T.C.
Marmara Üniversitesi
Eğitim Bilimleri Enstitüsü
Bilgisayar ve Öğretim Teknolojileri Ana Bilim Dalı
Bilgisayar ve Öğretim Teknolojileri Öğretmenliği Bilim Dalı

PROGRAMLAMA ÖĞRETİMİNDE GÖRSELLEŞTİRME ARAÇLARININ KULLANIMININ ÖĞRENCİ BAŞARI VE MOTİVASYONUNA ETKİSİ

Yüksek Lisans Tezi

Işıl GÜLMEZ

İstanbul, 2009

T.C.
Marmara Üniversitesi
Eğitim Bilimleri Enstitüsü
Bilgisayar ve Öğretim Teknolojileri Ana Bilim Dalı
Bilgisayar ve Öğretim Teknolojileri Öğretmenliği Bilim Dalı

PROGRAMLAMA ÖĞRETİMİNDE GÖRSELLEŞTİRME ARAÇLARININ KULLANIMININ ÖĞRENCİ BAŞARI VE MOTİVASYONUNA ETKİSİ

Yüksek Lisans Tezi

Işıl GÜLMEZ

Danışman: Yrd. Doç. Dr. Nesrin ÖZDENER

İstanbul, 2009

T.C

Marmara Üniversitesi

Eğitim Bilimleri Enstitüsü

Bilgisayar ve Öğretim Teknolojileri Öğretmenliği Ana Bilim Dalı

Bilgisayar ve Öğretim Teknolojileri Bilim Dalı

Işıl GÜLMEZ tarafından hazırlanan Programlama Öğretiminde Görselleştirme Araçlarının Kullanımının Öğrenci Başarısı Ve Motivasyonuna Etkisi başlıklı çalışma, ..26..10.2009 tarihinde yapılan savunma sınavı sonucunda başarılı bulunarak jürimiz tarafından Yüksek Lisans tezi olarak kabul edilmiştir.

İmzalar

Danışman : Yard.Doc.Dr. Nesrin ÖZDENER .....

Jüri Üyesi : Yrd. Doç Dr. Levent DENİZ .....

Jüri Üyesi : Prof. Dr. Servet BAYRAM .....

T.C.

Marmara Üniversitesi

Eğitim Bilimleri Enstitüsü

Bilgisayar ve Öğretim Teknolojileri Öğretmenliği Ana Bilim Dalı

Bilgisayar ve Öğretim Teknolojileri Bilim Dalı

Işıl GÜLMEZ tarafından hazırlanan Programlama Öğretiminde Görselleştirme Araçlarının Kullanımının Öğrenci Başarısı Ve Motivasyonuna Etkisi başlıklı çalışma, …..07.2009 tarihinde yapılan savunma sınavı sonucunda başarılı bulunarak jürimiz tarafından Yüksek Lisans tezi olarak kabul edilmiştir.

	İmzalar
Danışman: Yrd.Doç. Dr. Nesrin ÖZDENER	……
Jüri Üyesi: Yrd. Doç Dr. Levent DENİZ	……
Jüri Üyesi: Prof. Dr. Servet BAYRAM	……

ÖNSÖZ

Hızla değişen dünyada, başarılı olabilmek için gelişen teknolojiye uyum sağlamak gerekmektedir. İlerleyen teknoloji, günlük hayatta karşılaşılan problemlere hızlı ve kolay bir biçimde çözüm bulunabilmesine imkan sağlamaktadır. Bu nedenle toplumun teknolojiyi takip eden ve problemleri teknoloji yardımıyla çözebilen bireylere ihtiyacı vardır. Bu bağlamda okullarda öğrencilere, karşılaşılan problemleri küçük parçalara bölme ve teknolojiyi kullanarak probleme uygun çözüm stratejisi geliştirme konusunda eğitim verilmesi önemlidir. Bu durum, öğrencilere algoritmik düşünme eğitiminin yaygınlaştırılması gerekliliğini ortaya çıkarmaktadır. Öğrencilerin problem çözme ile ilgili becerilerini küçük yaşlarda edindikleri göz önüne alınırsa bu konuda verilecek eğitimin temellerinin ilköğretim okullarında atılması gereklidir. Bu nedenlerle ilköğretim çağından itibaren öğrencilere temel programlama eğitimi verilmektedir.

Programlama eğitimi ile ilgili yapılan araştırmalar programlama öğrenmeye yeni başlayan öğrencilerin zorlandıklarını ve başarısız olduklarını göstermektedir. Bu zorlukların en aza indirgenmesi ve programlama eğitiminde başarının artırılabilmesi için uygun yöntem ve tekniklerin kullanılması gerekmektedir. Bu konuda yapılan araştırmalar, belirli bir programlama diline bağlı kalınmadan, görsel yardımcı araçlar kullanılarak uygulanan yöntemlerin başarılı olduklarını göstermektedir. Programlama eğitiminde uygulanan yöntemlerin başarılı olabilmesi için kullanılacak olan görselleştirme araçların doğru belirlenmesi ve bu görsel araçlardan dersin hedeflerine uygun biçimde faydalanılması önemlidir. Literatürde programlama eğitimine yeni başlayanlar için görselleştirme araçlarından, akış şeması modelli araçların uygun olduğunu savunan görüşler olduğu gibi algoritmayı hikayeleştiren araçların uygun olduğunu savunan görüşler de vardır. Bu bağlamda, öğrencilere programlamanın temelleri öğretilirken; görselleştirme araçlarından hangisinin daha uygun olduğunun belirlenmesi, eğitimcilere ışık tutması açısından önemlidir.

Bu çalışmada programlama eğitiminde algoritmayı hikayeleştiren araçlar ile akış şeması modelli araçların öğrenci başarı ve motivasyonuna etkisi araştırılmıştır. Çalışmanın sonuçları, programlama eğitiminde kullanılacak olan görselleştirme araçlarının doğru belirlenmesinin öğrencilerin başarıları ve motivasyonları üzerinde etkili olduğunu vurgulamaktadır.

Bu çalışmada yol almamda, büyük emeği geçen, yardımlarını esirgemeyen değerli hocam ve danışmanım Yrd.Doç. Dr. Nesrin ÖZDENER başta olmak üzere, değerli hocam Prof. Dr. Servet BAYRAM'a, çalışmam için desteklerinden ötürü okul müdürüm Hasan PATAN'a, manevi destekleri ile yanımda olan annem Gevher GÜLMEZ, babam Kadir GÜLMEZ ve kardeşim Esra GÜLMEZ'e, teşekkürlerimi sunarım.

Kasım 2009 Işıl GÜLMEZ

ÖZET

PROGRAMLAMA ÖĞRETİMİNDE GÖRSELLEŞTİRME ARAÇLARININ KULLANIMININ ÖĞRENCİ BAŞARI VE MOTİVASYONUNA ETKİSİ

Bu çalışmada, programlama öğretiminde görselleştirme araçları kullanmanın, öğrenci başarı ve motivasyonuna olan etkisi araştırılmıştır. Çalışmada ayrıca, ilköğretim seviyesindeki öğrencilerin programlama başarılarının yordanmasına katkı sağlaması amacıyla algoritma geliştirme başarıları ile ilişkisi olan dersler belirlenmeye çalışılmıştır. Deneme ve ilişkisel tarama modelinin kullanıldığı çalışma, iki deney grubu ile yürütülmüştür. Gruplardan biri akış şeması modelli yazılımdan, diğeri ise algoritmayı hikayeleştiren yazılımdan faydalanmıştır. Gruplar, değişken, koşul ve döngü kullanımı konularındaki başarıları açısından karşılaştırılmıştır. Biri kağıt üzerinde diğeri uygulamalı olmak üzere iki adet son test sınavı kullanılmıştır. Grupları motivasyon açısından karşılaştırmada Özerbaş (2003) tarafından geliştirilen motivasyon testi kullanılmış, öğrencilerin algoritma geliştirme başarısı ile ilişkisi olan derslerin belirlenmesinde ise korelasyon testi sonuçlarından yararlanılmıştır.

Çalışma sonucunda döngü ve koşul kullanımı konusunda algoritmayı hikayeleştiren araç lehine anlamlı fark gözlenmiştir. Çalışma grupları arasında motivasyon açısından anlamlı fark olmadığı belirlenmiş, öğrencilerin algoritma geliştirme başarıları ile Türkçe, matematik, İngilizce ve bilişim teknolojileri dersleri arasında anlamlı ilişki tespit edilmiştir.

Anahtar Sözcükler: Akış şeması modelli araç, algoritmayı hikayeleştiren araç, algoritma geliştirme başarısı, motivasyon.

ABSTRACT

EFFECTS OF USING VISUALIZATION TOOLS IN PROGRAMMING INSTRUCTION ON STUDENT SUCCESS AND MOTIVATION

In this study, the effects of using visualization tools in programming instruction on student success and motivation were investigated. Lessons which significantly correlate with elementary school students' algorithm development success tried to be determined, as it is useful to predict their programming achievement. The study which used experimental and correlative survey model was conducted by two experimental groups. One group used flow-model tool, the other used narrative tool. Groups' success on using variables, conditions and loops were compared using paper-based and computer-based tests. Ozerbas (2003)'s motivation survey was used for comparing groups' motivation, correlation test results was used for determining the lessons which significantly correlate with algorithm development success.

Results revealed that there is significant difference on using conditions and loops in favor of narrative tools. There isn't any difference between groups' motivation. The lessons which significantly correlate with students' algorithm development success are Turkish, mathematics, English and computer technologies.

Key Words: Flow-model tools, narrative tools, algorithm development success, motivation.

İÇİNDEKİLER DİZİNİ

Sayfa No

ÖNSÖZ i

ÖZET iii

ABSTRACT iv

İÇİNDEKİLER DİZİNİ v

TABLOLAR DİZİNİ ix

ŞEKİLLER DİZİNİ xi

BÖLÜM I: GİRİŞ 1

1.1 PROBLEM 1
1.2 AMAÇ 5
1.3 ÖNEM 6
1.4 VARSAYIMLAR 6
1.5 SINIRLILIKLAR 7
1.6 TANIMLAR 7

BÖLÜM II: İLGİLİ ALANYAZIN 8

2.1 TÜRKİYE'DE VE DÜNYADA PROGRAMLAMA EĞİTİMİ 8
2.2 PROGRAMLAMA ÖĞRETİMİNDE KARŞILAŞILAN GÜÇLÜKLER 10
2.3 PROGRAMLAMA ÖĞRETİMİNDE BAŞARI VE MOTİVASYONU ARTIRMAK İÇİN YAPILAN ARAŞTIRMALAR 11
2.4 PROGRAMLAMA ÖĞRETİMİNDE KULLANILAN GÖRSELLEŞTİRME ARAÇLARI 14
2.4.1 Programlama Öğretiminde Kullanılabilecek Görselleştirme Araçlarının Sınıflandırılması 17
2.4.2 Programlama Öğretiminde Görselleştirme Araçlarının Kullanımı İle İlgili Yapılan Çalışmalar 22

2.5 PROGRAMLAMA ÖĞRETİMİNDE KULLANILABİLECEK YÖNTEMLER....28
2.6 PROGRAMLAMA ÖĞRETİMİNDE DEĞERLENDİRME....30
2.7 PROGRAMLAMA ÖĞRETİMİNDE ÖĞRENCİ BAŞARISINI ETKİLEYEN FAKTÖRLER....31
BÖLÜM III: YÖNTEM....33
3.1 ARAŞTIRMA MODELİ....33
3.2 ÇALIŞMA GRUBU....36
3.3 VERİLERİN TOPLANMASI....36
3.3.1 Veri Toplama Araçları....36
3.3.2 Kullanılan Materyal....41
3.4 VERİLERİN ÇÖZÜMLENMESİ....44
BÖLÜM IV: BULGULAR....46
4.1 UYGULAMA ÖNCESİ BULGULAR VE YORUMLANMASI....46
4.1.1 Çalışma Grubunun Ön Bilgi Testi Sonuçları ile İlgili Bulgular46
4.1.2 Çalışma Grubunun Motivasyon Ön Test Sonuçları İle İlgili Bulgular....47
4.2 UYGULAMA SONRASI BULGULAR VE YORUMLANMASI....47
4.2.1 Farklı Görselleştirme Araçları Kullanan Öğrencilerin Değişkenlerin Kullanımı İle İlgili Sorulardaki Başarılarının Karşılaştırılması....47
4.2.2 Farklı Görselleştirme Araçları Kullanan Öğrencilerin Problemin Çözümü İçin Koşul Kullanımı İle İlgili Başarılarının Karşılaştırılması....48
4.2.3 Farklı Görselleştirme Araçları Kullanan Öğrencilerin Problemin Çözümü İçin Döngü Kullanımı İle İlgili Başarılarının Karşılaştırılması....49
4.2.4 Farklı Görselleştirme Araçları Kullanan Öğrencilerin Motivasyon Ön Test – Son Test Sonuçlarının Karşılaştırılması....51

4.2.5 Öğrencilerin Bilişim Teknolojileri Matematik, Türkçe ve İngilizce Derslerindeki Akademik Başarıları İle Algoritma Geliştirme Başarıları Arasındaki İlişki 53
4.2.6 Öğrencilerin Algoritma Geliştirme Başarılarını Yordayan Dersler.....57

BÖLÜM V: SONUÇ, TARTIŞMA VE ÖNERİLER.....59

5.1 SONUÇ VE TARTIŞMA.....59
5.2 ÖNERİLER.....66

KAYNAKÇA 67

EKLER.....80

EK 1 ÖN BİLGİ TESTİ 80
EK 2 UYGULAMADA KULLANILAN GÜNLÜK PLANLAR.....82
1. Ders 1 (Algoritma Ve Programlamanın Temel Kavramları) 82
1.1 Scratch Algoritma Ve Programlama Çalışma Sayfası 1 84
1.1 Scratch Algoritma Ve Programlama Çalışma Sayfası 2 86
1.2 Fcpro Algoritma Ve Programlama Çalışma Sayfası 1 88
1.2 Fcpro Algoritma Ve Programlama Çalışma Sayfası 2 90
2. Ders 2 (Değişken Kavramı) 91
2.1 Scratch Değişken Dersi Çalışma Sayfası 1 92
2.1 Scratch Değişken Dersi Çalışma Sayfası 2 94
2.2 Fcpro Değişken Dersi Çalışma Sayfası 1.....96
2.2 FCPRO DEĞİŞKEN DERSİ ÇALIŞMA SAYFASI 2.....97
3. Ders 3 (Koşul Cümleleri).....99
3.1 Scratch Koşul Cümleleri Dersi Çalışma Sayfası 1 100
3.1 Scratch Koşul Cümleleri Dersi Çalışma Sayfası 2 102
3.2 Fcpro Koşul Cümleleri Dersi Çalışma Sayfası 1.....104
3.2 Fcpro Koşul Cümleleri Dersi Çalışma Sayfası 2.....106
4. Ders 4 (Döngü).....107
4.1 Scratch Döngü Dersi Çalışma Sayfası 1 108
4.1 Scratch Döngü Dersi Çalışma Sayfası 2 109
4.2 Fcpro Döngü Dersi Çalışma Sayfası 1.....111

4.2 Fcpro Döngü Dersi Çalışma Sayfası 2 113
5. SCRATCH ETKİNLİKLERİ / FCPRO ETKİNLİKLERİ 114
EK 3 SON TEST 115
EK 4 UYGULAMA SINAVI 119
EK 5 MOTİVASYON ÖLÇEĞİ 120

TABLOLAR DİZİNİ

Sayfa No

Tablo 1 Araştırma Kapsamında Yapılan Uygulamalar ve Uygulamalarda Geçen Süre 34

Tablo 2 Araştırma Deseni 36

Tablo 3 Çalışma Grubu Öğrencilerinin Sayıları Ve Yüzde Dağılımları....36

Tablo 4 Ön Bilgi Testi'ne İlişkin Madde Belirtke Tablosu 37

Tablo 5 Son Test 1 Madde Belirtke Tablosu....39

Tablo 6 Son Test 2 (Uygulama Sınavı) Madde Belirtke Tablosu 40

Tablo 7 Ön Bilgi Testi Sonuçlarına Yönelik Bağımsız Gruplarda t-Testi Sonuçları 46

Tablo 8 Motivasyon Ön Test Sonuçlarına Yönelik Bağımsız Gruplarda t-Testi Sonuçları....47

Tablo 9 Değişkenler İle İlgili Sorulara İlişkin Bağımsız Gruplarda t-Testi Sonuçları 48

Tablo 10 Koşul İle İlgili Sorulara İlişkin Bağımsız Gruplarda t-Testi Sonuçları 49

Tablo 11 Döngü İle İlgili Sorulara İlişkin Bağımsız Gruplarda t-Testi Sonuçları 49

Tablo 12 Son Test 1 (Yazılı Sınav) Sonuçları İle Son Test 2 (Uygulama Sınavı) Sonuçları Arasında Yapılan Pearson Korelasyon Testi Sonuçları 50

Tablo 13 Uygulama Sınavı Sonuçlarına Yönelik Bağımsız Gruplarda t-Testi51

Tablo 14 Grupların Motivasyon Ön Test ve Son Test Puanlarının Karşılaştırılmasına Yönelik Bağımlı Gruplarda t-Testi Sonuçları ...52

Tablo 15 Her İki Grubun Motivasyon Son Test Sonuçlarının Karşılaştırılması İle İlgili Bağımsız Gruplarda t-Testi Sonuçları....53

Tablo 16 Öğrencilerin Not Ortalamalarına İlişkin Betimsel İstatistik Sonuçları 54

Tablo 17 Programlama Son Test Başarısı Ve Bağımsız Değişkenlere Yönelik Spearman Rho Korelasyon Testi Sonuçları 55

Tablo 18 Programlama Son Test Başarısı ve Bağımsız Değişkenlere Yönelik Pearson Korelasyon Testi Sonuçları.....55

Tablo 19 Programlama Başarısını Yordayan Değişkenler.....57

ŞEKİLLER DİZİNİ

Sayfa No

Şekil 1 Ville Yazılımının Ekran Görüntüsü 15
Şekil 2 Jeliot 3 Yazılımı Ekran Görüntüsü 16
Şekil 3 Jhave Yazılımı Ekran Görüntüsü 17
Şekil 4 Flint Yazılımı İle Oluşturulan Akış Şeması Örneği.....21
Şekil 5 Scratch Yazılımının Ekran Görüntüsü 42
Şekil 6 Scratch İle Geliştirilen Bir Algoritma Örneği 42
Şekil 7 Fcpro Aracının Ekran Görüntüsü.....43

BÖLÜM I
GİRİŞ

1.1 PROBLEM

Dünyada programlama dersleri, mesleki yeterlilik için gerekli olmanın yanında bilgisayar okuryazarlığının önemli bir parçası olmaları nedenleriyle yaygınlaşmaktadırlar. Zira bu tür derslerin öğrencilerin analitik düşünme ve problem çözme becerilerinin gelişmesi açısından faydalı olduğu bilinmektedir (Sleeman ve diğerleri, 1984). Bu nedenlerden ötürü dünyada birçok ülkede programlamaya giriş dersleri ilk ve ortaöğretim müfredatlarına eklenmiştir (Tucker ve diğerleri, 2003). Tayvan'da son yıllarda okullarda Lego Mindstorms'a yer verilmeye başlanmış (Lin ve diğerleri, 2005), Kore'de ise 2010 yılında ortaokulların, 2011 yılında ise liselerin bilişim ders müfredatlarında değişiklikler yapılması kararı alınmıştır. Yapılan çalışmalara göre, ilkokul düzeyinde öğrencilerin bilgisayar işletmenliğinden çok algoritmaları anlamaya odaklanılması gerekmektedir (Choi ve diğerleri, 2008). Hindistan eğitim sisteminde programlama öğretimi, 3. sınıf düzeyinde LOGO programlama dili kullanılarak temel kavramların öğretimi ile başlamakta, daha sonra öğrencilerin 8. sınıfın sonuna kadar adım adım akış şeması çizmeleri ve BASIC dili kullanarak temel düzeyde program yazmayı öğrenmeleri amaçlanmaktadır (Curriculum and Teaching Material for K-12 Indian Schools, 2007). İsrail ve Kanada'da ise programlama eğitimi ile ilgili dersler liselerde verilmektedir (Tucker ve diğerleri, 2003).

K-12 Bilgisayar Bilimleri dersleri için geliştirilen ACM (Association for Computing Machinery) müfredatında algoritmik düşünmenin önemi vurgulanmıştır (Lin ve diğerleri, 2005). Ülkemizde ise ilköğretim bilişim teknolojileri dersinin amaçları arasında öğrencilere algoritma geliştirme becerileri kazandırabilmek yer almıştır. Bu yolla öğrencilere, doğru yaklaşımlar sergileyerek problemlere uygun çözüm yolu geliştirme becerisi kazandırabilmek amaçlanmaktadır (İnce, Şenyüzlü ve Uğur, B., 2007).

Programlama eğitiminin ilköğretim düzeyinde verilmeye başlanmasıyla çocuklara programlama eğitiminin nasıl verileceği sorunu ortaya çıkmıştır. Arabacıoğlu (2007)'na göre Türkiye'de programlama eğitimi teorik yöntemlerle verilmekte; ancak yapılan araştırmalar teorik yöntemlerin etkili olmadığını göstermektedir (Garner 2003; Arabacıoğlu, Bülbül ve Filiz, 2007; Gültekin 2007). Bu nedenle ilköğretim seviyesindeki öğrencilerde programlama öğretiminin hangi yöntemlerle ve nasıl yürütülmesi gerektiğinin araştırılması yararlı olacaktır.

Öte yandan ilköğretim öğrencileri için hazırlanan Bilişim Teknolojileri kitabı incelendiğinde, programlama öğretimine başlangıç aşamasında Visual Basic programı kullanarak geliştirilen örneklerin yer aldığı görülmekte ve kitapta herhangi bir programlama dilinden bağımsız biçimde bilgisayar uygulamasına dayalı örneklerin azlığı dikkat çekmektedir. Oysa yapılan araştırmalara göre; programlama eğitiminde öğrencilere belirli bir programlama dili kodlarını öğretmek yerine programlama mantığını öğretebilecek örnekler vermek önemlidir (Ramadhan 2000; Cooper, Dann ve Pausch 2003; Kelleher, Pausch ve Kiesler 2007; Malan ve Leitner 2007; Özdener 2008). Araştırmalar, yazılımda kullanılan görselliğin ve etkileşimin öğrenci başarısı üzerindeki etkisini vurgulamaktadır (Arabacıoğlu, Bülbül ve Filiz, 2007; Gültekin, 2006). Kelleher, Pausch ve Kiesler'in (2007), programlama eğitiminde görselliğin önemini vurgulamak amacı ile gerçekleştirdikleri araştırma sonucunda, görsel özelliklerin öğrenci başarı ve motivasyonunda önemli etkisi olduğunu belirlemişlerdir. Konu ile ilgili yapılan diğer araştırmalar, yazılımlarda grafik ve animasyon kullanımının öğrencilerin derslere olan ilgisini artırmada daha etkili olduklarını göstermektedir (Bishop Clark, Courte ve Howard, 2007; Brusilovsky ve Spring, 2004; Lin ve Zhang, 2003). Tüm bu nedenlerden ötürü, görsellik ve canlandırma içeren yardımcı araçlarla yapılacak uygulamaların uygun yöntemlerle derslere dahil edilmesi durumunda, ilköğretim öğrencilerinin programlamaya ilgi duymalarını ve sevmelerini sağlayabileceği söylenebilir.

Programlama öğretiminde öğrencilerin algoritma tasarlama yeteneklerini geliştirmek amacıyla kullanılabilecek çok sayıda yardımcı araç geliştirmiştir. Araştırmacılar programlama öğretiminde kullanılabilecek görselleştirme araçlarını kategorilere ayırarak özelliklerini açıklamışlardır (Bergin ve Martiez 1996; Cooper ve diğerleri 2006). Programlama dersine başlangıç aşamasında bu araçlardan hangilerinin kullanılması gerektiği ile ilgili çeşitli görüşler bulunmaktadır. Araştırmacılardan bazıları programlama eğitimine yeni başlayan kullanıcılar için akış şeması modelli araçların uygun olduğunu iddia ederken (Baldwin ve Kuljis, 2000; Crews ve Ziegler, 1998) bazıları ise algoritmayı hikayeleştiren araçların daha uygun olduğunu iddia etmektedirler (Grissom ve diğerleri, 2003; Malan ve Leitner, 2007). Bravo, Marcelino, Games, Esteves ve Mendes (2005), algoritmayı bir karakter yardımıyla oluşturmaya yardımcı olan araçların, öğrencilerin programları anlamaları ve analiz etmeleri konusunda yardımcı olabileceğini belirtmişlerdir. Ramadhan'ın (2000), üniversite öğrencileriyle yaptığı çalışmada programlama öğretiminde koşul ve döngü ile ilgili konularda görsel araçlardan destek almanın öğrenci başarısı üzerinde olumlu etkisi olduğu görülmüştür. Lahtinen, Ahoniemi ve Salo'nun (2007), araştırma sonuçlarına göre problemlerin karmaşıklığı arttıkça görselleştirme araçlarına daha çok ihtiyaç duyulmaktadır. Bu konuda yapılan araştırmalar üniversite düzeyinde gerçekleştirilmiş olup, ilköğretim öğrencilerine uygun görselleştirme aracının belirlenmesinde yetersiz kalmaktadır. Bu durumda ilköğretim öğrencilerine programlamanın temelleri öğretilirken, hangi görselleştirme araçlarını kullanmanın daha uygun olacağının araştırılmasında yarar vardır.

Araştırmaların yetersiz kaldığı bir başka nokta ise programlama eğitimi ile ilgili yapılan çalışmaların öğrenci seviyesine göre ayrılmamış olmasıdır. Nitekim dünyada birçok ülkede programlama ile ilgili konuların ilköğretim müfredatına yeni eklenmiş (Choi ve diğerleri, 2008; Lin ve diğerleri, 2005; Tucker ve diğerleri, 2003) olması nedeniyle öğrencilerinin yaş ve seviyelerine uygun olarak programlama öğretiminde kullanılabilecek örnek sayısı sınırlıdır. Bu bağlamda, ilköğretim seviyesine uygun araç gereçler kullanılarak uygun

yöntemlerin geliştirilmesinde yarar vardır. Ayrıca, algoritmayı görselleştirme biçimi farklı araçların öğrencilerin algoritma geliştirme başarı ve motivasyonuna olan etkisinin belirlenmesi de faydalı olacaktır.

