ve Stasko (2002) eğitsel programlama dilinin öğrenme üzerinde pozitif bir etkisi olduğunu ve öğrencilerin akademik başarılarını olumlu yönde etkilediği bulgularıyla bu sonucu desteklemektedir.

Çalışmanın bu sonucunu desteklemeyen Ramadhan (2000), eğitsel programlama dilinin kullanıldığı deney / kontrol gruplu çalışmasında, gruplar arasında son-test başarı puanlarında anlamlı bir farkın oluşmadığını; Tekerek ve Altan (2014), kontrol gruplu ön-test son-test deseninin kulanıldığı çalışmada, deney ve kontrol gruplarının son-test puanları arasında anlamlı bir fark oluşmadığını ve eğitsel programlama dilinin öğrenci başarısı üzerine olumlu bir etkisinin olmadığını belirlemişlerdir.

5.2. Ders İçi Performans

Bu çalışmada öğrencilerin ön-test son-test akademik başarıların ölçülmesinin yanı sıra uygulama süreci içerisinde de haftalık performanslarının belirlenmesine yönelik olarak beş adet performans testi uygulanmıştır. Bu testler başarı testleri ile içerik olarak aynı soru olarak farklılık gösteren haftalık olarak ders sonunda yapılanan testlerdir. Haftalık yapılan performans testleri ön-test son-test olarak uygulanan akademik başarı testinin sonuçlarının teyitini sağlamak ve güvenirliliğini artırmak için uygulanmıştır. Uygulanan bu testlerden elde edilen bulgular bu bölümde tartışılmıştır.

5.2.1. Eğitsel programlama dilinin derse entegrasyon şekli, ders içi performans testlerinin bilişsel beceri puanları arasında anlamlı bir fark oluşturmakta mıdır?

Performans Testi 1, 2, 3 ve 4'e göre gruplar arasında bilişsel beceriler acısından anlamlı farkın olduğu görülmektedir. Bu farkın hangi grup ya da gruplardan kaynaklandığını belirlemek amacı ile Kruskal-Wallis Post-Hoc testi yapılmış olup, Performans Testi 1, 2, 3 ve 4'in bilişsel beceriler yönünden anlamlı farklılığın TUEG' nin TEG' den olduğu görülmektedir. Eğitsel programlama dilinin öğrencilerin performansına etkisinin araştırıldığı bu çalışmada, eğitsel programlama dilinin hem teori hem de uygulama entegre grupta diğerlerine göre daha etkili olduğu, özellikle soyut olan programlama derslerinin somutlaştırılmasının, öğrenme üzerinde olumlu bir etkiye sahip olduğu anlaşılmaktadır. Ayrıca akademik başarı testinin bir nevi teyiti olan performans testlerindeki sorular, akademik başarı testinde olduğu gibi üst düzey bilişsel becerilere yönelik olduğundan, hem teori hemde uygulama entegre grup ile teori entegre grupta ders içi performans test puanları açısından anlamlı bir fark oluşturmuş olabilir. Ders içi performans testlerinde uygulama entegre grupla hem teori hem de uygulama entegre grup arasında performans test puanları açısından anlamlı bir farklılığın oluşmamasının sebebi ders içi performans test soru sayısının azlığı olabilir. Bu nedenlerle eğitsel programlama dilinin programlama öğretiminde ders içi performans başarısı açısından, derslerin işlenişinde dersin hem teori hem de uygulama kısmına entegre edilmesi gerektiği söylenebilir.

Performans Testi 5'e göre gruplar arasında bilişsel beceriler acısından anlamlı farkın olmadığı görülmektedir. Öğrencilerin eğitsel programlama dili ile uzun süre çalışmış olmaları ve bu eğitsel programlama dilinin daha etkili kullanmayı öğrenmiş olmaları gibi etkenler performansı her gurupta aynı düzeyde etkilemiştir. Dolayısıyla performansın her grupta aynı oranda etkilenmesi, testte gruplar arasında anlamlı bir farklılığın oluşmamasının sebebi olarak görülebilir.

Naser (2008), A1-Azhar Üniversitesi'nde bir grup öğrenci üzerinde gerçekleştirdiği ve eğitsel programlama dilinin öğrencilerin performansına etkisini araştırdığı çalışmasında, geleneksel yöntemlerle karşılaştırıldığında uygulanan yeni eğitsel programlama dillerinin, öğrenci performansına daha olumlu etki yaptığı ve

ayrıca yeni versiyon eğitsel programlama dillerinin, eski versiyon eğitsel programlama dillerine oranla öğrencilerin başarısı üzerinde daha etkili olduğu; Rizvi, Humphries, Major, Jones ve Lauzun (2011), zayıf matematik bilgisi olan öğrencilere, eğitsel programlama dilleri yardımıyla programlama derslerinin verilmesini sağlamış ve çalışma sonucunda öğrencilerin öz-yeterliliklerinde, performanslarında ve başarılarında bir önceki yıla oranla artış gözlendiği; Bishop-Clark, Courte ve Howard (2007), programlama dersinde eğitsel programlama dillerini kullanmışlar ve araştırma sonucunda eğitsel programlama dilini kullanan öğrencilerin programlama yapılarını ve kavramlarını daha iyi anladığı; Buche, Davis ve Vician (2007), bilgisayar kaygısı ile performans arasındaki ilişkinin incelendiği çalışmada, bilgisayara yönelik kaygının, bilgisayarla yoğun bir öğrenme ortamına maruz kalındığında değişikliğe uğradığını göstermektedir. Bilgisayar öğretim ortamında bireyin düşük kaygısı, yüksek kaygıya doğru giderse performansı gelişirken, bireyin yüksek kaygısı iyice yükselirse performans bundan negatif yönde etkilenmektedir. Bu araştırmaların bulguları yapılan çalışmanın performans testlerine yönelik sonucunu desteklemektedir.

Akademik başarı ve performans testlerinin puan ortalamaları karşılaştırıldığında başarı ön-test ve son-test ile performans testlerinin puanlarının birbirine paralel olduğu görülmektedir. Yine bu karşılaştırmada UEG, TEG ve TUEG gruplarının da kendi içlerinde birbirine yakın oldukları ve paralel hareket ettikleri görülmektedir. Gerek ön-test son-test, gerekse performans testlerinde olsun grupların başarı sıralaması değişmemiştir. Ön-test ve son-test'de TUEG en yüksek puana sahip, UEG puan olarak onu takip etmekte ve TEG ise en düşük puana sahiptir. Performans testleride incelendiğinde bu sıralamanın ön-test son-test ile benzerlik gösterdiği görülmektedir. Performans test 3' ün içeriğinde matematiksel ifadelerin çok olması ve öğrencilerin de bu konudaki yetersizlikleri, bu testin puanlarının düşük olmasına sebep olduğu düşünülmektedir. Ayrıca bu çalışmada araştırmacı, başarıyı etkileyebilecek olumsuz durumların etkisini belirlemek için performans testlerinden elde edilen sonuçlarla başarı testlerinden elde edilen sonuçların uyumuna bakarak akademik başarı testinin güvenirliliğini belirlemeyi hedeflemiştir. Hem akademik başarı ön-test son-testin hem de performans testlerinin sonuçlarının gruplar yönüyle uyumlu olması, performans testlerinin de akademik başarı testini sonuç olarak desteklediği bulgusunu ortaya koymaktadır. Bu bulgu, çalışmadaki akademik başarı ön-test son-test puanlarının

güvenilirliğini olumlu yönde etkilemiştir denilebilir.

5.3. Bilgisayar Parogramlama Kaygısı

Bu çalışmada eğitsel programlama dilinin derse entegrasyon şeklinin öğrencilerin bilgisayar programlamaya yönelik kaygı düzeyine etkisi ön-test son-test olarak belirlenmiş ve elde edilen bulgular bu bölümde tartışılmıştır.

5.3.1. Eğitsel programlama dilinin derse entegrasyon şekli, öğrencilerin bilgisayar programlamaya yönelik kaygıları arasında anlamlı bir fark oluşturmakta mıdır?

Tablo 4.29'da görüldüğü gibi, UEG ön test puan ortalamalarının 2,03 olurken, son-test puan ortalamalarının 1,19 olduğu; TEG grubunun ön-test puan ortalamalarının 1,37, son-test puan ortalamalarının ise 1,49 olduğu; TUEG grubunun ön-test puan ortalamalarının 2,14 ve son test puan ortalamalarının 1,34 olduğu görülmektedir. Bu ortalamalara bakıldığında UEG ve TUEG' nin ön-test kaygı ortalama puanlarının yüksek olduğunu, son-test puanlarının ise düşük olduğu görülmektedir. Bu bulgudan haraketle verilen eğitimin, bilgisayar programlama kaygısını azalttığı anlaşılmaktadır. Bununla beraber TEG' nin ön-test puan ortalamasının diğerlerine göre düşük olduğu ve eğitim sonucunda da düşme değil, az da olsa arttığı görülmektedir. Bu artışın, teori entegre grubunun çalışmanın yapıldığı süreç boyunca gerek akademik başarı gerekse performans test puanları yönüyle diğer gruplardan puanlarının düşük olması ve alanyazındaki araştırmalardan da anlaşılacağı gibi kız öğrenci sayısının diğer gruplara oranla daha fazla olmasından kaynaklı olmuş olabilir. Chua, Chen ve Wong (1999), yaptıkları meta-analiz çalışmasında kızların erkeklere oranla bilgisayar kaygı düzeylerinin fazla olduğunu; (McInerney,1990;Namlu ve Ceylan, 2002; Tzavara ve Komis, 2003) cinsiyet olarak kızların bilgisayar kaygısının erkeklerden fazla olduğunu belirtmişlerdir.

Ancak, teori entegre gruptaki kaygı düzeyinde bir artış söz konusu olsada bu artış istatistiksel olarak anlamlı düzeyde olmadığı görülmektedir. Connolly, Murphy ve Moore (2007) bu bulguyu destekleyen çalışmalarında, öğrencilerin özellikle bilgisayar programcılığına giriş dersindeki soyut ve karmaşık yapıları anlamakta ve bu soyut

yapıları somutlaştırmakta güçlük çektikleri, zorlandıkları ve stres ve kaygı yaşadıkları; Baloğlu ve Çevik (2008) öğrencilerin bilgisayar kaygılarının olduğunu ve buna bağlı olarak programlama öğretiminde de kaygı yaşayabilecekleri; Fesakis ve Serafeim (2009) Eğitsel programlama dillerinin stres ve kaygı yüzdesini azalttığı; Tzavara ve Komis (2003), kurslara katılımın, öğrencilerin stres seviyelerini aşağı çekmede yardımcı olacağını; Bishop-Clark, Courte ve Howard (2007), eğitsel programlama dillerinin programlamaya olan güveni artırdığı ve kaygıyı azalttığı bulgularıyla bu sonucu desteklemektedirler. Gürcan-Namlu (2003), bilgisayar programlama kaygısının başarı üzerindeki etkisini incelemiştir. Deney ve kontrol gruplu çalışması sonucunda, deney grubunda bulunan öğrencilerin bilgisayara yönelik kaygı düzeylerinde azalma olduğu bulgusuna ulaşmıştır.

Gruplar arasında bilgisayar programlama kaygısının olup olmadığına bakıldığında, programlama kaygısına yönelik olarak yapılan ön-test ve son-testler sonucunda UEG ve TUEG grupları kendi içlerinde kaygı düzeyi açısından anlamlı bir fark oluşurken, TEG grupta ise kendi içerisinde anlamlı bir fark olmadığı bulgusuna ulaşılmıştır. Bilgisayar eğitiminin programlamaya yönelik kaygıyı azalttığını belirten alanyazında birçok araştırma bulunmaktadır (Martocchio, 1992; Maurer ve Simonsen, 1993; Hakinken, 1994; Russell ve Bradley, 1996; Ayersman ve Reed, 1996; Bradley ve Russell, 1997). Bu araştırmalardan bir tanesinde Leso ve Peck (1992), ders işlenişinde programlama aracı kullanıldığı ve kullanılmadığı şekliyle iki gruba ayırmışlardır. Bu gruplara bilgisayara yönelik kaygı testi ön-test ve son-test şeklinde uygulanmış olup, programlama aracı kullanılan grubun kaygı düzeyinin, araç kullanılmayan gruba göre daha düşük olduğu bulunmuştur.

5.3.2. Eğitsel programlama dilinin derse entegrasyon şekli, öğrencilerin bilgisayar programlamaya yönelik ön kaygıları kontrol altına alındığında, son-test kaygı düzeyleri arasında anlamlı bir fark oluşturmakta mıdır?

Eğitsel programlama dilinin derse entegrasyon şeklinin UEG, TEG ve TUEG gruplarıda, ön kaygılar kontrol altına alındığında ön-test son-test kaygı düzeyleri arasındaki ilişkiye kovaryans analizi ile bakılmış olup, gruplar arasında anlamlı bir farkın oluşmadığı görülmektedir. Bu sonucu destekleyen çalışmalarında McInerney, McInerney ve Sinclair, (1994), bilgisayar ve programlama eğitiminin tek başına

bilgisayar ve programlamaya yönelik kaygı düzeyini düşürmede yeterli olmadığı; (Chua, Chen ve Wong 1999; Goldstein, Dudley, Erickson ve Richer, 2002; Namlu ve Ceyhan 2002) bilgisayar deneyimi ile bilgisayara yönelik kaygı arasında negatif yönlü bir ilişki olduğu; Necessary ve Parish (1996), öğrencilerin bilgisayar kullanım saatleri arttıkça buna bağlı olarak bilgisayar deneyimlerinin arttığı ve bunun sonucunda bilgisayara yönelik kaygının azaldığı görülmektedir.

Eğitsel programlama dilinin programlama dersine entegrasyon şeklinin öğrencilerin ön kaygıları kontrol altına alındığında bilgisayar programlama kaygı düzeyinde fark oluştuğu, fakat oluşan bu farklılığın anlamlı olmadığı görülmektedir. Bu bulgu eğitsel programlama dilinin, programlama derslerinde kullanımının bilgisayar programlama kaygı düzeyini olumlu yönde etkilediğini göstermektedir. Bu etkinin bütün gruplarda verilen eğitimin sonucunda oluşması, çalışmanın sağlıklı bir şekilde gerçekleştiğini göstermektedir.

5.4. Öneriler

Uygulayıcılara Öneriler

- Eğitsel programlama dilleri programlama öğretiminde sadece teori ya da sadece uygulamada kullanılmasından ziyade dersin hem teori hem de uygulama kısmına entegre edilerek kullanılmalıdır.
- Eğitsel programlama dilleri, programla öğretiminde, alt düzey bilişsel becerilerin geliştirilmesinden ziyade, üst düzey bilişsel becerilerin geliştrilmesinde kullanılması önerilmektedir.
- Programlamanın yapısı üst düzey bilişsel becerilere yönelik olduğundan, programlama derslerinde kullanılacak eğitsel programlama dilleri, dersin hem teori hem de uygulama kısmına entegre edilerek kullanılmalıdır.
- Bilgisayar programlama kaygısını azaltmak için eğitsel programlama dilleri dersin uygulama ve hem teori hem de uygulama kısmına entegre edilerek kullanılabilir.
- Ders içi performans çalışmalarında eğitsel programlama dilleri dersin hem teori hem de uygulama kısmına entegre edilerek kullanılabilir.

- Programlamaya ilk başlayan acemi kullanıcıların bilgisayar programlama kaygılarını azaltmaya yönelik olarak eğitsel programlama dilleri programlama derslerinin uygulama ve hem teori, hem de uygulama kısımlarına entegre edilerek kullanılabilir.
- Öğrencilerin program yazma söz dizimine geçmelerini kolaylaştırmak için programlama derslerinde eğitsel programalama dilleri kullanılabilir.
- Programlama öğretiminde, eğitsel programlama dilleri gibi teknolojik ortamların derslere nasıl entegre edileceği konusunda bu çalışma uygulayıcılara fikir verebilir.

Araştırmacılara Öneriler

- Acemi olan öğrenciler üzerinde gerçekleştirilen bu çalışma, programlama ön bilgisine sahip öğrenciler üzerinde gerçekleştirilebilir.
- Öğrencilerin kaygı seviyelerine (düşük, yüksek) göre eğitsel programlama dilinin derse entegrasyonuna bakılabilir.
- Yüksek seviyeli dillerde eğitsel programlama dilinin derse entegrasyonuna bakılabilir.
- Eğitsel programlama dilinin derse entegrasyon şeklinin öğrencilerin motivasyonuna ve derse katılımına (engagement) ve bilgisayarca düşünmeye (computational thinking) etkisi araştırılabilir.
- Eğitsel programlama dillerinin derse entegrasyon şeklinin üst düzey bilişsel becerileriler de neden etkili olduğu nitel yöntemler kullanılarak araştırılabilir.
- Eğitsel programlama dilinin programlama derslerinde bir katalizör olarak kullanılıp öğrencilerin bilişsel yüküne etkisi araştırılabilir.

KAYNAKÇA

Adams, J. C., & Webster, A. R. (2012). What do students learn about programming from game, music video, and storytelling projects?. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 643-648). ACM.

Adelmann, R., Bischoff, T., & Lauer, T. (2007). IDEA: A framework for the fastcreation of interactive animations by pen sketching. In J. Hughes, D. R. Peiris, P. T. Tymann (Eds.), *ITiCSE '07: Proceedings of the 12th annual SIGCSE conference onInnovation and Technology in Computer Science Education* (pp. 291-295). New York:ACM.

Akhu-Zaheya, L.M., Khater, W., Nasar M., & Khraisat O. (2012). Baccalaureate nursing students' anxiety related computer literacy: a sample from Jordan. http://jrn.sagepub.com/content/early/2011/07/16/1744987111399522.abstract adresinden 17.Haziran.2015 tarihinde edinilmiştir.

Al-Linjawi, A. A., & Al-Nuaim, H. A. (2010). Using Alice to Teach Novice Programmers OOP Concepts. *Journal of King Abdulaziz University: Science*, *22*(1), 59-68.

Allison, I. K., Orton, P., & Powell, H. (2002). A virtual learning environment for introductory programming. In *Proceedings of the 3rd Conference of the LTSN-ICS.*. Higher Education Academy Subject Centre for Information and Computer Sciences (HEA-ICS).

Almeida-Martínez, F. J., Urquiza-Fuentes, J., & Velázquez-Iturbide, J. Á. (2009). Software educativo en Procesadores de Lenguajes: del enfoque genérico al enfoque centrado en el estudiante. In *Actas del IX Simposio Internacional de Informática Educativa, SIIE*.

Alpar, R.(2010).*Uygulamalı istatistik ve geçerlik-güvenirlik*. Ankara: Detay Yayıncılık.

Alper, A., & Gülbahar, Y. (2009). Trends and issues in educational technologies: A review of recent research in TOJET. *The Turkish Online Journal of Educational Technology,* 8 (2), 124-135.

Arabacıoğlu, C., Bülbül, H. ve Filiz, A. (2007). Bilgisayar programlama öğretiminde yeni bir yaklaşım. *Akademik Bilişim 2007 Konferansı*, Dumlupınar Üniversitesi, Kütahya. http://ab.org.tr/ab07/bildiri/99.doc adresinden 27 Haziran 2014 tarihinde edinilmiştir.

Arıkan, R. (2004). *Araştırma teknikleri ve rapor hazırlama.* Ankara: Asil Yayın.

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

Atılgan,H., Kan,A. ve Doğan, N. (2006). *Eğitimde ölçme ve değerlendirme*. Ankara: AnıYayıncılık.

Aydın, C. H., Hoşcan, Y. ve Özkul, A. E. (2004). *Temel Bilgi Teknolojileri*. Anadolu Üniversitesi, Eskişehir.

Ayersman, D. J., & Reed, W. M. (1996). Effects of learning styles, programming and gender on computer anxiety. *Journal of Research on Computing in Educaiton, 28*(2),148-161.

Baecker, R. (1998). Sorting out sorting: A case study of software visualization for teaching computer science. In J. T. Stasko, J. Domingue, M. H. Brown, & B. Price (Eds.), *Software visualization—Programming as a multimedia experience* (pp. 369-382).Cambridge, MA: The MIT Press.

Baecker, R., & Price, B. (1998). The early history of software visualization. *Software Visualization*, 29-34.

Baldwin, L.P., & Kuljis, J. (2000). Visualization Techniques For Learning and Teaching Programming. *Journal of Computing and Information Technology 8*(4), 285–291. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=915833 adresinden 9 Ağustos 2015 tarihinde edinilmiştir.

Baloğlu, M., & Çevik, V. (2008). Multivariate effects of gender, ownership, and the :frequency of use on computer anxiety among high school students. *Computers in Human Behavior,* 24, 6, 2639-2648

Bassil, S., & Keller, R. K. (2001). A Qualitative and Quantitative Evaluation of Software Visualization Tools. In *Proceedings of the Workshop on Software Visualization* (pp. 33-37).

Batumlu, D. Z., & Erden, M. (2007). The relationship between foreign language anxiety and English achievement of Yıldız Technical University School of foreign languages preparatory students. *Theory and Practice in Education*, *3*(1), 24-38.

Bau, D., Dawson, M., & Bau, A. (2015). Using Pencil Code to Bridge the Gap between Visual and Text-Based Coding. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 706-706). ACM.

Beck, A. T., Emery, G., & Greenberg, R. L. (2005). *Anxiety disorders and phobias: A cognitive perspective*. Basic Books.

Beckers, J. J., Wicherts, J. M., & Schmidt, H. G. (2007). Computer anxiety: Trait or state. *Computers in Human Behavior,* 23 :2851–2862.

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. *ACM SIGCSE Bulletin*, *39*(2), 32-36.

Bergin, J., & Martinez, M. P. (1996). An overview of visualization: its use and design, *Report of the Working Group on Visualization,* Integrating Tech. into C.S.E. 6/96 Barcelona, Spain.

Bergin, S., & Reilly, R. (2005b). The influence of motivation and comfort level on learning to program. *Proceedings of the 17th Annual Workshop of the Psychologyof Programming Interest Group*, June 29– July 1, Brighton, UK.

Bishop- Clark, C., Courte, J., & Howard, E. V. (2007). A Quantitative and Qualitative Investigation of Using Alice Programming to Improve Confidence, Enjoyment and Achievement among Non-Majors. *Journal of Educational Computing Research*, *37*(2) 193-207.

Bozdoğan. A. E. ve Öztürk, Ç. (2008). Coğrafya ile ilişkili fen konuların öğretimine yönelik öz-yeterlik inanç ölçeğinin geliştirilmesi. *Necatibey Eğitim Fakültesi Elektronik Fen ve Matematik Eğitimi Dergisi(EFMED),2*(2),66-81.

Bozıonelos, N. (2001). Computer anxiety: Reliationship with computer experience and prevalence. *Computers in Human Behavior*, 17(2), :213-224.

Bradley, G., & Russell, G. (1997). Computer experience, school support and computer anxiety. *Educational Psychology, 17*, 267-285.

Bravo, C., Marcelino, M. J., Gomes, A. J., Esteves, M., & Mendes, A. J. (2005). Integrating Educational Tools for Collaborative Computer Programming Learning. *J. UCS*, *11*(9), 1505-1517.

Brown, M. H. (1988). Exploring algorithms using Balsa-II. *IEEE Computer*, 21(5),14-36.

Brown, M. H. (1991). ZEUS: A system for algorithm animation and multi-view editing. *In Proceedings of the 1991 IEEE workshop on visual languages* (pp. 4-9). LosAlamitos, CA: IEEE Computer Society.

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.

Buche, M. W., Davis, L. R., & Vician, C. (2007). A longitudinal investigation of the effects of computer anxiety on performance in a computing-intensive environment. *Journal of Information Systems Education*, *18*(4), 415.

Butler, J. (2007). *El género en disputa: el feminismo y la subversión de la identidad* (Vol. 168). Ediciones Paidós Ibérica.

Büyüköztürk, Ş. (2001). *Deneysel desenler*. Ankara: PegemA Yayıncılık.

Büyüköztürk, Ş. (2010). *Sosyal bilimler için veri analizi elkitabı*. Ankara: Pegem Akademi Yayınları.

Büyüköztürk, Ş., Çakmak, E. K., Akgün, Ö. E., Karadeniz, Ş. ve Demirel, F. (2008). *Bilimsel Araştırma Yöntemleri.* Ankara: Pegem A Yayıncılık.

Byrne, P., & Lyons, G. (2001). The Effect of Student Attributes on Success in Programming. *SIGCSE Bull. 33*(3), 49-52.

Caffarella, R. S., & Zinn, L. F. (1999). Professional development for faculty: A conceptual framework of barriers and supports. *Innovative Higher Education,23*(4), 241-254.

Calder, N. (2010). Using Scratch: An Integrated Problem-Solving Approach to Mathematical Thinking. *Australian Primary Mathematics Classroom*, *15*(4), 9-14.

Cañas, J. J., Bajo, M. T., & Gonzalvo, P. (1994). Mental models and computer programming. *International Journal of Human-Computer Studies, 40*(5), 795-811.

Canfora, G., & Mancini, L., et al. (1996). A workbench for program comprehension during software maintenance. *Proceedings of the Fourth Workshop on Program Comprehension*. Los Alamitos, CA, IEEE Computer Society: 30-39.

Carlisle, M. C., Wilson, T. A., Humphries, J. W., & Hadfield, S. M. (2005). RAPTOR: a visual programming environment for teaching algorithmic problem solving. In *ACM SIGCSE Bulletin,* Vol. 37, No. 1, pp. 176-180). ACM.

Ceyhan, E. (2006). Computer anxiety of teacher trainees in the framework of personality variables. *Computers in Human Behavior*. 22, 207–220

Ceylan, H. (2009). Mesleki Eğitimde Öğrenci Kalitesi. *1. Uluslararası 5. Ulusal Meslek Yüksekokulları Sempozyumu.* Selçuk Üniversitesi. Konya. p.580-585,

Ceylan, H. ve Elibol, H. (2009). Meslek Yüksekokullarının Öğrenci Kaynağı Açısından Mesleki Teknik Ortaöğretim. *1. Uluslararası 5. Ulusal Meslek Yüksekokulları Sempozyumu,* Selçuk Üniversitesi. Konya, p.464-470, 27-29

Ceylan, H. ve Erdoğan, İ. (2009). Meslek Yüksekokulları Nasıl Cazip Hale Getirilebilir?. *1. Uluslararası 5. Ulusal Meslek Yüksekokulları Sempozyumu*. Selçuk Üniversitesi. Konya. p.351-356.

Ceylan, H., Avan, M. ve Elibol, H. (2009). Mesleki ve Teknik Eğitimde Öğretim Elemanı Profili, Sorunlar ve Çözüm Önerileri. *1. Uluslararası 5. Ulusal Meslek Yüksekokulları Sempozyumu.* Selçuk Üniversitesi. Konya. p.464- 470.

Chang, C. K. (2014). Effects of Using Alice and Scratch in an Introductory Programming Course for Corrective Instruction. *Journal of Educational Computing Research*, *51*(2), 185-204.

Chen, S., & Morris, S. (2005). Iconic programming for flowcharts, java, turing, etc. *ACM SIGCSE Bulletin*, *37*(3), 104-107.

Chiu, C. F. (2015). Introducing Scratch as the Fundamental to Study App Inventor Programming. In *Learning and Teaching in Computing and Engineering (LaTiCE), 2015 International Conference on* (pp. 219-220). IEEE.

Choi, G., Ligon, J., & Ward, J. (2002). Computer Anxiety and Social Workers: Differences by Access, Use, And Training. Journal of Technology in Human Services. 19(1). http://www2.uta.edu/cussn/jths/JTHS_V19_1.pdf adresinden 29 Nisan 2015 tarihinde edinilmiştir.

Choi, S.K., Bell, T., Jun, S.j., & Lee, W.G. (2008). Designing offline computer science activities for the korean elementary school curriculum. *Proceedings of the 13th annual conference on Innovation and technology in computer science education,* ACM. (s. 338-338). New York, NY, USA.

Christensen, R. (2002). Effects of technology integration education on the attitudes of teachers and students. *Journal of Research on technology in Education*, *34*(4), 411- 433.

Chua S. L., Chen D. T., & Wong A. F. L. (1999). Computer anxiety and its correlates: a meta- analysis. *Computers in Human Behavior*, 15(5), 609-623.

Clark, R. E. (1983). Reconsidering research on learning from media. *Review of educational research*, *53*(4), 445-459.

Clark, R. E. (1994). Media will never influence learning. *Educational technology research and development*, *42*(2), 21-29.

Clements, D. H., & Battista, M. T. (1989). Learning of geometric concepts in a Logo environment. *Journal for Research in Mathematics Education*, 450-467.

Clements, D. H., & Sarama, J. (1997). Research on Logo: A decade of progress. *Computers in the Schools*, *14*(1-2), 9-46.

Connolly, C., Murphy, E., & Moore, S.(2007). Second Chance Learners, Supporting Adults Learning Computer Programming. *International Conference on Engineering Education – ICEE 2007,s.3-7.*

Conway, M. J. (1997). *Alice: easy-to-learn 3d scripting for novices*. Yayımlanmamış Doktora Tezi, University of Virginia, Charlottesville.

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. In *Journal of Computing Sciences in Colleges* (Vol. 15, No. 5, pp. 107-116). Consortium for Computing Sciences in Colleges.

Cooper, S., Dann, W., & Pausch, R. (2003). Teaching objects-first in introductory computer science. Proceedings of the 34th SIGCSE technical symposium on Computer Science Education, Reno, Nevada.http://www.alice.org/publications/ TeachingObjectsfirstInIntroductoryComputerScience.pdf adresinden 20 Eylül 2014 tarihinde edinilmiştir.

Cooper, S., Dann, W., & Pausch, R. (2003). Using animated 3D graphics to prepare novices for CS1. *Computer Science Education*, *13*(1) 3-30 http://beust.com/algorithms.pdf web adresinden 11 Haziran 2015 tarihinde edinilmiştir.

Cooper,S.,Dann,W., & Pausch.R. (2003). Evaluating the Effectiveness of a New Instructional Approach. İn SIGCSE'03 February 19-23, 2003, Reno,Nevada, USA.

Cornforth, D. M., Popat, R., McNally, L., Gurney, J., Scott-Phillips, T. C., Ivens, A., & Brown, S. P. (2014). Combinatorial quorum sensing allows bacteria to resolve their social and physical environment. *Proceedings of the National Academy of Sciences*, *111*(11), 4280-4284.

Costa, F. A. (2007). Educational technologies: Analysis of master dissertation carried out in Portugal. *Educational Sciences Journal*, 3, 7-24.

Crescenzi, P., Malizia, A., Verri, M. C., Diaz, P., & Aedo, I. (2012). Integrating algorithm visualization video into a first-year algorithm and data structure course. *Educational Technology ve Society*, 15 (2), 115–124.

Cutts, Q., Connor, R., Michaelson, G., & Donaldson, P. (2014). Code or (not code): separating formal and natural language in CS education. In*Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (pp. 20-28). ACM.

Çakıroğlu, Ü. , Sarı, E. ve Akkan, Y. (2011). Üstün Yetenekli Öğrencilere Programlama Öğretiminin Problem Çözmeye Katkısı Konusunda Öğretmen Görüşleri. *5 th International Computer & Instructional Technologies Symposium,* 353-359

Çatlak, Ş., Tekdal, M., ve Baz, F. Ç. (2015). Scratch Yazılımı İle Programlama Öğretiminin Durumu: Bir Doküman İnceleme Çalışması. *Journal of Instructional Technologies & Teacher Education*, *4*(3).

Çetin, I. (2013). Visualization: A tool for enhancing students' concept images of basic object-oriented concepts. *Computer Science Education*, 23(1), 1-23.

Çırakoğlu, C. O. (2004). Bilgisayar kaygısı. *Başkent Üniversitesi iİktisadi ve İdari Bilimler Fakültesi Eleştirel ve Yaratıcı Düşünme ve Davranış Araştırmaları Laboratuvarı*, (13), 15-18.

Çölkesen, R. (2002). *Bilgisayar Programlama ve Yazılım Mühendisliğinde Veri Yapıları ve Algoritmalar*. İstanbul. Papatya Yayıncılık.

Daly, T. (2009). Using introductory programming tools to teach programming concepts: A literature review. *The Journal for Computing Teachers*.

Dann, W., Cooper, S., & Pausch, R. (2000). Making the connection: programming with animated small world. In *ACM SIGCSE Bulletin* (Vol. 32, No. 3, pp. 41-44). ACM.

De Bonte, A. M. (1998). *Pet park: a virtual learning world for kids.* Yayımlanmamış Yüksek Lisans Tezi, http://llk.media.mit.edu/papers/archive/deBonte-MEng/ adresinden 17 Temmuz 2015 tarihinde erişilmiştir.

Dershem, H. L., & Brummund, P. (1998). Tools for Web-based sorting animation. In *ACM SIGCSE Bulletin* (Vol. 30, No. 1, pp. 222-226). ACM.

Diehl, S. (2005). Software visualization. In *Proceedings of the 27th international conference on Software engineering* (pp. 718-719). ACM.

Dogan, B., & Robin, B. (2008). Implementation of digital storytelling in the classroom by teachers trained in a digital storytelling workshop. In *Society for Information Technology & Teacher Education International Conference* (Vol. 2008, No. 1, pp. 902-907).

Doyle, E., Stamouli, I., & Huggard, M. (2005). Computer anxiety, self-efficacy, computer experience: An investigation throughout a computer science degree. In *Frontiers in Education, 2005. FIE'05. Proceedings 35th Annual Conference* (pp. S2H-3). IEEE.

D'Souza, D., Hamilton, M., Harland, J., Muir, P., Thevathayan, C., & Walker, C. (2008). Transforming learning of programming: A mentoring project. *Proceedings of the Tenth Conference on Australasian Computing Education*, January 1, Wollongong, Australia.

DuBoulay, B. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1), 57–73.

Dunican, E. (2002). Making the analogy: Alternative delivery techniques for first year programming courses. In *Proceedings from the 14th workshop of the psychology of programming interest group, Brunel University* (pp. 89-99).

Eby, K. K. (2001). Teaching and Learning from an Interdisciplinary Perspective. *Peer Review*.

Eckerdal, A., Mc Cartney, R., Moström, J. E., Ratcliffe, M., & Zander, C. (2006). Can graduating students design software systems? *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, March 1 – 5, Houston, Texas, USA.

Eckerdal, A., Thuné, M., & Berglund, A. (2005). What does it take to learn 'programming thinking'? *Pro- ceedings of the First international Workshop on Computing Education Research*, Seattle, October 1-2, WA, USA.

Edwards, J. R. (1991). *Person-job fit: A conceptual integration, literature review, and methodological critique*. John Wiley & Sons.

Eick, S. G., Steffen, J. L., & Sumner Jr, E. E. (1992). Seesoft-a tool for visualizing line oriented software statistics. *Software Engineering, IEEE Transactions on*, *18*(11), 957-968.

Erden, M. (1998). *Eğitimde Program Değerlendirme*. Ankara: Anı Yayıncılık

Erdoğmuş, F. U. (2009). *Research trends in CEIT MS and PhD theses in Turkey: A content analysis. Unpublished master's thesis,* Middle East Technical University, Ankara, Turkey

Ertmer, P. A. (1999). Addressing first-and second-order barriers to change: Strategies for technology integration. *Educational Technology Research and Development*, *47*(4), 47-61.

Ertmer, P. A. (2005). Teacher pedagogical beliefs: The final frontier in our quest for technology integration?. *Educational technology research and development*, *53*(4), 25-39.

Ertmer, P. A., & Hruskocy, C. (1999). Impacts of a university-elementary school partnership designed to support technology integration. *Educational Technology Research and Development*, *47*(1), 81-96.

Ertmer, P. A., Conklin, D., Lewandowski, J., Osika, E., Selo, M. & Wignall, E. (2003). Increasing preservice teachers' capacity for technology integration through the use of electronic models. *Teacher Education Quarterly*, 95-112.

Esteves, M., & Mendes, A., (2004), A Simulation Tool to Help Learning of Object Oriented Programming Basics. In *Proceedings of the 34th ASEE/IEEE Frontiers in Education Conference* (pp. 20–23). Savannah, GA.

Esteves, M., Fonseca, B., Morgado, L., & Martins, P. (2011). Improving teaching and learning of computer programming through the use of the Second Life virtual world. *British Journal of Educational Technology*, *42*(4), 624-637.

Fesakis, G., & Dimitrakopoulou, A. (2006). *Review of educational environments for programming*: Technological and Pedagogical Dimensions, in journal THEMES

in education, 7(3), pp. 279-304 in Greek11 MIT Media Lab. (2014). Scratch.http://scratch.mit.edu adresinden 10 Eylül 2015 tarihinde edinilmiştir.

Fesakis, G., & Serafeim, K. (2009). Influence of the familiarization with scratch on future teachers' opinions and attitudes about programming and ICT in education. In *ACM SIGCSE Bulletin* (Vol. 41, No. 3, pp. 258-262). ACM.

Fincher, S., Baker, B., Box, I., Cutts, Q., Raadt, M. d., Haden, P., & . . . Tutty, J. (2005). Programmed to succeed?. A multi-national, multi-institutional study of introductory programming courses *Technical Report*. Computing Laboratory: University of Kent.

Fisher, A., & Margolis, J. (2002). Unlocking the clubhouse: the Carnegie Mellon experience. *ACM SIGCSE Bulletin*, *34*(2), 79-83.

Fujiwara, K., Fushida, K., Tamada, H., Igaki, H., & Yoshida, N. (2012). Why Novice Programmers Fall into a Pitfall?: Coding Pattern Analysis in Programming Exercise. In *Empirical Software Engineering in Practice (IWESEP), 2012 Fourth International Workshop on* (pp. 46-51). IEEE.

Futschek, G., & Moschitz, J. (2010). Developing algorithmic thinking by inventing and playing algorithms. *Constructionism 2010:Constructionist approaches to creative learning, thinking and education*, 28-30 June, Paris, France.

Futschek, G., & Moschitz, J. (2011). Learning algorithmic thinking with tangible objects eases transition to computer programming. *5th International Conference on Informatics in Schools: Situation, Evolution and Perspectives, ISSEP 2011*, 26- 29 October, Bratislava, Slovakia.

Garner, S. (2003). Learning resources and tools to aid novices learn programming. In *Informing Science & Information Technology Education Joint Conference (INSITE)* (pp. 213-222).

Garner, S. (2009). Learning to program from Scratch. In *2009 Ninth IEEE International Conference on Advanced Learning Technologies* (pp. 451-452). IEEE.

Goldsteın, S. B., Dudley, E. A., Erıckson, C.M., & Rıcher, N. L. (2002), Personalitytraits and computer anxiety as predictors of y2k anxiety. *Computers in Human Behavior*,18(3), :271-284.

Gomes, A., & Mendes, A. J. (2007b) Learning to program – difficulties and solutions. *Proceedings of the 2007 international convergence on Engineering education,* September 3-7, Coimbra, Portugal.

Goold, A., & Rimmer, R. (2000). Indicators of performance in first-year computing. In *Computer Science Conference, 2000. ACSC 2000. 23rd Australasian* (pp. 74-80). IEEE.