Diğer yandan öğrencilerin programlama konusunda başarılı olabilmesini etkileyen faktörleri bilmek, onların programlama başarılarını öngörebilmek ve onları programlama ile ilgili bölümlere yönlendirebilmek konularında eğitimcilere yardımcı olacaktır. Programlama öğretiminde öğrenci başarısına etki eden faktörler ile ilgili yapılan araştırma sonuçları, programlamada başarılı olunabilmesi için matematiksel başarının yanı sıra okuduğunu anlayabilme, mantıklı biçimde yorumlayabilme ve analiz etme becerilerinin de önemli olduğunu göstermektedir. Yapılan araştırmalara göre öğrencilerin önceki akademik başarıları ile üniversitede almış oldukları programlama ders başarıları arasında pozitif yönde anlamlı ilişki vardır (Peterson ve Howe 1979, Leeper ve Silver 1982). Birçok çalışmada matematik sonuçları ve programlama başarıları arasında pozitif yönde anlamlı bir ilişki bulunmuştur (Erdoğan, 2005; Byrne ve Lyons, 2001; Soloway, Lochhead ve Clement, 1982; Webb, 1985; Fletcher, 1984; Pea ve Kurland, 1983). Nowaczyk (1983), araştırmasında öğrencilerin İngilizce derslerindeki akademik başarıları ile bilgisayar programlama başarıları arasında pozitif yönde anlamlı bir ilişki olduğu sonucunu ortaya çıkarmıştır (Chung 1988). Ancak Jones ve Burnett'in (2008), araştırma sonuçlarında İngilizce veya yabancı dil ders başarıları ile programlama başarısı arasında ilişki bulunamamıştır (Jones ve Burnettt, 2008). Bu araştırmalar özellikle programlamaya yeni başlayan öğrencilerin başarılarını yordayabilmek, onları uygun bölümlere yönlendirebilmek için akademik başarının programlama başarısına olan etkisini belirlemekte yetersiz kalmaktadır. Bu nedenle programlama dersini daha önce almayan ilköğretim öğrencilerinin bilişim teknolojileri, Türkçe, matematik ve İngilizce başarılarının programlama başarıları konusunda belirleyici olup olmadığının araştırılmasına ihtiyaç vardır.

Özetle, programlamaya yeni başlayan öğrenciler, verilen problemleri tanımlama ve program yardımıyla çözüm bulma konusunda zorlanmaktadırlar. Bu nedenle yeni başlayan öğrenciler için programlama dili öğrenmeyi kolaylaştırmak, öğrencilerin programlama konusundaki başarı ve motivasyonlarının artmasını sağlamak için kullanılabilecek araç-gereç ve yöntemlerin hangilerinin etkili olduğunun tartışılması gerekmektedir. Bunun yanında programlama eğitiminde başarılı olan öğrencilere uygun yönlendirme yapılabilmesi için öğrencilerin programlama başarısını etkileyen derslerin hangilerinin olduğu da araştırılması gereken problemlerdendir.

1.2 AMAÇ

Çalışmanın temel amacı, ilköğretim bilişim teknolojileri dersi kapsamında verilen programlama eğitiminde, yararlanılan algoritmayı hikayeleştiren araçlar ile akış şeması modelli araçları öğrenci başarısı ve motivasyonu açısından karşılaştırmaktır. Araştırmada ayrıca akademik başarının algoritma geliştirme başarısına etkisi de belirlenmiştir.

Araştırma kapsamında aşağıdaki hipotezler test edilmiştir.

1. Akış şeması modelli araçtan faydalanan öğrenciler ile algoritmayı hikayeleştiren araçtan faydalanan öğrenciler, değişkenlerin kullanımı konusundaki başarıları açısından karşılaştırıldığında, akış şeması modelli araç kullanan öğrenciler lehine anlamlı fark olacaktır.
2. Akış şeması modelli araçtan faydalanan öğrenciler ile algoritmayı hikayeleştiren araçtan faydalanan öğrenciler, koşul kullanımı konusundaki başarıları açısından karşılaştırıldığında, algoritmayı hikayeleştiren araçtan faydalanan öğrenciler lehine anlamlı fark olacaktır.
3. Akış şeması modelli araçtan faydalanan öğrenciler ile algoritmayı hikayeleştiren araçtan faydalanan öğrenciler, döngü kullanımı konusundaki başarıları açısından karşılaştırıldığında, algoritmayı hikayeleştiren araçtan faydalanan öğrenciler lehine anlamlı fark olacaktır.
4. Algoritmayı hikayeleştiren araçtan faydalanan öğrenciler ile akış şeması modelli araçtan faydalanan öğrencilerin uygulama sonundaki

motivasyonları arasında algoritmayı hikayeleştiren araçtan faydalanan öğrenciler lehine anlamlı bir fark vardır.

5. Öğrencilerin bilişim teknolojileri, matematik, Türkçe ve İngilizce derslerindeki akademik başarıları ile algoritma geliştirme başarıları arasında pozitif yönde anlamlı bir ilişki vardır.

6. Öğrencilerin almış oldukları dersler arasında bilişim teknolojileri, matematik, Türkçe ve İngilizce derslerindeki akademik başarıları, algoritma geliştirme başarılarını anlamlı bir şekilde yordamaktadır.

1.3 ÖNEM

Programlamanın temellerinin atıldığı ilköğretim okullarında öğrenciler, programlama konusuna yabancı oldukları için bu aşamada onların başarı ve motivasyonlarını artıracak araç, gereç ve yöntemlerin kullanılması gerekmektedir. Programlama öğretiminde yardımcı olarak geliştirilen görselleştirme araçları, öğrencilerin konuları anlamasını kolaylaştırmakta, başarı ve motivasyonlarında olumlu katkılar sağlamaktadır. Çalışmanın, eğitimcileri görselleştirme araçları konusunda bilgilendireceği düşünülmektedir. Çalışma sonuçlarının eğitimcilere dersin hedefine uygun görselleştirme aracının belirlenmesi, öğrenci yaş ve seviyesine uygun örneklerin geliştirilmesi ve kullanımı konularında yol göstereceğine inanılmaktadır.

Çalışmada programlama başarısı ile ilişkili olan derslerin araştırılması, öğrencilerin programlama konusunda başarılı olabilmesini etkileyen faktörleri bilmek açısından önem teşkil etmektedir. Araştırma sonuçlarının eğitimcilere öğrencilerin programlama başarılarını öngörme ve onları programlama ile ilgili bölümlere yönlendirme konularında rehberlik edeceği umulmaktadır.

1.4 VARSAYIMLAR

Araştırmada kabul edilen varsayımlar aşağıda sunulmuştur:

1. Araştırma kapsamında kullanılan bilgisayar laboratuarı koşullarından deney ve kontrol grubu öğrencilerinin eşit şekilde etkilendiği varsayılmaktadır.

2. Araştırmada her iki programı kullanan öğrenciler, öğretim konularıyla ilgili araştırmacıdan eşit ölçüde yararlanmışlardır.

3. Öğrenciler, motivasyon ölçeğini yanıtlarken beceri, duygu ve düşüncelerini doğru biçimde yansıtmışlardır.

1.5 SINIRLILIKLAR

1. Araştırma, 2008–2009 eğitim ve öğretim yılı ile sınırlıdır.
2. Araştırmada Sevindikli İlköğretim Okulu 6. ve 7. ve 8. sınıflardan oluşan çalışma grubuyla sınırlıdır.
3. Araştırma, görselleştirme araçlarından Scratch ve Fcpro araçlarının kullanımıyla sınırlıdır.
4. Araştırmada verilen örnekler, algoritma geliştirme ile ilgili temel kavramlar, değişkenleri kullanımı, koşul, döngü konularında verilen başlangıç düzeyindeki temel örneklerle sınırlıdır.
5. Araştırma, algoritma geliştirme başarısını ölçmede kullanılan son test ve uygulama sınavı ile sınırlıdır.

1.6 TANIMLAR

Algoritmayı hikayeleştiren araç: Programlamayı bir animasyon karakter yardımıyla hikayeleştirerek öğretmeyi amaçlayan görselleştirme aracıdır (Cooper ve diğerleri, 2006).

Akış Şeması Modelli Araç: Program parçalarını işlem sırasını gösterecek biçimde birbirine bağlayarak programları yapılandırmayı sağlayan görselleştirme aracıdır (Cooper ve diğerleri, 2006)

BÖLÜM II
İLGİLİ ALANYAZIN

2.1 TÜRKİYE'DE VE DÜNYADA PROGRAMLAMA EĞİTİMİ

Programlama eğitimi, öğrencilerin problem çözme ve analitik düşünme becerilerinin gelişmesini sağlamaktadır. (Sleeman ve diğerleri, 1984). Bu nedenle programlama eğitiminin önemi artmıştır. Dünyada birçok ülkede programlamaya giriş dersleri ilk ve ortaöğretim müfredatına eklenmiştir.

Ülkemizde, programlama ile ilgili temel kavramlar ilköğretim okullarında verilmektedir. İlköğretim bilişim teknolojileri kitabının 6–8. basamaklarında ele alınan kazanımlar arasında "Basit bir program için algoritma ve akış şeması oluşturma" ve "nesne tabanlı programlama dilinde nesneleri kullanarak basit programlar oluşturma" ifadeleri yer almaktadır (İnce, Şenyüzlü, Uğur, 2007). Buna göre bilişim teknolojileri dersinin amaçları arasında öğrencilere algoritma geliştirme becerilerini kazandırabilmek yer almıştır. Bu yolla öğrencilere problemlere doğru yaklaşımlar sergileyerek uygun çözüm yolu geliştirme becerilerinin kazandırılması amaçlanmaktadır.

Programlamanın temellerinin küçük yaşlarda öğretilmesinin gerekliliğine karşın, bazı ülkelerde programlama ile ilgili konuların ilköğretim müfredatına henüz eklenmemiş olması, önemli bir eksiklik olarak karşımıza çıkmaktadır. Örneğin İsrail'de programlama öğretimi, 10. 11. ve 12. sınıflarda bilgisayar bilimleri dersi kapsamında verilmeye başlanmaktadır. Bu ders müfredatı kapsamında algoritmalar ile ilgili temel düzeyde bilgi verilmektedir. Kanada'da ise programlama ile ilgili bilgiler, ortaöğretim kurumlarında bilgisayar mühendisliği ve bilgisayar bilimleri ile ilgili açılan dersler kapsamında verilmektedir (Stephenson, 2001).

Öte yandan birçok ülkede, öğrencilere programlama ile ilgili temel bilgiler ilköğretim okullarında verilmektedir. Tayvan'da son yıllarda ilköğretim okullarında Lego Mindstorms'a yer verilmeye başlanmış (Lin ve diğerleri, 2005), Kore'de ise 2010 yılında ortaokulların, 2011 yılında ise liselerin bilişim ders müfredatlarında değişiklikler yapılması kararı alınmıştır. Yapılan çalışmalar, ilkokul düzeyinde öğrencilerin bilgisayar işletmenliğinden çok algoritmaları anlamaya odaklanılması gerektiğini vurgulamaktadır (Choi ve diğerleri, 2008).

Hindistan, bilindiği gibi programlama ile ilgili büyük bir pazar payına sahiptir. Ülke, birçok başarılı bilgisayar mühendisi yetiştirmektedir. Bu başarının nedenlerinden biri olarak ilköğretim çağından itibaren öğrencilere programlama ile ilgili temellerin verilmesi düşünülebilir. Hindistan eğitim sisteminde k12 düzeyinde yer alan bilgisayar bilimleri ders müfredatında programlama konuları aşağıdaki biçimde yer almaktadır:

3. sınıf: Öğrencilerin temel programlama komutları ile ilgili bilgilendirilmesi amaçlanmaktadır.
4. sınıf: Bu aşamada öğrencilerin LOGO programlama dilinde sık kullanılan komutlar kullanılarak işlemler yapmaları amaçlanmaktadır.
5. sınıf: Bu aşamada öğrencilerin LOGO programlama dili ile programlamaya başlangıç yapmaları amaçlanmaktadır.
6. sınıf: Bu aşamada öğrencilerin BASIC programlama dili kullanarak programlama yapmaları amaçlanmaktadır.
7. sınıf: Bu aşamada öğrencilere yapısal programlama, sabitler, değişken ve döngü kavramları ile ilgili bilgiler verilmektedir. Öğrencilerin BASIC programlama dili kullanılarak uygulamalar yaptırılmaktadır.
8. sınıf: Bu aşamada ise yapısal programlama ile ilgili kavramlar genişletilmektedir. BASIC dili kullanılarak daha önce öğrenilen sabit, değişken, döngü gibi konularda alıştırmalar yaptırılmaktadır. 9. ve 10. sınıf düzeyindeki öğrencilere ise fonksiyon, sınıf ve dizi kavramları ile ilgili bilgiler verilmektedir (SSRVM Model Curriculum and Teaching Material for K–12

Indian Schools, 2007). Tucker ve diğerleri (2003), araştırmalarında Avrupa, Rusya, Asya, Güney Afrika, Yeni Zelanda ve Avustralya'da bilgisayar bilimleri dersinin k–12 müfredatına eklendiğini belirtmektedirler. İsrail ve Kanada'da ise programlama eğitimi ile ilgili dersler liselerde verilmektedir. (Tucker ve diğerleri, 2003).

K-12 düzeyinde Bilgisayar Bilimi dersleri için geliştirilen ACM (Association for Computing Machinery) müfredatına göre özellikle 8. sınıf öğrencileri bir görevi tamamlamak için kullanılabilecek uygun adımlarla algoritmik problem çözmeyi öğrenmelidirler. Bu adımlarda koşul ifadeleri ve tekrarlar kullanılabilir. (ACM, 2003, p.12. Aktaran: Lin ve diğerleri, 2005).

2.2 PROGRAMLAMA ÖĞRETİMİNDE KARŞILAŞILAN GÜÇLÜKLER

Lise ve üniversitelerde yer alan bilgisayar programlama dersleri, öğrencilerin belirli bir programlama dilinin komutlarını öğrenmelerine odaklanmaktadır. Bu durum, öğrenciler için programlama öğrenmeyi zorlaştırmaktadır. Halbuki bu dersler problem çözme metotlarını da kapsamalıdır. Problem çözme, problemin nasıl analiz edileceğini ve çözüm planının nasıl oluşturulacağını bilmeye bağlıdır. Bilgisayar programlamaya giriş dersleri, problemleri analiz etmeyi ve algoritmayı kullanarak çözmeyi öğretmelidir (Gilmore, 1990: Aktaran. Özdener, 2008).

Arabacıoğlu, Bülbül ve Filiz (2007) araştırmalarında, ülkemizde programlama öğretimini zorlaştıran etkenlerden biri olarak programlama öğretiminde yabancı dil kullanılmasını göstermektedirler. Arabacıoğlu, Bülbül ve Filiz (2007)'e göre, öğrencilerin, yabancı dil seviyelerinin yeterli olmayışı, programlama öğretiminin önündeki en büyük engellerden biridir.

Programlama öğretimi ile ilgili yapılan araştırmalar, yeni başlayan öğrenciler için bilgisayar programlama ile ilgili kavram ve bilgileri anlamak zor olduğunu vurgulamaktadır. Araştırmalara göre programlama öğrenmeye yeni başlayan öğrenciler, genellikle programlama mantığını kavramakta güçlük çekmektedir. (Dunican, 2002; Jenkins, 2002; Proulx, 2000). Programlama öğretiminde geleneksel öğretim metotları, öğrencilerin programlama öğrenme konusundaki sıkıntılarını gidermede başarılı olamamaktadır. Programlama becerileri en iyi deneyimle kazandırılabilmektedir (Traynor ve Gibson, 2004).

Hagan, Sheard ve Macdonald (1997), Monash Üniversitesi, 1. sınıf öğrencilerinin programlamaya giriş derslerinde başarısız olduklarını dile getirmişlerdir. Başarısızlığın nedenini programlama ile ilgili konuların sıralanışına bağlayan araştırmacılar, konuların kolaydan zora doğru sıralanmasının başarıyı artırabileceğini belirtmişlerdir.

Öğrencilerin yeni bir programlama dili öğrenirken en çok zorlandıkları konulardan biri değişken atamadır. Öğrenciler için özellikle döngü içerisinde değişkenleri kullanmak oldukça zordur (Samurçay 1989. Aktaran: Pane ve Mayers, 1996).

Naser (2008)'e göre öğrenciler, algoritmalar arasında ilişki kurma ve algoritmanın parametrelerinde değişiklikler olduğunda sonucun nasıl değişeceğini tahmin etme gibi konularda güçlük çekmektedirler (Naser, 2008).

2.3 PROGRAMLAMA ÖĞRETİMİNDE BAŞARI VE MOTİVASYONU ARTIRMAK İÇİN YAPILAN ARAŞTIRMALAR

Eğitimciler, programlama öğretiminde genel olarak geleneksel metin tabanlı yöntemleri tercih etmektedirler. Oysa Hansen, Schrimpsher ve Narayanan (1998)'ın araştırmaları çoklu ortam kullanılarak oluşturulan görsellerin ve animasyonların metin tabanlı anlatımlara göre daha etkili olduğunu göstermektedir. Ayrıca Arabacıoğlu, Bülbül ve Filiz (2007) programlama

mantığı öğretiminde kullanılmak üzere bir uygulama dili tasarladıkları araştırma sonucunda teorik yöntemlerin, programlamaya yeni başlayanların sürekli bir başkasının desteğine ihtiyaç duymasına neden olduğu için etkili olmadığını vurgulamışlardır.

Programlama öğretiminde öğrenci başarı ve motivasyonunu artırabilmek amacıyla yapılan araştırmalar, görselliğin önemini ve öğrencilerin uygulama yapmasını sağlayacak yardımcı materyal kullanımının faydalarını vurgulamaktadır (Arabacıoğlu, Bülbül ve Filiz, 2007; Gültekin, 2006). Gültekin (2006), Hacettepe Üniversitesi öğrencileri ile yaptığı araştırmada programlama öğretiminde çoklu ortama dayalı eğitim yazılımı kullanımının öğrenci başarısı üzerindeki etkisini araştırmıştır. Araştırma sonucunda çoklu ortama dayalı eğitim yazılımı kullanımının, öğrenci başarısı üzerinde istatistiksel olarak anlamlı bir etkisi bulunduğu saptanmıştır.

Naser (2008), A1-Azhar Üniversitesi'nde yaptığı deneysel çalışmada, görselleştirme araçlarının öğrencilerin performansına etkilerini araştırmıştır. Araştırmacı, çalışmasında programlama öğretiminde geleneksel yöntem ile eski ve yeni versiyon görselleştirme araçları kullanarak uygulanan yöntemleri karşılaştırmıştır. Yeni versiyon görselleştirme araçları, eski versiyon görselleştirme araçlarına yeni görsel özellikler eklenerek oluşturulmuştur. Araştırma sonuçları, görselleştirme araçları kullanılarak uygulanan yöntemlerin geleneksel öğretim yöntemine göre öğrenci performansı açısından daha etkili olduğunu göstermektedir. Araştırma sonuçları ayrıca yeni versiyon görselleştirme araçlarının, eski versiyon görselleştirme araçlarına göre öğrenci başarısı üzerinde daha etkili olduğunu göstermektedir (Naser, 2008).

Klassen (2006)'in çalışmasına göre, programlama konularının görseller veya animasyonlar yardımı ile sunumu, öğrencilerin motivasyonunu artırmaktadır. Araştırma, programlama derslerinde animasyon ve görsellerin kullanımının, öğrencilerin derse katılımını artırdığını ve öğrencilerin konuları akılda tutmasını kolaylaştırdığını vurgulamaktadır.

Grissom, McNally ve Naps (2003), araştırmalarında çeşitli üniversitelerdeki çalışma sonuçlarına değinmişlerdir. Çalışmada, görselleştirme aracı ile etkileşimin, öğrenmeye olan etkileri araştırılmıştır. Araştırma sonuçları, öğrenci etkileşimi artıkça öğrenmenin arttığını göstermektedir. Araştırmaya göre, görselleştirme araçları, öğrencilerin hem görselleri izleyebilecekleri, hem de görsellerle etkileşime girebilecekleri şekilde kullanıldığında daha etkili olmaktadır.

Urquiza-Fuentes ve Velázquez-Iturbide (2007), araştırmalarında öğrencilerin başarılı olmaları için animasyonları yalnızca izlemelerinin yanı sıra animasyonlarla etkileşime girmelerinin de gerekli olduğunu belirtmektedirler. Araştırmacılar, çalışmalarında öğrencileri yapılandırma grubu ve izleme grubu olmak üzere iki gruba ayırmışlardır. İzleme grubu animasyonları sadece izleyerek öğrenirken, yapılandırma grubu animasyonlar üzerinde değişiklikler yaparak öğrenmektedir. Araştırma sonucunda grupların etkinlikleri değerlendirildiğinde yapılandıran grupta yer alan öğrencilerin, izleyen grupta yer alan öğrencilerden daha başarılı olduklarını göstermektedir.

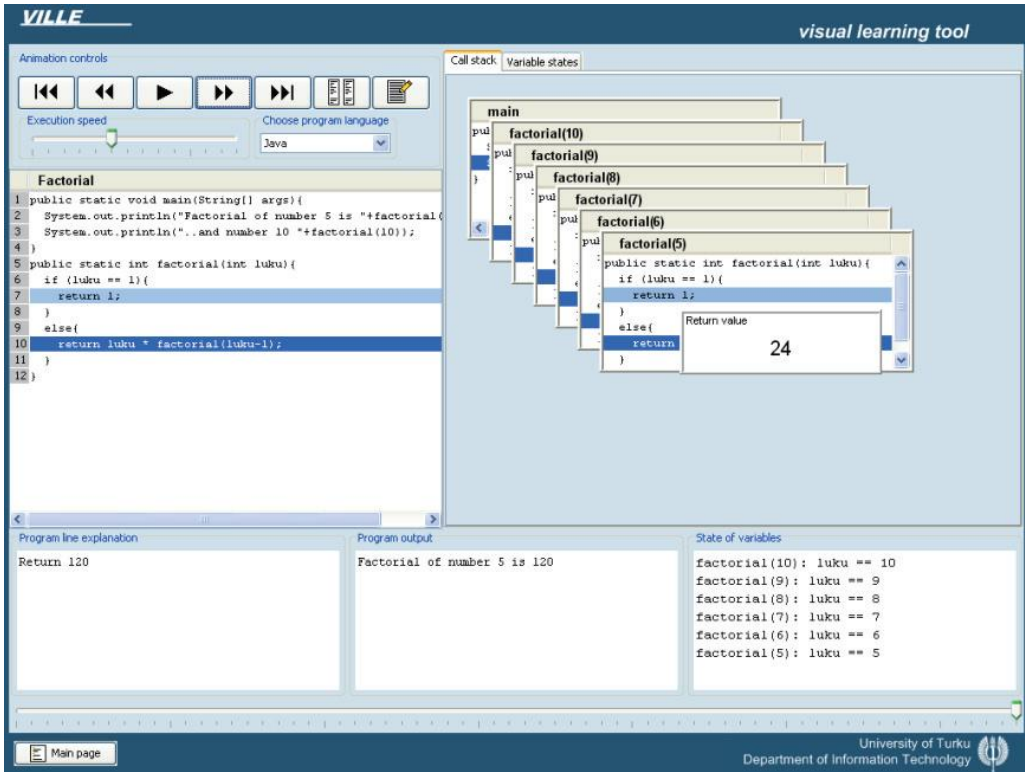
Buraya kadar incelenen araştırmalar, programlama öğretiminde görselliğin ve etkileşimin öğrencilerin başarı ve motivasyonları açısından önemini vurgulamaktadırlar. Bu sebeple öğrencilerin programlama öğrenmelerini kolaylaştırmak amacıyla çeşitli görselleştirme araçları geliştirilmiştir. Bu araçlar, programlama öğrenmeye yeni başlayanların etkileşimli bir ortamda basit komutlar kullanarak temel düzeyde programlar oluşturmalarını sağlar. Yapılan araştırma sonuçlarına göre programlama öğretiminde görsel araçların kullanımı, öğrenci başarısını artırmaktadır. (Malan ve Leitner, 2007; Peppler ve Kafai, 2007; Hu, 2004; Cooper, Dann ve Pausch, 2003; Hyrskykari, 1993). Programlama öğretimine yardımcı araçlarda yer alan görsel özelliklerin önemini vurgulamak amacıyla yapılan araştırmaların sonuçları, görsel özelliklerin öğrenci başarısı ve motivasyonunda önemli bir etkisi olduğunu göstermektedir (Kelleher, Pausch ve Kiesler, 2007; Ramadhan, 2000).

2.4 PROGRAMLAMA ÖĞRETİMİNDE KULLANILAN GÖRSELLEŞTİRME ARAÇLARI

Literatürde programlama öğretimine yardımcı araçlarla ilgili birçok araştırma bulunmaktadır. Begel (1997), araştırmasında çocukların Java programının özelliklerini kolaylıkla kullanmasını sağlayan bir programlama dili ve ortamı olan Bongo'yu açıklamıştır. Bongo, çocukların kendi video oyunlarını oluşturmalarını ve bu oyunları Internet üzerinden paylaşmalarını sağlamaktadır. Programlama öğretiminde kullanılabilecek diğer bir yazılım da Toontalk adlı yazılımdır. Toontalk yazılımında kaynak kodları, animasyon ile gösterilmektedir. Programlama ortamı bir video oyunudur. Toontalk, öğrencilerin bir karakteri kontrol ederek onun programları kurması, çalıştırması, hataları ayıklaması ve değiştirmesini sağlamaktadır (Kahn, 1996).

Scratch bilgisayar programlamanın temellerini görsellik yoluyla öğreten diğer bir yazılımdır. Kullanıcının kendi animasyonlarını oluşturmasını sağlayan bu yeni programlama dili, Java script olarak bilinen algoritma programının gelişmiş hali olarak karşımıza çıkmaktadır.

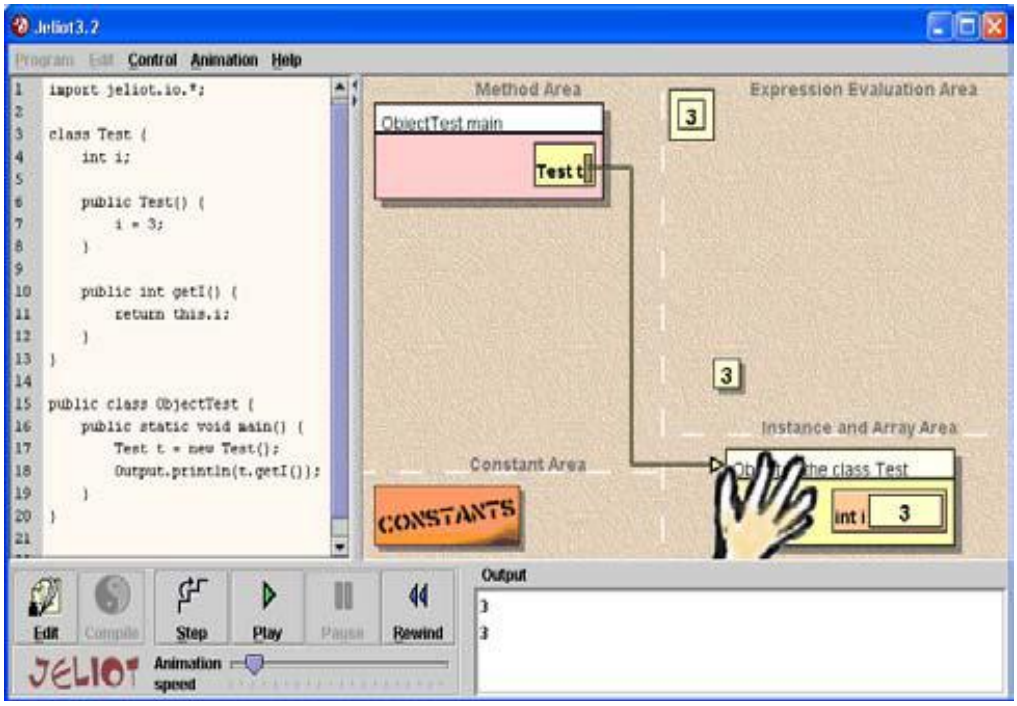
Rajala, Laakso, Kaila ve Salakoski (2008), öğrencilerin programlama ile ilgili konuları anlamalarını kolaylaştırmak için Şekil-1'de görülen Ville adlı bir görselleştirme aracı geliştirmiş ve bu aracı araştırmalarında kullanmışlardır.