Göktaş, Y., Yildirim, Z., & Yildirim, S. (2008). The keys for ICT integration in K-12 education: Teachers' perceptions and usage. *Hacettepe Üniversitesi Eğitim Fakültesi Dergisi*, *34*(34).

Grant, N. S. (2003). *A study on critical thinking, cognitive learning style, and gender in various information science programming classes*. Paper presented at the Proceedings of the 4th conference on Information technology curriculum, Lafayette, Indiana, USA.

Grissom, S., McNally, M. F., & Naps, T. (2003). Algorithm visualization in CS education: comparing levels of student engagement, 2003 ACM *Software Visualization Symposium*, San Diego, California.

Guzdial, M. (2004). Programming environments for novices. *Computer science education research, 2004*, 127-154.

Gültekin, K. (2006). *Çoklu ortamın bilgisayar programlama başarısı üzerine etkisi*. Yayımlanmamış Yüksek Lisans Tezi, Hacettepe Üniversitesi, Ankara.

Gültekin, K. (2006). *Çokluortamın programlama başarısı üzerindeki etkisi*. Yüksek Lisans Tezi, Hacettepe Üniversitesi Fen Bilimleri Enstitüsü.

Gülümbay, A. A. (2006). *Yükseköğretimde Web'e dayalı ve yüzyüze ders alan öğrencilerin öğrenme stratejilerinin, bilgisayar kaygılarının ve başarı durumlarının karşılaştırılması.* Eskisehir: Anadolu Üniversitesi Egitim Fakültesi Yayınları No:99.

Gümüşay, M.Ü. (2012). *Temel Bilgisayar Bilimleri,* Y.T.Ü., http://www.yildiz.edu.tr/~gumusay/Dersler/Lisans/TBB/Dokumanlar/VisualBasicProgAdim.pdf adresinden 11 Şubat 2015 tarihinde edinilmiştir.

Güneş, A. ve Karabak, D. (2013). Ortaokul Birinci Sınıf Öğrencileri İçin Yazılım Geliştirme Alanında Müfredat Önerisi.

Gürcan - Namlu, A. ve Ceyhan, E. (2003). Bilgisayar Kaygısı: Öğretmen Adayları Üzerinde Çok Yönlü Bir İnceleme. *Educational Sciences: Theory ve Practice*, *3*(2).

Gürcan- Namlu, A. (2003). The effect of learning strategy on computer anxiety. *Computers in Human Behavior*, 19, 565–578.

Gürcan, A. (2005). Bilgisayar özyeterliği algısı ile bilişsel öğrenme stratejileri arasındaki ilişki. *Eğitim Araştırmaları Dergisi*, 19,179-193.

Haajanen, J., Pesonius, M., Sutinen, E., Tarhio, J., Terasvirta, T., & Vanninen, P. (1997). *Animation of user algorithms on the Web* (p. 356). IEEE.

Hagan, D., Sheard, J., & Macdonald, I. (1997). Monitoring and evaluating a redesigned first year programming course [Özet]. *ACM SIGCSE Bulletin, Proceedings of the 2nd conference on Integrating technology into computer science education ITiCSE '97, 29* (3) 37-39.

Häkkinen, P. (1994). Changes in computer anxiety in a required computer course. *Journal of Research on Computing in Education*, *27*(2), 141-153.

Hansen, S., Schrimpsher, D., & Narayanan, N.H. (1998). From Algorithm Animations to Animation- embedded Hypermedia Visualizations. Visual Information, Intelligence and Interaction Research Group Auburn University Department of Computer Science ve Engineering, Auburn, AL ftp://ftp.eng.auburn.edu/pub/ techreports/cse/98/CSE98- 05.pdf adresinden 15 Mart 2015 tarihinde edinilmiştir.

Harel, I. E., & Papert, S. E. (1991). *Constructionism*. Ablex Publishing.

Harel, I., & Papert, S. (1990). Software design as a learning environment. *Interactive learning environments*, *1*(1), 1-32.

Hew, K. F., Kale, U., & Kim, N. (2007). Past research in instructional technology: Results of a content analysis of empirical studies published in three prominent

instructional technology journals from the year 2000 through 2004.*Journal of Educational Computing Research, 36*(3), 269-300.

Hirakawa, M., Tanaka, M., & Ichikawa, T. (1990). HI-VISUAL Iconic programming environment, In T. Ichikawa, E. Jungert and R.R. Korfhage (Eds.) *Visual Languages and Applications. Plenum Press*, New York, 12, 1-145.

Hooshyar, D., Ahmad, R. B., Yousefi, M., Yusop, F. D., & Horng, S. J. (2015). A flowchart-based intelligent tutoring system for improving problem-solving skills of novice programmers. *Journal of Computer Assisted Learning*.

Horn, M. S., & Jacob, R. J. K. (2006). *Tangible programming in the classroom: a practical approach*. Paper presented at the CHI '06 Extended Abstracts on Human Factors in Computing Systems, Montrve#233;al, Quve#233;bec, Canada.

Horn, M. S., & Jacob, R. J. K. (2007). *Designing tangible programming languages for classroom use*. Paper presented at the Proceedings of the 1st international conference on Tangible and embedded interaction, Baton Rouge, Louisiana.

Hoyles, C., & Noss, R. (1992). A pedagogy for mathematical microworlds. *Educational studies in Mathematics*, *23*(1), 31-57.

Hranstinski, S., & Keller, C. (2007). An examination of research approaches that underlie research on educational technology: A review from 2000 to 2004. *Journal of Educational Computing Research*, 36 (2), 175-190.

Hu, M. (2004). Teaching novices programming with core language and dynamic visualisation. *Proceedings of the 17th NACCQ*, 94-103..

Hundhausen, C.D., Douglas, S.A., & Stasko, J.T. (2002). A Meta-Study of Algorithm Visualization Effectiveness. Journal of Visual Languages and Computing, 13(3), 259-290.

Ismail, M. N., Ngah, N. A., & Umar, I. N. (2010). Instructional strategy in the teaching of computer programming: a need assessment analyses. *TOJET*, *9*(2), 125-131.

Işık, F. (2015). İlkokullara 'kodlama dersi' geliyor. http://www.memurlar.net/haber/516181/ adresinde 3 Haziran 2015 tarihinde edinilmiştir.

İspir, K. ve Ceylan, H.(2009). Meslek Yüksekokullarında Elektrik Eğitimi Sorunlar ve Çözüm Önerileri. *1. Uluslararası 5. Ulusal Meslek Yüksekokulları Sempozyumu*. Selçuk Üniversitesi. Konya. p.393-402.

Jenkins, T. (2002). On the difficulty of learning to program. In*Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences* (Vol. 4, pp. 53-58).

Kafai, Y. B., Fields, D. A., Roque, R., Burke, W. Q., & Monroy-Hernandez, A. (2012). Collaborative agency in youth online and offline creative production in Scratch. *Research and Practice in Technology Enhanced*.

Kafai, Y. B., Franke, M., Ching, C., & Shih, J. (1998). Games as interactive learning environments fostering teachers' and students' mathematical thinking. *International Journal of Computers for Mathematical Learning*, *3*(2), 149-193.

Kapıkıran, Ş. (2002). Üniversite öğrencilerinin sınav kaygısının bazı psiko-sosyal değişkenlerle ilişkisi üzerine bir inceleme. *Pamukkale Üniversitesi Eğitim Fakültesi Dergisi, 1*(11), 34-43.

Karal, H., & Berigel, M. (2006). Eğitim fakültelerinin öğretmenlerin teknolojiyi eğitimde etkin olarak kullanabilme yeterlilikleri üzerine etkileri ve çözüm önerileri. *Çukurova Üniversitesi Eğitim Fakültesi Dergisi*, *2*(32), 60-66.

Karasar, N. (2004). *Bilimsel araştırma yöntemi: kavramlar, ilkeler, teknikler*. Nobel Yayın Dağıtım.

Karasar, N. (2005). *Bilimsel arastirma yöntemi*. Ankara: Nobel Yayin Dagitim, s.157.

Kaučič, B., & Asič, T. (2011). Improving introductory programming with Scratch?. In *MIPRO, 2011 Proceedings of the 34th International Convention*(pp. 1095-1100). IEEE.

Kehoe, C., Stasko, J., & Taylor, A. (2001). Rethinking the evaluation of algorithm animations as learning aids: an observational study. *International Journal of Human-Computer Studies*, *54*(2), 265-284.

Kelleher, C., Pausch, R., & Kiesler, S. (2007). Storytelling Alice motivates middle school girls to learn computer programming. *Proceedings of the SIGCHI Conference On Human Factors İn Computing Systems Table Of Contents.* San Jose, California, USA.

Kenar, N. (2015). Türkiye'de Mesleki Eğitim ve Meslek Yüksekokulları. İnternet Adresi: http://www.messegitim.com.tr/ti/799/0/TURKIYEDE-MESLEKI-EGITIM-VE- MYOLAR Erişim Tarihi: 07.01.2015.

Kesici, T. ve Kocabaş, Z. (2007) *Bilgisayar-1 Liseler için* Ankara: MEB, s.35.

Kinnunen, P., & Malmi, L. (2008). CS minors in a CS1 course. In*Proceedings of the Fourth international Workshop on Computing Education Research* (pp. 79-90). ACM.

Kinnunen, P., McCartney, R., Murphy, L., & Thomas, L. (2007). Through the eyes of instructors: a phenomenographic investigation of student success. *ICER '07: Proceedings of the third international workshop on Computing education research*, September 15-16, Georgia, USA.

Klassen, M. (2006). Visual Approach for Teaching Programming Concepts. 9th International Conference on Engineering Education. http://public.clunet.edu//~mklassen/ICEE2006.pdf web adresinden 26 Nisan 2015 tarihinde edinilmiştir.

Klein, J. (1997). ETR&D-Development: An analysis of content and survey of future direction. *Educational Technology Research and Development,* 45 (3), 57-62

Konvalina, J., Stephens, L., & Wileman, S. (1983). Identifying factors influencing computer science aptitude and achievement. *AEDS Journal*, *16*(2), 106-112.

Kozma, R. B. (1994). Will media influence learning? Reframing the debate.*Educational technology research and development*, *42*(2), 7-19.

Kölling, M., & Rosenberg, J. (2001). Guidelinesfor Teaching Object Orientationwith Java. in Proceedingsof6thconferenceon Information Technologyin Computer Science Education (ITiCSE2001), Canterbury,2001.

Krpan, D., Mladenović, S., & Rosić, M. (2015). Undergraduate Programming Courses, Students' Perception and Success. *Procedia-Social and Behavioral Sciences, 174*, 3868-3872.

Kuşat, N. (2014). Meslek Yüksekokullarında Öğrenci Başarısı Üzerine Bir Çalışma: Eğirdir Meslek Yüksekokulu Muhasebe Programı Örneği. *Muhasebe ve Finansman Dergisi*. Cilt 61. p.65-79.

Laakso, M. J., Rajala, T., Kaila, E., & Salakoski, T. (2008). The impact of prior experience in using a visualization tool on learning to program. *Appeared in Cognition and Exploratory Learning in Digital Age (CELDA 2008)*, 13-15.

Lahtinen, E., Ahoniemi, T., & Salo, A. (2007). Effectiveness Of Integrating Program Visualizations To A Programming Course. Proceedings of the 7th Baltic Sea Conference on Computing Education Research (s.195- 198). Koli, Finland.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H. M. (2005). A study of the difficulties of novice programmers. In *ACM SIGCSE Bulletin* (Vol. 37, No. 3, pp. 14-18). ACM.

Lai, A. F., & Guo, S. H. (2011). The flow and self efficacy of sixth grade students under Scratch programming learning. In *Electrical and Control Engineering (ICECE), 2011 International Conference on* (pp. 6915-6919). IEEE.

Latchem, C. (2006). Editorial: A content analysis of the British Journal of Educational Technology. *British Journal of Educational Technology*, 37 (4), 503-511

Lawrence, A. W., Badre, A. M., & Stasko, J. T. (1994). Empirically evaluating the use of animations to teach algorithms. In *Visual Languages, 1994. Proceedings., IEEE Symposium on* (pp. 48-54). IEEE.

Lemieux, F., & Salois, M. (2006). Visualization techniques for program comprehension. In *New Trends in Software Methodologies, Tools and Techniques: Proceedings of the Fifth SoMeT 06* (Vol. 147, p. 22). IOS Press.

Leso, T., & Peck, K. L. (1992). Computer anxiety and different types of computer courses. *Journal of Educational Computing Research, 8*(4), 469-478.

Levy, R. B. B., Ben-Ari, M., & Uronen, P. A. (2003). The Jeliot 2000 program animation system. *Computers & Education, 40*(1), 1-15.

Lewis, C. M. (2010). How programming environment shapes perception, learning and goals: logo vs. scratch. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 346-350). ACM.

Lin, C., & Zhang, M. (2003). The use of computer animation in teaching discrete structures course. *MICS 2003 Proceedings The 36th Annual Midwest Instruction and Computing Symposium.* http://www.micsymposium.org/apache2-default/mics_2003/Lin.PDF adresinden 27 Haziran 2014 tarihinde edinilmiştir.

Lin, J. M. C., Yen, L. Y., Yang, M. C., & Chen, C. F. (2005). Teaching computer programming in elementary schools: A pilot study. In *National Educational Computing Conference*.

Lindbeck, J. S., & Dambrot, F. (1986). Measurement and reduction of math and computer anxiety. *School science and mathematics, 86*(7), 567-577.

Liu, M., & Reed, W. M. (1992). Teacher education students and computers: Gender, major, prior computer experience, occurrence. *Journal of Research on Computing in Education*, 24,457-466.

Loyd, B. H., & Gressard, C. (1984). Reliability and factorial validity of computer attitude scales. *Educational and Psychological measurement, 44*(2), 501-505.

Ma, Y. (2000). *Research in educational communications and technology at the University of Wisconsin: A study of dissertation completed since the inception of the program.* Paper presented at the 22'rd National Convention of the Association for Educational Communiations and Technology.

Maheshwari, P. (1997). Teaching programming paradigms and languages for qualitative learning. In *Proceedings of the 2nd Australasian conference on Computer science education* (pp. 32-39). ACM.

Mahor, D., Henderson, R., & Deane, F. (1997) The effects of computer anxiety, state anxiety and computer experience on users'performance of computer based tasks. *Personalityand Individual differences*, 22, : 683-692.

Malan, D.J. & Leitner, H.H. (2007). Scratch for budding computer scientists. *38th ACM Technical Symposium on Computer Science Education.* Covington, Kentucky.

Malcom, S., Teich, A. H., Jesse, J. K., Campbell, L. A., Babco, E. L., & Bell, N. E. (2005). Preparing women and minorities for the IT workforce: The role of nontraditional educational pathways. *Washington, DC: American Association for the Advancement of Science*.

Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: urban youth learning programming with scratch. *ACM SIGCSE Bulletin*, *40*(1), 367-371.

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004). Scratch: a sneak preview [education]. In *Creating, Connecting and Collaborating through Computing, 2004. Proceedings. Second International Conference on*(pp. 104-109). IEEE.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), 16.)

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), 16.

Marcoulides, G. A. (1989). Measuring computer anxiety: The computer anxiety scale. *Educational and Psychological Measurement*, *49*(3), 733-739.

Maria, H., Misra, A., Rutter, M., & Mercuri, R. (2003). Identifying and correcting Java programming errors for introductory computer science students. *ACM SIGCSE Bulletin*, *35*(1), 153-156.

Martocchio, J. J. (1992). Microcomputer usage as an opportunity: The influence of context in employee training. *Personnel Psychology, 45*, 529-553.

Masood, M. (1997). A ten year analysis: Trends in traditional educational technology literature. Malaysian Online Journal of Instructional Technology, 1 (2), 1823-1844.

Matthiasdottir, A. (2006). How to teach programminglanguages to novicestudents? Lecturing or not? İn International Conference on Computer Systems and Technologies, 15-16 June 2006, University of Veliko Tarnovo, Bulgaria).

Maurer, M. M., & Simonson, M.R. (1993). The reduction of computer anxiety: Its relation to relaxation training, previous computer. *Journal of Research on Computing in Education, 26*, 205-220.

McCartney, R., Eckerdal, A., Moström, J. E., Sanders, K., & Zander, C. (2007). Successful students' strategies for getting unstuck. *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, June 23- 27, Dundee, Scotland.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B. D., & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin*, *33*(4), 125-180.

McCue, C. (2010). *Fearless Flash: Use Adobe InDesign CS5 and the Tools You Already Know to Create Engaging Web Documents*. Pearson Education.

McGettrick, A., Boyle, R., Ibbett, R., Lloyd, J., Lovegrove, G., & Mander, K. (2005). Grand challenges in computing: Education - A summary. *The Computer Journal*, *48*(1), 42-48.).

McInerney, O., McInerney, D. M., & Sinclair, K. E (1994). Student teachers, computer anxiety and computer experience. *Journal of Educational Computing Research*, *11*, 27-50.

McInerney, V. (1990). Computer Anxiety and Student Teachers: Interrelationships between Computer Anxiety, Demographic Variables and an Intervention Strategy.

Mcmillan, H., & Schumacher, S. (2010). Researcher in Education.

McNerney, T. (2004). From turtles to Tangible Programming Bricks: explorations in physical language design. *Personal and Ubiquitous Computing*, *8*(5), 326-337. doi: 10.1007/s00779-004-0295-6.

Mead, J., Gray, S., Hamer, J., James, R., Sorva, J., Clair, C. S., & Thomas, L. (2006, June). A cognitive approach to identifying measurable milestones for programming skill acquisition. In *ACM SIGCSE Bulletin* (Vol. 38, No. 4, pp. 182-194). ACM.

Meisalo, V., Suhonen, J., Torvinen, S., & Sutinen, E. (2002, June). Formative evaluation scheme for a web-based course design. In *ACM SIGCSE Bulletin*(Vol. 34, No. 3, pp. 130-134). ACM.

MIT, (2015). Massachusetts Institute of Technology, http://wiki.scratch.mit.edu/wiki/Scratch adresinden 02 Aralık 2015 tarihinde edinilmiştir.

MIT, (2015). Massachusetts Institute of Technology, https://scratch.mit.edu/ adresinden 10 Kasım 2015 tarihinde edinilmiştir.

Michael, K., & John, R., (2001). Guidelines for teaching object orientation with Java. In *ACM SIGCSE Bulletin* (Vol. 33, No. 3, pp. 33-36). ACM.

Michael, K., (1999). The problem of teaching object-oriented programming. *Journal of Object Oriented Programming*, *11*(8), 8-15.

Michael, M., & Desmond, C., (2002) Evaluation of students attitudes to learning the Java language. ACM Intl. Conf. Proc. 25, 125-130.

Michaelson, G. (2015). Teaching Programming with Computational and Informational Thinking. *Journal of Pedagogic Development*, *5*(1).

Mihalca, L., & Miclea, M. (2007). Current trends in educational technology research. *Cognition, Brain, Behavior*, *11*(1), 115-129.

Milic, J. (2009). Predictors of success in solving programming tasks. . *The Teaching of Mathematics*, *12*(1), 25-31.

Mohamad, S. N. H., Patel, A., Tew, Y., Latih, R., & Qassim, Q. (2011). *Principles and dynamics of block-based programming approach.* Paper presented at the Computers ve Informatics (ISCI), 2011 IEEE Symposium on.

Monroy-Hernández, A., & Resnick, M. (2008). Empowering kids to create and share programmable media. *Interactions*, 15(2), 50-53.

Moreno, A., Myller, N., Sutinen, E., & Ben-Ari, M. (2004, May). Visualizing programs with Jeliot 3. In *Proceedings of the working conference on Advanced visual interfaces* (pp. 373-376). ACM.