Şekil 1 Ville Yazılımının ekran görüntüsü (Rajala, Laakso, Kaila ve Salakoski, 2008).

Ville öğrencilere programların çalışmasını inceleme imkanı sunmaktadır. Programın ana özelliği; dil bağımsızlığıdır. Yani program, uygulamalarının iki farklı dilde yapılabilmesine ve yeni programlama dillerinin tanımlanmasına olanak vermektedir. Kullanıcı, programı çalıştırma işlemi sırasında değişkenlerin değerlerindeki değişimi ve program çıktısındaki değişimi gözlemleyebilmektedir. Rajala, Laakso, Kaila ve Salakoski (2008), geleneksel yöntemler ile Ville aracının kullanıldığı öğretim yöntemlerini karşılaştırdıkları çalışmalarının sonucunda Ville'nin öğrencilerin temel programlama kavramlarını öğrenmelerini kolaylaştırdığını, ayrıca programlama öğrenmeye yeni başlayan öğrenciler üzerinde daha etkili olduğunu vurgulamaktadırlar.

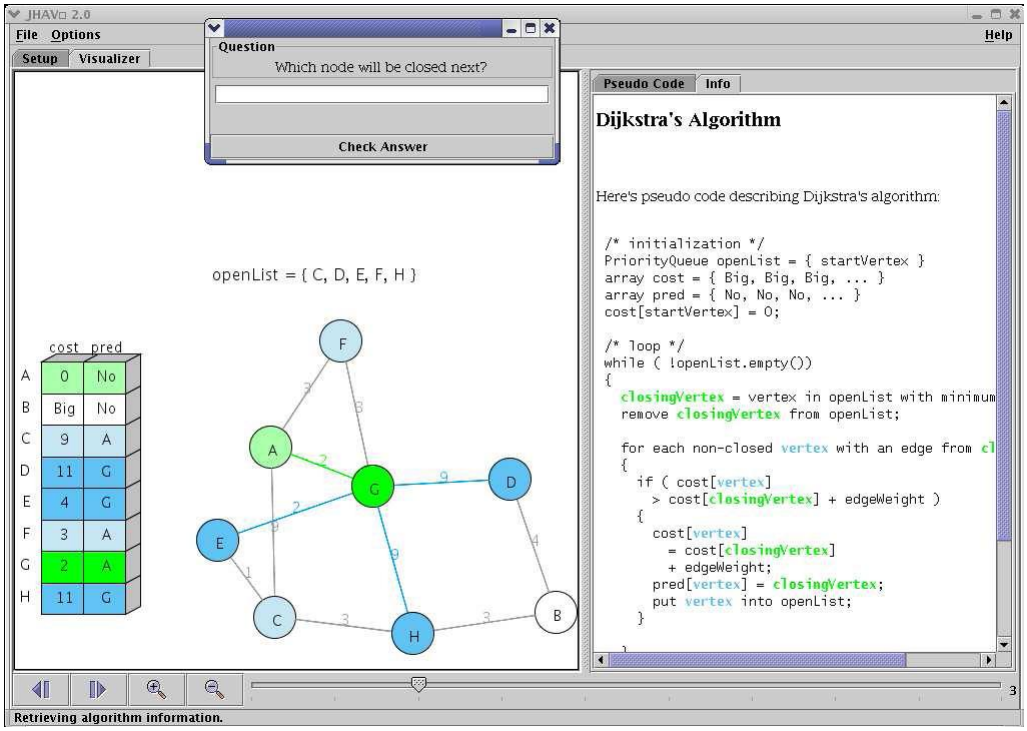
Moreno, Myller, Sutinen ve Ben-Ari (2004), araştırmalarında Şekil-2'de görülen Jeliot 3 adlı bir program görselleştirme aracını tanıtmışlardır.



Şekil 2 Jeliot 3 Yazılımı Ekran Görüntüsü (30 Mayıs 2008 tarihinde http://cs.joensuu.fi/jeliot/description.php web adresinden edinilmiştir.)

Jeliot 3, programlamayı yeni öğrenen öğrencilere yapısal ve nesne yönelimli programlama öğretimi için tasarlanmış bir program görselleştirme aracıdır. Jeliot 3, öğrencilerin kendi programlarını oluşturmalarına katkı sağlamakla kalmayıp aynı zamanda programlarının çalışmasını görsel olarak incelemelerine yardımcı olmaktadır. Araştırmacılar, Jeliot 3 aracını, Joensuu Üniversitesi'nde programlama dilleri 2 dersinde kullanmışlardır. Araştırmada öğrencilerin Jeliot 3 ile ilgili düşüncelerine yer verilmiştir. Araştırmanın sonucunda Jeliot 3 aracını kullanan öğrencilerin daha başarılı olduğu görülmüştür. Araştırma sonucunda öğrenciler, görselleştirme aracında yer alan dönütleri; koşul cümleleri, döngü ve nesneleri anlamayı kolaylaştırması nedeniyle olumlu bulmuşlardır. Öğrenciler, ayrıca animasyonların yavaş çalıştığı ve aracın gelişmiş dersler için uygun olmadığı yönünde görüş belirtmişlerdir. (Moreno, Myller, Sutinen ve Ben-Ari, 2004).

Naps (2005), algoritma görselleştirme ile ilgili yaptığı araştırmada Şekil-3'te ekran görüntüsü yer alan Jhave adındaki bir programı tanıtmaktadır.



Şekil 3 Jhave Yazılımı Ekran Görüntüsü (Naps, 2005).

Araştırmaya göre Jhave programının faydaları şunlardır:

1. Öğrencilerin adım adım ilerleyerek çalışmalarını sağlar.
2. Bilgi ve kod pencereleri sayesinde öğrencilerin algoritmayı oluşturan kodları anlamalarına da yardımcı olur.
3. Nesneler yardımıyla öğrencilerin öğrenmesini kolaylaştırır.
4. Jhave yardımıyla öğrencilere bir süre sonra hangi ekranın geleceği ile ilgili sorular sorularak öğrencilerden tahmin yürütmeleri istenebilir.
5. Tasarımcılar, hem grafiksel görüntüleme hem de etkileşimli araçlar oluşturabilir (Naps, 2005).

2.4.1 Programlama Öğretiminde Kullanılabilecek Görselleştirme Araçlarının Sınıflandırılması

Bergin ve diğerleri (1996)'ne göre programlama eğitiminde üç farklı görselleştirme tipi vardır. Bunlar:

1. Program Görselleştirme: Bu tip görselleştirme araçları, bir programın nasıl çalıştığını grafiksel olarak sunmak için kullanılır.
2. Algoritma Animasyonları: Temel algoritmaların hangi işlemleri tamamladığını görseller yardımıyla göstermek amacıyla kullanılır. Algoritma animasyonları, algoritmanın bir görsele yönlendirilmesi ve algoritma yürütüldüğünde işlem sırasına göre bu görselin hareket ettirilmesi mantığıyla çalışır.
3. Veri Görselleştirme: Bu tip görselleştirme araçları ise program kodu içerisinde veri tiplerinin neler olduğu ve programda tanımlanan işlemlerden sonra verilerin son durumlarının ne olduğu hakkında bilgi verir (Gültekin, 2006).

Program görselleştirme: Bu tip görselleştirme araçları çalıştırılan programın ve verilerinin grafiksel gösterimi üzerine odaklanmaktadır. Bu araçlar yardımıyla veri, kod ve olaylar görselleştirilmektedir. Veri ve kodların önceden tanımlı görselleştirmeleri vardır. Kullanıcı ile etkileşim, programın istenilen yerde durdurulması ve girdilerde değişiklik yapılabilmesi şeklinde sağlanmaktadır (Bergin ve diğerleri, 1996). Rajala, Laakso, Kaila ve Salakoski (2008), araştırmalarında program görselleştirme araçlarının, öğrencilerin programların davranışlarını anlamalarına yardımcı olduğunu belirtmektedirler. Araştırmalarında program görselleştirme araçlarına örnek olarak Ville, Jeliot3 ve Bluej araçlarını göstermektedirler.

Algoritma animasyonları: Algoritma animasyonları, algoritma için gerekli işlemleri görselleştirir. Görselleştirme, kod ve verilerle sınırlı değildir. Algoritma animasyonlarında kullanıcı, neyin görselleştirileceğine kendisi karar verir. Kullanıcı, açıklayıcı notlar yardımıyla algoritma ve nesneler arasındaki ilişkiyi tanımlayabilir ve ilgili görselleri seçerek ne zaman görselleştirileceğini ayarlar. Algoritma animasyonlarında kullanıcı etkileşimi, programcı tarafından tanımlanabilmektedir. Bu tür araçlar, kullanıcı ile yüksek düzeyde etkileşim sağlamaktadır (Bergin ve diğerleri, 1996). Gültekin (2006), araştırmasında algoritma animasyonlarının bilgisayar algoritmalarını görselleştirerek anlamayı

kolaylaştırdığı için bir eğitimsel araç olarak kullanılabileceğini belirtmiştir. Samba, bu amaçla yazılmış programlardan birisidir. Samba, etkileşimli bir biçimde komutları yorumlayarak animasyonları oluşturan bir araçtır. Algoritma animasyonlarına diğer bir örnek ise Dershem ve Brummund (1988. Aktaran: Gültekin, 2006) tarafından tasarlanan "The Sort Animator" isimli araçtır. Naps, Eagon ve Norton (2000) tarafından geliştirilen Jhave ise işlemci-sunucu mimarisine dayanan bir algoritma animasyonu aracı olup Internet üzerinde çalışmaktadır. Cooper, Dann ve Pausch (2003), araştırmalarında Xtango, Balsa gibi algoritma animasyonlarına değinmişler ve bu araçların öğrencilerin algoritmaları analiz etmelerine yardımcı olması açısından programlama derslerinde kullanılabileceğini belirtmişlerdir. Bu tür yazılımlarda öğretmen, sık kullanılan algoritmalar için bir animasyon geliştirebilmektedir. Öğrenciler ise önceden hazırlanan animasyonu değişik girdilerle çalıştırmakta ve algoritmanın döndürdüğü sonuçları görebilmektedirler.

Veri Görselleştirme: Bu tip görselleştirme araçları, verilerin aldıkları değerleri görselleştirir. Veri görselleştirme araçları, sadece veri değerlerini göstermekle kalmayıp ayrıca veriyi organize ederler. Bu tür yazılımlarda kullanıcı ile düşük düzeyde etkileşim sağlanmaktadır (Bergin ve diğerleri, 1996). Chen ve Sobh (2001), araştırmalarında Javamy adında bir veri görselleştirme aracını tanıtmışlardır. Veri görselleştirme araçları, veri yapıları üzerinde çok genel işlemler olan veri ekleme çıkarma gibi işlemler için kullanılmaktadır.

Cooper ve diğerleri (2006), araştırmalarında programlama öğretiminde kullanılan görselleştirme araçlarını kategorilere ayırmışlardır. Bu kategoriler: algoritmayı hikayeleştiren araçlar (narrative tools), görsel programlama araçları (visual programming tools), akış şeması modelli araçlar (flow-model tools), program çıktılarını görselleştiren araçlar (specialized output realizations) ve sıraya dizilmiş programlama dili araçları (tiered language tools)'dır.

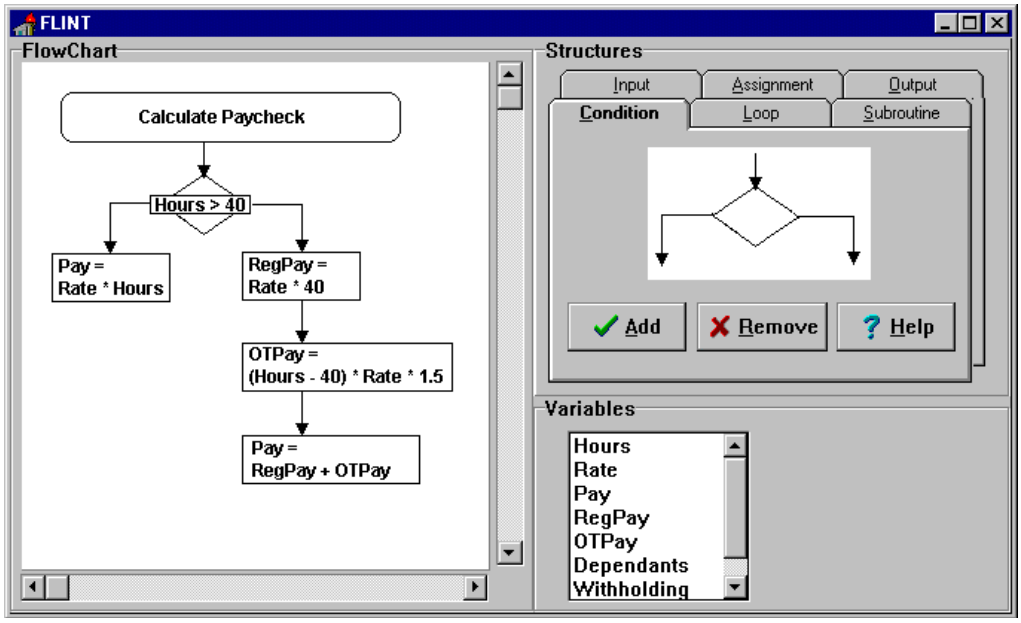
Algoritmayı hikayeleştiren araçlar: Programlamayı bir animasyon karakter yardımıyla hikayeleştirerek öğretmeyi amaçlayan araçlardır. Storytelling (hikayeleştirme) kullanımının iki önemli faydası vardır: Birincisi öğrencilerin kendi filmini yönetebilmeleri, onların ilgisini çekmektedir. İkincisi ise öğrencilerin programı hikaye şeklinde oluşturmaları, onların görsel senaryo ile tanışmalarını sağlamaktadır. Alice ve Jeroo, bu tür araçlara örnek olarak gösterilebilir. Bu tür araçlardan JPie, hareketli bir karakter aracılığıyla sürükle bırak yöntemini kullanarak öğrencilere programlama öğretmeyi amaçlar (Cooper ve diğerleri, 2006). Bu tür yazılımların kullanımının programlama öğretiminde sağladığı kolaylıklar aşağıda sıralanmaktadır.

1. Kod yazımı olmadığı için yazım hatalarından kaynaklanan problemler azalır.
2. Veri ve nesne kavramları, görselleştirme ile öğretilir.
3. Program değişiklikleri görülebilir. Basit biçimde hata ayıklama işlemi yapılabilir.
4. Programlama öğretimi, önce senaryo (storyboard) sonra programlama mantığı ile verilebilir (Klassen, 2006).

Görsel Programlama Araçları: Programları bir ara yüz yardımıyla, sürükle – bırak biçiminde yapılandırmayı sağlayan araçlardır. Görsel programlama araçları, programcıkların yazım hataları ile ilgili sorunlar olmaksızın geliştirilebilmesine imkan vermektedir. Bunun yanı sıra kullanıcıların, programlama içeriklerinin grafiksel gösterimleri üzerinde değişiklikler yapabilmesini sağlamaktadırlar. Görsel programlama araçları, ayrıca programların çalıştırılıp dönüt alınabilmesini sağlamaktadırlar. Bu araçlara örnek olarak JPie, Alice ve Karel Universe gösterilebilir (Cooper ve diğerleri, 2006).

Akış Şeması Modelli Araçlar: Program parçalarını işlem sırasını gösterecek biçimde birbirine bağlayarak programları yapılandırmayı sağlayan araçlardır. Raptor, Iconic Programmer ve VisualLogic, bu tür araçlara örnek olarak gösterilebilir (Cooper ve diğerleri, 2006). Bu tür araçlar, programlamanın akış

şeması tabanlı öğrenimini sağlar. Akış şeması, bir işin adım adım gösterimidir. Bu programlardan biri olan Flowchart Interpreter (Flint), Şekil-4'te görülmektedir. Flint, sürükle bırak yöntemiyle akış şemaları oluşturmayı sağlar.



Şekil 4 Flint yazılımı ile oluşturulan akış şeması örneği (Garner, 2003).

Flint yardımıyla kolaylıkla bilgisayar programları tasarlanabilir, geliştirilebilir. Yazım hatalarına odaklanılmadığı için problem analiz ve tasarımı için harcanan zaman en aza indirgenmektedir. Öğrenciler, bu araç sayesinde oluşturdukları programlar ile ilgili dönüt alabilmekte ve sonuçları inceleyebilmektedirler (Crews ve Ziegler, 1998). Raptor da Flint gibi akış şeması modelli görselleştirme aracıdır. Raptor, öğrencilerin ikonlar ve program elemanları arasındaki bağlantıyı değiştirerek program inşa edebilecekleri bir ortam sağlar (Klassen, 2006). Akış şeması modelli (Flow model) araçlar, yazım hatası karmaşasını azaltmaları ve öğrencilerin eksik satırlar yerine problem çözmeye odaklanmalarını sağlamaları bakımından kullanışlıdırlar (Cooper ve diğerleri, 2006). Akış şemasının kullanımının programlama öğrenmeye yeni başlayan öğrenciler üzerindeki etkisini araştıran bir çalışma sonuçları, öğrencilerin akış şeması kullandıklarında daha az hata yaptıkları, özellikle basit problemlerin

çözümünde daha başarılı olduklarını ve kendine daha fazla güvendiklerini göstermektedir (Crews ve Ziegler, 1998).

Program Çıktılarını Görselleştiren Araçlar (Specialized Output Realizations): Oluşturulan programlar yürütüldüğünde çoklu ortam kullanarak dönüt veren araçlardır. Bu araçları kullanarak, öğrenciler kendi programlarını kodlayıp test edebilmektedirler. Daha da önemlisi öğrenciler, anında görsel dönüt alabilmektedirler. Lego MindStorms ve JES bu tür araçlara örnek olarak gösterilebilir (Cooper ve diğerleri, 2006).

Sıraya Dizilmiş Programlama Dili Araçları: Programlama dilini öğrenci seviyesine uygun biçimde aşamalı olarak öğreten araçlardır. Programlama öğretimi, yazım kuralı ve karmaşıklığın en aza indirgenmiş şekli ile başlar. Programda yer alan hata mesajları, öğrenciler ilerledikçe seviyelerine uygun biçimde değiştirilir. Daha sonraki aşamada programlama diline yeni ve daha karmaşık özellikler eklenir. Başlangıç düzeyinde öğrenciler, sadece belirgin özelliklere odaklanırken, programlama diline yeni özellikler eklendikçe aşamalı olarak karmaşık özellikler de öğrenmeye başlar. Bu tür araçlara ProfessorJ ve RoboLab örnek olarak gösterilebilir (Cooper ve diğerleri, 2006).

2.4.2 Programlama Öğretiminde Görselleştirme Araçlarının Kullanımı İle İlgili Yapılan Çalışmalar

Literatürde programlama öğretimine başlangıç aşamasında hangi araçların kullanılması gerektiği ile ilgili farklı görüşler vardır. Araştırmanın bu bölümünde bu görüşlere yer verilecektir.

Peppler ve Kafai (2007), çalışmalarında katılımcılara Scratch programı yardımıyla bilgisayarda video oyunu programlama öğretmişlerdir. Katılımcılar, Scratch yazılımı 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 Scratch yazılımı kullanılarak video oyunları hazırlamanın, programlama öğrenmeye yardımcı olduğu belirtilmektedir.

Malan ve Leitner (2007), yaptıkları çalışmada Scratch programını yüksek öğretimde Harvard Yaz Okulu'nun Computer Science dersinde kullanmışlardır. Araştırma kapsamında dönem başında öğrencilere önceki programlama bilgileri üzerine anket yapılmıştır. 25 katılımcı üzerinden % 52'sinin herhangi bir ön bilgisi yoktur, %32'si az bilgiye sahiptir. % 16'sının ise programlama ile ilgili güçlü alt yapısı vardır. Dönem sonunda Scratch programının, önceden programlama deneyimi olan öğrencilerin başarısını nasıl etkilendiği araştırılmıştır. 25 katılımcı içerisinden 19 (% 76)'u pozitif etkisi olduğunu, 2 (% 8)'si negatif etkisi olduğunu, 4 (%16)' ü ise Scratch yazılımının herhangi bir etkisi olmadığını savunmaktadır.

Cooper, Dann, ve Pausch (2003) yaptıkları bir araştırmada Ithaca College ve Saint Joseph Üniversitesi'nde 2001–2002 eğitim - öğretim yılında programlama bilgisi olmayan 21 öğrenciyle çalışmıştır. Bu öğrencilerden 11'i Alice programını kullanmıştır. Araştırma sonucunda programı kullanan grup, Computer Science 1 dersinde programı kullanmayan gruba göre daha başarılı olmuşlardır. Programını kullanan öğrenciler arasında Computer Science 2 kursuna devam edenlerin oranı % 91'dir.

Bishop-Clark, Courte ve Howard (2007), yaptıkları çalışmada görselleştirme aracı olarak "Alice" isimli programlama dilini kullanmışlardır. Araştırmada yer alan öğrenciler, programlama dersi kapsamında 2,5 hafta boyunca Alice ortamında grafiksel programlama uygulamalarına katılmışlardır. Araştırmanın sonucunda programı kullanan öğrencilerin programlama kavramlarını daha iyi anladığı ve programlama ile ilgili kendilerine güvenlerinin arttığı belirtilmiştir.

Alice programı ile ilgili benzer bir çalışma Kelleher, Pausch ve Kiesler (2007) tarafından yapılmıştır. Çalışmalarında Storytelling (3 boyutlu hikayeleştirme özelliği olan) Alice programı ve aynı programın hikayeleştirme özelliği olmayan versiyonunu öğrencilerin programlamayı öğrenmeye etkileri açısından karşılaştırmışlardır. Storytelling Alice programı, ortaokul öğrencilerine bilgisayar programlamayı 3 boyutlu anime edilmiş hikayelerle tanıtan bir programdır. Çalışmanın sonucunda Storytelling Alice kullanıcılarının program yazma konusunda daha fazla motivasyona sahip olduğu görülmüş ve programlamada %42 daha fazla zaman harcadıkları kaydedilmiştir.

Literatürde programlama dersine başlangıç aşamasında görselleştirme araçlarının kullanımında hangi yolun izlenmesi gerektiği ile ilgili çeşitli görüşler vardır. Hundhausen, Douglas ve Stasko (2002), yaptıkları çalışmada görselleştirmenin öğrenme üzerinde istatistiksel açıdan anlamlı pozitif etkisi olduğunu vurgulamışlardır. Hundhausen, Douglas ve Stasko (2002), ayrıca tek başına görselleştirme kullanımının öğrenmeyi kolaylaştırma konusunda yeterli olmadığını, önemli olanın görselleştirme aracı yardımıyla öğrencileri konuya dahil etmek olduğunu belirtmişlerdir. Araştırmacılar, çalışmalarında programlama öğretiminde görselleştirme araçlarının etkileşimli kullanımı için çeşitli senaryolara yer vermişlerdir. Bu senaryolar şunlardır:

1. Algoritmanın nasıl çalıştığının anlaşılması için grafiksel gösterim kullanılabilir.
2. Öğrenciler, kendi görselleştirmelerini oluşturmaya çalışabilirler.
3. Öğrenciler, kendilerine verilen görevleri tamamladıktan sonra kendi hazırladıkları görselleştirmeleri arkadaş ve öğretmenlerine sunarak fikir ya da dönüt almaya çalışabilirler.
4. Öğrenciler laboratuar alıştırmalarını yaparken etkileşimli bir biçimde algoritmaları keşfedebilir.
5. Öğrenciler, kendi görselleştirmelerini çizerek, başkalarının hazırladıkları görselleştirmeleri kağıt üzerinde inceleyerek ya da etkileşimli bir görselleştirme yazılımı kullanarak çalışabilirler.

6. Öğretmenler, öğrencilerin hatalarını tespit edebilir ve sorularına cevap verebilirler.

7. Öğrencilerin bilgileri değerlendirilirken, öğrencilerin algoritmaları isimlendirmeleri, algoritma içerisindeki görsellerin tanımlamaları ve algoritmanın nasıl çalışacağını tahmin etmeleri ya da çizmeleri istenebilir (Hundhausen, Douglas ve Stasko, 2002).

Naps ve diğerleri (2003)'ne göre bilgisayar bilimleri dersinde kullanılan görselleştirmeler, öğrencileri aktif olarak öğrenmeye katmalıdır. Görselleştirmeler, öğrencileri aktif olarak öğrenmeye katmıyorsa nasıl tasarlanmış olursa olsun eğitimsel değeri düşük olacaktır. Araştırmacılar, programlama dersinde öğrencilerin yapabilecekleri etkinlikleri Bloom'un taksonomisine dayanarak düzeylere göre sunmuşlardır. Araştırmada ayrıca öğrencilere görselleştirme aracı kullanarak verilebilecek görevlere de yer verilmiştir. Düzeylere göre programlama dersinde yer verilecek kazanımlardan bazıları aşağıdaki gibidir.

1. Bilgi (Knowledge): Algoritmanın kendine özgü kavramlarını hatırlama ve bilgi verici biçimde bu kavramları ifade etme.
2. Kavrama (Comprehension):
   a. Algoritmanın arkasındaki genel prensibi anlama ve algoritmaların nasıl çalıştığını kelime ve resimlerle nasıl çalıştığını açıklama
   b. İçeriği formel olarak açıklayabilme ve sunma
   c. Algoritmanın içerdiği anahtar kavramları ve bu kavramların algoritmadaki rollerini anlama
   d. Bazı programlama dillerini kullanarak algoritmayı test etme ve temsil etme
   e. Algoritmanın nasıl davrandığını anlama
   f. Algoritmayı takip etme
3. Uygulama (Application):
   a. Önceden çalışılmış bir algoritmayı özel bir uygulamaya adapte etme

4. Analiz

a. Algoritmanın diğer algoritmalarla ilişkisini anlama

b. Algoritma kodundaki sabitleri anlama

c. Algoritma ile ilgili sorgulama yapma ve / veya algoritmanın doğruluğunu kanıtlama

d. Karmaşık bir problemi analiz etme, gerekli nesneleri tanımlayarak küçük problemlere bölme.

5. Sentez

a. Karmaşık bir probleme değişik veri yapıları, algoritma ve gerekli tekniklerle çözümler tasarlama

b. Karmaşık algoritmaların verilerini analiz etme

c. Değişik algoritma çözümlerini karşılaştırmak için kriter oluşturmak.

6. Değerlendirme

a. Yeni ve karmaşık bir problemi etkin biçimde çözebilmek için bazı algoritmaların niçin ve nasıl değiştirilip diğer algoritmalarla bütünleştirilmesi gerektiğini tartışma.

b. Aynı veya benzer problemlerin çözümü için kullanılan farklı algoritmaların iyi ve kötü yönlerini kararlaştırma.

c. Tasarım ve analiz değerlendirmesi yapma.