Morrison, G. (1994). The media effects question: Unresolvable or asking the right question. *Educational Technology Research and Development,* 42(2),41--44.

Moser, R. (1997, June). A fantasy adventure game as a learning environment: why learning to program is so difficult and what can be done about it. In *ACM SIGCSE Bulletin* (Vol. 29, No. 3, pp. 114-116). ACM.

Muller, O., Haberman, B., & Averbuch, H. (2004, June). (An almost) pedagogical pattern for pattern-based problem-solving instruction. In *ACM SIGCSE Bulletin* (Vol. 36, No. 3, pp. 102-106). ACM.

Myers, J. E. (2006). *Child protection in America: Past, present, and future*. Oxford University Press.

Namlu, A. G. (2003). The effect of learning straregy on computer anxiety. *Computers in Human Behaviour, 19* (5), 565-578.

Namlu, A. G. ve Ceyhan E.. (2002). *Bilgisayar kaygısı: Üniversite öğrencileri üzerinde bir çalışma.* Eskişehir: Anadolu Üniversitesi Yaymlar1 No: 1353, Eğitim Fakültesi Yayınları No: 84.

Namlu, A., ve Ceyhan, E. (2002). Bilgisayar Kaygısı. (Üniversite öğrencileri üzerinde bir araştırma). Eskisehir: Anadolu Üniversitesi Yayınları (In English: Computer anxietya study on university students). Eskisehir: Anadolu UniversityPublishing.

Naps, T. L., Eagan, J. R., & Norton, L. L. (2000). JHAVE- An Environment to Actively Engage Students in Web-based Algorithm Visualisations", ACM

SIGCSE Bulletin, *Proceeding of the 31th SIGCSE technical symposium on Computer Science Education*, V32, I 1.

Naps, T., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., & Hunhausen, C. vd. (2003). Exploring the role of visualization and engagement in computer science education.. SIGCSE Bulletin,35(2),131–152.

Naps, T., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., & Velázquez-Iturbide, J. (2002). Exploring the role of visualization and engagement in computer science education. Proceedings from ITiCSE-WGR '02: *Working group reports from ITiCSE on Innovation and technology in computer science education*. (pp. 131-152). New York, NY: Association for Computing Machinery.

Naser, S. S. (2008). Developing Visualization Tool for Teaching AI Searching Algorithms. *Information Technology Journal, 7*(2), 350-355.

Navarro-Prieto, R., & Cañas, J. J. (2001). Are visual programming languages better? The role of imagery in program comprehension. *International Journal of Human- Computer Studies, 54*(6), 799-829.

Necessary, J. R., & Parish, T. S. (1996). The relationships between computer usage and computer-related attitudes and behaviors. *Education, 116*, 384-388.

Nedzad, M., & Yasmeen, H. (2001). Challenges in teaching java technology. *Informing Sci*. pp:365-371.

Nelson, M. L., & Rice, D. (2000). Introduction to algorithms and problem solving. In *Frontiers in Education Conference, 2000. FIE 2000. 30th Annual*(Vol. 2, pp. S2C-5). IEEE.

Nevalainen, S., & Sajaniemi, J. (2006, September). An experiment on short-term effects of animated versus static visualization of operations on program perception. In *Proceedings of the second international workshop on Computing education research* (pp. 7-16). ACM.

Olsen, K. A., Pedersen, B., Harnes, P., & Tosse, O.J., (1990). A visual system to support teaching of programming, In S-K. Chang (ed.) *Visual Languages and Visual Programming.* Plenum Press, New York 277-288.

Ousterhout, J. K. (1998). Scripting: Higher level programming for the 21st century. *Computer*, *31*(3), 23-30.

Pane, J., & Myers, B. (1996). Usability Issues in the Design of Novice Programming Systms. *Human Computer Interaction Institute Technical Report CMU-HCII-96-101*. School of Computer Science Technical Reports, Carnegie Mellon university, CMU-CS-96-132. http://www.cs.cmu.edu/~pane/ftp/CMU-CS-96-132.pdf web adresinden 25 Mart 2015 tarihinde edinilmiştir.

Papert, S. (1971). A computer laboratory for elementary schools.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

Papert, S. (1984). Computer as Mudpie. *Classroom Computer Learning*, *4*(6), 36.

Papert, S. (1990). *A critique of technocentrism in thinking about the school of the future*. Epistemology and Learning Group, MIT Media Laboratory.

Papert, S. (1993). The children's machine. *Technology Revıew-Manchester Nh-*, *96*, 28-28.

Pea, R. D. (1986). Language-independent conceptual “bugs” in novice programming. *Journal of Educational Computing Research*, 2(1), 25-36.

Peppler, K. A., & Kafai, Y. B. (2007). Collaboration, computation, and creativity: media arts practices in urban youth culture. In *Proceedings of the 8th iternational conference on Computer supported collaborative learning* (pp. 590-592). International Society of the Learning Sciences.

Peppler, K. A., & Kafai, Y. B. (2007). What video game making can teach us about learning and literacy: Alternative pathways into participatory cultures. *Paper to be presented at the Digital International Games Research Association meeting* in Tokyo, Japan. http://scratch.mit.edu/files/DiGRA07_games_kafai.pdf adresinden 2 Ekim, 2014 tarihinde edinilmiştir.

Peppler, K., & Kafai, Y. (2005). Creative coding: The role of art and programming in the K-12 educational context. http://download.scratch.mit.edu/CreativeCoding.pdf adresinden24 Ağustos 2014 tarhinde edinilmiştir.

Perry, G. (2009). *Yeni Başlayanlar İçin Programlama Kılavuzu* (çev. T. Aksoy). İstanbul: Sistem Yayıncılık.

Petre, M. (2007). *Computer science education research.* New York, NY: Taylor & Francis.

Pillay, N. (2003). Developing intelligent programming tutors for novice programmers. *ACM SIGCSE Bulletin*, *35*(2), 78-82.

Pillay, N., & Jugoo, V. R. (2005). An Investigation into Student Characteristics Affecting Novice Programming Performance. *SIGCSE Bulletin*, *37*(4), 107-110.

Powers, K., Gross, P., Cooper, S., McNally, M., Goldman, K. J., & Proulx, V. (2006). Tools for teaching introductory programming: what works? Proceedings from SIGCSE ’06: *The 37th SIGCSE Technical Symposium on Computer Science Education.* (pp. 560-561). New York, NY: Association for Computing Machinery.

Prensky, M. (2001). Digital natives, digital immigrants part 1. *On the horizon*,*9* (5), 1-6.

Price, B. A., Baecker, R. M., & Small, I. S. (1993). A principled taxonomy of software visualization. *Journal of Visual Languages & Computing*, *4*(3), 211-266.

Proulx, V. K. (2000). Programming patterns and design patterns in the introductory computer science course. In *ACM SIGCSE Bulletin* (Vol. 32, No. 1, pp. 80-84). ACM.

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81-106.

Ragonis, N., & Ben-Ari, M. (2005, February). On understanding the statics and dynamics of object-oriented programs. In *ACM SIGCSE Bulletin* (Vol. 37, No. 1, pp. 226-230). ACM.

Ramadhan, H. A. (2000). Programming by discovery. Journal of Computer Assistes Learning 16, 83-93. (ERIC Document Reproduction Service).

Reiser, R. (1994). Clark's invitation to the dance: An instructional designer's response. *Educational Technology Research and Development,* 42(2), 45--48.

Renda, C., & Sprouse, D. (2010). Giving Experiential Learning a Digital Makeover: A Case Study in Using Digital Storytelling and Web 2.0 Applications to Promote

Greater. Technological Competency in K-12 Teachers. Proceedings of *International Conference on Education* 2010, Hawaii.

Resnick, M. (1996, July). Distributed constructionism. In *Proceedings of the 1996 international conference on Learning sciences* (pp. 280-284). International Society of the Learning Sciences.

Resnick, M. (1997). *Turtles, termites, and traffic jams: Explorations in massively parallel microworlds*. Mit Press.

Resnick, M. (2007). Sowing the Seeds for a More Creative Society. *International Society for Technology in Education. 18-22*. US&Canada.

Resnick, M., & Ocko, S. (1990). *LEGO/logo--learning through and about design*. Epistemology and Learning Group, MIT Media Laboratory.

Resnick, M., Kafai, Y., & Maeda, J. (2003). ITR: A Networked, Media-Rich Programming Environment to Enhance Technological Fluency at After-School Centers. Proposal {funded} to the National Science Foundation, Washington, DC.

Resnick, M., Kafai, Y., Maloney, J., Rusk, N., Burd, L., & Silverman, B. (2003). A networked, mediarich programming environment to enhance technological fluency at after-school centers in economically-disadvantaged communities. *Proposal to National Science Foundation*.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: programming for all. *Communications Of The Acm*, 52(11), 60-67.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A.,Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). International Society of the Learning Sciences *Proceedings of the Tenth Conference on Australasian Computing Education*, January 1, Wollongong, Australia.

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

Rizvi, M., Humphries, T., Major, D., Jones, M., & Lauzun, H. (2011). A CS0 course using scratch. *Journal of Computing Sciences in Colleges, 26*(3), 19-27.

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

Rogerson, C., & Scott, E. (2010). The fear factor: How it affects students learning to program in a tertiary environment. *Journal of Information Technology Education: Research, 9*(1), 147-171.

Romeike, R. (2007). Applying creativity in CS high school education: criteria, teaching example and evaluation. In *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research-Volume 88*(pp. 87-96). Australian Computer Society, Inc..

Ropp, M. M. (1999). Exploring individual characteristics associated with learning to use computers in preservice teacher preparation. *Journal of Research on Computing inEducation, 31*, 402-416.

Ross, S. M., Morrison, G. R., & Lowther, D. L. (2010). Educational technology research past and present: Balancing rigor and relevance to impact school learning. *Contemporary Educational Technology, 1*(1), 17-35.

Rössling, G., & Freisleben, B. (2002). ANIMAL: A system for supporting multiple roles in algorithm animation. *Journal of Visual Languages and Computing*, 13, 341-354.

Rusk, N., Resnick, M., & Maloney,J.(2012). Learning with scratch:21st century learning skills. Retrieved Oct. 12, 2012 from M.I.T., *Lifelong Kindergarten Group* Web site: http://info.scratch.mit.edu/sites/infoscratch.media.mit.edu/docs/ Scratch- 21st Century Skills.pdf adresinden 12 Temmuz, 2015 tarihinde edinilmiştir.

Russell, G., & Bradley, G. (1996). Computer anxiety and student teachers: Antecedent and intervention. *Asia-Pacific Journal of Teacher Education, 24*, 245-258.

Sajaniemi, J., & Hu, C. (2006). Teaching Programming: Goingbeyond "Objects First".in 18th Work shopof the Psychologyof Programming Interest Group, Universityof Sussex, September 2006.

Sanwar, S. (2005). *kuasa khusus nonmuslım dalam perkara perceraıan dı pengadılan agama menurut hukum ıslam (Studi Kasus di Pengadilan Agama Blora)* (Doctoral dissertation, Universitas Muhammadiyah Surakarta).

Scholfield, S. (2012). Learning through sharing and doing. *Incite*, *33*(7), 8.

Schuck, J. A., DeHaan, R. L., & McCray, R. A. (Eds.). (2003). *Improving Undergraduate Instruction in Science, Technology, Engineering, and Mathematics:: Report of a Workshop*. National Academies Press.

Schwartzman, L. (2006). A qualitative analysis of reflective and defensive student responses in a software engineering and design course.*Proceedings of the 6th Baltic Sea Conference on Computing Education Research*, February 01, Uppsala, Sweden.

Scott, E. C. (2008). From requirements to code: Issues and learning in IS students systems development projects. *Journal of Information Education Technology (JITE) - Innovations in Practice (IIP)*, 7, 1-13. Retrieved from http://www.jite.org/documents/Vol7/JITEv7IIP001-013Scott.pdf.

Scovel, C. (1991). On symplectic lattice maps. *Physics Letters A*, *159*(8), 396-400.

Sewchurran, K. (2008*)*. Toward an approach to create self-organizing and reflexive information systems project practitioners. *International Journal of Managing Projects in Business*, *1*(3), 316-333.

Shaffer, C. A., Cooper, M. L., Alon, A. J. D., Akbar, M., Stewart, M., Ponce, S., & Edwards, S. H. (2010). Algorithm visualization: The state of the field. *ACM Transactions on Computing Education (TOCE)*, *10*(3), 9.

Shute, V. J. (1991). Who is likely to acquire programming skills? *Journal of Educational Computing Research*, *7*(1), 1-24.

Simon, S., Fincher, S., Robins, A., Baker, B, Box, I., Cutts, Q., de Raadt., M., Haden, P., Hamer, J., Hamil- ton, M., Lister, R., Petre, M., Sutton, K., Tolhurst, D., & Tutty, J. (2006). Predictors of success in a first programming course. *ACM International Conference Proceeding Series*; *Proceedings of the 8th Australian conference on Computing education*, January 16 – 19, Hobart, Tasmania, Australia.

Sivasakthi, M., & Rajendran, R.(2011).Learningdifficultiesof'object-oriented programming paradigm using Java':students'perspective. *Indian Journal of Science and Technology*, 8(4). s.983-985.

Sivilotti, P. A. G., & Laugel, S. A. (2008). Scratching the surface of advanced topics in softwareengineering: a workshop module for middle school students. *39th Technical Symposium on Computer Science Education*, March 12-15, Portland, OR, USA.

Sleeman, D., Putnam, R. T., Baxter, J., & Kuspa, L. (1986). Pascal and high-school students: A study of misconceptions. *Journal of Educational Computing Research*, 2(1), 5-23 .

Smith, B. J. (2009). Conceptual graphs as a visual programming language for teaching programming.

Stasko, J. T., Badre, A., & Lewis, C. (1993). Do algorithm animation assist learning?An empirical study and analysis. In S. Ashlund, K. Mullet, A. Henderson, E. Hollnagel,& T. White (Eds.), *Proceedings of the SIGCHI conference on human factors incomputing systems (ACM INTERCHI'93)* (pp. 61-66). New York: ACM.

Şahin, İ. ve Fındık, T. (2008). Türkiye'de Mesleki ve Teknik Eğitim: Mevcut Durum, Sorunlar ve Çözüm Önerileri. *TSA*. 12(3). p.65-86.

Şencan, H. (2008). Türkiye'de Mesleki ve Teknik Eğitim, Sorunlar-Öneriler. *(Rapor No.55)*, İstanbul: MÜSİAD.

Şimşek, A., Özdamar, N., Becit, G., Kılıçer, K., Akbulut, Y. ve Yıldırım, Y. (2008). Türkiye'deki eğitim teknolojisi araştırmalarında güncel eğilimler. *Selçuk Üniversitesi Sosyal Bilimler Enstitüsü Dergisi*, 19, 439-458

Tan, P. H., Ting, C. Y., & Ling, S. W. (2009). Learning Difficulties in Programming Courses: Un dergraduates 'Perspective and Perception. *İn 2009 International Conference on Computer Technology and Development*, Kota Kinabalu, Malaysia.p188.

Tekerek, M., & Altan, T. (2014). The effect of Scratch environment on students achievement in teaching algorithm. *World Journal on Educational Technology*,*6*(2).

Thomas, L., Ratcliffe, M., & Thomasson, B. (2004). Scaffolding with object diagrams in first year programming classes: Some unexpected results. *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education*, March 3 – 7, Norfolk, Virginia, USA.

Traynor, D., & Gibson, P. (2004). Towards the development of a cognitive model of programming; A software engineering approach. *16th PPIG Workshop*, Carlow, Ireland. http://www.ppig.org/papers/16thtraynor. pdf adresinden 24 Mart 2015 tarihinde edinilmiştir.

Tsai, M. J., & Tsai, C. C. (2003). Student computer achievement, attitude, and anxiety: The role of learning strategies. *Journal of Educational Computing Research*, *28*(1), 47- 61.

Tzavara, A., & Komis, V. (2003). *ICT in the education of preschool future teachers*: Examples of multidisciplinary units, In proceedings of the 2nd Pan-Hellenic Conference on "ICT in Education" (Syros, Greece, May, 9-11, 2003).

Uçar, C. ve Özerbaş, M. A. (2013). Mesleki ve Teknik Eğitimin Dünyadaki ve Türkiye'deki Konumu. *Eğitim ve Öğretim Araştırmaları Dergisi.* 2(2). p.242-253.

Utting, I., Cooper, S., Kölling, M., Maloney,J., & Resnick, M. (2010). Alice, greenfoot, and scratch–a discussion. *ACM TransactionsonComputing Education*,10(4),1-11.

Ünver, H. M., Yaylı, H. ve Ceylan, H.(2009). Taşra Meslek Yüksekokullarının Sorunları ve Çözüm Önerileri. *1. Uluslararası 5. Ulusal Meslek Yüksekokulları Sempozyumu.* Selçuk Üniversitesi. Konya. p.3063-3070

Van Haaster, K., & Hagan, D. (2004, June). Teaching and learning with BlueJ: An evaluation of a pedagogical tool. In *Information Science+ Information Technology Education Joint Conference, Rockhampton, QLD, Australia* (pp. 455-470).

Velasquez, N. F., Fields, D., Olsen, D., Martin, T., Shepherd, M. C., Strommer, A., & Kafai, Y. B. (2014). Novice programmers talking about projects: What automated text analysis reveals about online Scratch users' comments. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on* (pp. 1635-1644). IEEE.

Vobornik, P. (2011). Teaching algorithms using multimedia tools. *8th International Conference onEfficiency and Responsibility in Education*, June 9-10 Prague, Czech Republic.

Wang, D., Zhang, C., & Wang, H. (2011). *T-Maze: a tangible programming tool for children*. Paper presented at the Proceedings of the 10th International Conference on Interaction Design and Children, Ann Arbor, Michigan.

Wang, X., & Zhou, Z. (2011). The research of situational teaching mode of programming in high school with Scratch. In *Information Technology and Artificial Intelligence Conference (ITAIC), 2011 6th IEEE Joint International*(Vol. 2, pp. 488-492). IEEE.

Wettel, R., & Lanza, M. (2008, May). Codecity: 3d visualization of large-scale software. In *Companion of the 30th international conference on Software engineering* (pp. 921-922). ACM.

Wiedenbeck, S. (2005). *Factors Affecting the Success of Non-Majors in Learning to Program.* Paper presented at the First international workshop on Computing education research, Seattle, WA, USA.

Wiedenbeck, S., Labelle, D., & Kain, V. N. (2004, April). Factors affecting course outcomes in introductory programming. In *16th Annual Workshop of the Psychology of Programming Interest Group* (pp. 97-109).

Wiedenbeck, S., Ramalingam, V., Sarasamma, S., & Corritore, C. (1999). A comparison of the comprehension of object-oriented and procedural programs by novice programmers. *Interacting with Computers*, *11*(3), 255-282.

Wilson, B. C., & Shrock, S. (2001). *Contributing to success in an introductory computer science course: a study of twelve factors*. Paper presented at the

Proceedings of the thirty-second SIGCSE technical symposium on Computer Science Education, Charlotte, North Carolina, USA.

Winslow, L. E. (1996). Programming pedagogy - A psychological overview. *SIGCSE Bulletin*, *28*(3), 17- 22.

Wolz, U., Maloney, J., & Pulimood, S. M. (2008). 'scratch'your way to introductory cs. In *ACM SIGCSE Bulletin* (Vol. 40, No. 1, pp. 298-299). ACM.

Yadin, A. (2011). Reducing the dropout rate in an introductory programming course. *Acm Inroads*, *2*(4), 71-76.