Araştırmada ayrıca etkileşimlerin öğrencilerin başarısı üzerindeki etkisi karşılaştırılmıştır. Buna göre görselleştirme araçları kullanılarak yapılabilecek olan etkileşimler ve öğrenme üzerindeki etkisi aşağıdaki gibidir:

1. Görüntüleme (Viewing): Görüntüleme pasif biçimde yapıldığında öğrenci başarısında artış gözlenmemektedir. Görüntüleme, öğrencinin de katıldığı biçimde yapıldığında başarıda artış gözlenmektedir.

2. Yanıtlama - Görüntüleme (Responding vs. Viewing): Yanıtlamanın kullanıldığı etkinlikler, görüntülemenin kullanıldığı etkinliklere göre başarı üzerinde daha etkilidir.

3. Değiştirme - Yanıtlama (Changing vs. Responding): Değiştirmenin kullanıldığı etkinlikler, yanıtlamanın kullanıldığı etkinliklere göre başarı üzerinde daha etkilidir.

4. Yapılandırma – Değiştirme (Constructing vs. Changing): Yapılandırmanın kullanıldığı etkinlikler, yanıtlamanın kullanıldığı etkinliklere göre başarı üzerinde daha etkilidir.

5. Sunma – Yapılandırma (Presenting - Constructing): Sunum yapma, yapılandırmadan daha olumlu sonuçlar vermektedir.

6. Çoklu Bütünleştirme (Multiple Engagement): Etkileşim ne kadar çok kullanılırsa öğrenme de o kadar artmaktadır (Naps ve diğerleri, 2003).

Bravo, Marcelino, Games, Esteves ve Mendes (2005), görselleştirme araçlarının derslerde kullanımı ile ilgili yaptıkları araştırmada, akış şeması modelli araçların öğrencilerin algoritma yapılandırma yeteneklerini geliştirebileceğini belirtmişlerdir. Algoritmayı bir karakter yardımıyla sürükle – bırak yöntemiyle oluşturan araçların ise öğrencilere nesne yönelimli programları anlamaları ve analiz etmeleri konusunda yardımcı olabileceği belirtmiştir.

Ramadhan (2000), yaptığı çalışmasında öğrencileri deney ve kontrol grubu olmak üzere iki gruba ayırmıştır. Gruplara ön test uygulamıştır. Deney grubunda DISCOVERY adlı bir görselleştirme aracı kullanmıştır. Kontrol grubunda ise aynı programın görsel özelliklerinin gizlendiği, değiştirilmiş versiyonunu kullanmıştır. Yapılan uygulamanın sonunda DISCOVERY yazılımını görsel özellikleriyle kullanan deney grubu son testteki döngü ve koşul içerikli sorulara kontrol grubundan daha başarılı olmuşlardır. Araştırma sonuçları, programlama öğretiminde görsel araçlardan destek almanın öğrenci başarısı üzerinde olumlu etkisi olduğunu vurgulamaktadır.

Lahtinen, Ahoniemi ve Salo (2007), program görselleştirmenin programlama derslerinde nasıl kullanılabileceği ile ilgili yaptıkları çalışmada VIP (Visual Interpreter) isimli bir araç geliştirmişlerdir. Bu aracı kullanarak öğrencilerle

koşul, döngü, dizi ve fonksiyonlar konularıyla ilgili çalışma yapmışlardır. Öğrenciler, koşul cümlelerini kolay bulurken, fonksiyonlarda zorlanmışlardır. Araştırma sonuçları, problemlerin karmaşıklığı arttıkça görselleştirme araçlarına daha çok ihtiyaç duyulduğunu göstermektedir.

Klassen (2006), programlama öğretiminde önce senaryo sonra programlama mantığı kullanılmak istendiğinde algoritma animasyonlarının kullanılmasını önermiştir. Araştırmacı, öğrencilerin programlama öğretiminde program elemanları arasında bağlantı kurabilmesi ve bu yolla akış şeması oluşturabilmesi içinse akış şeması modelli araçlar kullanılabileceğini belirtmiştir.

Baldwin ve Kuljis (2000) ise programlama öğrenimine yeni başlayanlar için algoritmik programlama öğretiminde öğrencilerin değişkenleri, işlemleri, akışı öğrenmesini sağlayacak bir program kullanılmasının önemli olduğunu belirtmişlerdir.

Garner (2003), yazılım döngüsünün 4 temel aşamasını kullanarak görselleştirme araçlarının hangi aşamalarda kullanılabileceği ile ilgili örnekler vermiştir. Buna göre Garner (2003), akış şeması yazılımlarının program tasarım, geliştirme ve algoritma oluşturma aşamalarında; algoritma animasyonlarının ise programın test etme ve gözden geçirme aşamasında kullanılabileceğini öne sürmüştür.

2.5 PROGRAMLAMA ÖĞRETİMİNDE KULLANILABİLECEK YÖNTEMLER

Programlama öğretiminde görselleştirme araçlarıyla kullanılabilecek yöntemler ile ilgili çok sayıda araştırma bulunmaktadır. DipICT (Diploma in Information and Computer Technology) derslerine katılan öğrencilerle 2003 yılında yapılan bir araştırmaya göre öğrencilerin % 90'ı programlama öğrenimi konusunda teori ve pratik uygulamayı eş zamanlı olarak öğrenmeyi tercih ederlerken,

öğrencilerin % 10'u ise önce mantıksal tasarımın verilmesini daha sonra pratik uygulamanın da yapılmasını tercih etmişlerdir (Hu, 2004).

Lawrence, Badre ve Stasko (1994), yaptıkları çalışma sonucunda öğrencilerin programlama öğretiminde bir yazılım yardımıyla kendi animasyonlarını oluşturabilmelerinin ve animasyonlara sınıf dışında erişebilmelerinin önemini vurgulamışlardır. Nelson ve Rice (2000) ise yaptıkları araştırmada programlama öğretiminde öğrencilerin programlama dilinden bağımsız biçimde algoritma oluşturabilmelerinin önemine değinmişlerdir. Çalışmada ayrıca öğrencilerin yazdıkları kodlarla ilgili dönüt alabilmelerinin önemli olduğu vurgulanmıştır.

Lin ve Zhang (2003), çalışmalarında programlama öğretiminde kullanılan yöntem ve tekniklerin öğrenci motivasyonu üzerindeki etkisini araştırmışlardır. Bu yöntem ve teknikler; öğrenci merkezli yaklaşım, işbirliğiyle öğrenme, algoritma tasarlama, yardımcı bir yazılım kullanımı, animasyon oluşturma ve erişilebilir kodlardan yararlanmadır. Çalışma sonunda araştırmacılar, öğrencilerin kendi animasyonlarını oluşturmalarının onların motivasyonunu artırdığı sonucuna ulaşmışlardır (Lin ve Zhang, 2003).

Naps ve diğerleri (2003), programlama öğretiminde görsel teknolojinin kullanımı ile ilgili yaptıkları taksonomiye göre programlama öğretimi, altı adımda gerçekleşmektedir: 1. Görselleştirme aracı olmaksızın anlatım, 2. Görselleştirme aracının kullanımı 3. Yanıtlama, 4. Değişiklik yapabilme, 5. Kendi programını oluşturma, 6. Sunma. Araştırmacılar, çalışmalarının sonucunda programlama öğretiminde etkileşimin önemini vurgulamaktadırlar.

Araştırmalar, etkileşimli görsellerin, kara tahta ve projektör kullanımı gibi materyallerle ile birleştirildiğinde çok daha başarılı olduğunu kanıtlamıştır (Brusilovsky ve Spring, 2004). Bu bağlamda görsel sunumlarla desteklenen bir anlatım yönteminin kullanımı önemlidir (Felder ve Silverman, 1988).

Korhonen (2003) programlama dersinde öğrenme ortamının nasıl düzenlemesi gerektiği ile ilgili yaptığı çalışmasında, öğrencilerin görselleştirme araçları kullanarak etkileşimli biçimde kendilerine verilen program örneklerini düzenleyebilmeleri ve kendi tasarımlarını yapabilmelerinin önemli olduğunu vurgulamaktadır. Korhonen (2003)'e göre, programlama öğretimi özetle beş farklı yöntemle yapılabilmektedir: 1. Öğretici materyallerin kullanımı 2. Görselleştirme aracının kullanımı 3. Sınıf ortamında öğretmen rehberliğinde laboratuar uygulamalarının yapılması 4. Öğrencilerin Internet ortamında kendi hızlarına göre ilerleyebilmeleri 5. Yukarıdaki maddelerin beraberce kullanımı. Yani her madde bir önceki maddeyi takiben kullanılabilir (Korhonen, 2003).

Yapılan araştırmalara göre programlama öğretiminde önce konuların genel hatlarıyla verilmesi daha sonra uygulamaların yapılması yerine teori ve pratik uygulamanın iç içe kullanımı daha başarılı olmaktadır (Crews ve Murphy, 2004, Ziegler ve Crews, 1999. Aktaran: Hu, 2004). Bu bağlamda öğretime programlama dillerinden bağımsız biçimde başlanmalı, daha sonra bir programlama dili kullanılarak uygulama yapılmalıdır. Programlama mantığı öğretilirken, basitten karmaşığa doğru adım adım gidilmesi gerektiği unutulmamalıdır (Hu, 2004).

2.6 PROGRAMLAMA ÖĞRETİMİNDE DEĞERLENDİRME

Algoritma geliştirme konusunda verilen eğitimin etkililiğinin ölçülmesi, öğrencilerin başarılı olup olmadığı konusunda yorum yapılabilmesi ve onların gerektiğinde bu konuda uygun bölümlere yönlendirilebilmesi için değerlendirme yapılması gerekmektedir. Bu durum, algoritma geliştirme konusunda öğrencilerin nasıl değerlendirilmesi gerektiği sorununu ortaya çıkmaktadır. Türkiye'de ilköğretim okullarında bilişim teknolojileri dersi seçmeli olduğu için notla değerlendirme yapılmamaktadır. Onun yerine performans ve proje ödevleri verilerek öğrencilerin konuyu ne derece anladıkları ölçülebilmektedir. Fakat bu tür değerlendirmeler, öğrencilere verilen eğitimin ve uygulanan yöntemin ne derece etkili olduğunu ölçmek konusunda yetersiz kalmaktadır. Bilgisayar derslerinde değerlendirme, kağıt

üzerinde yapılabildiği gibi bilgisayar üzerinde uygulama sınavı şeklinde de yapılabilmektedir. Bilgisayar programlama ile ilgili değerlendirme yapılırken bu değerlendirme yöntemlerinden hangisinin kullanılması gerektiğinin araştırılması önemlidir.

Literatürde derslerin bilgisayar üzerinde değerlendirilmesi ile kağıt üzerinde değerlendirilmesinin öğrenci performansı üzerindeki etkilerini araştıran birçok çalışma bulunmaktadır. Bu çalışmalardan bir kısmı bilgisayar tabanlı değerlendirme sonucunda öğrenci başarısının kağıt üzerindeki değerlendirmelere göre daha yüksek olduğunu göstermektedir (Clariana ve Wallace, 2002; Pomplun, Frey ve Becker, 2002; DeAngelis, 2000; Bojic ve Greasley, 1999; Russell ve Haney, 1997). Araştırmacıların bir kısmı ise kağıt üzerinde değerlendirme sonuçlarının, bilgisayar tabanlı değerlendirme sonuçlarına göre daha yüksek olduğunu göstermektedir (Russell, 1999; Mazzeo, Druesne, Raffeld, Checketts ve Muhlstein, 1991; Lee, Moreno ve Sympson, 1986). Diğer çalışmalar ise bilgisayar tabanlı değerlendirme sonuçları ile kağıt üzerindeki değerlendirme sonuçları arasında fark bulamamışlardır (Mason, Patry ve Bernstein, 2001; Weinberg, 2001; Neuman ve Baydoun, 1998; Mead ve Drasgow, 1993; Ward, Hooper ve Hannafin, 1989) .

2.7 PROGRAMLAMA ÖĞRETİMİNDE ÖĞRENCİ BAŞARISINI ETKİLEYEN FAKTÖRLER

Öğrencilere programlama dersi verirken onların başarı ve motivasyonlarına etki eden faktörleri bilmek ve ona göre yöntem ve teknikler geliştirmek önemlidir. Bu bakımdan programlama başarısına etki eden faktörleri inceleyen araştırmacılar, programlama öğretimi için ders planlarının yapılabilmesinde yol gösterebilecek çeşitli faktörler belirlemişler ve bu faktörlerin başarı üzerindeki etkilerini incelemişlerdir.

Leeper ve Silver (1982) ve Petersen ve Howe (1979), öğrencilerin önceki akademik başarılarıyla üniversitede almış oldukları bilgisayar programlama

ders başarıları arasında akademik olarak başarılı olan öğrenciler lehine pozitif ilişki bulmuşlardır. Araştırma sonuçları ayrıca öğrencilerin önceki bilgisayar deneyimlerinin de programlama başarıları üzerinde etkili olduğunu göstermektedir.

Nowaczyk (1983), araştırmasında öğrencilerin İngilizce dersindeki akademik başarıları ile bilgisayar programlama başarıları arasında pozitif yönde anlamlı bir ilişki olduğu sonucuna ulaşmıştır. Leeper ve Silver (1982), Ingilizce ve lisede alınan yabancı dil ile programlama başarısı arasındaki ilişkiyi araştırmışlar ve sonuç olarak Ingilizce ile programlama başarısı arasında anlamlı bir ilişki olduğunu bulmuşlardır. Ancak Byrnes ve Lyons (2001. Aktaran: Jones ve Burnettt, 2008) araştırmalarında yabancı dil veya İngilizce sonuçları ve programlama başarısı arasında ilişki bulamamışlardır (Jones ve Burnettt, 2008). Bu araştırmalarda İngilizce'nin ana dil olması, bu araştırma sonuçları ile ilgili yorum yapılırken dikkate alınması gereken önemli bir noktadır.

Birçok çalışmada öğrencilerin matematik ders başarıları ile programlama başarıları arasında ilişki bulunmuştur (Byrnes ve Lyons, 2001. Aktaran: Jones ve Burnettt, 2008; Erdoğan, 2005; Fletcher, 1984; Pea ve Kurland, 1983; Soloway, Lochhead ve Clement, 1982. Aktaran: Reed ve
Burton, 1988; Webb, 1985).

BÖLÜM III
YÖNTEM

Araştırmanın modeli, çalışma grubu, veri toplama araçları, verilerin toplanması, çözümlenmesi ve araştırmada yapılan tüm uygulamalar bu bölümde ele alınmıştır.

3.1 ARAŞTIRMA MODELİ

Deneme modeli ve ilişkisel tarama modelinin kullanıldığı çalışma, iki deney grubu ile gerçekleştirilmiştir. Çalışmada görselleştirme araçlarının öğrencilerin algoritma geliştirme başarılarına olan etkisini karşılaştırabilmek amacıyla son test deneme modeli, görselleştirme araçlarının öğrencilerin motivasyonları üzerindeki etkisinin belirlenmesinde ise ön test-son test deneme modeli kullanılmıştır. Araştırmada öğrencilerin algoritma geliştirme başarıları ile akademik başarıları arasındaki ilişkinin saptanmasında için ilişkisel tarama modelinden yararlanılmıştır.

Araştırma öncesinde yapılan ön bilgi testi, öğrencilerin algoritma geliştirme ile ilgili ön bilgilerinin ölçülmesi ve öğrencilerin eş gruplara ayrılması amacıyla kullanılmıştır. Araştırma süresince yapılan uygulamalar ve uygulamada geçen süreler Tablo-1'de görülmektedir.

Tablo 1 Araştırma Kapsamında Yapılan Uygulamalar ve Uygulamalarda Geçen Süre

Uygulama öncesi	
Etkinlik	Süre
Pilot Çalışma	2 saat
Ön Bilgi Testi	1 saat
Motivasyon Ön Test	1 saat
Toplam	4 saat
Uygulama	
Etkinlik	Süre
Ders 1	2 saat
Ders 2	2 saat
Ders 3	2 saat
Ders 4	2 saat
Toplam	8 saat
Uygulama Sonrası	
Etkinlik	Süre
Son Test 1 (Yazılı Sınav)	2 saat
Son Test 2 (Uygulama Sınavı)	1 saat
Motivasyon Son Test	1 saat
Toplam	4 saat

Araştırmada, uygulama kapsamında ele alınacak konuları belirlemek ve uygulamalar için ayrılacak süreyi tespit edebilmek amacı ile 10 kişilik küçük bir gruba görselleştirme araçları yardımıyla uygulamalar yaptırmak suretiyle pilot çalışma yapılmıştır.

Yapılan pilot çalışmadan sonra çalışma grubunda yer alan öğrenciler, algoritma geliştirme ile ilgili ön bilgi testi (Ek 1) sonuçları dikkate alınarak öğrenci başarısı açısından iki eş deney gruba ayrılmıştır.

Araştırmada Cooper ve diğerleri (2006)'nin çalışmalarında kullandıkları sınıflandırmada yer alan algoritmayı hikayeleştiren araç ve akış şeması modelli araçlardan yararlanılmıştır. Deney gruplarından birine (Grup 1) akış şeması modelli araç, diğerine ise (Grup 2) algoritmayı hikayeleştiren araç ile uygulamalar yapılmıştır.

Araştırma toplam 16 hafta sürmüş olup haftada 2'şer saat etkinlikler yapılmıştır. Çalışmada yer alan her iki grubun derslerine araştırmacı girmiştir. Bu iki gruba görselleştirme aracı kullanılarak programlamanın temel kavramları, değer atama, döngü ve koşul ile ilgili uygulamalar yaptırılmıştır. Yapılan uygulamalara Ek 2'de yer verilmiştir. Yapılan uygulamalar sonrasında; araştırmada kullanılan iki yardımcı aracın algoritma geliştirme başarısına olan etkilerini karşılaştırabilmek amacı ile son test (Ek 3, Ek 4) sonuçlarından yararlanılmıştır. Son test ile elde edilen sonuçların güvenirliğini arttırmak amacı ile biri kağıt üzerinde(Ek 3) diğeri uygulama sınavı (Ek 4) olmak üzere iki sınav yapılmış ve bu sınav sonuçlarının ortalamaları kullanılmıştır.

Çalışmada ayrıca görselleştirme araçlarının öğrencilerin motivasyonuna olan etkisini belirleyebilmek amacıyla Özerbaş (2003) tarafından geliştirilen motivasyon testi (Ek5) uygulama öncesi ve sonrası iki kez uygulanarak sonuçlar karşılaştırılmıştır.

Uygulama sonrasında öğrencilerin algoritma geliştirme başarıları ile ilişkisi olan derslerin belirlenebilmesi için korelasyon testi yapılmıştır. Daha sonra algoritma geliştirme ile ilişkisi olduğu belirlenen derslerin algoritma geliştirme başarısını ne kadar iyi yordadığı incelenmiştir. Araştırmanın deseni Tablo-2'de gösterilmiştir.

Tablo 2 Araştırma Deseni

Gruplar	Grup 1	Grup 2
N	42	42
Program	Scratch	Fcpro
Algoritma Geliştirme Ön Bilgi Testi	Uygulandı	Uygulandı
Motivasyon Ön Test	Uygulandı	Uygulandı
Son Test 1 (Yazılı Sınav)	Uygulandı	Uygulandı
Son Test 2 (Uygulama Sınavı)	Uygulandı	Uygulandı
Motivasyon Son Test	Uygulandı	Uygulandı

3.2 ÇALIŞMA GRUBU

Araştırmanın çalışma grubunu 2008-2009 eğitim-öğretim yılında Kocaeli ili Körfez ilçesi Sevindikli İlköğretim Okulu 6 ve 7. ve 8. sınıf öğrencileri oluşturmuştur. Uygulama öncesinde yapılan ön bilgi testine (Ek 1) göre çalışma grubunda yer alan öğrenciler 2 eş gruba ayrılmıştır. Çalışma grubu Tablo-3'te gösterilmiştir.

Tablo 3 Çalışma Grubu Öğrencilerinin Sayıları Ve Yüzde Dağılımları

Gruplar	Program	Kişi Sayısı	%
1	Fcpro	42	50
2	Scratch	42	50

3.3 VERİLERİN TOPLANMASI

3.3.1 Veri Toplama Araçları

Araştırmada verilerin toplanması için algoritma geliştirme becerilerini ölçen ön bilgi testi, motivasyon ölçeği, 2 adet son test sınavı ve öğrenci akademik ders notu ortalamaları kullanılmıştır. Araştırmada kullanılan ön bilgi testi, son test ve uygulama sınavının kapsam geçerliliğini sağlamak üzere hedef davranışlar

ile ilgili madde belirtke tablosu hazırlanmış ve dört ayrı bilgisayar öğretmeninden mevcut soruları, bu hedef davranışları ölçüp ölçmedikleri konusunda değerlendirmeleri istenmiştir. Bunun için öğretmenlere madde belirtke tablosu verilerek öğretmenlerden her bir maddeyi ölçen soru numarasını hedef davranışların yanlarına yazmaları istenmiştir.

3.3.1.1 Ön Bilgi Testi

Araştırmacı tarafından geliştirilen ön bilgi testi (Ek 1), farklı görselleştirme araçları kullanacak olan deney gruplarının eş gruplar olup olmadığının belirlenmesi amacı ile kullanılmıştır. Ön bilgi testine ait madde belirtke tablosu Tablo-4'te verilmiştir.

Tablo 4 Ön Bilgi Testi'ne İlişkin Madde Belirtke Tablosu

Ölçülmek İstenen Hedef Davranış	Sorular
Verilen işlemleri değer vererek çözümler	5
Bir problemin çözümlenmesi için gerekli olan işlem sırasını doğru biçimde sıralayabilir.	1, 2
Değişkenlerin veri türlerini doğru biçimde belirleyebilir	3
Verilen bir algoritmada belli bir süre ya da sayıca tekrar edilen bir olayı döngü kullanarak ifade edebilir.	7
Verilen ifadeleri doğru yapan değerleri bulabilir.	8
Belli bir koşulda belli eylemlerin yapıldığı bir olayı koşul kullanarak ifade edebilir.	4
Operatörleri kullanarak verilen işlemleri yapabilir.	6

Ön bilgi testi için yapılan kapsam geçerlilik çalışması sonunda öğretmenlerin değerlendirme sonuçları arasındaki tutarlılığın %87.5 olduğu belirlenmiştir. Ön

bilgi testi, 8 adet sorudan oluşmaktadır. Sorular hazırlanırken, algoritma geliştirme ile ilgili temel kavramlar göz önüne alınmıştır.

3.3.1.2 Motivasyon Ölçeği

Araştırmada Özerbaş (2003) tarafından geliştirilen motivasyon ölçeği (Ek 5) öğrenmeye ilişkin güdülemeyi belirlemek amacıyla uygulanmıştır. Ölçeğin bütünü 15 olumlu, 15 olumsuz olmak üzere likert tipi 30 ifadeden oluşmaktadır. "Motivasyon Ölçeği"nde yer alan her bir madde beşli likert ile değerlendirilmektedir. Elde edilen sonuçlarda 90 puanın üzerindeki puanlar olumlu tutumlara, bu puanın altındaki puanlar ise olumsuz tutumlara yöneliktir. Özerbaş (2003) tarafından motivasyon ölçeğinin geçerlilik ve güvenirlilik çalışması yapılmış olup aracın Cronbach Alpha güvenirlilik değeri 0.88 olarak bulunmuştur. Özerbaş (2003) çalışmasında ayrıca "Motivasyon Ölçeği"nin yapı geçerliliği için faktör analizi çalışması yapmıştır.

3.3.1.3 Son Test 1 (Yazılı Sınav)

Uygulama sonunda, öğrencilerin algoritma geliştirme ile ilgili başarılarını değerlendirmek üzere araştırmacı tarafından geliştirilen bir başarı testi, son test olarak kullanılmıştır (Ek 3). Araştırmada kullanılan son testin hazırlanmasında algoritma geliştirme becerilerinin değerlendirilmesine ilişkin literatür taranmış, Naps ve diğerleri (2003)'nin araştırmalarında programlama öğretiminde kullanılması için önerdikleri Bloom taksonomisine uygun olarak düzenlenen kazanımlardan yararlanılmıştır. Soruların hazırlanmasında ayrıca MEB, Bilişim Teknolojileri 6, 7, 8.sınıf öğretmen kitabındaki alıştırma ve örneklerden yararlanılmıştır (İnce, Şenyüzlü ve Uğur, 2007). Araştırmanın amacı doğrultusunda öğrencilerin verilen bir problemi algoritma kullanarak çözebilme düzeylerinin ölçülmesinde çoktan seçmeli ya da kısa cevaplı sorular yeterli olamamaktadır. Çünkü öğrenciler, bu tür sınavlarda kendilerini özgürce ifade edememektedir. Bu nedenle öğrencilerin farklı çözüm yolları üretmesini sağlayacak bir değerlendirmenin yapılabilmesi açısından yazılı sınav yapılmıştır. Güvenirliği artırmak açısından soru sayısı olabildiğince

çoğaltılmış, sonuçların güvenirliğini artırmak açısından da uygulama sınavı ikinci bir son test olarak kullanılmıştır. Son test sınav sorularına ait madde belirtke tablosu Tablo-5'te verilmiştir.

Tablo 5 Son Test 1 Madde Belirtke Tablosu

Ölçülmek İstenen Hedef Davranış	Sorular
Verilen bir algoritmada değişkenlere değer vererek gerekli işlemleri yapar	7
Bir problemin çözümlenmesi için gerekli olan işlem adımlarını doğru biçimde sıralayabilir.	3
Değişkenlerin veri türlerini doğru biçimde belirleyebilir	5
Bir problemi küçük problemlere ayırarak çözüm yolu geliştirme	4
Verilen bir problemin çözülmesi için belli bir süre ya da sayıca tekrar edilen bir olayı küçük adımlar halinde döngü kullanarak algoritmaya dönüştürebilir.	9,14,15,16
Verilen bir algoritmayı inceleyerek algoritma ile ilgili sorgulama yapabilir.	10
Belli bir koşulda belli eylemlerin yapıldığı bir olayın algoritmasını oluşturabilir.	6,11,12,13
Operatörleri kullanarak verilen işlemleri yapabilir.	8
Algoritma ile ilgili temel kavramları bilir ve bunları bilgi verici bir biçimde ifade edebilir.	1
Verilen bir algoritmayı takip ederek döndüreceği sonucu tahmin eder.	2

Uygulanan son test, algoritma geliştirme ile ilgili 14 açık uçlu sorudan oluşmaktadır. Son testte değişkenlerin kullanımı ile ilgili 3, koşul konusu ile

ilgili 4, döngü konusu ile ilgili 4 soru yer almaktadır. Kağıt üzerinde yapılan sınav 16 sorudan oluşmakta olup 100 üzerinden değerlendirilmiştir. Son testin kapsam geçerliliğini belirlemek için yapılan çalışma sonucunda yapılan değerlendirmede dört bilgisayar öğretmeninin değerlendirme sonuçların arasındaki tutarlılığın % 93.75 olduğu belirlenmiştir.

3.3.1.4 Son Test 2 (Uygulama Sınavı)

Uygulama bittikten sonra araştırmacı tarafından hazırlanan uygulama sınavı, öğrencilerin algoritma geliştirme ile ilgili başarılarını değerlendirmek üzere kullanılmıştır (Ek 4). Bilgisayar laboratuarında gerçekleştirilen sınavda değişkenlerin kullanımı ile ilgili 3, koşul konusu ile ilgili 1, döngü konusu ile ilgili 1 soru olmak üzere 5 soru sorulmuş ve 100 üzerinden değerlendirilmiştir. Uygulama sınavı, çalışmanın güvenirliğini artırmak amacıyla yapılmıştır. Uygulama sınavının kapsam geçerliliğini sağlamak üzere yapılan çalışmada dört bilgisayar öğretmeninin değerlendirme sonuçları arasındaki tutarlılığın % 90 olduğu belirlenmiştir. Uygulama sınavına ait madde belirtke tablosu Tablo-6'da görülmektedir.

Tablo 6 Son Test 2 (Uygulama Sınavı) Madde Belirtke Tablosu

Ölçülmek İstenen Hedef Davranış	Sorular
Verilen bir algoritmada bir eylemin belli bir süre ya da sayıca tekrar edilen bir olayı akış şemasında uygun yere yerleştirir.	5
Belli bir koşulda belli eylemlerin yapıldığı bir olayı akış şemasına aktarır.	4
Değişkenleri kullanarak verilen işlemleri yapabilir.	1,2,3

3.3.1.5 Öğrenci Not Ortalamaları

Çalışma grubunda yer alan öğrencilerin ders not ortalamaları ile ilgili veriler, ilgili ders öğretmenlerinden edinilmiştir.

3.3.2 Kullanılan Materyal

Uygulamalar kapsamında iki farklı görselleştirme aracı kullanılmıştır. Yapılan incelemeler doğrultusunda uygulamalar, kullanımları kolay ara yüz tasarımları ile yeni başlayan öğrenciler için uygun olduğu düşünülen Fcpro ve Scratch görselleştirme araçları ile yürütülmüştür. Araştırma süresince her iki araç kullanılarak yapılan etkinlikler (Ek 2), araştırmacı tarafından hazırlanıp, uygulanmıştır.

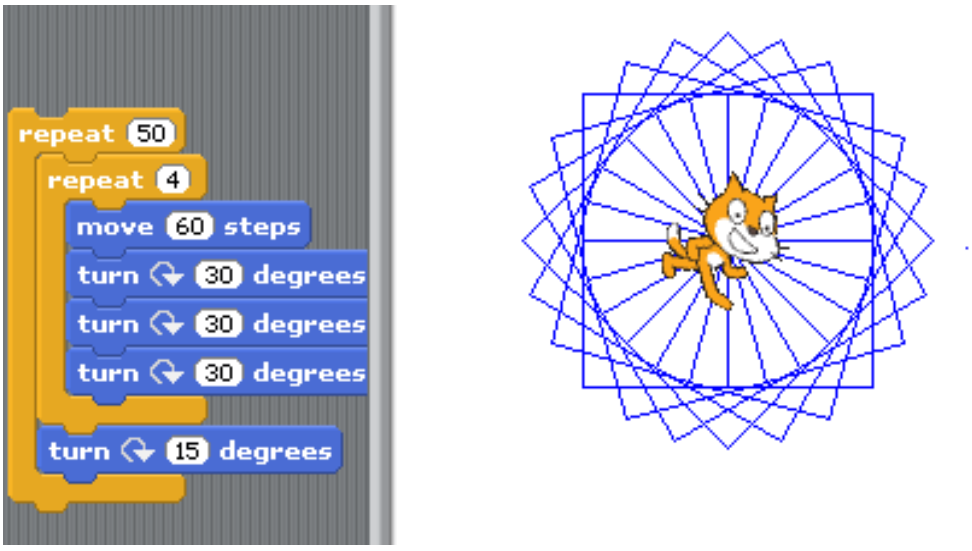
3.3.2.1 Scratch

Araştırma süresince 1. deney grubunda yer alan öğrenciler, ders kapsamında Scratch adlı yazılımı kullanmışlardır. Scratch, öğrencilere programlamayı tanıtmak amacıyla geliştirilmiş bir görsel programlama yazılımıdır. Scratch yardımıyla öğrenciler, sürükle – bırak yöntemiyle oluşturdukları kodların dönütlerini animasyonlar ile görsel biçimde alabilmektedirler. Çocuklar, bu sayede kişisel olarak anlamlı projeler (animasyonlu hikayeler, oyunlar ve etkileşimli resimler gibi) oluşturabilmektedir (Maloney v.d., 2004). Scratch programı indirildikten sonra tüm düzenekler, script yani metin yazısının altına çekilerek oluşturulmaktadır (Malan ve Leitner, 2007). Scratch ile program geliştirmek için grafiksel blokları arka arkaya dizmek yeterlidir. Bloklar yazım hatalarını engellemek için birbirlerine uyumlu olarak tasarlanmışlardır. Program çalışırken yeni bloklar eklenebilmekte ve kodların sonuçları ekranda görüntülenebilmektedir (Resnick, 2007).

Scratch yardımıyla öğrenciler, sürükle – bırak yöntemiyle oluşturdukları kodların dönütlerini animasyonlar ile görsel biçimde alabilmektedirler. Scratch yazılımının ekran görüntüsü Şekil-5'te verilmiştir. Şekil-6'da ise

programlama aracı yardımıyla karakterin programlanarak ekranda verilen kodlara uygun biçimde hareket etmesi sağlanmıştır.

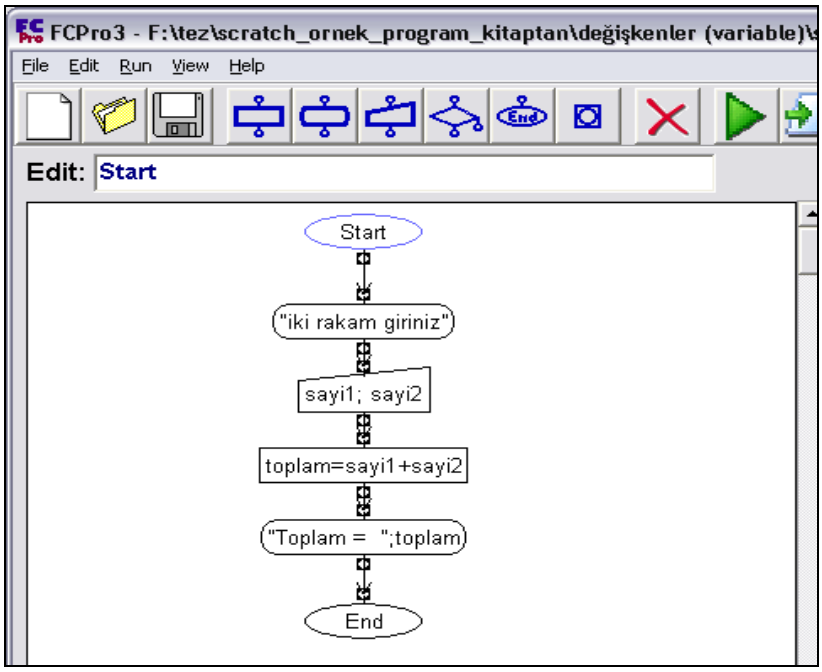
Şekil 5 Scratch Yazılımının Ekran Görüntüsü



Şekil 6 Scratch İle Geliştirilen Bir Algoritma Örneği (19 Mayıs 2008 tarihinde http://www.redware.com/scratch/iteration.html web adresinden edinilmiştir).

3.3.2.2 Fcpro

Araştırma süresince 2. deney grubunda yer alan öğrenciler, ders kapsamında Fcpro aracını kullanmışlardır. Fcpro aracı ise sürükle bırak yöntemi ile kodları bir araya getirerek algoritma oluşturmayı sağlayan bir akış şeması modeli araçtır. Fcpro yazılımının ekran görüntüsü Şekil-7'de verilmiştir.



Şekil 7 Fcpro Aracının Ekran Görüntüsü

Kullanıcılar, Fcpro yazılımını kullanarak, algoritma parçalarını birbirine bağlayıp uygun şekilde dizerek bir algoritma geliştirebilmektedirler. Fcpro aracılığıyla yapılan işlemler sonucunda algoritmanın hangi sonucu döndüreceğini metin tabanlı dönütler aracılığıyla görüntülemektedir.

3.3.3 Uygulama

Her iki yazılımda da algoritma geliştirme ile ilgili aşağıdaki kazanımları vermek üzere uygulamalar hazırlanmıştır.

Kazanımlar

1. Verilen işlemleri değer vererek çözümler
2. Bir problemin çözümlenmesi için gerekli olan işlem sırasını doğru biçimde sıralayabilir.
3. Değişkenlerin veri türlerini doğru biçimde belirleyebilir
4. Verilen bir algoritmada bir eylemin belli bir süre ya da sayıca tekrar edilen bir olayı akış şemasında uygun yere yerleştirir.
5. Verilen ifadeleri doğru yapan değerleri bulabilir.

6. Belli bir koşulda belli eylemlerin yapıldığı bir olayı akış şemasına aktarır.
7. Operatörleri kullanarak verilen işlemleri yapabilir.

Araştırmanın uygulama aşaması toplam 8 saat sürmüştür. Uygulama aşamasında öğrencilere Ek 2'de yer alan konular anlatılarak etkinlikler yaptırılmıştır. 2 ders saatinden biri konu anlatımı, diğeri ise etkinlikler için ayrılmıştır.

3.4 VERİLERİN ÇÖZÜMLENMESİ

Araştırmada öncelikle eş grupların belirlenmesi için ön bilgi testi kullanılmıştır. Araştırmada kullanılan istatiksel analizler aşağıda verilmiştir.

1. Akış şeması modelli araç kullanan öğrencilerle algoritmayı hikayeleştiren araçtan faydalanan öğrencilerin değişkenlerin kullanımı konusundaki başarılarının karşılaştırılması amacıyla bağımsız gruplarda t-testi kullanılmıştır.

2. Algoritmayı hikayeleştiren faydalanan öğrencilerle akış şeması modelli araç kullanan öğrencilerin problemin çözümüne uygun koşul cümlelerinin kullanımı konusundaki başarılarının karşılaştırılması amacıyla bağımsız gruplarda t-testi kullanılmıştır.

3. Algoritmayı hikayeleştiren araçtan faydalanan öğrencilerle akış şeması modelli araç kullanan öğrencilerin problemin çözümüne uygun döngü cümlelerinin kullanımı konusundaki başarılarının karşılaştırılması amacıyla bağımsız gruplarda t-testi kullanılmıştır.

4. Programlama öğretiminde öğrencilerin motivasyon puanlarındaki değişimin kullanılan araca göre anlamlı bir farklılık gösterip göstermediğini belirlemek amacıyla yapılacak istatistiksel işlemler şunlardır: Grupların ön test puanlarını karşılaştırmak amacıyla bağımsız gruplarda t-testi, grupların kendi içinde ön test ve son test puanlarını karşılaştırmak amacıyla bağımlı gruplarda

t-testi, grupların son test puanlarını karşılaştırmak için de bağımsız gruplarda t-testi kullanılmıştır.

5. Öğrencilerin bilişim teknolojileri, matematik, Türkçe ve İngilizce dersindeki akademik başarı ile algoritma geliştirme başarısı arasında ilişki olup olmadığını tespit amacıyla korelasyon testi yapılmıştır.

6. Öğrencilerin bilişim teknolojileri, matematik, Türkçe ve İngilizce dersindeki akademik başarılarının birlikte algoritma geliştirme başarısını anlamlı biçimde yordayıp yordamadığının test edilmesi için regresyon analizi yapılmıştır.

BÖLÜM IV
BULGULAR

Araştırmanın bu bölümünde, yöntem kısmında belirtilen istatistiksel analizler sonucunda elde edilen bulgulara ve bu bulguların yorumlanmasına yer verilmiştir.

4.1 UYGULAMA ÖNCESİ BULGULAR VE YORUMLANMASI

Bu bölümde araştırmaya katılan grupların eş gruplar olup olmadığını belirlemek amacıyla yapılan istatiksel analizler ve bu analizlerin bulgularına yer verilmiştir.

4.1.1 Çalışma Grubunun Ön Bilgi Testi Sonuçları ile İlgili Bulgular

Çalışma gruplarının, uygulama kapsamında işlenen konu hakkındaki ön bilgileri açısından eş gruplar olup olmadıklarını belirlemek amacıyla öncelikle öğrencilerin ön bilgi testi sonuçlarının normal dağılım sergileyip sergilemediklerini belirlenmiştir. Yapılan Shapiro-Wilk testi sonucuna göre gruplar normal dağılım sergilemektedir ($p > .05$). Buna göre ön bilgi testi puanları bağımsız gruplarda t-testi kullanılarak karşılaştırılmış ve sonuçları Tablo-7'de verilmiştir.

Tablo 7 Ön Bilgi Testi Sonuçlarına Yönelik Bağımsız Gruplarda t-Testi Sonuçları

Program	n	x	ss	sd	t	p
Fcpro	42	34.83	22.71	82	.33	.74
Scratch	42	36.36	18.93			

Yapılan bağımsız gruplarda t-testi sonucunda ön bilgi testi sonucunda grup başarıları arasında istatistiksel açıdan anlamlı bir fark bulunmamıştır [$t_{(82)} = .33, p > .05$].

Uygulama öncesi yapılan istatistiksel analizlerin bulguları ışığında grupların uygulama kapsamında işlenen konu bilgisi açısından eş gruplar olduğu belirlenmiştir.

4.1.2 Çalışma Grubunun Motivasyon Ön Test Sonuçları İle İlgili Bulgular

Çalışmanın başında algoritma geliştirme başarısı açısından eş olduğu belirlenen grupların motivasyon açısından da eş grup olup olmadığının test edilebilmesi için öncelikle normal dağılım testi uygulanmıştır. Yapılan Shapiro-Wilk testi sonucuna göre gruplar normal dağılım sergilemektedir ($p > .05$). Grupların motivasyon ön test sonuçları, bağımsız gruplarda t-testi kullanılarak karşılaştırılmıştır. Sonuçlar Tablo-8'de verilmiştir.

Tablo 8 Motivasyon Ön Test Sonuçlarına Yönelik Bağımsız Gruplarda t-Testi Sonuçları

Program	n	x	ss	sd	t	p
Fcpro	42	103.09	12.97	82	1.50	.14
Scratch	42	98.33	16.01			

Yapılan bağımsız gruplarda t-testi sonucunda grupların motivasyon ön test sonuçları karşılaştırıldığında gruplar arasında istatistiksel açıdan anlamlı bir fark bulunmamıştır [$t_{(82)} = 1.50, p > .05$].

4.2 UYGULAMA SONRASI BULGULAR VE YORUMLANMASI

4.2.1 Farklı Görselleştirme Araçları Kullanan Öğrencilerin Değişkenlerin Kullanımı İle İlgili Sorulardaki Başarılarının Karşılaştırılması

Son testte yer alan değişkenlerle ilgili sorular için yapılan Shapiro-Wilk testi sonucuna göre her iki görselleştirme araçlarını kullanan öğrencilerin sınav sonuçlarının normal dağılım sergiledikleri görülmektedir ($p > .05$). Akış şeması

modelli araç kullanan öğrencilerin değişkenler ile ilgili sorulardaki başarılarının algoritmayı hikayeleştiren araçtan faydalanan öğrencilere göre yüksek olup olmadığının belirlenmesi amacıyla, gruplara bağımsız gruplarda t-testi uygulanmış ve sonuçlar Tablo-9'da verilmiştir.

Tablo 9 Değişkenler İle İlgili Sorulara İlişkin Bağımsız Gruplarda t-Testi Sonuçları

SON TEST						
Program	n	x	ss	sd	t	p
Fcpro	42	7.40	3.55	82	1.65	.10
Scratch	42	8.54	2.76			

Tablo-9 incelendiğinde gruplar arasında değişkenler kullanımı konusunda istatistiksel açıdan anlamlı bir fark bulunmadığı ortaya çıkmıştır [$t(82) = 1.65$, $p > .05$].

1.2.2 Farklı Görselleştirme Araçları Kullanan Öğrencilerin Problemin Çözümü İçin Koşul Kullanımı İle İlgili Başarılarının Karşılaştırılması

Son testte yer alan koşul ifadelerinin kullanımı ile ilgili sorular için yapılan Shapiro-Wilk testi sonucuna göre her iki görselleştirme araçlarını kullanan öğrencilerin sonuçlarının normal dağılım sergiledikleri görülmektedir ($p > .05$). Algoritmayı hikayeleştiren araçtan faydalanan öğrencilerin koşul ile ilgili sorulardaki başarılarının akış şeması modelli araç kullanan öğrencilere göre yüksek olup olmadığının belirlenmesi amacıyla gruplara bağımsız gruplarda t-testi uygulanmıştır. Sonuçlar Tablo-10'da verilmiştir.

Tablo 10 Koşul İle İlgili Sorulara İlişkin Bağımsız Gruplarda t-Testi Sonuçları

SON TEST						
Program	n	x	ss	sd	t	p
Fcpro	42	12.54	7.21	82	2.57	.01
Scratch	42	16.28	6.06			

Tablo-10 incelendiğinde son test puanları arasında yapılan bağımsız gruplarda t-testi sonucunda, gruplar arasında koşul kullanımı konusunda algoritmayı hikayeleştiren araçtan faydalanan öğrenciler lehine istatistiksel açıdan anlamlı bir fark bulunduğu ortaya çıkmıştır [$t(82) = 2.57$, $p < .05$].

4.2.3 Farklı Görselleştirme Araçları Kullanan Öğrencilerin Problemin Çözümü İçin Döngü Kullanımı İle İlgili Başarılarının Karşılaştırılması

Öğrencilerin döngü kullanımı ile ilgili son test 1 (yazılı sınav) sonuçlarının normal dağılım durumunun incelenmesi için yapılan Shapiro-Wilk testi sonucuna göre her iki grupta yer alan öğrencilerin döngü kullanımı ile ilgili sonuçlarının normal dağılım sergiledikleri görülmektedir ($p > .05$). Algoritmayı hikayeleştiren araçtan faydalanan öğrencilerin döngü ile ilgili sorulardaki başarılarının akış şeması modelli araç kullanan öğrencilere göre yüksek olup olmadığının belirlenmesi amacıyla gruplara bağımsız gruplarda t-testi uygulanmış ve sonuçlar Tablo-11'de verilmiştir.

Tablo 11 Döngü İle İlgili Sorulara İlişkin Bağımsız Gruplarda t-Testi Sonuçları

SON TEST						
Program	n	x	ss	sd	t	p
Fcpro	42	11.14	5.26	82	2.02	.04
Scratch	42	13.40	4.97			

Tablo-11 incelendiğinde son test sonuçları arasında yapılan bağımsız gruplarda t-testi sonucunda gruplar arasında döngü kullanımı konusunda algoritmayı hikayeleştiren araçtan faydalanan öğrenciler lehine istatistiksel açıdan anlamlı bir fark bulunduğu ortaya çıkmıştır [$t_{(82)} = 2.02$, $p < .05$].

Araştırma sonunda son test 1 (yazılı sınav) sonuçlarının karşılaştırılmasıyla elde edilen sonuçların güvenirliliğini artırmak amacıyla ikinci bir son test olarak uygulama sınavı yapılmıştır. Yapılan uygulama sınavı sonuçları ile son test 1 (yazılı sınav) sonuçları arasında ilişki olup olmadığının belirlenmesi amacıyla her iki sınav sonucuna korelasyon testi uygulanmıştır. Bunun için öncelikle yazılı sınav ve uygulama sınavı sonuçları için normal dağılım testi uygulanmıştır. Yapılan Shapiro-Wilk testi sonucuna göre sonuçların normal dağılım sergilediği görülmektedir ($p > .05$). Yazılı sınav ve uygulama sınav sonuçları arasındaki ilişkinin belirlenebilmesi için yapılan korelasyon testi sonucuna göre gruplar arasında orta düzeyde pozitif ve anlamlı bir ilişki olduğu gözlenmektedir ($r = .45$, $p < .01$). Yapılan korelasyon testi sonuçları, Tablo-12'de görüntülenmektedir.

Tablo 12 Son Test 1 (Yazılı Sınav) Sonuçları İle Son Test 2 (Uygulama Sınavı) Sonuçları Arasında Yapılan Pearson Korelasyon Testi Sonuçları

	Son Test 2 (Uygulama Sınavı)	
	r	p
Son Test 1	.45	.00

Uygulama sınavında yapılan bağımsız gruplarda t-testi ile elde edilen sonuçlar Tablo-13'te gözlenmektedir.

Tablo 13 Uygulama Sınavı Sonuçlarına Yönelik Bağımsız Gruplarda t-Testi

Değişkenler						
Program	N	X	S	sd	t	p
Fcpro	42	45.04	7.86	82	.721	.473
Scratch	42	46.26	7.55			
Koşul						
Program	N	X	S	sd	t	p
Fcpro	42	13.07	3.40	82	4.166	.000
Scratch	42	15.85	2.68			
Döngü						
Program	N	X	S	sd	t	p
Fcpro	42	11.61	3.29	82	7.088	.000
Scratch	42	16.19	2.58			

Tablo-13 incelendiğinde uygulama sınavı sonuçları arasında yapılan bağımsız gruplarda t-testi sonucunda gruplar arasında değişkenler kullanımı konusunda istatistiksel açıdan anlamlı bir fark bulunmadığı ortaya çıkmıştır [$t(82) = 0.72$, $p > .05$].

Koşul kullanımı ve döngü kullanımı konusunda ise Scratch kullanan öğrenciler lehine istatistiksel açıdan anlamlı bir fark bulunduğu ortaya çıkmıştır [$t(82) = 4.17$, $p < .05$], [$t(82) = 7.09$, $p < .05$].

4.2.4 Farklı Görselleştirme Araçları Kullanan Öğrencilerin Motivasyon Ön Test – Son Test Sonuçlarının Karşılaştırılması

4.2.4.1 Grupların Motivasyon Ön Test - Son Test Puanlarının Karşılaştırılması

Motivasyon ön testi açısından eş grupların kendi içinde ön test ve son test puanlarının karşılaştırılması için öncelikle motivasyon son test sonuçlarının normal dağılım sergileyip sergilemediğine bakılmıştır. Yapılan Shapiro-Wilk testi sonucuna göre gruplar normal dağılım sergilemektedir ($p > .05$). Öğrencilerin motivasyon ön test puanları ile motivasyon son test puanları arasındaki ilişkinin belirlenebilmesi için bağımlı gruplarda t-testi yapılmıştır. Yapılan bağımlı gruplarda t-testi sonucunda motivasyon ön test ile motivasyon son test puanları karşılaştırıldığında son test sonuçları lehine anlamlı bir farklılık göstermektedir [$t_{(83)} = 2.54$, $p < .05$]. Yapılan bağımlı gruplarda t-testi sonuçları Tablo-14'te gözlenmektedir.

Tablo 14 Grupların Motivasyon Ön Test ve Son Test Puanlarının Karşılaştırılmasına Yönelik Bağımlı Gruplarda t-Testi Sonuçları

	n	x	ss	sd	t	p
Motivasyon Ön Test	84	100.71	14.68	83	2.54	.01
Motivasyon Son Test	84	105.35	15.76			

Yapılan bağımlı gruplarda t-testi sonucunda motivasyon ön test ve son test puanları karşılaştırıldığında puanlar arasında motivasyon son test sonuçları lehine anlamlı bir farklılık olduğu sonucu ortaya çıkmıştır [$t(83) = 2.54$, $p < .05$].

4.2.4.2 Her İki Grubun Motivasyon Son Test Puanlarının Karşılaştırılması

Grupların motivasyon son test puanlarını karşılaştırmak için bağımsız gruplarda t-testi uygulanmıştır. Sonuçlar Tablo-15'te gösterilmiştir.

Tablo 15 Her İki Grubun Motivasyon Son Test Sonuçlarının Karşılaştırılması İle İlgili Bağımsız Gruplarda t-Testi Sonuçları

Program	n	x	ss	sd	t	p
Fcpro	42	106.14	16.41	82	.45	.65
Scratch	42	104.57	15.24			

Yapılan Bağımsız Grup T testi sonucunda motivasyon son test sonuçlarında gruplar arasında istatistiksel açıdan anlamlı bir fark bulunmadığı ortaya çıkmıştır [$t_{(82)} = .455$, $p > .05$].

4.2.5 Öğrencilerin Bilişim Teknolojileri Matematik, Türkçe ve İngilizce Derslerindeki Akademik Başarıları İle Algoritma Geliştirme Başarıları Arasındaki İlişki

Araştırmanın bu kısmında öğrencilerin programlama başarısı ile ilişkisi olan derslerin belirlenmesinde öğrencilerin yapılan son test sınavları sonucunda aldıkları puanların ortalamaları kullanılmıştır. Tablo-16'da öğrencilerin ders not ortalamaları ile ilgili betimsel istatistik sonuçları görüntülenmektedir.

Tablo 16 Öğrencilerin Not Ortalamalarına İlişkin Betimsel İstatistik Sonuçları

Değişkenler	x	ss	sh
Programlama Son Test Ortalaması	61.10	11.80	1.30
Bilişim Teknolojileri	57.91	10.33	1.13
Türkçe	57.46	15.36	1.68
Matematik	57.00	18.17	1.98
İngilizce	64.14	17.64	1.92
Fen ve Teknoloji	77.93	14.47	1.58
Teknoloji ve Tasarım	74.93	14.34	1.57
Görsel Sanatlar	90.36	6.16	0.67
Beden Eğitimi	86.92	4.94	0.54
Müzik	82.93	12.50	1.36
Din Kültürü	78.58	12.37	1.35

Araştırmada öncelikle yapılan karşılaştırmalar için kullanılacak olan sınav sonuçlarının ve ders notlarının normal dağılım sergileyip sergilemediği incelenmiştir. Yapılan Kolmogorov-Smirnov testi sonuçlarına göre öğrencilerin son test ortalamaları ile bilişim teknolojileri, Türkçe, matematik, İngilizce ve din kültürü not ortalamaları normal dağılım sergilemekte ($p > .05$); teknoloji ve tasarım, beden, müzik, fen ve teknoloji ders not ortalamaları ise normal dağılım sergilememektedir ($p < .05$).

Öğrencilerin algoritma geliştirme son test ortalamaları ile teknoloji ve tasarım, beden, müzik, fen ve teknoloji ders başarıları arasında anlamlı bir ilişki olup olmadığının araştırılması için spearman rho korelasyon testi yapılmıştır. Yapılan korelasyon testi sonuçları Tablo – 17'de gözlenmektedir.

Tablo 17 Programlama Son Test Başarısı Ve Bağımsız Değişkenlere Yönelik Spearman Rho Korelasyon Testi Sonuçları

	Programlama Başarısı (Son Test 2)	
	r	p
Teknoloji ve Tasarım	.11	.33
Beden	-.08	.46
Müzik	.02	.86
Fen ve Teknoloji	.03	.80

Tablo incelendiğinde öğrencilerin son test başarıları ile ilişkisi araştırılan teknoloji ve tasarım, beden, müzik, fen ve teknoloji dersleri arasında anlamlı bir ilişki olmadığı gözlenmiştir ($p > .05$).