Yazıcı, C., ve Kültür, C. (Çağıltay, K. Ve Göktaş, Y.). (2013). Medya mı Yöntem mi? Bitmeyen tartışma. Kürşat Çağıltay, Yüksel Göktaş. *Öğretim Teknolojilerinin Temelleri: Teoriler, Araştırmalar, Eğilimler.* Ankara: PegemA Yayıncılık.

Yıldırım, A., ve Şimşek, H. (2006). *Sosyal bilimlerde nitel arastirma yöntemleri*. Seçkin Yayıncılık. S.227.

Yorulmaz, M. (2008) Internet Kafelerin Daha Faydalı Kullanılabilmeleri İçin Bir Öneri: Scratch

Yükseltürk, E., ve Top, E. (Çağıltay, K. Ve Göktaş, Y.). (2013). Web 2.0 Teknolojilerinin Öğretmen Eğitiminde Kullanımı. *Öğretim Teknolojilerinin Temelleri: Teoriler, Araştırmalar, Eğilimler.* Ankara: PegemA Yayıncılık.

EKLER

EK 1. Akademik Başarı Testi

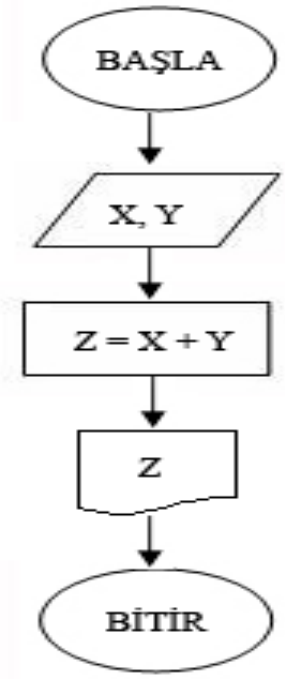
1. Aşağıdaki yapılardan hangisi veri girişi olarak kabul edilir.
   a) Yazıcıdan çıktı almak
   b) Klavyeden bir tuşa basmak
   c) Hoparlörden ses almak
   d) Yazının ekranda görünmesi

```
CLS
INPUT "VİZE=", V
INPUT "FİNAL=", F
ORT = V * 0.4 + F * 0.6
IF ORT <= 60 THEN SONUC$="KALDI"
ELSE
SONUC$="GEÇTİ"
PRINT ORT, SONUC$
```

2. Yukardaki programın V=70 ve F=60 için ekran çıktısı aşağıdakilerden hangisi olur?
   a) 59.4 KALDI
   b) 64 KALDI
   c) 64 GEÇTİ
   d) 59.6 GEÇTİ

3. Bir basketbol takımı seçmelerinde seçmeye katılacak kişilerin boyları (B) 1.70 cm az olmamalı veya kilosu (K) 115kg geçmemelidir cümlesinin programlamada (söz dizimi) yazılışı, aşağıdaki seçeneklerden hangisidir?

   a) K <= 115 AND B <= 1.70
   b) K <= 115 OR B <= 1.70
   c) K < 115 AND B <= 1.70
   d) K <= 115 OR B >= 1.70



Şekil_1

4. Şekil_1 deki programın açıklaması aşağıdakilerden hangisidir?

   a) Ekrandan girilen iki sayının toplamını bulan bir program.
   b) Sabit iki sayının ortalamasını bulan bir program.
   c) Sabit iki sayının toplamını bulan bir program
   d) Ekrandan girilen iki sayının ortalamasını bulan program.

5. Şekil_1 deki programda ekrana yazdırılan değişken aşağıdakilerden hangisidir?

   a) Y değişkeni
   b) X değişkeni
   c) Z değişkeni
   d) X ve Y değişkenleri

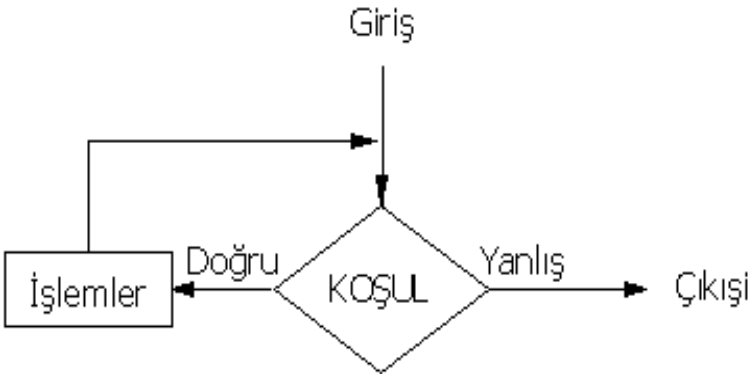
6. Şekil_1 deki programda X=5 ve Y=9 değerleri için programın üreteceği çıktı ne olur?

   a) 45 b) 14 c) 7 d) 4

7. Bir yük kamyonunun darası A ve yüklü ağırlığı ise B dir. Darası yüklü ağırlığının iki kartından fazla olmamalıdır cümlesinin programlama yazılışı karşılığı aşağıdaki seçeneklerden hangisidir?

   a) A <= 2B
   b) A+B<= 2B
   c) A <= 2*B
   d) A+B <= 2*B

8. Aşağıdaki şekil programlamanın hangi yapısıyla alakalıdır?



a) Döngüsel yapı
b) Operatör yapısı
c) Dizi yapısı
d) Değişken belirleme yapısı

9. İf (sayi1>0) || (sayi1<5) ifadesindeki koşul aşağıdakilerden hangisidir?

a) sayi1 büyüktür sıfırdan veya sayi1 büyüktür 5'ten.
b) sayi1 büyüktür sıfırdan veya sayi1 küçüktür 5'ten.
c) sayi1 büyüktür sıfırdan ve sayi1 büyüktür 5'ten.
d) sayi1 büyüktür sıfırdan ve sayi1 küçüktür 5'ten.

10. Aşağıdaki program parçasının bellek görüntüsü ve ekran çıktısını hangisidir?

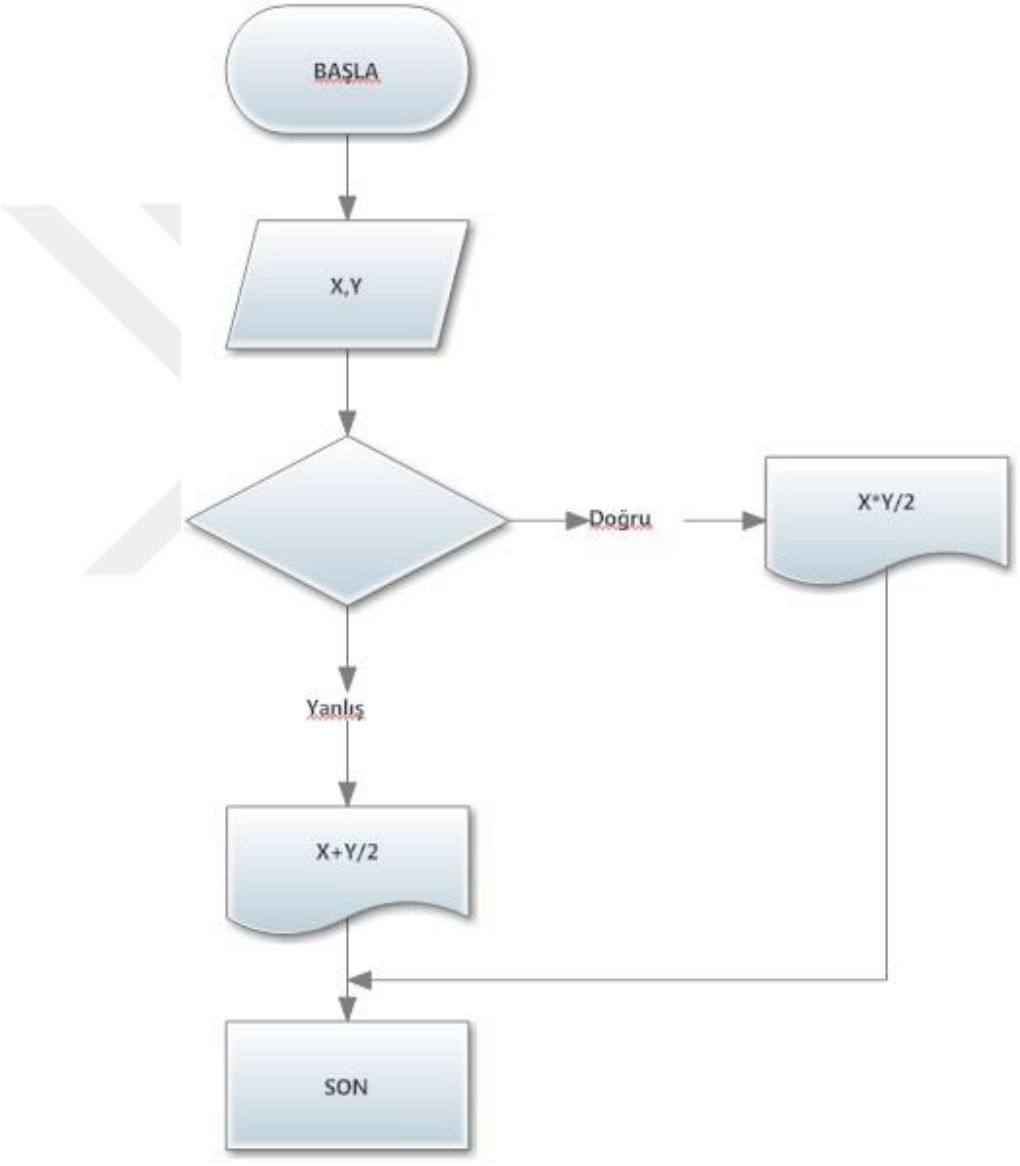
```
FOR  P = 2  TO  4
FOR  Y = 7  TO 9
PRINT  Y;
NEXT  Y
PRINT  P
NEXT  P
```

a)	b)	c)	d)
7 8 9 2	8 7 2 9	7 9 2 7	7 8 8 2
7 8 9 3	8 8 3 8	7 9 3 7	8 8 7 3
7 8 9 4	8 8 4 7	7 9 4 7	9 8 8 4

```
CLS
A = 5
B = 2
TOPLAM = A + B
ORT = TOPLAM / 2
PRINT ORT
END
```

11. Yukardaki programın açıklaması aşağıdakilerden hangisidir?

a) Sabit iki sayının ortalamasını bulup yazdıran algoritma.
b) Klavyeden girilen iki sayının toplamını ve ortalamasını bulup yazdıran algoritma.
c) Klavyeden girilen iki sayının ortalamasını bulup yazdıran algoritma.
d) Sabit iki sayının toplamını ve ortalamasını bulup yazdıran algoritma.



Şekil_4

12. Şekil_4 de boş bırakılan yere X<Y ifadesi getirildiğinde sırasıyla 2 ve 3 değerleri için elde edilen çıktı değeri ne olur?

a) 3 b) -0,5 c) 5 d) 2,5

13. Şekil_4 de hangi programlama yapısı bulunmamaktadır?
    a) Döngü yapısı
    b) Giriş – Çıkış yapısı
    c) Değişken atama yapısı
    d) Koşul yapısı

14. Şekil_4 için girilen 8 ve 2 değerleri ile 5 sonucunu çıktı olarak alabilmek için boş bırakılan alana aşağıdakilerden hangisini getirmek doğru olabilir?

    a) Y<2
    b) X=X/2+Y+2
    c) X<Y
    d) X/2>Y*2

15. Bir robota merdiven sonuna kadar her bir basamağı çıkmak için ayağını 10 cm kaldırıp 20 cm ileri götürmesi komutu verilmek istenmektedir. Her bir basamak için tekrar tekrar verilmesi gereken bu komutları daha kısa şekilde ifade etmek için aşağıdaki yapılardan hangisinin kullanılması uygun olabilir?

    a) FOR
    b) IF…..ELSE
    c) SELECT …… CASE
    d) FOREACH

16. Bir arabanın hızı H ve A ve B arasında alacağı yol için geçen süre T olan bir yolculuk için önerilen şart arabanın hızı 110 km geçmemeli ve A ve B arasındaki mesafeyi 180 dakikanın altında almamalı cümlesinin programlama karşılığı olan yazılışı aşağıdaki seçeneklerden hangisidir?

    a) if h< 110 AND T >=180
    b) if h< 110 AND T >180
    c) if h<= 110 AND T<=180
    d) if h<= 110 AND T >=180

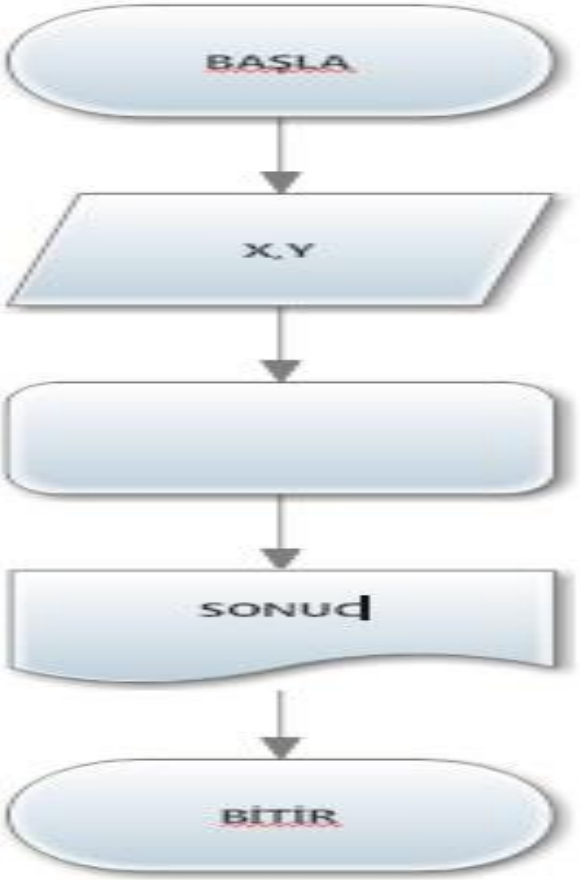
17. Herhangi bir problem için oluşturulan yapının, görsel olarak simge ya da sembollerle ifade edilmiş şekline ..... denir.

    a) Sabit c. Operatör
    b) Algoritma d. Giriş

```
CLS
FOR  A = 0  TO  5
PRINT  A ;
NEXT  A
```

18. Yukardaki programın açıklaması aşağıdakilerden hangisidir?

a) 5 ten büyük olan sayıları ekranda alt alta ve yan yana gösteren programı yazınız.
b) Sıfırdan 5'e kadar olan sayıları ekranda alt alta ve yan yana gösteren programı yazınız.
c) Sıfırdan 5'e kadar olan sayıları ekranda alt alta gösteren programı yazınız.
d) Sıfırdan 5'e kadar olan sayıları ekranda yan yana gösteren programı yazınız.



Şekil_3

19. Şekil_3 de boş bırakılan alana SONUC=X+X*Y ifadesi yerleştirildiğinde ve girdi olarak da 4 ve 5 değerleri alındığında programın sonuç çıktısı ne olur?

a) 45 b) 24 c) 40 d) 49

20. Şekil_3 deki akış diyagramında kaç tane değişken vardır?

    a)4 b)3 c)2 d)1

21. Şekil_3 deki gösterim sonucu iki sayının ortalamasının alınması istenmektedir. Bu görüş doğrultusunda boş alana aşağıdaki seçeneklerden hangisi getirilmelidir?

    a) SONUC=(x+y)/2
    b) SONUC=x+y/2
    c) SONUC=x/2
    d) SONUC=x/y

22. Aşağıdaki cümlelerin başında boş bırakılan parantezlere, cümlelerde verilen bilgiler doğru ise D, yanlış ise Y yazınız.

    ( ) For döngüsünde koşul sağlanmadığı müddetçe döngü çalıştırılmaz.

    ( ) İç-içe birden fazla if ifadesi kullanılamaz.

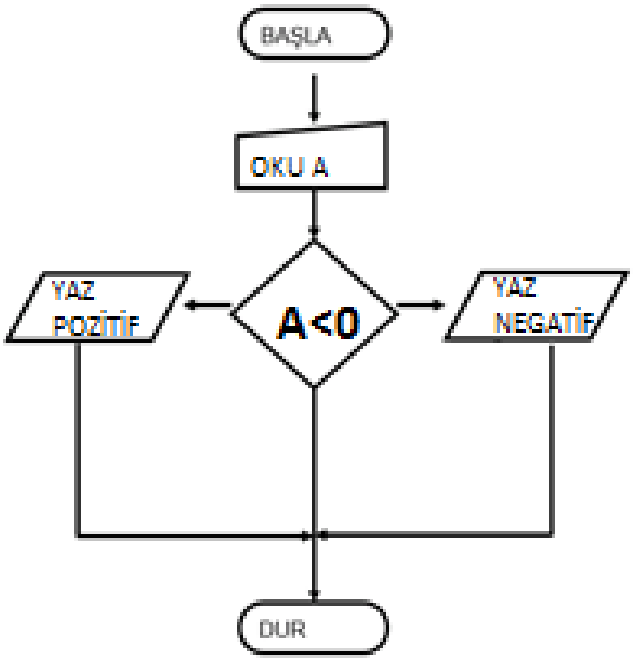
    ( ) Döngüler yalnızca ileri dönük sayma işlemi gerçekleştirir, geriye doğru sayma işlemi gerçekleştiremezler.

    ( ) Goto anahtar sözcüğü ile bir döngü ve koşul bloğu içerisine dallanma işlemi gerçekleştirilebilir.

    ( ) For döngüsünde sayaç char türünde de olabilir.

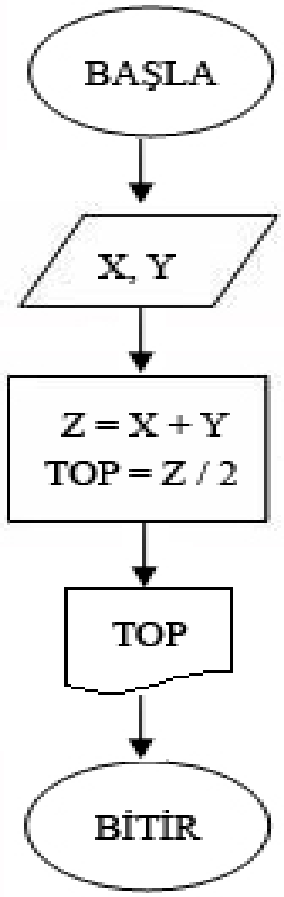
    ( ) Dışardan alınan değerin değişkene atanması “ ; ” işaretiyle olur.

    ( ) Eğer koşulun sağlanmaması durumunda işlem yapılması isteniyorsa, else ifadesine gerek duyulur.



23. Yukardaki akış şeması verilen programın editördeki karşılığı aşağıdakilerden hangisidir?

a)
```
INPUT "Sayiyi Giriniz:"; A
    IF A>0 THEN PRINT "Pozitif"
    ELSE IF A<0 THEN "Negatif"
    END
```
b)
```
    INPUT "Sayiyi Giriniz:"; A
    IF A>0 THEN PRINT "Pozitif"
    IF A>=0 THEN PRINT "Negatif"
    END
```
c)
```
    INPUT "Sayiyi Giriniz:"; A
    IF A>0 THEN PRINT "Pozitif"
    ELSE THEN PRINT "Negatif"
    END
```
d)
```
   INPUT "Sayiyi Giriniz:"; A
   IF A>0 THEN  "Pozitif"
   IF A<0 THEN  "Negatif"
   END
```



Şekil_2

24. Şekil_2 deki programın açıklaması aşağıdakilerden hangisidir?

a) Klavyeden girilen iki sayının ortalamasını bulan program.
b) Sabit olan iki sayının toplamını bulan program.
c) Klavyeden girilen iki sayının toplamını bulan bir program
d) Sabit iki sayının ortalamasını bulan bir program

25. Şekil_2 deki program ekrana çıktı olarak aşağıdakilerden hangisi yazdırır?

a) TOPLAM değeri
b) Z değeri
c) TOP değeri
d) Ortalama değeri

26. Şekil_2 deki programda X=6 ve Y=8 değerleri için ekran çıktısı aşağıdakilerden hangisi olur?

a) 42
b) 14
c) 7
d) 6

27. Aşağıdaki programın çıktısı hangisidir?

```
CLS
A=19
C=A-2
B=C+6
IF B>C THEN PRINT “İYİ”
IF C<A THEN PRINT “KÖTÜ”
```

a) İYİ
KÖTÜ

b) İYİ

c) BOŞ

d) KÖTÜ

28. Aşağıdaki programın çıktısı nedir?

```
CLS
FOR G= 1  TO  4
PRINT  11+G
NEXT G
```

a) 4
5
6
7

b) 4
3
2
1

c) 1
2
3
4

d) 12
13
14
15

29. Klavyeden boy bilgisi giriliyor. Boy,150cm’den kısa ise “Boyunuz kısa”; 150-170cm arasında ise “Boyunuz Normal”; 170cm’den uzun ise “Boyunuz Uzun” mesajını ekranda görüntüleyen programın içerisinde aşağıdaki programlama yapılarından hangisi bulunur.

a) İç içe döngü yapısı
b) İç içe şart yapısı
c) Döngü yapısı
d) Sayaç yapısı

```
A1 :sayac = 0
A2 :a değerini gir.
A3 :Eğer a < 0 A2'ye git.
A4 :toplam = toplam + a
A5 :…….
A6 :Toplam değerini yaz.
A7 :Bitir.
```

30. Yukardaki programda, klavyeden girilen 5 adet pozitif sayının toplamını ekrana yazdırması için, boş bırakılan satıra aşağıdaki seçeneklerden hangisi yazılmalıdır?

a) Eğer sayac < 5 A4’e git
b) Eğer sayac < 6 A2’ye git
c) Eğer sayac > 5 A4’ e git
d) Eğer sayac < 5 A2' ye git

31. Aşağıdakilerden yapılardan hangisi veri çıkışı olarak kabul edilir.

a) Cep telefonunda numara çevirme
b) Bankamatiğe şifre yazmak
c) Mouse ile resmi çift tıklama
d) Programın hata mesajı vermesi

Adı:			
Soyadı:			
Grup	A	B	C

EK 2. Bilgisayar Programlama Kaygı ölçeği

Lütfen, programlama ile ilgili kendinizi nasıl hissettiğinizi en iyi açıklayan numarayı yuvarlak içine alarak tüm ifadeleri işaretleyiniz. Teşekkürler.

Hiçbir zaman 0 doğru	Nadiren 1 doğru	Bazen 2 doğru	Sıkla 3 doğru	Her zaman 4 doğru

1.	Bilgisayarlar kendimi tedirgin ve rahatsız hissetmeme sebep oluyor.	0	1	2	3	4
2.	Programlarda hata ayıklama benim için büyük bir endişe kaynağıdır.	0	1	2	3	4
3.	Program yazmayı başaramayarak kendimi arkadaşlarımın önünde komik duruma düşürmekten tedirgin oluyorum.	0	1	2	3	4
4.	Birçok bilgisayar terimi yüzünden aklım karışıyor.	0	1	2	3	4
5.	Programlamayı anlayabileceğimi düşünmüyorum.	0	1	2	3	4
6.	Yazdığım programlarda birçok hata çıkıyor.	0	1	2	3	4
7.	Programımda hata yaptığım anda öğretmenim tarafından seçilmekten endişe duyuyorum.	0	1	2	3	4
8.	Eğer programım çalışmazsa, sonraki aşamada ne yapacağımı bilemiyorum.	0	1	2	3	4
9.	Programım çalışmadığı zaman endişeleniyorum.	0	1	2	3	4
10.	Öğretmenimin benim iyi program yazamadığımı biliyor olması beni endişelendiriyor.	0	1	2	3	4
11.	Programımın satırlarında hata olduğunda düzgün düşünemiyorum.	0	1	2	3	4
12.	Programımdaki hataları tekrar tekrar düzeltmek zorunda kaldığımda kendimi gergin hissediyorum.	0	1	2	3	4
13.	Program satırları daha karmaşık olmaya başladığında yoğunlaşamadığımı hissediyorum.	0	1	2	3	4
14.	Daha yetenekli bilgisayar programlama öğrencilerinin varlığı beni geriyor.	0	1	2	3	4
15.	Arkadaşlarım programlama yapabildiği fakat benim yapamadığım zamanlarda kendimi üzgün hissediyorum.	0	1	2	3	4
16.	Bir bilgisayar sorununu çözemediğimde zihnim boşalmış gibi oluyor.	0	1	2	3	4
17.	Bilgisayar programımda hataların olduğu her zaman kendimi gergin hissediyorum.	0	1	2	3	4
18.	Arkadaşlarım benim anlamadığım programlama konularında konuştuğunda sıkıntı yaşıyorum.	0	1	2	3	4
19.	Bilgisayarla çalışma konusunda her zaman kendimi güvensiz hissediyorum.	0	1	2	3	4

EK 3. Örnek Ders Tasarımı

DERS PLANI

ŞARTLI YÖNLENDİRME [EGER (İF)] YAPISI

Dersin Anlatımı:

Karar verme aşaması programcılıkta neredeyse bütün programlarda mevcuttur. Programların genelinde kullanılan bu yapının öğrenenler tarafından bilinmesi çok önemlidir. IF olarak isimlendirilen bu yapı Türkçe karşılık olarak EĞER ifadesiyle adlandırırlar. Türkçe konuşmalardada kullandığımız bu sözcük bir durumun belirli bir şarta bağlı olarak gerçekleşeceğini anlatır. Programcılıkta kullanılan IF (EĞER) yapısının Tek durumlu ve iki durumlu olmak üzere iki kullanımı vardır.

(İF) EĞER 'in (tek durumlu) kullanımı:

" Trafik lambası sarı yandıysa bekle."

Burada kişinin yapması gereken **EĞER** trafik lambası sarı **İSE** bekle.

"Okula devam etmezsen derslerden başarılı olamazsın"

Kişinin göstermesi gereken davranış **EĞER** okula devam etmez **İSE** derslerden başarılı olamaz.

Belirtilen örnekler İF (EĞER) yapısının tek durumluk kullanımına örnek teşkil etmektedir. Burada önemli olan şartın gerçekleşme durumudur. Dolayısı ile gerçekleşmeme durumu söz konusu değildir. Bu kullanım iekli İF (EĞER) yapısının basir kullanımıdır.

Program yazılımı:

EĞER ŞART gerçekleşiyor İSE şunu/şunları

Örnek;

EĞER X <= 15 İSE YAZ "X SAYISI 15'E EŞİT VEYA 15'DEN KÜÇÜKTÜR"

> Program satırı açıklaması; X değişkenin değeri 15 veya 15 den küçükse ekrana "X SAYISI 15'E EŞİT VEYA 15'DEN KÜÇÜKTÜR" yazmasıdır.

(İF) EĞER 'in (iki durumlu) kullanımı:

"Sıcaklık 16 dan yüksekse hava sıcak, sıcaklık 16 dan düşükse hava soğuk"

Yapılacak şey havanın durumuna bakmak EĞER YÜKSEKSE hava sıcak, DÜŞÜKSE hava soğuk.

"Asansörü çağırma düğmesine bas EĞER kabini görüyorsan asansöre bin, YOKSA beklemeye devam et."

Yapılacak şey asansörü çağırma düğmesine basmak EĞER gelmişse kabin VARSA asansöre binmeliyiz, YOKSA beklemeye devam etmeliyiz.

Program Yazma Aşamaları:
Program yazmak için aşağıdaki adımlar sırayla takip edilir:

Problemi analiz etmek;

Problem: Klavyeden girilen sayının negatifmi pozitifmi olduğunu bize döndüren bir algoritma tasarlayarak programını yazınız ?

Çözüm yolları düşünmek,

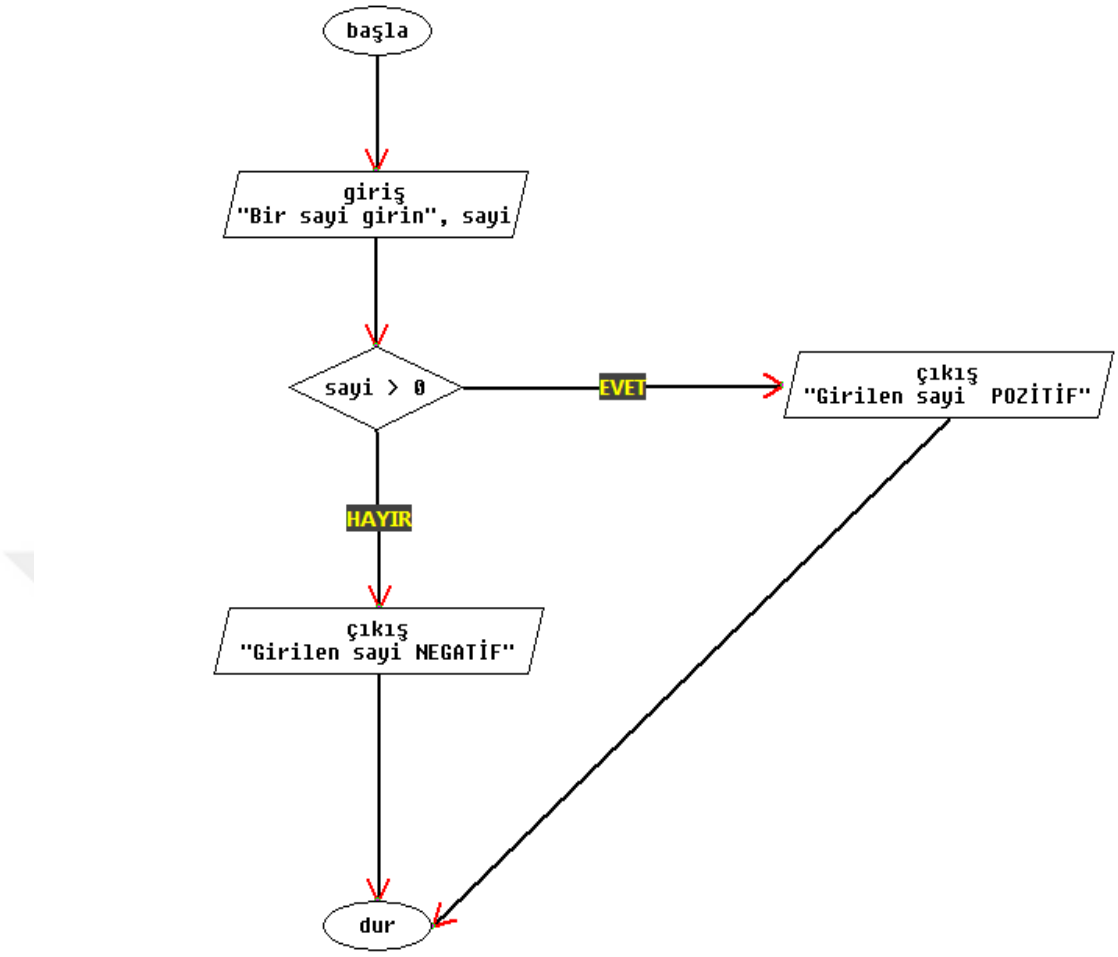
Nasıl Çözeriz: İlk olarak problem çözme aşamalarını belirleriz.

a) Klavyeden girilen dediğine göre input kullanılacak.
b) Bir sayı dediğine göre bir tane değişken gerekli
c) Bir değişken girilen değere atanmalı
d) Kıyaslama olduğuna göre şartlı yapı var
e) Sonuçu yazdırmak içinde print kullanılacak

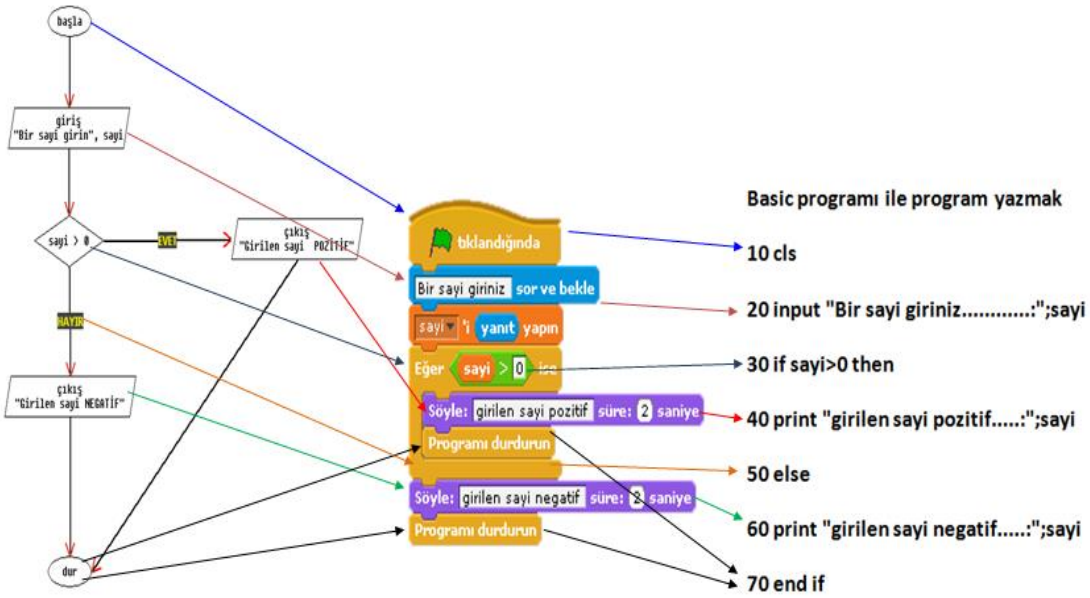
Uygun çözüm yolları bularak **Algoritma** oluşturmak,

Değişkenler
Sayımız: sayı
Algoritma
Adım 1:Başla
Adım 2:Birinci sayıyı gir.(sayı)
Adım 3:Eğer sayı > 0 ise ekrana Girilen sayı " pozitif " yaz.
Adım 5:Değilse. Girilen sayı " Negatif " yaz.
Adım 6:Bitir

Akış Diyagramı çizmek



Uygun bir programlama dilinde **Programı** yazmak



Program Yazma Aşamaları:
Program yazmak için aşağıdaki adımlar sırayla takip edilir:
Problemi analiz etmek;

Problem:

Çözüm yolları düşünmek,

Nasıl Çözeriz:

Uygun çözüm yolları bularak **Algoritma** oluşturmak,

Değişkenler
Algoritma

Akış Diyagramı çizmek

Uygun bir programlama dilinde **Programı** yazmak

Problem: Klavyeden girilen iki sayıdan büyük olanını ekrana yazan algoritmayı tasarlayarak programını oluşturunuz ?

Nasıl Çözeriz: İlk olarak problem çözme aşamalarını belirleriz.

a) Klavyeden girilen dediğine göre input kullanılacak.
b) İki sayı dediğine göre iki tane değişken gerekli
c) İki değişken girilen değerlere atanmalı
d) Kıyaslama olduğuna göre şartlı yapı var
e) Sonuçu yazdırmak içinde print kullanılacak.

EK 4. Program Görselleştirme Ortamı Tanıtımı

Program Görselleştirme Aracı (Scratch)'nin Tanıtımı

Sctratch programı eğlenceli bir ortamda resim, ses, müzik gibi çeşitli medya araçlarını bir araya getirebileceğimiz, kendi animasyonlarımızı, bilgisayar oyunlarımızı tasarlayabileceğimiz ya da interaktif hikâyeler anlatabileceğimiz ve paylaşabileceğimiz bir eğitsel programlama dilidir.

Sctratch ana ekranı tanıyalım.

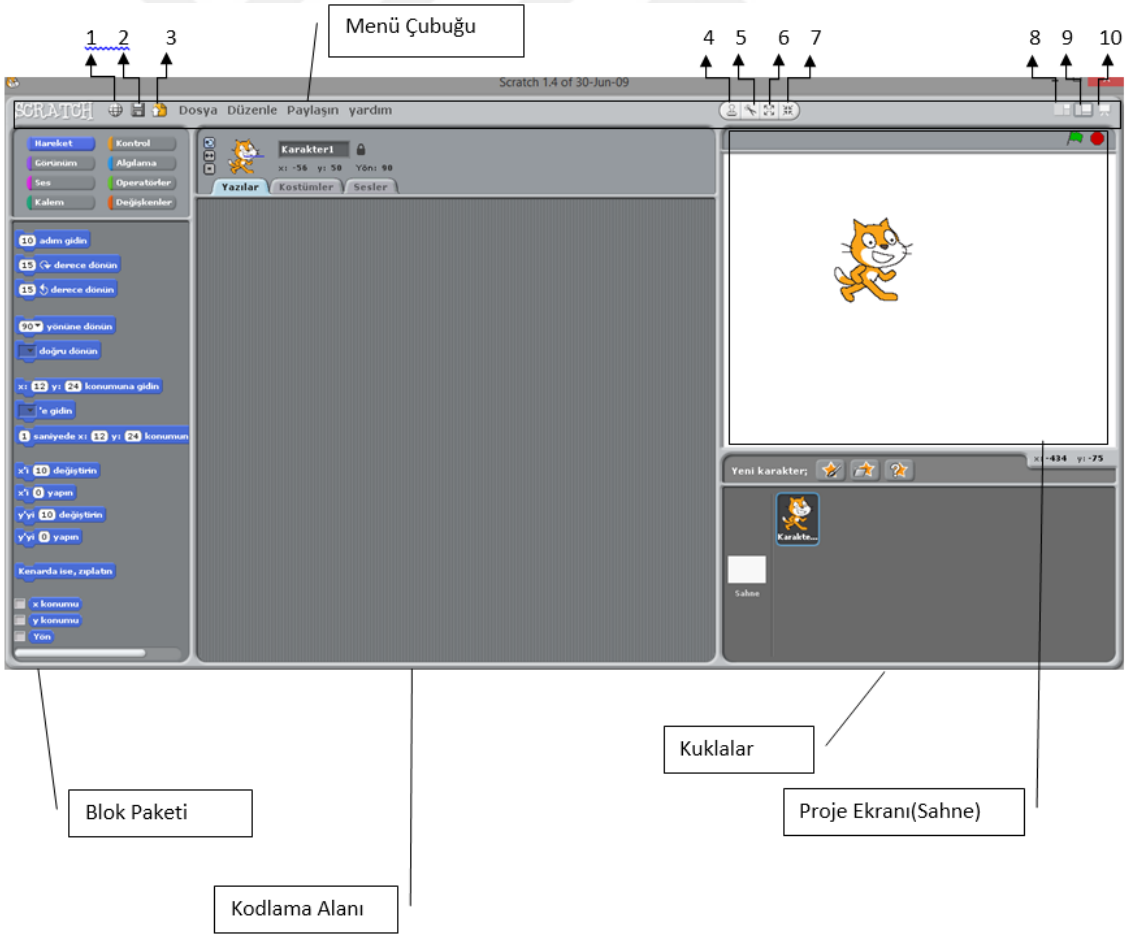
1: Dili Ayarla **6:** Karakter Büyültün

2: Projeyi Kaydet **7:** Karakter Küçültün

3: Projeyi Paylaş **8:** Küçük Ekrana Geçin

4: Çoğaltma **9:** Tam Ekrana Geçin

5: Kesme **10:** Sunum Ekrana Geçin



MENÜ ÇUBUĞU

Dil ayarla: Bu ikonu kullanarak kullandığımız blok dilini değiştirebilir farklı dillerde kullanabiliriz.