Öğrencilerin Bilişim Teknolojileri Türkçe, Matematik, İngilizce ve din kültürü derslerindeki akademik başarıları ile algoritma geliştirme son test ortalamaları arasındaki ilişkinin belirlenebilmesi için pearson korelasyon testi yapılmıştır. Yapılan korelasyon testi sonuçları Tablo-18'de gözlenmektedir.

Tablo 18 Programlama Son Test Başarısı ve Bağımsız Değişkenlere Yönelik Pearson Korelasyon Testi Sonuçları

	Algoritma Geliştirme Son Test Ortalaması	
	r	p
Bilişim Teknolojileri	.55	.00
Türkçe	.66	.00
Matematik	.60	.00
İngilizce	.74	.00
Din Kültürü	.11	.31

Tabloya bakıldığında öğrencilerin bilişim teknolojileri, Türkçe matematik ve İngilizce dersleri ile son test ortalamaları arasında istatistiksel olarak anlamlı ilişki olduğu ($p < .05$), din kültürü dersi ile son test ortalamaları arasında anlamlı bir ilişki olmadığı gözlenmiştir ($p > .31$).

Tabloda öğrencilerin bilişim teknolojileri akademik başarıları ile son test ortalamaları arasında orta düzeyde pozitif ve anlamlı bir ilişki olduğu görülmektedir ($r = .55$, $p < .05$). Determinasyon katsayısı ele alındığında ($r^2 = 0.30$) son test sonuçlarındaki toplam varyansın %30'unun bilişim teknolojilerindeki akademik başarıdan kaynaklandığı söylenebilir.

Tablo incelendiğinde öğrencilerin Türkçe dersindeki akademik başarıları ile son test sonuçları arasında orta düzeyde pozitif ve anlamlı bir ilişki olduğu görülmektedir ($r = .66$, $p < .05$). Determinasyon katsayısı ele alındığında ($r^2 = 0.44$) son test sonucundaki toplam varyansın %44'ünün Türkçe dersindeki akademik başarıdan kaynaklandığı söylenebilir.

Tablo incelendiğinde ayrıca öğrencilerin matematik dersindeki akademik başarıları ile son test sonuçları arasında orta düzeyde pozitif ve anlamlı bir ilişki olduğu görülmektedir ($r = .60$, $p < .05$). Determinasyon katsayısı ele alındığında ($r^2 = 0.36$) son test sonucundaki toplam varyansın %36'sının matematik dersindeki akademik başarıdan kaynaklandığı söylenebilir.

Tabloda öğrencilerin İngilizce dersindeki akademik başarıları ile son test sonuçları arasında yüksek düzeyde pozitif ve anlamlı bir ilişki olduğu görülmektedir ($r = .74$, $p < .05$). Determinasyon katsayısı ele alındığında ($r^2 = 0.55$) son testteki toplam varyansın %55'inin İngilizce dersindeki akademik başarıdan kaynaklandığı söylenebilir.

4.2.6 Öğrencilerin Algoritma Geliştirme Başarılarını Yordayan Dersler

Programlama başarısını yordayan değişkenleri belirleyebilmek için çoklu regresyon analizi yapılmıştır. Yapılan regresyon analizi sonuçlarına göre algoritma geliştirme başarısını yordayan değişkenler Tablo-19'da gösterilmektedir.

Tablo 19 Programlama Başarısını Yordayan Değişkenler

Değişkenler	β	t	p
Sabit	21.57	4.30	.00
Bilişim Teknolojileri	.16	1.51	.13
Türkçe	.16	1.68	.10
Matematik	-.02	-.22	.82
İngilizce	.34	3.84	.00
$R = 0.76$, $R^2 = .57$; $F = 26.77$, $p = 0.00$			

Algoritma geliştirme başarısını yordadığı düşünülen bilişim teknolojileri, Türkçe, matematik ve İngilizce ders başarıları birlikte, öğrencilerin algoritma geliştirme başarısı ile pozitif ve orta düzeyde anlamlı bir ilişki vermektedir ($R = 0.76$, $R^2 = 0.57$, $p < .01$). Adı geçen değişkenlerin tamamı, algoritma geliştirme başarısı üzerindeki toplam varyansın yaklaşık % 57'sini açıklamaktadır.

Yordayıcı değişkenlerin standardize edilmiş regresyon katsayısı (β)'na göre, sıralanışı; İngilizce (.34), Türkçe (.16), bilişim teknolojileri (.16) ve matematiktir (-.02) şeklindedir.

Regresyon katsayısının anlamlılığına ilişkin t testi incelendiğinde ise Öğrencilerin İngilizce dersindeki başarılarının programlama başarısı üzerinde etkili olduğu görülmektedir. İngilizce başarısı, öğrencilerin İngilizce komutların kullanıldığı bu tür görselleştirme araçlarını kullanma becerilerini

artırdığı için etkili olmaktadır. Regresyon analizi sonuçlarına göre programlamanın yordanmasına ilişkin regresyon eşitliği aşağıda verilmiştir.

PROGRAMLAMA BAŞARISI = 21.57 + 0.16 BİLİŞİM TEKNOLOJİLERİ + 0.16 TÜRKÇE + .34 İNGİLİZCE - .02 MATEMATİK

BÖLÜM V
SONUÇ, TARTIŞMA VE ÖNERİLER

5.1 SONUÇ VE TARTIŞMA

Bu çalışmada, algoritma geliştirmede görselleştirme araçları kullanmanın, öğrenci başarı ve motivasyonuna olan etkisi belirlenmeye çalışılmıştır. Çalışmada ayrıca öğrencilerin programlama başarılarının yordanmasına katkı sağlaması açısından algoritma geliştirme başarıları ile ilişkisi olan dersler incelenmiştir. Elde edilen bulgular doğrultusunda aşağıdaki sonuçlara ulaşılmıştır.

Araştırmanın birinci hipotezinde akış şeması modelli araç kullanan öğrenciler ile algoritmayı hikayeleştiren araçtan faydalanan öğrencilerin, değişkenlerin kullanımı konusundaki başarıları arasında akış şeması modelli araç kullanan öğrenciler lehine anlamlı fark olduğu öngörülmektedir. Bu hipotezin doğruluğunun test edilebilmesi amacıyla algoritmayı hikayeleştiren araçtan faydalanan grup ile akış şeması modelli araç kullanan grubun değişkenlerin kullanımı konusunda başarıları karşılaştırılmıştır. Araştırma sonuçlarına göre, her iki görsel aracı kullanan ilköğretim öğrencilerinin değişkenlerin kullanımı konusundaki başarıları arasında farklılık yoktur. Ancak bu konuda yapılan çalışmalar, programlama öğretimine başlangıç aşamasında temel kavramların öğretilmesinde akış şeması modelli araçların kullanılmasının faydalı olabileceğini göstermektedir (Baldwin ve Kuljis, 2000; Crews ve Ziegler, 1998). Yapılan bu çalışmalar, üniversite öğrencilerine yöneliktir. Bu durumda literatürde incelenen çalışmalar ile araştırmanın hedef kitlesi ve dolayısıyla ön bilgileri birbirinden farklıdır. Araştırma sonuçları ile literatürde incelenen çalışmalardan elde edilen sonuçlar arasındaki farklılığın bu nedenden kaynaklandığı düşünülebilir. Hedef kitlenin farklı olması nedeniyle literatürde incelenen araştırmalarda programlama ile ilgili daha karmaşık etkinlik ve örnekler yer almaktadır. Bu durum, akış şeması tabanlı araçların, değişkenlerin kullanımı ile ilgili ileri seviyede örneklerin pekiştirilmesinde daha faydalı olabileceği düşüncesini akla getirmektedir.

Ortaya çıkan sonucun, kullanılan görselleştirme aracının yapısından da kaynaklanmış olabileceği düşünülmektedir. Zira araştırmada kullanılan Fcpro aracı komutların çalıştırılarak sonuçlarının izlenmesi mantığıyla çalışmaktadır; Scratch aracı ise oluşturulan bir nesne yardımıyla komutların görselleştirilmesini sağlamaktadır. Araştırmada uygulamalar esnasında yapılan gözlemler ışığında bu iki yazılım, değişkenler kullanımı konusu ile ilgili olarak karşılaştırıldığında Fcpro programı görselleştirme açısından yetersiz olmasına karşın değişkenlerin oluşturulması ve kullanılması açısından daha avantajlı olduğu görülmektedir. Fcpro aracında oluşturulan tüm değişkenlerin isimleri ve değişkenlerle ilgili yapılan işlemler aynı anda ekranda görülebilmektedir. Scratch aracında ise değişkenler, aritmetik işlemler, koşul ve döngü cümleleri ayrı sekmelerde gerçekleştirilmektedir. Bu durum öğrencilerin değişkenlerle ilgili işlem yaparken değişkenler ve algoritmanın bütününü aynı anda görememesine neden olmaktadır. Bu durum, değişken sayısının az olduğu basit örneklerde fark yaratmamaktadır. Ancak değişken sayısının daha fazla olduğu örneklerde; öğrencilerin, algoritma içerisindeki değişkenlerin isimlerini doğru ve tutarlı biçimde kullanabilme konusunda karmaşa yaşayabileceği düşünülmektedir. Değişkenler konusu ile ilgili daha ileri düzeyde etkinlikler ve örneklerde görselleştirmenin ve bütünü görebilmenin önemi artacağından daha farklı bir sonuç ortaya çıkabileceği düşünülmektedir. Araştırmada elde edilen bulgulardan hareketle değişkenlerin kullanımı konusunda basit etkinlik ve örneklerde her iki aracın da faydalı olabileceği sonucu çıkarılabilir.

Araştırmada programlama öğretiminde öğrencilerin koşul ve döngü konularıyla ilgili başarıları ayrı ayrı incelenmiştir; zira döngü, ilköğretim öğrencilerin yabancı oldukları bir kavramdır. Bu durumun araştırma sonuçlarını da etkileyebileceği düşünülerek koşul ve döngü konuları ayrı ayrı ele alınmıştır. Literatürde ulaşılan kaynaklarda ise her iki konu birlikte ele alınarak öğrenci başarılarının karşılaştırıldığı gözlenmiştir. Bu nedenle araştırmanın ikinci ve üçüncü hipotezlerinin test edilmesi sonucu elde edilen bulgular bir araya getirilmiş; literatürden elde edilen bilgiler ışığında yorumlanmıştır.

Bu kısımda araştırmanın ikinci ve üçüncü hipotezinin test edilmesi sonucunda elde edilen bulgular değerlendirilecektir. Araştırmanın ikinci hipotezinde akış şeması modelli araç kullanan öğrenciler ile algoritmayı hikayeleştiren araçtan faydalanan öğrencilerin, koşul kullanımı konusundaki başarıları arasında algoritmayı hikayeleştiren araçtan faydalanan öğrenciler lehine anlamlı fark olduğu savunulmaktadır. Araştırma sonucunda yapılan karşılaştırmalar sonucunda koşul kullanımı konusunda algoritmayı hikayeleştiren araç lehine anlamlı bir fark bulunmuş ve 2. hipotez doğrulanmıştır. Araştırmanın üçüncü hipotezinde ise akış şeması modelli araç kullanan öğrenciler ile algoritmayı hikayeleştiren araçtan faydalanan öğrencilerin, döngü kullanımı konusundaki başarıları arasında algoritmayı hikayeleştiren araçtan faydalanan öğrenciler lehine anlamlı fark olduğu savunulmaktadır. Araştırma sonucunda döngü kullanımı konusunda algoritmayı hikayeleştiren araç lehine anlamlı bir fark bulunmuştur. Bu sonuca göre 3. hipotez de doğrulanmıştır. Araştırmanın 2. ve 3. hipotezinden elde edilen sonuçlar; yazılımda görsellik, animasyon ve etkileşim kullanımının öğrenci başarısı üzerindeki etkisini vurgulayan araştırmaları desteklemektedir. (Arabacıoğlu, Bülbül ve Filiz, 2007; Kelleher, Pausch ve Kiesler, 2007; Gültekin, 2006; Ramadhan, 2000). Literatürde yer alan bu çalışmalar, üniversite düzeyindedir. Araştırmada elde edilen sonuçlar, literatürde incelenen araştırma sonuçları ile örtüşmektedir. Bu durum, programlama öğretiminde görselliğin kullanımının ilköğretim öğrencilerinin de koşul ve döngü kullanımı konusundaki başarılarını olumlu yönde etkilediğini göstermektedir. Araştırma sonucunda döngü ve koşul ile ilgili sorularda algoritmayı hikayeleştiren araçtan faydalanan öğrenciler lehine oluşan farkın nedeninin, koşul ve döngü ile ilgili ifadelerin animasyon kullanılarak görselleştirilmesinin öğrencinin ilgisini çekerek, öğrenmesini ve akılda tutmasını kolaylaştırdığından kaynaklandığı düşünülebilir. Bravo, Marcelino, Games, Esteves ve Mendes (2005), bu konuya değinmişler, algoritmayı bir karakter yardımıyla oluşturmaya yardımcı olan araçların, öğrencilerin programları anlamaları ve analiz etmeleri konusunda yardımcı olabileceğini belirtmişlerdir. Bu durumda ilköğretim öğrencilerine algoritma geliştirme ile ilgili eğitim verilirken koşul ve döngü gibi konularda görselliğin

daha fazla ön planda olduğu bir yardımcı araç kullanımının öğrencilerin konuyu öğrenmelerini kolaylaştırdığı söylenebilir. Ayrıca araştırmada değişkenlerin kullanımı konusunda fark çıkmazken, koşul ve döngü konularında fark ortaya çıkması sonucu, Lahtinen, Ahoniemi ve Salo (2007)'de yaptıkları araştırmadan elde ettikleri sonuçlarla örtüşmektedir. Araştırmacılar, araştırmalarında problemlerin karmaşıklığı arttıkça görselleştirmeye daha çok ihtiyaç duyulduğu sonucuna ulaşmışlardır.

Araştırmanın 4. hipotezinde algoritmayı hikayeleştiren araçtan faydalanan öğrencilerle, akış şeması modelli araç kullanan öğrencilerin uygulama sonundaki motivasyonları arasında algoritmayı hikayeleştiren araçtan faydalanan öğrenciler lehine anlamlı bir fark olduğu savunulmaktadır. Yapılan istatistiksel karşılaştırmalar sonucunda grupların motivasyon son test puanları arasında istatistiksel açıdan anlamlı bir farklılık görülmemiştir. Bu sonuçlara göre 4. hipotez doğrulanamamıştır. Oysa konu ile ilgili yapılan araştırmalar, yazılımlarda grafik ve animasyon kullanımının öğrencilerin derslere olan ilgisini artırmada daha etkili olduklarını göstermektedir (Bishop-Clark, Courte ve Howard, 2007; Kelleher, Pausch ve Kiesler, 2007; Brusilovsky ve Spring, 2004; Lin ve Zhang, 2003; Ramadhan, 2000). Bu nedenle algoritmayı hikayeleştiren aracın, akış şeması modelli araca göre öğrencilerin motivasyonunu artırma konusunda daha fazla etkiye sahip olacağı düşünülmüştür. Zira algoritmayı hikayeleştiren araçlar daha görsel, ilköğretim öğrencileri için daha çekicidir. Ancak araştırma sonucunda motivasyon son test sonuçları arasında beklenen fark çıkmamıştır. Araştırmada grupların motivasyon ön test ve motivasyon son test puanları karşılaştırıldığında ise son test sonuçları lehine anlamlı fark olduğu ve bu farkın çok büyük olduğu gözlenmektedir. Araştırmada elde edilen sonuçların, her iki grupta yer alan öğrenciler, daha önce bu tür araçlarla etkinlik yapmadığından, kendileri için yeni bir görselleştirme aracının sunulmasının onların ilgisini çekmesinden kaynaklandığı söylenebilir. Bu görsel araçların, daha uzun süre kullanılması halinde öğrenciler için görselliğin öneminin artabileceği, bu durumun onların motivasyonlarını etkileyebileceği düşünülmektedir.

Araştırmanın 5. hipotezinde öğrencilerin bilişim teknolojileri, matematik, Türkçe ve İngilizce derslerindeki akademik başarıları ile algoritma geliştirme başarıları arasında pozitif yönde anlamlı bir ilişki olduğu öne sürülmektedir. Yapılan karşılaştırmalar sonucunda öğrencilerin algoritma geliştirme başarıları ile bilişim teknolojileri, matematik, Türkçe ve İngilizce ders başarıları arasında istatistiksel olarak pozitif ve anlamlı bir ilişki olduğu görülmektedir. Bu sonuçlara göre 5. hipotez doğrulanmıştır. Araştırma sonucunda öğrencilerin bilişim teknolojileri ders başarıları ile algoritma geliştirme başarıları arasında anlamlı bir ilişki olması sonucu bu konuda yapılan araştırmaları desteklemektedir (Leeper ve Silver, 1982; Petersen ve Howe, 1979). Araştırmacılar, öğrencilerin bilgisayar deneyimlerinin programlama başarıları üzerinde etkili olduğunu göstermektedirler. Öğrencilerin bilgisayar deneyimleri, programları daha etkili kullanabilmesini, konuyu daha iyi anlamasını ve uygulama sınavında daha başarılı olmalarını sağlamaktadır. Ancak, araştırma sonucunda bilişim teknolojileri ders başarısı ile programlama başarısı arasındaki ilişkinin orta düzeyde çıkması ilginç bir sonuçtur. Bu sonuç göstermektedir ki bilgisayar başarısı algoritma geliştirme başarısı üzerinde etkili olmakla birlikte çok fazla paya sahip değildir. Bu da bilişim teknolojileri ders içeriğinden kaynaklanmaktadır. Araştırmada elde edilen bu sonuç, öğrencilerin bilişim teknolojileri ile ilgili konularla ilgili çok fazla bilgi sahibi olmadan da algoritma geliştirme konusunda başarılı olabileceğini göstermektedir. Araştırmada elde edilen diğer bir sonuca göre, ilköğretim öğrencilerinin matematik başarıları, algoritma geliştirme başarılarını etkileyen faktörlerden biri olarak karşımıza çıkmaktadır. Bu sonuç, konu ile ilgili daha önce yapılan çalışmalarla paralellik göstermektedir. Matematik başarısının programlama başarısı üzerindeki etkisini araştıran ve üniversite düzeyinde yapılan çalışmalarda öğrencilerin matematik ve programlama başarıları arasında anlamlı ve pozitif ilişki bulunmuştur (Erdoğan, 2005; Byrnes ve Lyons, 2001. Aktaran: Jones ve Burnettt, 2008; Soloway, Lochhead, ve Clement, 1982. Aktaran: Reed ve Burton, 1988; Webb, 1985; Fletcher, 1984; Nowaczyk, 1983; Pea ve Kurland, 1983). Matematik başarısının algoritma

başarısı üzerindeki etkisi, ilköğretim düzeyinde algoritma geliştirme konusunda verilen eğitimde öğrencilerin basit matematiksel işlem gerektiren soruları anlama, yorumlama ve sorulara uygun çözüm önerileri geliştirmeye odaklanmaları ve yapılan değerlendirme sonuçları üzerinde öğrencilerin bu tür becerilerinin de etkili olması ile açıklanabilir. Araştırmada ortaya çıkan bir diğer sonuç olan, algoritma geliştirme başarısı ile Türkçe başarısı arasındaki ilişki ise algoritma geliştirme ile ilgili soruların öğrencilerin okuma, anlama ve yorumlama yeteneği gerektirmesi, öğrencilerin Türkçe başarılarının soruları daha iyi anlayarak yorumlamalarına katkı sağlamasından kaynaklandığı düşünülebilir. Türkçenin ana dil olması bu konuda önemli bir etkendir. Konuyla ilgili yapılan araştırmalar, öğrencilerin kendi anadillerinde verilen eğitimde elde ettikleri başarının programlama ile ilgili konuları anlama ve yorum yapma konusundaki başarılarını etkilediğini göstermektedir. Nowaczyk (1983)'ın, bilgisayar programlama başarısına etki eden faktörleri belirlemeye yönelik yaptığı çalışmanın sonuçları, öğrencilerin programlama başarıları ile ana dilleri olan İngilizce derslerindeki başarıları arasında önemli bir ilişki olduğu sonucunu ortaya çıkarmıştır. Leeper ve Silver (1982) ise İngilizce ve lisede alınan yabancı dil ile programlama başarısı arasındaki ilişkiyi araştırdıkları çalışmanın sonucunda İngilizce ile programlama başarısı arasında anlamlı bir ilişki olduğunu bulmuşlardır. Ancak Jones ve Burnett (2008), yaptıkları araştırmada farklı bir sonuca ulaşmışlardır. Öğrencilerin ana dili olan İngilizce veya yabancı dil sonuçları ve programlama başarısı arasında ilişki bulunamamıştır (Jones ve Burnettt, 2008). Araştırmada elde edilen bir diğer sonuç, öğrencilerin İngilizce başarıları ile programlama başarıları arasında anlamlı bir ilişki olduğunu göstermektedir. Araştırma sonucunda elde edilen bulgularda İngilizce ve algoritma geliştirme başarısı arasındaki ilişki, algoritma geliştirmede kullanılan komutların çoğunlukla İngilizce olması ve İngilizce bilen öğrencilerin bu komutları daha rahat anlayabilmeleri ile açıklanabilir. Araştırma sonuçları, İngilizce başarısının, öğrencilerin programlama başarısını artırdığını göstermemektedir. Araştırma sonuçları, öğrencilerin İngilizce başarılarının görselleştirme araçlarını kullanmalarını kolaylaştırdığını göstermektedir.

Araştırmanın 6. hipotezinde öğrencilerin bilişim teknolojileri, matematik, Türkçe ve İngilizce derslerindeki akademik başarıları, algoritma geliştirme başarılarını anlamlı bir şekilde yordamakta olduğu öne sürülmektedir. Araştırmanın sonucunda, öğrencilerin algoritma geliştirme başarısını etkileyen bilişim teknolojileri, matematik, Türkçe ve İngilizce ders başarılarının birlikte algoritma geliştirme başarısını anlamlı biçimde açıkladıkları görülmektedir. Buna göre 6. hipotez doğrulanmaktadır. Araştırmada öğrencilerin akademik başarılarının programlama başarısına katkısı incelendiğinde bu derslerin önem sırası; İngilizce, Türkçe, Bilişim teknolojileri ve matematik şeklindedir. Araştırma sonucuna göre İngilizce başarısı, öğrencilerin görselleştirme araçlarını kullanmalarını kolaylaştırdığı için etkili olmaktadır. Bu bakımdan İngilizce ders başarıları komutları ve yönergeleri İngilizce hazırlanmış olan görselleştirme araçlarının kullanılması konusunda dikkate alınması gereken önemli bir faktör olarak karşımıza çıkmaktadır. Bu durum, İngilizceyi iyi olan öğrencilerin görselleştirme araçlarında yer alan komutları daha iyi anlayarak etkili biçimde kullandıkları şeklinde yorumlanabilir.

Sonuç olarak programlama öğretiminde döngü ve koşul gibi temel kavramların öğretilmesinde algoritmayı hikayeleştiren aracın kullanımı, akış şeması modelli araç kullanımına göre öğrenci başarısı üzerinde daha etkilidir. Araştırma sonuçları ile daha önce yapılan araştırma sonuçları birlikte yorumlanırsa üniversite düzeyinde matematik başarısının, ilköğretimde de Türkçe başarısının daha çok önem kazandığını göstermektedir.

Hipotezlerle sınanmış olmamakla birlikte araştırmada ortaya çıkan bir diğer sonuç ise, kağıt üzerinde ve uygulamalı sınav sonuçları değişken, koşul ve döngü ile ilgili sonuçlar açısından karşılaştırıldığında; uygulama sınavı sonuçlarında, araştırmada yer alan iki grup arasında daha büyük fark çıkmış olmasıdır. Uygulama sınavı sonucunda oluşan bu farklılık, öğrencilerin derste uygulama yaparak öğrendikleri bilgilerini yine uygulama ile yapılan değerlendirme sırasında daha iyi kullanabildikleri şeklinde açıklanabilir. Her

iki yazılım da öğrencilerin motivasyonunun artmasında benzer etkiye sahiptir. Ayrıca elde edilen sonuçlar, öğrencilerin bilişim teknolojileri, matematik, Türkçe ve İngilizce derslerindeki akademik başarılarının algoritma geliştirme başarısı üzerinde olumlu etkisinin bulunduğunu göstermektedir.

5.2 ÖNERİLER

Programlama öğretiminde kullanılan yöntemlerin değiştirilmesi, ilköğretim öğrencilerine hitap edebilecek araç ve yöntemlerin kullanılması önemlidir. Araştırmalar, ilköğretim seviyesindeki öğrencilere programlama öğretebilmek için farklı yöntemlerin kullanılmasının öğrenci başarı ve motivasyonunu artırdığını göstermektedir. Bu nedenle programlama dersi veren öğretmenler, başarı ve motivasyonu artırabilecek yöntemler hakkında bilgilendirilmelidirler.

Eğitimcilerin, programlama öğretiminde görselleştirme araçlarını kullanırken, konuya uygun araç seçiminin önemli olduğunu göz önünde bulundurmaları ve dersten önce uygun aracın belirlenmesi ve derste hangi örnekler üzerinde durulacağı ile ilgili ön çalışma yapmaları gerekmektedir.

Araştırmada ayrıca öğrencilerin akademik başarılarının programlama başarıları üzerinde etkili olduğu sonucu ortaya çıkmıştır. Bu bakımdan programlama konusunda başarılı olan öğrenciler, ilgili bölümlere ve mesleklere yönlendirilirken akademik ders başarıları dikkate alınabilir. Öğrencilerin akademik başarıları incelenerek programlama ile ilgili bölüm ve mesleklerde başarılı olup olmayacağı hakkında öngörüde bulunulabilir.

Araştırma kapsamında program görselleştiren araçlar, öğrenci başarısı açısından karşılaştırılmıştır. Daha sonra yapılacak araştırmalarda program görselleştirme araçlarında yer alan farklı özelliklerin nasıl kullanılması gerektiği ve görselleştirme araçları kullanırken hangi yöntem ve tekniklerin kullanılması gerektiği ile ilgili çalışmaların yapılmasının faydalı olabileceği söylenebilir.

KAYNAKÇA

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 web adresinden 4 Eylül 2007 tarihinde edinilmiştir.

Baldwin, L.P. ve Kuljis, J. (2000). Visualization Techniques For Learning and Teaching Programming. *Journal of Computing and Information Technology* *8*(4), 285–291. http://cit.zesoi.fer.hr/downloadPaper.php?paper=305 web adresinden 9 Eylül 2008 tarihinde edinilmiştir.

Begel, A. (1997). *Bongo: A Kids' Programming Environment for Creating Video Games on the Web.* Master's Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science http://research.microsoft.com/~abegel/mit/begel-meng-thesis.pdf web adresinden 4 Ekim 2007 tarihinde edinilmiştir.