Kopyasını Çıkar: Karakterimizi çoğaltabiliriz.

Sil: İstediğimiz bir karakteri silebiliriz.

Büyüt: Karakterimizin üzerine tıkladığımızda karakterimizi istediğimiz kadar büyültebiliriz.

Ufalt: Bu simgeyi seçerek karakterimizin üzerine tıkladığımızda karakterimizi istediğimiz kadar küçültebiliriz.

Yardım: Scratch ve komut blokları hakkında zorlandığımızda yardım sağlar.

DOSYA MENÜSÜ

1.Yeni: Yeni bir boş proje oluşturmamızı sağlar.

2.Aç: Seçeneği daha önce yaptığımız kayıtlı bir projeyi yeniden açıp üzerinde değişiklik yapmanı sağlar.

3.Kaydet: Seçeneği üzerinde çalıştığın projeyi daha sonra açabilmek için bilgisayara kaydetmeni sağlar.

4.Farklı Kaydet: Seçeneği daha önce kaydettiğin bir projenin bir kopyasını, başka bir isimle kaydetmeni sağlar.

5.Projeyi İçeri Aktar: Seçeneği daha önce kaydettiğimiz başka bir projedeki karakterleri ve yazıları o anda açık olan projenin içine aktarmanı sağlar.

6.Dışa Karakter Aktar: Seçeneği seçtiğin bir karakteri, daha sonra başka projelerde kullanmak için kaydetmeni sağlar.

7.Proje Notları: Seçeneği projeni daha sonra açtığında hatırlamak istediklerini yazmanı sağlar.

8.Çık: Seçeneği Scratch'ı tamamen kapatmanı sağlar.

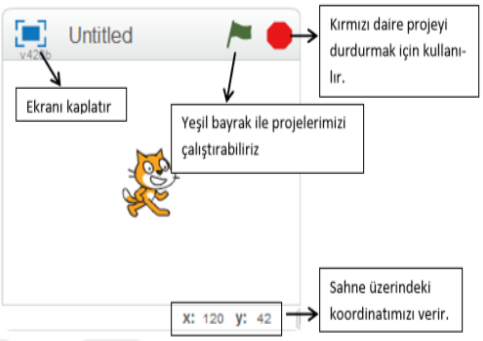
DÜZENLE MENÜSÜ

Silmeyi Geri Al: Seçeneği yanlışlıkla sildiğin blokları geri getirmeyi sağlar.

Adım Adım Çalışmayı Başlat: Eklediğimiz her bloğu(kodu) ayrı ayrı görerek çalıştırmayı sağlar.

SAHNE

Hazırladığımız projenin oynatıldığı yerdir. Oyunlarımız ve animasyonlarımız bu ekranda canlanır.



Ekran 480 birim genişlikte ve 360 birim uzunluktadır. Scratch ekranı aslında bir koordinat düzlemidir. Scratch programı açıldığında karşımıza çıkan kedi karakteri başlangıçta (0,0) noktasındadır.

SAHNE VE DEKOR AYARLARI

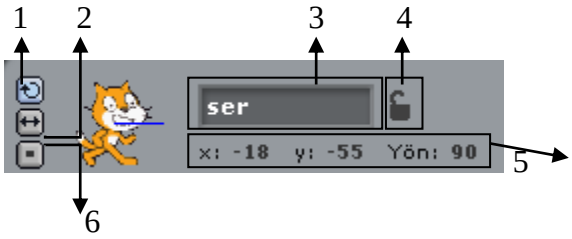
1. Yeni Dekor Çiz: Bir çizim ara yüzü sayesinde dekoru siz çizersiniz.

1 2 3 **2. Dekoru Kütüphaneden Seç:** Kütüphaneden dekor eklemek için kullanılır.

3. Sürpriz Dekor: Sahne alanına sürpriz dekorlar getirir.

KUKLA BİLGİSİ

Kuklanın sol üst köşesindeki tuşuna tıklanarak açılan bölümde kuklayla ilgili ayarlar yapmak için kullanılır. Bu ayarlar;



1. Tıklanırsa karakter kostümü 360° dönebilme özelliği kazanır.

2. Dönme Özellikleri: Karakterin nasıl döneceğini belirlendiği bölümdür. Bunlar;

2.1. Tıklanırsa karakter kostümü sağa-sola dönebilme

3. Kukla adı: Kuklanın (karakter)adını değiştirmek için kullanılır.

4. Web sitesinde taşınabilir mi?: Projemizi web sayfasına taşımamızı sağlar.

5. Konum ve Yön bilgisi: Kuklanın koordinatlarını(konum) ve yönünü belirtir bu bölümden karekterimizin yönü değiştirilebilir.

6.Döndürmeyin: Proje çalıştırıldığında kuklanın dönmesini engeller.

KILIKLAR (KOSTÜMLER)

Karakterimizin başka görünümlerinin olmasını istiyorsak kütüphaneden ekleyebiliriz veya kendimiz de çizebiliriz, bilgisayarımızda kayıtlı bir resmi içe aktarabiliriz ya da kamera ile çekebiliriz. Var olan kostümleri kostümün sağ üst köşesindeki 'x' butonuna basarak silebiliriz, seçili kostümü sağdaki düzenleme alanında düzenleyebiliriz. Kostüm üzerinde sağ tık yaparak kopyalayabilir ve silebiliriz.

SESLER

Karakter için var olan sesleri görebilmek için bu bölüme tıklarız. Bu bölümden yeni sesler ekleyebilir (sırasıyla kütüphaneden ses ekler, ses kaydeder, bilgisayarda kayıtlı bir sesi ekler), var olan sesi dinleyebilir, düzenle ve etkiler altındaki komutlarla düzenleyebilir veya silebiliriz.

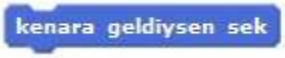
BLOK PAKETİ (DİZİLER)

Blok Paletinde karakterleri programlayabilmek için kullanılan bloklar vardır. 10 ana kategoride toplanmış blok grubu bulunmaktadır.

HAREKET

Hareket bloğunda yer alan blokların işlevleri tablodaki gibidir.

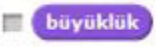
	Karakteri istediğimiz şekilde öne ya da arkaya doğru hareket ettirir.
	Karakterimizi istediğimiz derece kadar saat yönünde döndürür.
	Karakterin istenilen derece kadar saat yönü tersine dönmesini sağlar.
	Karakterin hangi yöne döneceğini belirler. (0=yukarı, 90=sağ,180=aşağı, -90=sol)
	Karakteri 'mause işaretçisi' veya başka karakterler varsa istenilen 'karakter' e doğru döndürülür.
	Karakteri istediğimiz x ve y konumuna getirir.
	Karakter 'mause işaretçisi' veya başka karakterler varsa istenilen 'karakter' ile aynı konuma gönderilir.

	Karakter belli bir süre içerisinde belirlenen x ve y konumuna göre hareket eder.
	Karakterin bulunduğu konumun x değeri istenilen değer kadar arttırılır veya azaltılır.
	Karakterin x konumunu istenilen değere çeker.
	Karakterin bulunduğu konumun y değeri istenilen değer kadar arttırılır veya azaltılır.
	Karakterin y konumunu istenilen değere çeker.
	Karakter ekranın kenarına değdiği an karakteri ters yöne döndürür.
	Karakterin sağa-sola dönme, etrafında dönebilme ve hiç dönememe ayarlarını yapar Karakterin x pozisyonu bilgisini verir.
	Bu seçenek işaretlendiğinde x değerini ekranda görebiliriz.
	Karakterin y pozisyonu bilgisini verir. Bu seçenek işaretlendiğinde y değerini ekranda görebiliriz. Karakterin yön bilgisini verir.
	Bu seçenek işaretlendiğinde yön bilgisini ekranda görebiliriz.

GÖRÜNÜM

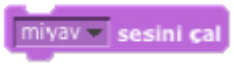
Görünüm bloğunda yer alan blokların işlevleri tablodaki gibidir

	Karakter istenilen süre boyunca ‘Hello!’ yazan kutudaki değeri ekranda konuşma balonu içerisinde gösterir.
	Karakter ‘Hello!’ yazan kutudaki değeri ekranda balon içerisinde gösterir. Kutu içerisi boş bırakılırsa konuşma balonu oluşmaz.
	Karakterimiz istenilen süre boyunca ‘Hmm…’ yazan kutudaki değeri ekranda düşünme balonu içeri- sinde gösterir.
	Karakterimiz ‘Hmm…’ yazan kutudaki değeri ekranda balon içerisinde gösterir.

	Karakterimizi sahne alanında görünür hale getirir.
	Karakterimizi sahne alanında görünmez yapar.
	Karakteri isteğimiz kostüme geçmesini sağlar.
	Karakterin o anki kostümünden bir sonraki kostümüne geçmesini sağlar.
	Seçili sahne dekorunu istediğimiz dekorla değiştiririz.
	Karakterin rengi, parlaklık, hayalet efekti sayısı yazan yerdeki değer kadar değişir.
	Karakterin rengi, parlaklık, hayalet efekti sayı yazan yerdeki değer olur.
	Karakter üzerinde uygulanmış bütün efektleri geri alır.
	Karakterin boyunu istediğimiz değer kadar değiştirir.
	Karakterin boyunu istediğimiz %' ye kadar getirir.
	Karakterin diğer bir üst katmana çıkmasını sağlar. Böylece karakter diğer karakterlerin önün- de görünebilir.
	Karakterin istenilen katman değeri kadar geri gitmesini sağlar. Böylece karakterimiz diğer karakterlerin arkasında gizlenebilir.
	Karakterin o anki kostüm bilgisini verir. Yan tarafındaki kutu işaretlendiğinde ekranda kostüm bilgisi görülür.
	Sahnede o anda gösterilen dekorun adını verir.
	Karakterimizin ebat değerlerini % olarak verir. Yan tarafındaki kutu işaretlendiğinde ebat değeri ekranda görünür.

SES

Ses bloğunda yer alan blokların işlevleri tablodaki gibidir.

	Seçtiğimiz olan müzik çalmaya başlar. Program müziğin bitmesini beklemeden diğer blokları çalıştırmaya devam eder.
	Seçili olan müzik çalmaya başlar ve bitene kadar diğer bloğa geçilmez.
	Çalmaya devam eden müzikler bu blok çalıştığında durur.
	48 (davul) sesini istenilen istenen sürede bir çalar. (18 sesinin bulunduğu yer açılır menüdür ve ses buradan değiştirilebilir.)
	İstenilen vuruşun bitmesini bekler.
	İstediğimiz notayı istenilen sürede bir çalmamızı sağlar.
	Enstrümanı(çalgıyı) değiştirmeyi sağlar.
	Sesi azaltır ya da arttırır.
	Sesin yüksekliğini belli bir % ye getirir.
	O anki ses seviyesinin bilgisini verir. Yan taraftaki kutu işaretlendiğinde ekranda ses bilgisi % olarak görürüz.
	Tempo değeri istediğimiz değere göre değiştirebiliriz.
	Tempo değeri saniyede istediğimiz vuruş değeri yapılır.
	O anki tempo değerinin bilgisini verir. Yan taraftaki kutu işaretlendiğinde ekranda tempo değeri görünür. (20-500)

KALEM

Kalem bloğunda yer alan blokların işlevleri tablodaki gibidir.

	Ekrandaki kalem ile çizdiklerimizi hepsini siler.
	Karakterin o anki konumunda ekrana kendi imajını (görüntüsünü) yapıştırır.
	Karakterin hareket ettiği doğrultuda kalem ile ekranı çizer.
	Karakterin hareket ettiği yerleri çizmez.
	Kalem rengi istediğimiz renk değerine geçer.
	Kalem rengi bu blok çalıştığında istenilen değer kadar değişir.
	Kalem renginin istediğimiz değer olmasını sağlar.
	Kalem tonu istenilen değer kadar değiştirilir. (kalem tonu 0’ dan 100’ e doğru açılır, 100’ den 200’e doğru koyulaşır.)
	Kalem tonu istediğimiz değer olur.
	Kalem kalınlığı istenen değer kadar değişir.
	Kalem kalınlığı istenen değer kadar olur.

OLAYLAR

Olaylar bloğunda yer alan blokların işlevleri tablodaki gibidir.

	Projemizin içerisinde yeşil bayrağa tıkladığımızda bu blok altında sıralanmış kodları çalışır.
	Seçilmiş olan tuşa (burada boşluk tuşu seçilmiş) basıldığında bu blok altında sıralanmış kodları çalıştırır.
	Karaktere tıklandığında bu blok altında sıralanmış kodları çalıştırır.
	Sahne dekoru burada seçilen dekor olunca bu blok altında sıralanmış kodları çalıştırır.
	Ses şiddeti belirtilen rakamdan yüksek olunca bu blok altında sıralanmış kodları çalıştırır.
	Yapılan duyurunun ardından bu blok altındaki kodları çalışır.
	Bu blok ile yeni haber yazarak komut kümelerini çalıştırmak için yeni bir blok oluştur muş oluruz. Bu blok çalıştığında yapılan duyurunun ardından bloktaki diğer komutlar çalışmaya devam ederken bu duyuruya ait kodlar da çalışmaya başlar.
	Yapılan duyurunun ardından bu bloğun al- tında kod kümesi var ise bu duyuruya ait kodları çalışmadan bekler, bitiminden sonra kendi çalışmasına devam eder.

KONTROL

Kontrol bloğunda yer alan blokların işlevleri tablodaki gibidir.

	Projemizi istediğimiz süre kadar bekletir, süre tamamlanınca sıradaki bloktan çalışmaya devam eder.
	Bu blok içerisinde bulunan bloklar sırasıyla istenilen değer kadar tekrar çalıştırılır.
	Bu blok içerisinde bulunan bloklar sırasıyla program durdurulana kadar tekrar tekrar çalıştırılır (bu da sonsuz döngü olur).
	Eğer' den sonra gelen koşul doğru ise program bu blok içerisine girer ve bu blokları çalıştırır, ardından bu blok içerisinden çıkarak sonraki bloktan çalışmaya devam eder.
	Eğer koşul doğru ise hemen altında yer alan kod blokları icra edilir ve bu kod bloğundan çıkılarak program icraya devam eder. Eğer koşul yanlış ise 'öyle değil ise' kısmında yer alan bloklar icra edilir, icradan sonra bu bloktan çıkılarak program akışı kaldığı yerden devam eder.
	Koşul doğru olana kadar beklenir, koşul doğru olunca aşağıdaki bloklar çalışmaya başlar. .
	Koşul doğru olunca blok içerisine girilerek kodlar icra edilir. Koşulun her döngüde doğru olması halinde icra devam eder. Koşul doğrulanmadığı halde altındaki kodlar çalıştırılır
	'Hepsini', 'bu diziyi', ' kuklanın kendi dizilerini' seçenekleri ile istenen kod dizilerini (bloklarının) çalışması durdurulur.
	Bir kuklanın ikizi program akışı esnasında komutla oluşturulduğunda bu blok altındaki komutlar çalışır
	Program akışı esnasında kendinin ya da seçilecek başka bir kuklanın ikizi oluşturulur.
	Oluşturulan kukla ikizin silinmesini sağlar

ALGILAMA

Algılama bloğunda yer alan blokların işlevleri tablodaki gibidir.

	Karakterimizin karaktere/ Mouse işaretçisine/ kenara değip değmediği kontrolünü yapar. (seçim yapmak için açılır menüye tıklayın.)
	Karakterin seçilen renge değip değmediği kontrolünü yapar. (Rengi değiştirmek için renk üzerine tıklayıp ekrandan renk seçin.)
	Karakterin üzerindeki seçtiğiniz rengin seçti- ğiniz bir başka renge değip değmediği kontrolünü yapar.
	Karakterin açılır menüden seçtiğimiz başka bir karakter ya da mause işaretçisiyle arasın- da olan mesafe bilgisini verir.
	Karakterin açılır menüden seçtiğimiz başka bir karakter ya da mause işaretçisiyle arasın- da olan mesafe bilgisini verir.
	Diye Sor ve bekle komutu ardından verdiğin yanıt bu 'yanıt' bloğunda tutulur. Blok yanındaki kutuya tıkladığında vermiş olduğun yanıt proje ekranında görünür.
	Seçili olan tuşa basılı olma durumunu kontrol eder.
	'Fare basılı' mı kontrolü yapar.
	Farenin x koordinatını verir.
	Farenin y koordinatını verir.
	Ses yüksekliği bilgisini barındırır.
	Web kamerasındaki görüntünün hareket oranını ve yönünü tespit eder.
	Web kamerasının görüntüsünü alarak sahne- de görünmesini sağlar.
	Web kamerasın görüntüsünü istenilen oran- da saydamlaştırır.

	Saniye cinsinden geçen süre bilgisini tutar.
	Süreölçeri sıfırlar.
	Seçili olan karakterin/sahnenin seçenekler dahilindeki (x koordinatı/ y koordinatı/ yön/ şu anda görünen kostüm/ ebat/ ses seviyesi) bilgisini kullanabilmemizi sağlar.
	Bilgisayarın tarih ve saatine göre yapılan seçim doğrultusunda şimdiki yıl, ay, gün, haftanın kaçıncı günü olduğu(Pazardan başlayarak), saat, dakika ve saniye bilgilerini verir.
	01.01.2000 tarih ve 00:00 saatten şu anki tarih ve saate kadar geçen gün sayısını verir.
	Tarih ve saate kadar geçen gün sayısını verir. Scratch kullanıcı adınızı içerir. Çalışması için siteye kullanıcı adınız ve şifrenizle giriş yapmış olmanız gerekir.

İŞLEMLER

İşlemler bloğunda yer alan blokların işlevleri tablodaki gibidir.

	Boş kutulara yazılan iki değeri toplar.
	Boş kutulara yazılan ilk kutunun içindeki değerden ikinci kutudaki değeri çıkarır.
	Boş kutulara yazılan iki değeri çarpar.
	İlk kutudaki değeri ikinci kutudaki değere böler.
	Belirtilen iki değer arasında rastgele bir sayı tutar.
	İlk kutudaki değerin ikinci kutudaki değerden küçük durumu kontrolü yapılır.
	İki değerin bir biri iler eşitliği durumu kontrolü yapılır.
	İlk kutudaki değerin ikinci kutudaki değerden büyük durumu kontrolü yapılır.
	Belirtilen iki koşulunda doğruluk kontrolünü yapar.
	Belirtilen iki koşuldan en az birinin doğru olması durumunu arar.
	Koşulda verilen durumun olmaması istendiği durumlar- da bu blok kullanılır.
	İstenilen iki değeri metin olarak birleştirilir.
	İstenilen değerin hangi sırasındaki harfi isteniyor ise bu blok ile bu bilgi elde edilir.
	Değerin uzunluk bilgisini bu blok ile elde edilir.
	Bu blok ilk kutuya yazılan değerin ikinci kutuya yazılan değere göre modunu bulurken elde edilen kalanı verir.
	İstenilen değerin en yakın tam sayıya yuvarlanmasını sağlar.
	Matematiksel terimleri bulabileceğimiz (mutlak değer, karekök, sin, cos, tan, asin, acos, atan, ln, log, e^, 10^) bu blok ile kutuya yazılan değerin karşılığını bilebilir ve bu değeri sayısal işlemlerde kullanabiliriz.

VERİ

Veri bloğunda yer alan komutların ve blokların işlevleri tablodaki gibidir.

Bir Değişken Oluştur	Bu komut ile yeni bir değişken oluştururuz.
☑ Puan	Oluşturulan 'Puan' isimli değişken verisidir. (Kutu işaretlendiğinde ekranda değişken değerinin ne olduğu görünür.)
Puan , 0 olsun	Var olan değişkenlerden istediğimiz seçilerek istenilen değer yapılır.
Puan 'i 1 arttır	Var olan değişkenlerden istenilen seçilerek istenilen değer kadar artırılabilir ya da azaltılabilir.
Puan değişkenini göster	Var olan değişkenlerden istenilen değişkeni ekranda gösterir.
Puan değişkenini gizle	Var olan değişkenlerden istenilen değişkeni ekranda gizler.
Bir Liste Oluştur	Bu komut ile yeni bir liste oluştururuz.
☑ Sınıf	Sınıf adlı liste bilgisini barındırır. Kutu işaretlendiğinde liste içeriği ekranda görünür.
Ahmet i Sınıf e ekle	İstenilen değeri seçili olan listeye ekler.
sil 1 ini Sınıf in	Listenin istenen sırasındaki kaydı, son sıradaki kaydı ya da tüm kayıtları siler.
1 sırasındaki Sınıf 'in	İstenen listenin ilk/son veya herhangi bir sıradaki kayıt bilgisini barındırır.
Sınıf in uzunluğu	İstenen listenin toplam kayıt sayısını verir.
Sınıf listesini göster	İstenen listeyi sahne alanında gösterir.
Sınıf listesini gizle	İstenen listeyi sahne alanında gizler.
Sınıf listesinde thing var (mı?)	İstenilen değerin seçilen liste demiz de olup olmadığını kontrol eder.
koy Ahmet 'yi, 1 sırasına Sınıf 'nin	İstenen listenin ilk/son veya herhangi bir sırasına yeni bir nesne eklenir.(Eski bilgi silinmez, bu bilgi bir sonraki eleman olarak kaydırılır.)
değiştir 1 sırasındakini Sınıf listesinin Ahmet ile	İstenen listenin ilk/son veya herhangi bir sırasındaki elemanı yerine yeni nesnemiz eklenir.

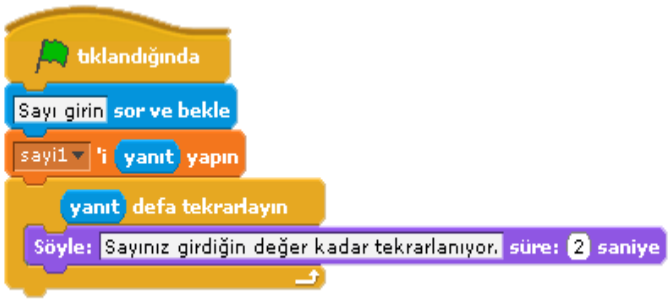
SCRATCH ÖRNEKLERİ

Toplama: 	Scratch Sonuç Ekranı: 
Çıkarma: 	Scratch Sonuç Ekranı: 
Çarpma: 	Scratch Sonuç Ekranı: 
Bölme: 	Scratch Sonuç Ekranı: 

Toplama:

İf Döngüsü:

For Döngüsü:



EK 5. Performans Testi

Performans Testi 1

ALGORITMA | GIRIŞ BIRIMI | ÇIKIŞ BIRIMI | AKIŞ DIYAGRAMI | KESINLIK

1. Yukarıdaki kelimeleri aşağıdaki boş yerlere uygun gelecek şekilde yerleştiriniz.

a) Bir çözümün daha iyi anlaşılabilmesi için kağıt üzerinde şemalar halinde gösterilmesi gerekir bu gösterime.....denir.
b) Algoritmaların başlangıç noktaları.....leridir.
c) Algoritmaların oluşturulmasında kullanılan şekillere.....denir.
d) Algoritmaların bitiş noktaları.....leridir.
e) Algoritmanın her adında net ve açık olması o algoritmanın .....özelliği ile ilgilidir.

2. Aşağıdaki akış diyagramı şekillerinin isimlerini altlarına yazınız.

		
:	:	:
		
:	:	:

3. Yukardaki akış diyagramı şekillerini kullanarak aşağıda problemin akış şemasını oluşturunuz.
Problem: Bir asansörün nasıl çalıştığını gösteren bir akış şeması çiziniz.

Asansör çağrı geldiği zaman hareket eder. Çağrıda iki durumda gelir: ya kabin içinde düğmeye basılır veya katta iseniz düğmeye basar çağırırsınız ikisi de aynı durumdur. Asansör çağrılan veya istenilen kata gider.

Performans Testi

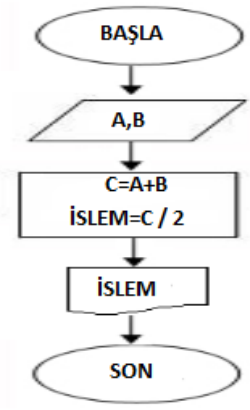
2

1. Girdiğimiz değerleri alan veya programın çalışmasıyla bazı değerlerin atandığı veri tutulduğu yapılara ..... denir.
   a. Operatörler c. Akış diyagramları
   b. Değişkenler d. Sabit

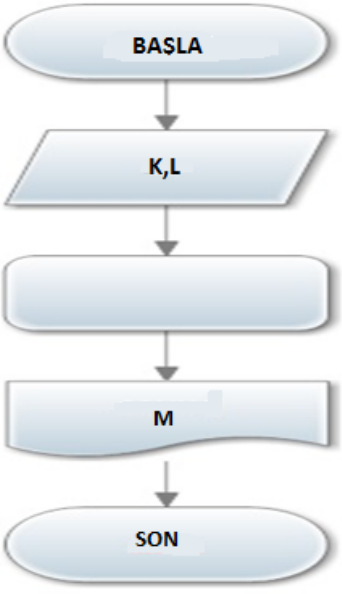
2. Bir programda tekrar edilme işleminin ne kadar yapıldığını tutan bir değişkenlere ..... denir.
   a. Döngü c. Sayaç
   b. Giriş d. İnput

3. Aşağıdaki program satırlarının hangisinde değişken atama şekli doğru verilmiştir.
   a. print "bir sayı giriniz"-a
   b. print "bir sayı giriniz"=a
   c. input "bir sayı giriniz"=a
   d. input "bir sayı giriniz";a

4. Aşağıdaki akış diyagramının karşılığı olan programı yan taraftaki kutucuğa yazınız.



5. Aşağıdaki şekle göre yandaki taraftaki soruları cevaplayınız.



Soru 1) Yan taraftaki akış diyagramında boş olarak gösterilen akış diyagramı yapısı aşağıdakilerden hangisidir.

a. İnput b. Print c. Döngü d. İşlem

Soru 2) Aşağıdaki tabloyu yandaki akış diyagramına göre boş olan kısımları tamamlayınız

	K	L	Boş Kısım	Sonuç (M)
	37	17	K+L	
	4	8	K+L/2	
	5	9		45
		4	K*L+L	16
	8		K-L/5	6
	34	29		5
	12	6	K-L/K+L	
	2	5	K/L	
	9	13	(K+L)/2	
	50	90	K*0.4+L*0.6	
	2	5	K^L	

Performans Testi

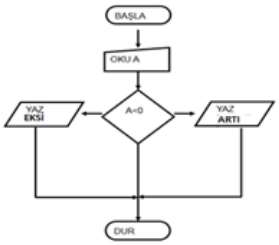
3

a b c d

1. Yukardaki akış diyagramı şekillerinin isimleri, hangi seçenekte doğru sıralama ile verilmiştir?

   a. İnput-Şart-İşlem-Print
   b. Şart-İnput-İşlem-Print
   c. Şart-Print-İnput-İşlem
   d. Şart-Print-İşlem-İnput

2. Aşağıdaki akış diyagramının karşılığı olan program hangi seçenekte doğru verilmiştir.




```
INPUT "Sayiyi Giriniz:"; A
IF A>0 THEN  "ARTI"
IF A<0 THEN  "EKSI"
END
```

```
INPUT "Sayiyi Giriniz:"; A
IF A>0 THEN PRINT "ARTI"
IF A<0 THEN PRINT "EKSI"
END
```

```
INPUT "Sayiyi Giriniz:"; A
IF A>0 THEN PRINT "ARTI"
ELSE THEN PRINT "EKSI"
END
```

```
INPUT "Sayiyi Giriniz:"; A
IF A>0 THEN PRINT "ARTI"
ELSE IF A<0 THEN "EKSI"
END
```

3. Aşağıdaki cümlelerin hangisi if (şart) ifadesi için **doğru değildir?**

   a. İç-içe birden fazla if ifadesi kullanılabilir.
   b. Goto anahtar sözcüğü ile bir döngü ve koşul bloğu içerisine dallanma işlemi gerçekleştirilebilir.
   c. Eğer koşulun sağlanmaması durumunda işlem yapılması isteniyorsa, else ifadesine gerek duyulur.
   d. İf yapısı birden çok işlemin gerçekleştirilmesinde kullanılan döngüsel bir yapıdır.

4. if H<= 100 AND T =>140 olan şart kelimesinin anlama hangi seçenekte doğru olarak verilmiştir?
   a. H değeri 100 den küçük ve T değeri 140' dan büyükse
   b. H değeri 100 den küçük veya T değeri 140' dan büyükse
   c. H değeri 100 eşit ve 100 den küçük ve T değeri 140 'a eşit ve 140' dan büyükse
   d. H değeri 100 eşit ve 100 den küçük veya T değeri 140 'a eşit ve 140' dan büyükse

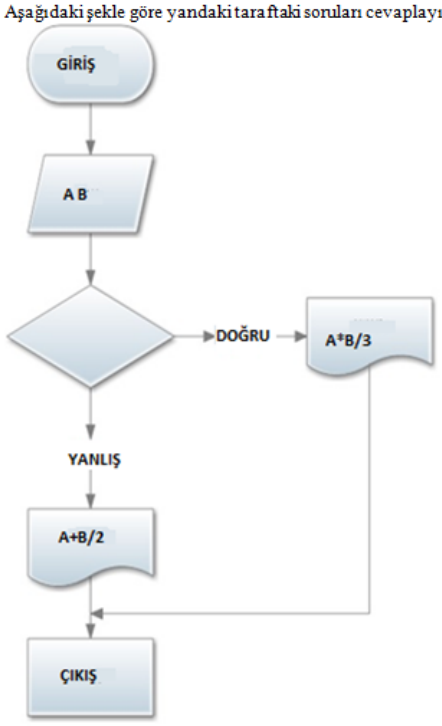
5. if(sayi1>0) || (sayi1<5) ifadesindeki koşul aşağıdakilerden hangisidir?
   a) sayi1 büyüktür sıfırdan ve sayi1 büyüktür 5'ten.
   b) sayi1 büyüktür sıfırdan ve sayi1 küçüktür 5'ten.
   c) sayi1 büyüktür sıfırdan veya sayi1 büyüktür 5'ten.
   d) sayi1 büyüktür sıfırdan veya sayi1 küçüktür 5'ten.

6. Aşağıdaki programın çıktısı hangisidir?

```
A=10
C=2*A-2
B=C+6
IF B>C THEN PRINT "KURU"
IF A>C THEN PRINT "NEMLİ"
IF B>A THEN PRINT "ISLAK"
```

   a. ISLAK c. KURU
   b. KURU ISLAK d.KURU NEMLI

7. Aşağıdaki şekle göre yandaki taraftaki soruları cevaplayınız



I-) Yan taraftaki şekilde boş olarak gösterilen akış diyagramı yapısı aşağıdakilerden hangisidir.

a. Input b. Print c. Döngü d. Şart

II-) Aşağıdaki tabloyu yandaki akış diyagramına göre boş olan kısımları tamamlayınız

A	B	KOŞUL(ŞART)	Sonuç
5	9	A<B	
4	8	A>B	
5	9	A=B	
3	7	A<=B	
8	2	A>=B	
6		A< B	16
	8	A> B	24
6	6		12
7	12	A<=B	
2	6	A>=B	
4	8		8

Performans Testi

4

1. Herhangi bir şart satırı ile etiket satırı arasında programın tekrar ettirilmesine …….. denir.

   a) Operatörler
   b) Akış diyagramları
   c) Değişkenler
   d) Döngü

2. Basic'te bazı işlemlerin tekrarlanmasını istiyorsak bu işlemi……ile yapabiliriz. Fakat bu işlemler ayni zamanda istenirse ……ile de yapılabilir.

   a) FOR - GOTO
   b) İNPUT-PRİNT
   c) FOR-NEXT
   d) İF-ELSE

```
10 CLS
20 FOR A=1 to 5
30 PRINT "IYI INSAN"
40  EXT A
```

3. Yukardaki program aşağıdaki seçeneklerden hangisinde doğru yorumlanmıştır.

   a) 4 defa "IYI INSAN" yazan bir programdır.
   b) 4 defa "INSANIYI" yazan bir programdır.
   c) 5 defa "IYI INSAN" yazan bir programdır.
   d) 5 defa "INSAN IYI" yazan bir programdır.

4. Aşağıdakilerden hangisi döngü gibi düşünülebilir.

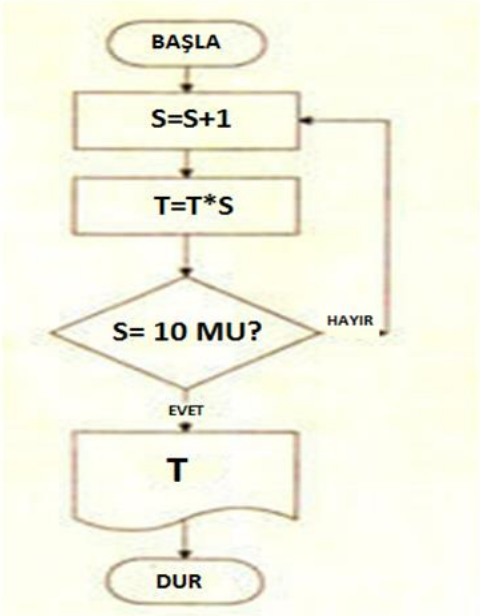
   a) Giriş
   b) Çıkış
   c) Sayaç
   d) Koşul

```
CLS
FOR G= 1 TO 3
PRINT 8+G
NEXT G
```

5. Yukardaki programın çıktısı aşağıdakilerden hangisidir.

   a) 1 b) 4 c) 9 d) 3
      2 5 10 4
      3 6 11 5

6. Aşağıdaki şekle göre yandaki taraftaki soruları cevaplayınız.



S=10 ŞARTI	T

Performans Testi

5

1. Aşağıdaki tabloda operatörlerin karşısındaki boşluklara yaptığı işlemi yazınız ?

Operatör	İşlem
/	
+	
*	
^	
( )	
–	

2. Yukardaki operatörlerin işlem önceliği, hangi seçenekte doğru sıralama ile verilmiştir?

a. /+*^( )–
b. * ^ ( ) – / +
c. ( ) ^ * / + -
d. ( ) ^ + - / *

3. Aşağıdaki tabloda matematiksel ifadelerin Program karşılığına boş kısımlara yazınız.

Matematiksel ifade	Program Karşılığı
$\frac{a+b}{7}$	
$\frac{Y}{XY}$	
$\frac{AB}{XY}$	

4. Aşağıdaki işlemleri işlem önceliğini göz önüne alarak cevaplayınız?
a) **3 + 12 – 5 ^ 2 * 2 =**
b) **(3 + (12 – 5) ^ 2) * 2 =**
c) **((19 – 3) / (3 – 1) ^ 2) * 6 =**
d) **6 * 4 / 2 * 4 =**

5. IF A > B AND A > C THEN PRINT "A SAYILARIN EN BÜYÜK OLANIDIR" ifadesinin açıklaması hangi seçenekte doğru verilmiştir.
a) Eğer A değeri B değerinden büyük veya A değeri C değerinden küçük ise ekrana A SAYILARIN EN BÜYÜK OLANIDIR yazar

b) Eğer A değeri B değerinden büyük ve A değeri C değerinden büyük ise ekrana A SAYILARIN EN BÜYÜK OLANIDIR yazar

c) Eğer A değeri B değerinden büyük ve A değeri C değerinden küçük ise ekrana A SAYILARIN EN BÜYÜK OLANIDIR yazar

d) Eğer A değeri B değerinden büyük veya A değeri C değerinden büyük ise ekrana A SAYILARIN EN BÜYÜK OLANIDIR yazar

6. IF NOT < 0 OR NOT > 100 THEN PRINT "NOT YÜZLÜK SİSTEME UYGUN DEĞİLDİR." ifadesinin açıklaması hangi seçenekte doğru verilmiştir.
a) Eğer NOT değişkeni sıfırdan küçük ve 100 den büyükse ekrana NOT YÜZLÜK SİSTEME UYGUN DEĞİLDİR yazar

b) Eğer NOT değişkeni sıfırdan büyük veya 100 den küçükse ekrana NOT YÜZLÜK SİSTEME UYGUN DEĞİLDİR yazar

c) Eğer NOT değişkeni sıfırdan büyük ve 100 den küçükse ekrana NOT YÜZLÜK SİSTEME UYGUN DEĞİLDİR yazar

d) Eğer NOT değişkeni sıfırdan küçük veya 100 den büyükse ekrana NOT YÜZLÜK SİSTEME UYGUN DEĞİLDİR yazar

7. İf H<= 100 & T =>140 olan şart kelimesinin anlama hangi seçenekte doğru olarak verilmiştir?
a. H değeri 100 den küçük ve T değeri 140' dan büyükse
b. H değeri 100 den küçük veya T değeri 140' dan büyükse
c. H değeri 100 eşit ve 100 den küçük ve T değeri 140 'a eşit ve 140' dan büyükse
d. H değeri 100 eşit ve 100 den küçük veya T değeri 140 'a eşit ve 140' dan büyükse

8. if(sayi1>0) || (sayi1<5) ifadesindeki koşul aşağıdakilerden hangisidir?
a) sayi1 büyüktür sıfırdan ve sayi1 büyüktür 5'ten.
b) sayi1 büyüktür sıfırdan ve sayi1 küçüktür 5'ten.
c) sayi1 büyüktür sıfırdan veya sayi1 büyüktür 5'ten.
d) sayi1 büyüktür sıfırdan veya sayi1 küçüktür 5'ten.

9. Aşağıdaki programın çıktısı hangisidir?

	A	B	OPERATÖR	SONUÇ
A)	**3**	**56**	**A+B**	
B)	**5**	**7**	**A*B**	
C)	**9**	**90**	**B/A**	
D)	**16**	**2**	**A-B**	
E)	**6**	**3**	**A^B**	
F)	**38**	**8**	**(A-B)*B**	

ÖZGEÇMİŞ

1976 Gümüşhane ili merkez Tekke Köyü doğumlu olan Faruk DEMİR ilk, orta ve lise tahsilini Erzurumda tamamladı. 2000 yılında Fırat Üniversitesi Teknik Eğitim Fakültesi Bilgisayar Öğretmenliği Bölümünden mezun oldu. Aynı yıl Mili Eğitim Bakanlığına Bilgisayar Öğretmeni olarak atandı. 2005 yılında Karadeniz Teknik Üniversitesine bağlı Gümüşhane Mühendislik Fakültesine muvafakatla geçiş yaptı. 2008 yılında Karadeniz Teknik Üniversitesi Eğitim Yönetimi Teftişi Planlaması ve Ekonomisi programında Yüksek Lisans'a başlayarak 2010 yılında yüksek lisansını tamamladı. 2012 yılında Atatürk Üniversitesi Eğitim Bilimleri Enstitüsü Bilgisayar ve Öğretim Teknolojileri Eğitimi Anabilim Dalında başladığı doktora eğitimini 28.12.2015 tarihinde tamamladı. Halen Gümüşhane Üniversitesi Meslek Yüksekokulu Bilgisayar Teknolojileri Bölümünde görev yapan DEMİR, evli ve iki çocuk babasıdır.