Bergin, J., Brodlie, K., Goldweber, M., Jiménez-Peris, R., Khuri, S., Patiño-Martínez, M. ve diğerleri (1996). An overview of visualization: its use and design**.** *Proceedings of the 1st conference on Integrating technology into computer science education* (s**.**192-200). Barcelona, Spain. http://lsd.ls.fi.upm.es/lsd/papers/1996/iticse96-wg.pdf web adresinden 17 Aralık 2007 tarihinde edinilmiştir.

Bishop-Clark, C., Courte, J., ve 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.

Bojic, C. ve Greasley, A. (1999). Can Computer-Based Testing Achieve Quality and Efficiency in Assessment?. *International Journal of Educational Technology*, *1*(1). http://www.outreach.uiuc.edu/ijet/v1n1/v1n1articles.html web adresinden 19 Mart 2009 tarihinde edinilmiştir.

Brusilovsky, P. ve Spring, M. (2004). Adaptive, Engaging, and Explanatory Visualization in a C Programming Course. *Proceedings of ED-MEDIA'2004 - World Conference on Educational Multimedia, Hypermedia and Telecommunications* (s. 1264-1271). Lugano, Switzerland.

Chen, T., ve Sobh T., (2001). A Tool For Data Structure Visualization and User-Defined Algorithm Animation, *31st ASEE IEEE Frontiers in Education Conference*. Reno, NV.

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

Chung, C. (1988). Correlates of problem-solving in programming. *CUHK Education Journal* *16*(2) http://sunzi1.lib.hku.hk/hkjo/view/33/3300413.pdf web adresinden 13 Temmuz 2008 tarihinde edinilmiştir.

Clariana, R. ve Wallace, P. (2002). Paper–based versus computer–based assessment: key factors associated with the test mode effect. *British Journal of Educational Technology, 33*, 593-602.

Cooper, S., Dann, W. ve 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/TeachingObjects-firstInIntroductoryComputerScience.pdf web adresinden 20 Eylül 2007 tarihinde edinilmiştir.

Cooper, S., Dann, W. ve Pausch, R. (2003). Using animated 3D graphics to prepare novices for CS1. *Computer Science Education*, *13*(1) 3-30 http://www.sju.edu/~scooper/alice/scooper_csej.pdf web adresinden 1 Eylül 2008 tarihinde edinilmiştir.

Cooper, S., Powers, K., McNally, M., Goldman, K.J. Proulx, V. Carlisle ve M. (2006). Tools for Teaching Introductory Programming: What Works? *in 37th SIGCSE Technical Symposium on Computer Science Education*, 2006, (s.560-561). Housten, Texas, USA. http://dsys.cse.wustl.edu/resources/papers/gross-sigcse-2006.pdf web adresinden 26 Eylül 2008 tarihinde edinilmiştir.

Crews, T. ve Ziegler, U. (1998). The Flowchart Interpreter for introductory programming course. *Frontiers in Education Conference*. Tepme Mission Palms Hotel, Tempe, Arizona. http://fie-conference.org/fie98/papers/1107.pdf web adresinden 30 Mayıs 2009 tarihinde edinilmiştir.

DeAngleis, S. (2000). Equivalency of Computer-Based-and-Pencil Testing. *Journal of Allied Health*, *29*(3). 161-164.

Dunican, E. (2002). Making the analogy: Alternative delivery techniques for first year programming courses. In J. Kuljis, L. Baldwin ve R. Scoble (Eds). *Proceedings from the 14th Workshop of the Psychology of Programming Interest Group*. (s.89-99). Brunel University.

http://ppig.org/papers/14th-dunican.pdf web adresinde 30 Mayıs 2009 tarihinde edinilmiştir.

Erdoğan, B. (2005). Programlama başarısı ile akademik başarı, genel yetenek, bilgisayara karşı tutum, cinsiyet ve lise türü arasındaki ilişkilerin incelenmesi. Yüksek Lisans Tezi, Marmara Üniversitesi Eğitim Bilimleri Enstitüsü.

Felder, R.M. ve Silverman, L. (1988). Learning styles and teaching styles in engineering education. *Engineering Education, 78* (7), 674-681.

Fletcher, S.H. (1984). *Cognitive Abitities & Computer Programming* [Özet]. (ERIC Document Reproduction Service No. ED259700)

Garner, S. (2003). Learning Resources and Tools to Aid Novices Learn Programming. Proceedings of *Informing Science ve Information Technology Education Joint Conference* (s. 213-222). Pori, Finland. http://proceedings.informingscience.org/IS2003Proceedings/docs/036Garne.pdf web adresinden 10 Eylül 2008 tarihinde edinilmiştir.

Grissom, S., McNally, M. F., ve Naps, T. (2003). Algorithm visualization in CS education: comparing levels of student engagement, *2003 ACM Yazılım Görselleştirme Sempozyumu*, San Diego, California.

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

Hagan, D., Sheard, J. ve 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.

Hansen, S., Schrimpsher, D. ve 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 web adresinden 30 mayıs 2009 tarihinde edinilmiştir.

Hu, M. (2004). Teaching Novices Programming with Core Language and Dynamic Visualization. *Papers from the Proceedings of the 17th NACCQ (National Advisory Committee on Computing Qualifications)* (s.94-103). http://www.naccq.ac.nz/conference05/proceedings_04/hu.pdf web adresinden 5 Eylül 2008 tarihinde edinilmiştir.

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

Hyrskykari, A. (1993) Development of program visualization systems. *2nd Czech British Symposium of Visual Aspects of Man-Machine Systems*, Praha. http://www.cs.uta.fi/reports/pdf/A-1995-3.pdf web adresinden 17 Ağustos 2008 tarihinde edinilmiştir.

İnce, İ., Şenyüzlü, B. ve Uğur, B. (2007). *İlköğretim Bilişim Teknolojileri 6, 7 ve 8. Basamak Öğretmen Kılavuz Kitabı*, MEB

Jenkins, T. (2002). On the difficulty of learning to program. *Proceedings of the 3rd annual Conference of the LTSN Centre for information and Computer Sciences* (s.53-58). http://www.psy.gla.ac.uk/~steve/localed/jenkins.html web adresinden 30 Mayıs 2009 tarihinde edinilmiştir.

Jones, S. ve Burnett, G. (2008). Spatial ability and learning to program. An *Interdisciplinary Journal on Humans in ICT Environments 4* (1), 47–61. ISSN: 1795-6889. http://www.humantechnology.jyu.fi/articles/volume4/2008/jones-burnett.pdf web adresinden 8 Ağustos 2008 tarihinde edinilmiştir.

Kahn, K. (1996). ToonTalk-An Animated Programming Environment for Children. *Journal of Visual Languages ve Computing*, *7*(2), 197-217(21).

Kelleher, C., Pausch, R. ve 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.

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 Eylül 2008 tarihinde edinilmiştir.

Korhonen, A. (2003). *Visual Algorithm Simulation.* Dissertation for the degree of Doctor of Science in Technology. Helsinki University of Technology Department of Computer Science and Engineering. Espoo, Finland. http://www.cs.hut.fi/Research/SVG/publications/tkoa40.pdf web adresinden 27 Ağustos 2008 tarihinde edinilmiştir.

Lahtinen, E., Ahoniemi, T. ve 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.

Lawrence, A.W., Badre, A.M. ve Stasko, J.T. (1994). Empirically evaluating the use of animations to teach algorithms Visual languages, *Proceedings of IEEE symposium* (s. 48-54). ftp://ftp.cc.gatech.edu/pub/gvu/tech-reports/1994/94-07.pdf web adresinden 30 Mayıs 2009 tarihinde edinilmiştir.

Lee, J., Moreno, K.E., ve Sympson, J.B. (1986). The effects of mode of test administration on test performance [Özet]. *Educational and Psychological Measurement 46*, 467-473.

Leeper, R.R. ve Silver, J. L. (1982). Predicting success in a first programming course. *13th SIGCSE Technical Symposium on Computer Science Education* (s.147-150). Indianapolis, Indiana, United States.

Lin, C. ve Zhang, M. (2003, April). 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 web adresinden 9 Eylül 2008 tarihinde edinilmiştir

Lin, J. M.C., Yen, L.Y., Yang, M.C., ve Chen, C.F. (2005). Teaching computer programming in elementary schools: A pilot study. [NECC pdf.] *National Educational Computing Conference Center*. Philadelphia, PA. Pennysylvania, Convention Center. http://center.uoregon.edu/ISTE/uploads/NECC2005/KEY_6749874/Lin_NECC2005_Paper_RP.pdf web adresinden 25 aralık 2008 tarihinde edinilmiştir.

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

http://www.eecs.harvard.edu/~malan/publications/fp079-malan.pdf web adresinden 3 Kasım 2007 tarihinde edinilmiştir.

Mason, B.J., Patry, M. ve Bernstein, D.J. (2001). An examination of the equivalence between non-adaptive computer-based and traditional testing [Özet]. *Journal of Educational Computing Research 24*, 29-39.

Mazzeo, J., Druesne, B., Raffeld, P.C., Checketts, K.T., ve Muhlstein, A. (1991). Comparability of Computer and Paper-and-Pencil Scores for Two CLEP General Examinations. College Board Report No. 91-5. New York. (ERIC Document Reproduction Service No.ED344902). http://professionals.collegeboard.com/profdownload/pdf/RR%2091-5.PDF web adresinden 29 Mayıs 2009 tarihinde edinilmiştir.

Mead, A.D. ve Drasgow, F. (1993). Equivalence of computerized and paper-and-pencil cognitive ability tests: A meta-analysis [Özet]. *Psychological Bulletin, 114*, 449-458.

Moreno, A., Myller, N., Sutinen, E., ve Ben-Ari, M. (2004). Visualizing programs with Jeliot 3. *Proceedings of the International Working Conference on Advanced Visual Interfaces AVI 2004*, Gallipoli, Italy.

Naps, T., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hunhausen, C., ve diğerleri (2003). Exploring the role of visualization and engagement in computer science education.. *SIGCSE Bulletin, 35 (2)*, 131–152. http://www.cs.hut.fi/Research/SVG/publications/exploring_role_of_vis_and_engagement_in_cse.pdf web adresinden 8 Eylül 2008 tarihinde edinilmiştir.

Naps, T. (2005). Jhave – Adressing The Need to Support Algorithm Visualization With Tools for Active Engegement. *IEEE Computer Graphics and Applications 25* (5), 49-55.

http://www.uwosh.edu/faculty_staff/naps/article.pdf web adresinden 24 Kasım 2008 tarihinde edinilmiştir.

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

Nelson, M. ve Rice, D. (2000). Introduction to algorithms and problem solving. *30th ASEE/IEEE Frontiers in Education Conference*, IEEE. http://fie-conference.org/fie2000/papers/1068.pdf web adresinden 30 Mayıs 2009 tarihinde edinilmiştir.

Neuman, G. ve Baydoun, R. (1998). Computerization of paper-and-pencil tests: When are they equivalent? [Özet]. *Applied Psychological Measurement 22,* 71-83. (ERIC Document Service No.EJ594309)

Nowaczyk, R.H. (1983). *Cognitive Skills Needed In Computer Programming* [Özet]. (ERIC Document Reproduction Service No. 236466).

Özdener, N. (2008). A comparison of the misconceptions about the time-efficiency of algorithms by various profiles of computer programming students. *Computers ve Education* 51, 1094–1102.

Özerbaş, A. (2003). *Bilgisayar destekli bağlaşık öğretimin öğrenci başarısı, motivasyonu ve transfer becerilerine etkisi.* Doktora Tezi. Ankara Üniversitesi, Eğitim Bilimleri Fakültesi, Ankara.

Pane, J. ve 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 29 Mayıs 2009 tarihinde edinilmiştir.

Pea, R.D. ve Kurland, D.M. (1983). On The Cognitive Prerequisites Of Learning Computer Programming *(Technical Report No.18).* New York: Bank Street College of Education, Center for Children and Technology.

Peppler, K. A. ve 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 web adresinden 2 Ekim, 2007 tarihinde edinilmiştir.

Petersen, C. G., & Howe, T. G. (1979). Predicting academic success in Introduction to Computers. *Association for Educational Data Systems Journal*, *12*(4), 182-191.

Pomplun, M. Frey, S. ve Becker, D. F. (2002). The Score Equivalence of Paper-and-of Pencil and Computerized Versions of a Speeded Test of Reading Comprehensions. *Educational and Psychological Measurement, 62*, 337-354.

Proulx, V. (2000). Programming patterns and design patterns in the introductory computer science course. *SIGCSE Bulletin, 32*(1), 80-84. http://www.ccs.neu.edu/home/vkp/Papers/Patterns-sigcse2000.doc web adresinden 30 Mayıs 2009 tarihinde edinilmiştir.

Rajala, T., Laakso, M., Kaila, E., ve Salakoski, T.(2008).Effectiveness of Program Visualization:A Case Study with the ViLLE Tool. *Journal of Information Technology Education:* Innovations in Practice. *7*,16-32

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

Resnick, M. (2007). All I Really Need to Know (About Creative Thinking) I Learned (By Studying How Children Learn) in Kindergarten. *Proceedings of the SIGCHI Conference on Creativity and Cognition*, Washington, D.C. http://web.media.mit.edu/~mres/papers/kindergarten-learning-approach.pdf web adresinden 2 Kasım 2007 tarihinde edinilmiştir.

Reed, W.M. ve Burton J.K. (1988). *Educational computing and problem solving.* Haworth Press ISBN 086656781X, 9780866567817

Russell, M. (1999). Testing on computers: A follow-up study comparing performance on computer and on paper. *Education Policy Analysis Archieves,* 7 (20). http://escholarship.bc.edu/intasc/6/ web adresinden 29 Mayıs 2009 tarihinde edinilmiştir.

Russell, M. ve Haney, W. (1997). Testing Writing on Computers: An Experiment Comparing Student Performance on Tests Conducted via Computer and via Paper-and-Pencil. *Education Policy Analysis Archives 5* (3), 1-19.

Sleeman, D. ve diğerleri (1984). Pascal and High-School Students: A Study of Misconceptions. *Technology Panel Study of Stanford and the Schools.* (ERIC Document Reproduction Service No. ED258552).

Sri Sri Ravishankar Vidya Mandir (SSRVM) Model Curriculum and Teaching Material for K-12 Indian Schools (2007), Computer Science. http://www.it.iitb.ac.in/~sri/papers/SSRVM-CS-March07.pdf web adresinden 15 Mayıs 2009 tarihinde edinilmiştir.

Stephenson, C. (2001). Knowing What to Teach—Computing in High Schools Panel (Özet Kitabı). *The Computer Science and Information Technology Symposium Issues and Trends in High School Computing* (s. 15). Chicago, Illinois.

Traynor, D., ve 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/16th-traynor.pdf web adresinden 30 Mayıs 2009 tarihinde edinilmiştir.

Tucker, A. Deek, F., Jones, J., McCowan, D., Stephenson, C. ve Verno, A. (2003). A Model Curriculum for k12 Computer Science: *Final Report of the Association for Computing Machinery (ACM),* New York. http://www.isp.org.pl/podstawa/podstawa_files/K12_Computer_Science.pdf web adresinden 25 Aralık 2008 tarihinde edinilmiştir.

Urquiza-Fuentes, J., ve Velázquez-Iturbide, J.A. (2007). An Evaluation of the Effortless Approach to Build Algorithm Animations with WinHIPE. *Elektronic Notes In Theoretical Computer Science. 178*, 3-13

Ward, T.J, Hooper, S.R. ve Hannafin, K.M. (1989). The effect of computerized tests on the performance and attitudes of college students [Özet]. *Journal of Educational Computing Research, 5,* 327-333. (ERIC Document Reproduction Service No. EJ399487)

Webb, N. M. (1985). Cognitive requirements of learning computer programming in group and individual settings. *AEDS Journal, 18*(3), 183-193. (ERIC Document Reproduction Service No. EJ317206)

Weinberg, A. (2001). Comparaison de deux versions d'une test de classement: Version papier-crayon et version informatisee [Comparison of two versions of a placement tests: Paper-pencil version and computer-based

version] [Özet]. *Canadian Modern Language Review, 57*(3) 607-627. (ERIC Document Service No. EJ628043).

EKLER

EK 1 ÖN BİLGİ TESTİ

1. Aşağıda Ali'nin otobüsle bir yerden bir yere giderken yapacaklarının adımları verilmiştir. Adımları doğru biçimde sıralamak için aşağıdaki boşluklara numara veriniz.

__ İneceğin yere yaklaştığında arkaya yürü.
__ Otobüs durunca in.
__ Durakta gideceğin yöndeki otobüs gelinceye kadar bekle.
__ Otobüse bin.
__ Otobüs durağına yürü.
__ İneceğini belirten ikaz lambasına bas.

2. Bilgisayarınızda ödevinizi yazdıracakken yazıcınız bozuldu. Bununla ilgili olabilecek problemlerden birini seçiniz ve çözüm yolunu adımlar halinde sıralayınız

1. Adım:
2. Adım:
…
…
…

3. Sizi tanımlayacak beş farklı alandaki bilgilerinizi doldurun ve alandaki değişkenlerin veri türlerini yazın. (10 puan)

Alan Adı	Bilgiler	Veri Türü
Adınız:	…	Metin
Yaşınız:	…	…
Doğum Tarihiniz (gün/ay/yıl):		
Oturduğunuz Yer:		…
Tel. No:	…	

4. Bilgisayarınız 1. yazılı puanınızı soracak, bu puana uygun bir değer verecektir. Eğer puan 45'ten küçükse ekrana "Dersten kaldınız" eğer puan 45'ten büyükse "Dersten geçtiniz." yazdıracaktır. Bilgisayarınızın yapacağı işlemleri adımlar halinde sıralayınız

1. Adım:
2. Adım:
…

5. Aşağıdaki işlemleri x e 1'den 8'e kadar olan değerleri vererek uyguladığınızda sonuç ne olacaktır?

```
X = 1'den 8'e kadar tekrarla
{
      Eğer x  tek sayı ise
            x sayısını 2 ile çarp
            sonucu yaz
      Eğer x çift sayı ise
            x sayısını 3 ile çarp
            sonucu yaz
}
```

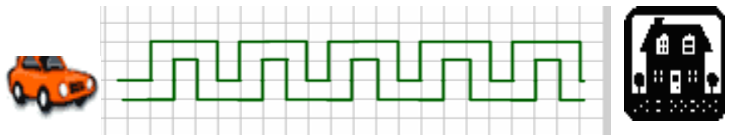
Örn: x = 1 için
Sonuç 2

6. Aşağıdaki ifadelerin yanlarına doğru ise (D) yanlış ise (Y) yazınız.

a. x = ((125 / 5) + (3 + 4) ise x = 5'dir. (…)

b. x = 2 için ((x >= 2) ve (x <= 2)'dir. (…)

7. Aşağıda yolunu kaybetmiş bir sürücünün eve gitmek için izlemesi gereken yol verilmiştir. Buna göre sürücünün izleyeceği adımları en kısa biçimde sıralayınız.



1. Adım:
2. Adım:
3. Adım:
...
...

8. Aşağıdaki ifadeleri doğru yapan i ve j değerlerini yazınız.

a. i>=1 ve i<=5 ise i = _____

b. j>=3 ve j<=7 ise j = _____

EK 2 UYGULAMADA KULLANILAN GÜNLÜK PLANLAR

Ders 1 (Algoritma Ve Programlamanın Temel Kavramları)
Kazanım

İşlemlerin ve problemlerin çözümünü yaparken algoritmanın ve programlamanın genel kavramlarını anlar.

1. Bütün bir problemi bileşenlerine ayırır.
2. Tanımlanmış bir probleme bilişim teknolojileri çözümlerini uygular.

Açıklamalar

Problemin çözümü için yazılması gereken programın temel aşamaları (problemin tanımı, çözüm yolunun tespiti, algoritmanın hazırlanması diyagramının çizilmesi) açıklanır.

Süre: 2 ders saati

Ders Öncesi Hazırlık

Öğrencilere günlük hayatlarında karşılarına çıkan farklı yollarla çözülebilecek birkaç problem getirmeleri söylenir.

Öğrenme – Öğretme Süreci

1. Dikkat Çekme

Öğrencilere günlük hayatta bilerek ya da bilmeyerek algoritma kurdukları söylenir. Örneğin yazın hava çok sıcak ve sıcaktan bunalıyorsunuz. Sıcaktan bunalmanız bir problemdir. Bu problemin birçok çözüm yolu olabilir. Örneğin bol sıvı tüketmek bu problemin çözüm yollarından biri olabilir.

2. İşleniş

- Belirtilen problem için her adım öğrencilerle birlikte okunur ve tartışılır. Problemin çözüm yolunun bu şekilde adım adım ifade edilmesinin algoritmaya bir örnek olduğu belirtilir ve algoritma kavramı açıklanır. Akış şemasında kullanılan şekiller ve anlamları açıklanır.
- Öğrencilerden algoritmaya günlük hayattan örnekler vermeleri istenir. Akış şeması kullanılarak günlük hayattan bir örnek için bir algoritma oluşturulur.
- Uygulamalar
    - Scratch programı kullanan öğrenciler için Scratch programında algoritma oluşturmada kullanılan şekiller ve anlamları açıklanır.

        1.1 Scratch ile ilgili etkinlik 1 yapılır.

- FCPRO programı kullanan öğrenciler için FCPRO programında algoritma oluşturmada kullanılan şekiller ve anlamları açıklanır.

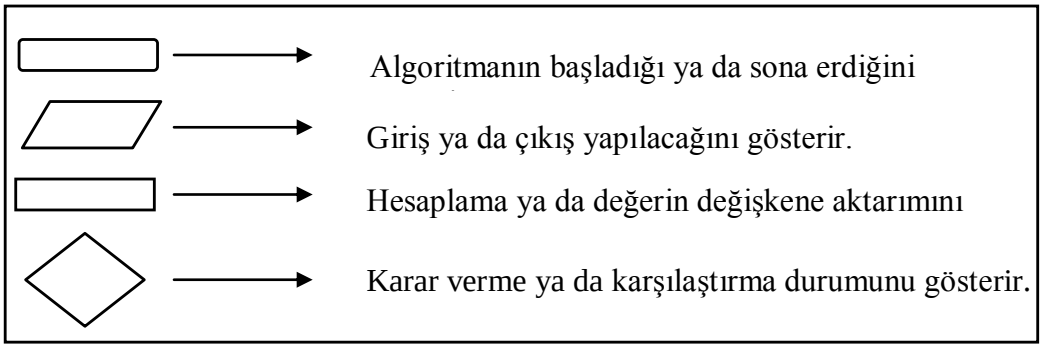
1.2 FCPRO ile ilgili etkinlik 1 yapılır

1.1 Scratch Algoritma Ve Programlama Çalışma Sayfası 1

Algoritma: Problemin çözümünün adım adım yazılmasına algoritma denir.

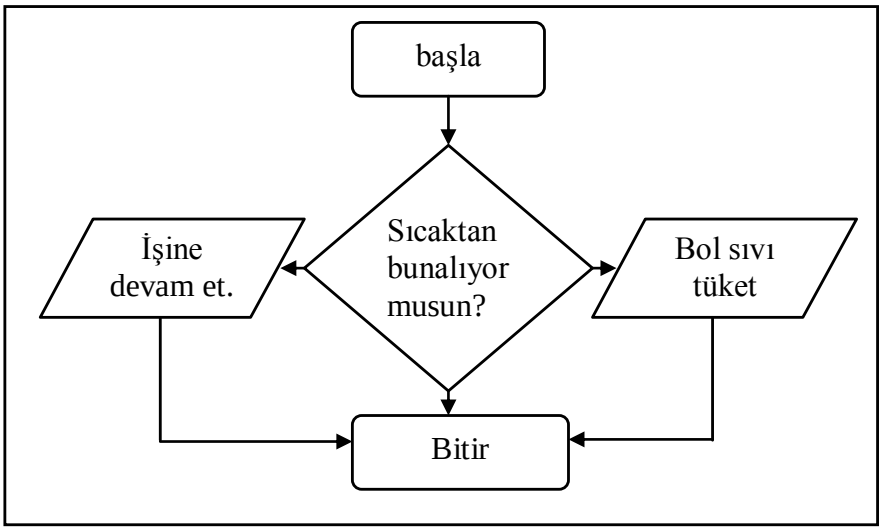
Akış şeması: Algoritmanın sembollerle ifade edilmesidir.

Akış Şemasında Kullanılan Şekiller Ve Anlamları



Şekil 1 Akış Şemasında Kullanılan Şekiller

Akış Şeması Örneği



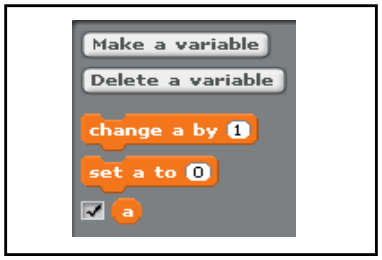
Şekil 2 Akış Şeması Örneği

SCRATCH Programı Akış Şeması Sembolleri

İşlemler

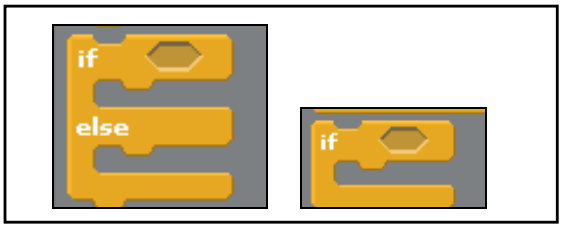
Şekil 3 Scratch Programında Yer Alan İşlemler

Değişkenler



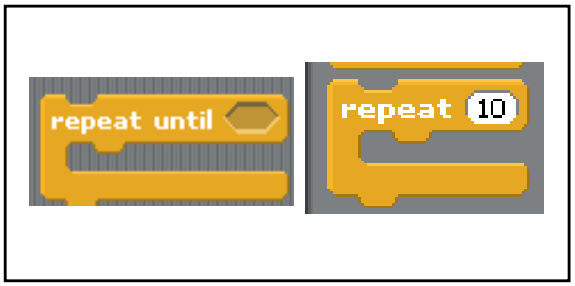
Şekil 4 Scratch Programında Değişken Ornegi

Koşul Cümleleri



Şekil 5 Scratch Programında Koşul Cümlesi Örnekleri

Döngü komutları



Şekil 6 Scratch Programında Döngü Örneği

1.1 Scratch Algoritma Ve Programlama Çalışma Sayfası 2

Etkinlik Numarası: 1
Konu: Scratch programında akış şeması oluşturma
Amaç: 1. Bütün bir problemi bileşenlerine ayırır. 2.Tanımlanmış bir problem için Scratch programında bir algoritma hazırlayabilme

Bu etkinlikte sizlerle Scratch programındaki kedi karakterinin algoritma yardımıyla istediğimiz gibi konuşmasını sağlayacağız. İşlemin sonucunda Şekil-8'de sahnede görülen kedi karakteri 200 ve 30'un toplamını görüntüleyecektir.

Bunun için gerekli kodları kod alanına sürüklemeniz ve algoritmayı çalıştırmanız gerekiyor. Bu iş için gerekli adımlar aşağıdadır. Başarılar.

Yönerge

1) Algoritmayı başlatmak için 

komutunu kullanınız.
2) Scratch programındaki hazır kodlardan yararlanarak
    a. karakterin "Toplama yapiyorum" deyip 2 saniye beklemesini,
    b. karakterin ""a sayisi = 200 ve b sayisi = 30 ise" demesini,
    c. "iki sayinin toplami = 200 + 30 = 230'dur" deyip 2 saniye beklemesini sağlayın
3) Karakterin hareketlerindeki değişimi izlemek için şekildeki flag düğmesine tıklayın.

Şekil 7 Flag, programı başlatma düğmesi

4) Algoritma tamamlandıktan sonra ekranda aşağıdaki gibi bir görüntü oluşacaktır.

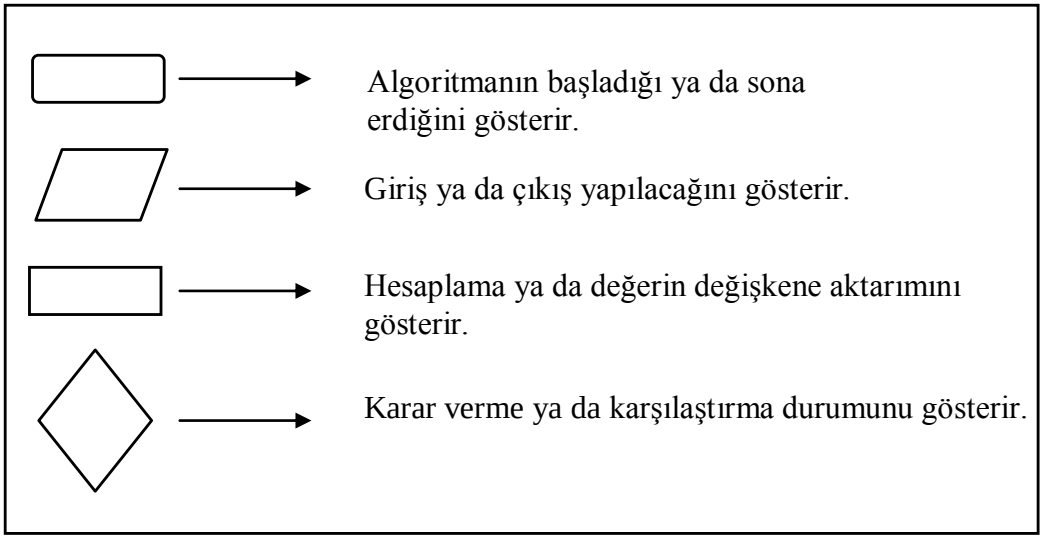
Şekil 8 Scratch Etkinlik 1 ekran görüntüsü

1.2 Fcpro Algoritma Ve Programlama Çalışma Sayfası 1

Algoritma: Problemin çözümünün adım adım yazılmasına algoritma denir.

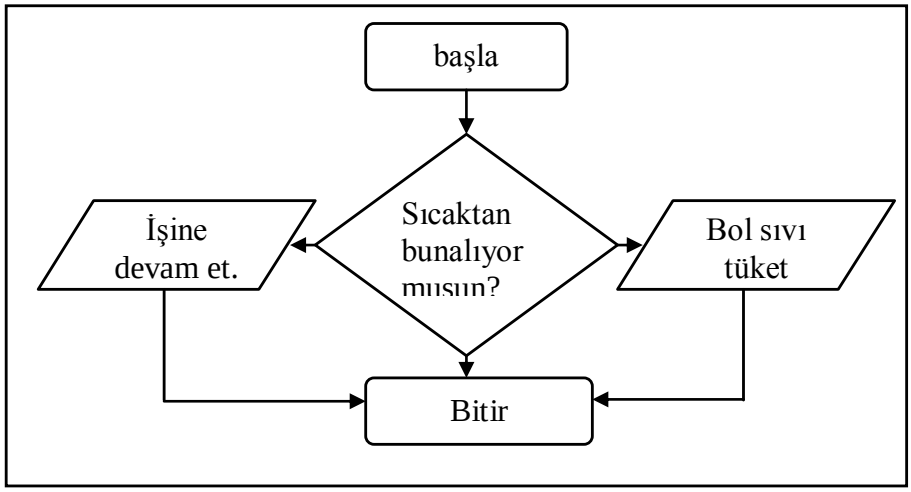
Akış şeması: Algoritmanın sembollerle ifade edilmesidir.

Akış Şemasında Kullanılan Şekiller Ve Anlamları



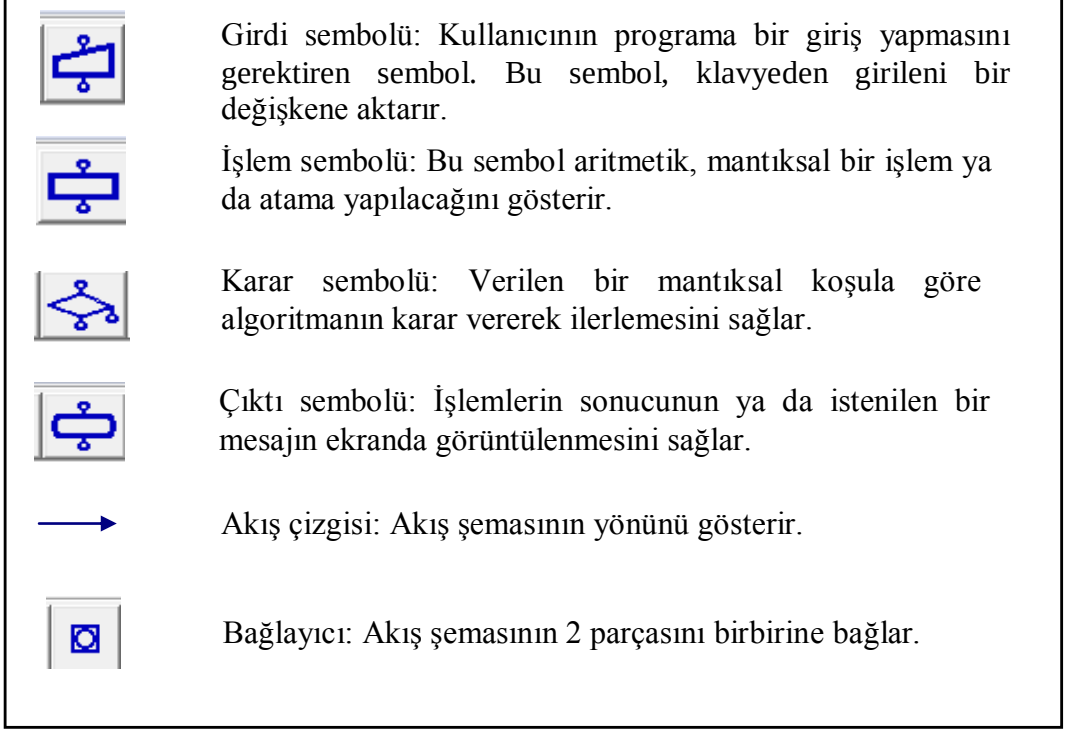
Şekil 1 Akış Şemasında Kullanılan Şekiller

Akış Şeması Örneği



Şekil 2 Akış Şeması Örneği

FCPRO PROGRAMI AKIŞ ŞEMASI SEMBOLLERİ VE ANLAMLARI



Şekil 3 FCPRO programı akış şeması sembolleri ve anlamları

1.2 Fcpro Algoritma Ve Programlama Çalışma Sayfası 2

Etkinlik Numarası: 1

Konu: FCPRO programında akış şeması oluşturma

Amaç:

1. Bütün bir problemi bileşenlerine ayırır.

2.Tanımlanmış bir probleme bilişim teknolojileri çözümlerini uygular.

Bu etkinlikte sizlerle FCPRO programında algoritma yardımıyla iki sayının toplamını görüntüleyeceğiz. İşlemin sonucunda çıktı ekranında algoritma, Şekil-5'te gösterildiği gibi olmalıdır. Bunun için gerekli kodları kod alanına sürüklemeniz, gerekli kodları yazarak algoritmayı çalıştırmanız gerekiyor. Bu iş için gerekli adımlar aşağıdadır. Başarılar.

Yönerge

1) Algoritmanın ekrana önce a sayisi = 189 yazdırmasını sağlayınız.

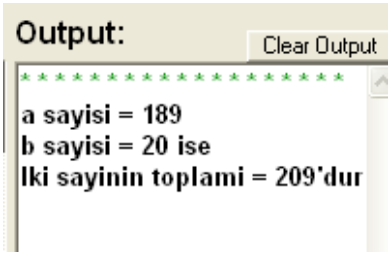
2) Algoritmanın ekrana b sayisi = 20 yazdırmasını sağlayınız.

3) Algoritmanin "İki sayinin toplami = 209" yazdırarak sonucun görüntülenmesini sağlayınız.

4) Programı çalıştırmak için aşağıdaki şekilde gösterilen düğmeye basınız.

Şekil 4 Programı çalıştırma

5) Sonuç ekranı aşağıdaki gibi görünmelidir.



Şekil 5 FCPRO Etkinlik 1 ekran görüntüsü

Ders 2 (Değişken Kavramı)

Kazanımlar

Bilgileri ifade etmek ve üzerlerinde işlem yapmak için basit değişkenler tanımlar ve kullanır.

Süre: 2 ders saati

Kelime bilgisi

Değişken, veri, veri türleri

Öğrenme – Öğretme Süreci

1. Dikkat Çekme

Öğrencilerden bir arkadaşıyla grup olmalarını istenir. Her ikisinin de akıllarından bir sayı tutmaları istenir. Bu sayıları "Sayı Tutmaca" etkinliğinde kullanacaklarını belirtilir ve bu etkinlikte değişkenler kullanarak bir oyun oynayacaklarını öğrencilere söylenir.

2. İşleniş

- Öğrencilerden yanlarındaki arkadaşlarının tuttuğu sayı ile işlem yapmalarını istenir. Bu işlemi yaparken bilmedikleri bu sayıya bir değişken ismi vermeleri gerektiğini arkadaşlarının tuttuğu sayının bir bilinmeyen olduğunu öğrencilere belirtilir.
- Değişkenin ne demek olduğu ve ne amaçla kullanıldığını öğrencilere açıklanır.
- Değişkenlerle işlem yapılırken matematik dersinde gördükleri işlem önceliklerini hatırlatılır. Açılan parantezlerin sırasıyla kapatılması ve parantez içlerine her durumda öncelik verilmesi gerektiği belirtilir.
- Öğrencilere "Veri" kavramı açıklanır. Veri türlerinin neler olduğu belirtilir.
- Uygulamalar
    - Scratch programı kullanan öğrenciler için Scratch programında değişken oluşturma ve değer atama açıklanır.
        - **2.1** Scratch ile ilgili etkinlik 2 yapılır.
    - FCPRO programı kullanan öğrenciler için FCPRO programında değişken oluşturma ve değer atama açıklanır.
        - **2.2** FCPRO ile ilgili etkinlik 2 yapılır.

2.1 Scratch Değişken Dersi Çalışma Sayfası 1

Değişken: Problemleri çözerken ya da programlama yaparken üzerinde işlem yapılacak nesneleri tanımlamak için kullanılır.

Değişkenlerle matematiksel işlem yaparken işlem öncelikleri şu şekilde olmalıdır:

1- Parantez içleri, 2- Üs alma, 3-Çarpma/Bölme, 4-Toplama/Çıkarma

Eğer yan yana işlemlerde parantez yoksa çarpma ve bölme işlemleri için öncelikle soldan başlayarak işlemleri yapmanız gerekir.

= (a * 30 / 100) + (b * 30 / 100) + (c * 40 / 100)

Veri: Olgu, kavram ya da komutların iletişim, yorum ve işlem için elverişli biçimli gösterimi

Veri Türleri: Verilerin girileceği alanlara hangi türde yazıldığıdır.

Metin: içeriği metinlerden oluşan alanlar için kullanılır.

Sayı: içeriği sayılardan oluşan alanlar için kullanılır.

Tarih/saat: içeriği tarih ya da saatten oluşan alanlar için kullanılır.

Para birimi: içeriği para miktarlarından oluşan alanlar için kullanılır.

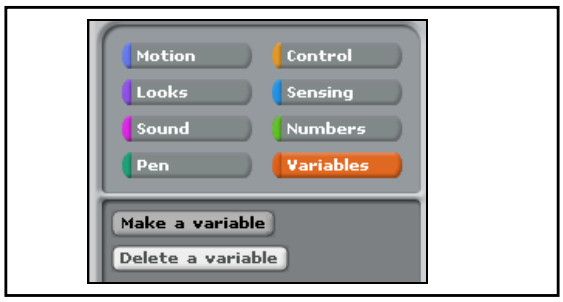
Otomatik sayı: içeriği otomatik olarak veritabanı tarafından artırılan sayılardan oluşan

alanlar için kullanılır.

Evet/hayır: içeriği evet ya da hayır kelimelerinden oluşan alanlar için kullanılır

Scratch Programında Değişken Oluşturma

Scratch Programında değişken oluşturmak için variables sekmesi kullanılır.



Şekil 1. Scratch programında değişken oluşturma 1

Make a variable düğmesi tıklandığında aşağıdaki gibi değişken adi sorulur. Bu adimda değişkene isim verilerek Ok düğmesine tıklandığında yeni değişken oluşturmuş oluruz.

Şekil 2. Scratch programında değişken oluşturma 2

2.1 Scratch Değişken Dersi Çalışma Sayfası 2

Etkinlik Numarası: 2

Konu: Değişkenler

Amaç: 1. Bilgileri ifade etmek ve üzerlerinde işlem yapmak için basit değişkenler tanımlar ve kullanır.

Bu etkinlikte sizlerle Scratch programında değişken oluşturacağız. Bu değişkene değer atayarak değişkenin program çalıştırıldığında nasıl değiştiğini gözlemleyeceğiz. Etkinliğin sonunda Şekil-4'teki gibi değişkenin değiştiği gözlemlenecektir.

Bu iş için gerekli adımlar aşağıdadır. Başarılar.

Yönerge

1) Algoritmayı başlatmak için 

komutunu kullanınız.

2) "a" ve "b" adında iki değişken oluşturunuz.

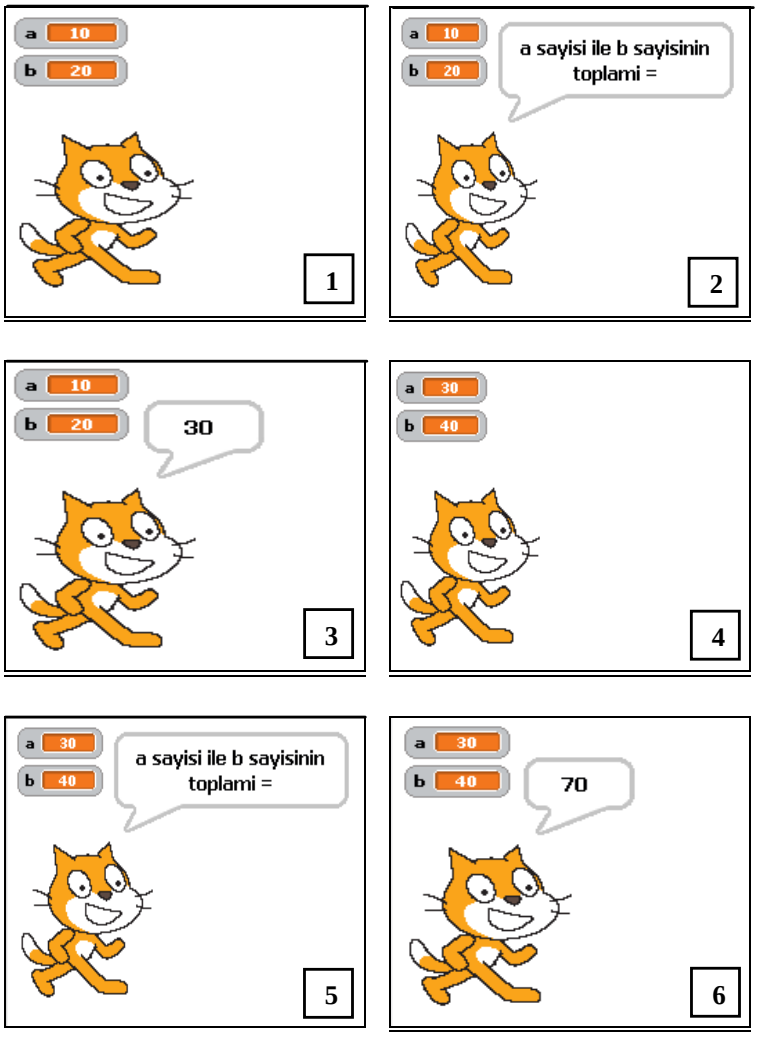
4) "a" değişkenine 10 değerini, "b" değişkenine ise 20 değerini atayınız.

5) Karakterin "a" ve "b" değişkeninin değerinin toplamını söylemesini sağlayın.

6) "a" değişkenine 30 değerini, "b" değişkenine ise 40 değerini atayınız.

7) Karakterin "a" ve "b" değişkeninin yeni değerlerinin toplamını söylemesini sağlayınız.

8) Program çalıştırıldığında aşağıdaki gibi görünecektir.



Şekil 3 Scratch Etkinlik 2 ekran görüntüsü

2.2 Fcpro Değişken Dersi Çalışma Sayfası 1

Veri: Olgu, kavram ya da komutların iletişim, yorum ve işlem için elverişli biçimli gösterimi

Veri Türleri: Verilerin girileceği alanlara hangi türde yazıldığıdır.

Metin: içeriği metinlerden oluşan alanlar için kullanılır.

Sayı: içeriği sayılardan oluşan alanlar için kullanılır.

Tarih/saat: içeriği tarih ya da saatten oluşan alanlar için kullanılır.

Para birimi: içeriği para miktarlarından oluşan alanlar için kullanılır.

Otomatik sayı: içeriği otomatik olarak veritabanı tarafından artırılan sayılardan oluşan alanlar için kullanılır.

Evet/hayır: içeriği evet ya da hayır kelimelerinden oluşan alanlar için kullanılır.

Değişken: Problemleri çözerken ya da programlama yaparken üzerinde işlem yapılacak nesneleri tanımlamak için kullanılır.

Değişkenlerle matematiksel işlem yaparken işlem öncelikleri şu şekilde olmalıdır:

1- Parantez içleri, 2- Üs alma, 3-Çarpma/Bölme, 4-Toplama/Çıkarma

Eğer yan yana işlemlerde parantez yoksa çarpma ve bölme işlemleri için öncelikle soldan başlayarak işlemleri yapmanız gerekir.

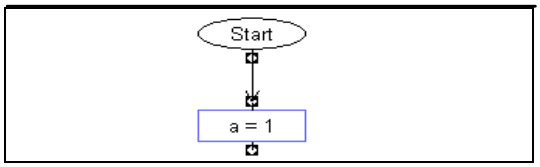
= (a * 30 / 100) + (b * 30 / 100) + (c * 40 / 100)

FCPRO Programında Değişken Oluşturma

FCPRO programında değişken oluşturmak aşağıdaki şekilde düğme kullanılır.

Şekil 1 FCPRO programında değişken oluşturma 1

Değişken oluşturmak için şeklin içine değişkenin adı = değer yazmak yeterlidir.



Şekil 2 FCPRO programında değişken oluşturma 2

2.2 Fcpro Değişken Dersi Çalışma Sayfası 2

Etkinlik Numarası: 2

Konu: Değişkenler

Amaç: 1. Bilgileri ifade etmek ve üzerlerinde işlem yapmak için basit değişkenler tanımlar ve kullanır.

Bu etkinlikte sizlerle FCPRO programında iki değişken oluşturacağız. Bu değişkenlere değerler atayarak değerleri görüntüleyeceğiz. Değişkenleri toplayarak sonucu görüntüleyeceğiz. Daha sonra değişkenlerin değerlerini değiştirerek değişkenlerin değerini yeniden görüntüleyeceğiz. Değişkenleri toplayarak yeni sonucu görüntüleyeceğiz.

Algoritma çalıştırıldığında sonuç ekranında iki değişkenin değerleri Şekil-5'teki gibi görüntülenmelidir.

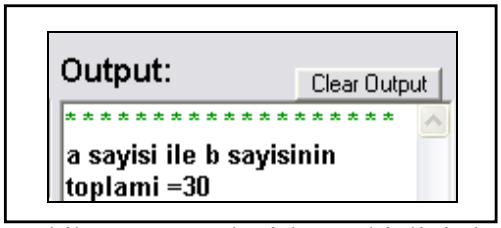
Bu iş için gerekli adımlar aşağıdadır. Başarılar.

Yönerge

1) FCPRO programında aşağıdaki komutu gibi a adında bir değişken oluşturarak 10 değerini atayınız.

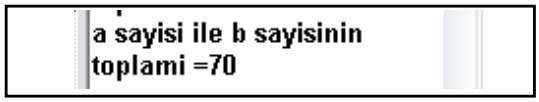
2) b adında bir değişken oluşturarak 20 değerini atayınız.

3) a ve b değişkenlerinin toplamının output ekranında şekildeki gibi görüntülenmesini sağlayın.



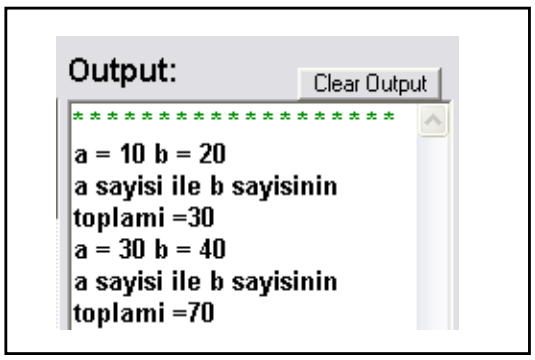
Şekil 3 FCPRO değişken etkinliği ekran görüntüsü 1

4) a değişkenine 30, b değişkenine 40 değerini veriniz. a ve b değişkenlerinin toplamının output ekranında şekildeki gibi görüntülenmesini sağlayın.



Şekil 4 FCPRO değişken etkinliği ekran görüntüsü 2

5) Programı çalıştırdığınız algoritmanın sonuç ekranı aşağıdaki gibi görünmelidir.



Şekil 5 FCPRO değişken etkinliği ekran görüntüsü 3

Ders 3 (Koşul Cümleleri)

Kazanımlar

Amacına uygun karşılaştırmaları yapmak için koşul cümlelerini kullanma

Süre: 2 Ders Saati

Açıklamalar

Öğrencilere eğer komutu ve <(küçüktür), <= (küçük eşit), > (büyüktür), >= (büyük eşit), =(eşittir), <>(eşit değildir) operatörleri kullanılarak mantıksal karşılaştırma yaptırılır.

Kelime bilgisi

"Eğer" (If) komutu

Öğrenme – Öğretme Süreci

1. Dikkat Çekme

Tahtaya günlük hayattan birkaç tane koşul cümlesi yazılır. Örneğin, "Bugün yağmur yağmazsa parka gidelim." ya da "Sınıfını geçersen sana bilgisayar alacağım." gibi. Öğrencilere bazı kararların verilmesinde mantıksal karşılaştırmalar yapıldığı söylenir.

2. İşleniş

- Hesaplamaları yaparken bilgisayarda uygun teknolojiler kullanmanın gerekliliği vurgulanır. Programlamada mantıksal karşılaştırmalar yapılırken "=", "<", ">" gibi karşılaştırma operatörleri ve "Eğer" (If) komutunun kullanıldığı öğrencilere söylenir.
- If komutunun ve gerekli operatörlerin kullanımını basit bir örnek üzerinde açıklanır.
- Uygulamalar
  - Scratch programı kullanan öğrenciler için Scratch programında koşul cümlelerinin nasıl oluşturulduğu açıklanır.

    3.1 Scratch ile ilgili etkinlik 5 yapılır.
  - FCPRO programı kullanan öğrenciler için FCPRO programında koşul cümlelerinin nasıl oluşturulduğu açıklanır.

    3.2 FCPRO ile ilgili etkinlik 6 yapılır.

3.1 Scratch Koşul Cümleleri Dersi Çalışma Sayfası 1

Günlük hayatta bazı kararların verilmesinde mantıksal karşılaştırmalar yapılır. Örneğin, notlarınıza göre sınıfı geçip geçmediğinize karar verilirken 45'ten küçük ortalamaların “Kaldı”; 45 ve daha büyük ortalamaların “Geçti” olarak belirlenir.

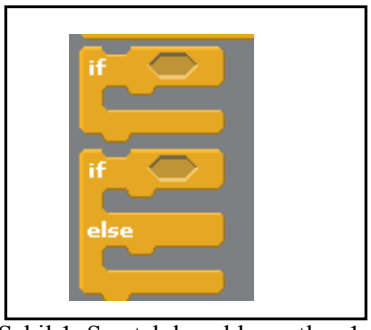
IF (Eğer): Koşula bağlı olarak hangi işlemlerin yapılacağını belirtir. Yapısı şöyledir:

```
if koşul1 then
        komutlar1
else if koşul2 then
        komutlar2
else
        komutlar3
end if
```

Koşul1 doğruysa komutlar1 çalışacaktır. Koşul1 yanlış, koşul2 doğruysa komutlar2 çalışacaktır. Koşul1 ve Koşul2 yanlışsa komutlar3 çalışacaktır.

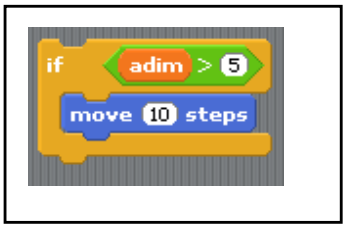
SCRATCH Programında Koşul Cümleleri

Scratch programında koşul ifadeleri aşağıda verilen şekildeki gibidir.



Şekil 1. Scratch koşul komutları 1

Şekilde gösterilen bu ifadeler if komutunun yanına koşul, koşulun içerisine de yapılacak eylem sürüklenerek oluşturulur. Koşul oluşturmak için aşağıdaki şekilde gösterildiği gibi bu sembolün içine sınanacak olan durumu yazarız. Daha sonrasında koşulun doğru ya da yanlış olması için yapılmasını istediğimiz işlemler aşağıdaki şekilde olduğu gibi belirtilir.



Şekil 2. Scratch koşul komutları 2

Üstteki şekilde “adim”, adım sayısını ifade eden bir değişkendir. Üstteki şekilde “adim > 5” ifadesi doğruluğu sınanacak olan koşul olurken, “move 10 steps” ifadesi ise koşulun doğru olması durumunda yapılacak eylemi belirtir.

3.1 Scratch Koşul Cümleleri Dersi Çalışma Sayfası 2

Etkinlik Numarası: 3

Konu: Koşul cümleleri

Amaç: 1. Amacına uygun karşılaştırmaları yapmak için koşul cümlelerini kullanma

Bu etkinlikte sizlerle Scratch programında iki değişken oluşturarak değişkenlerin değerini 0 ile karşılaştırarak 0'dan büyük ya da küçük olduğunu bize söylemesini sağlayacağız. Şekil-3'te program çalıştığında karakterin hareketlerindeki değişiklikler görüntülenmiştir. Şekli inceleyiniz.

Bunun için gerekli kodları kod alanına sürüklemeniz ve algoritmayı çalıştırmanız gerekiyor. Bu iş için gerekli adımlar aşağıdadır. Başarılar.

Yönerge

1) Algoritmayı başlatmak için 

komutunu kullanınız.

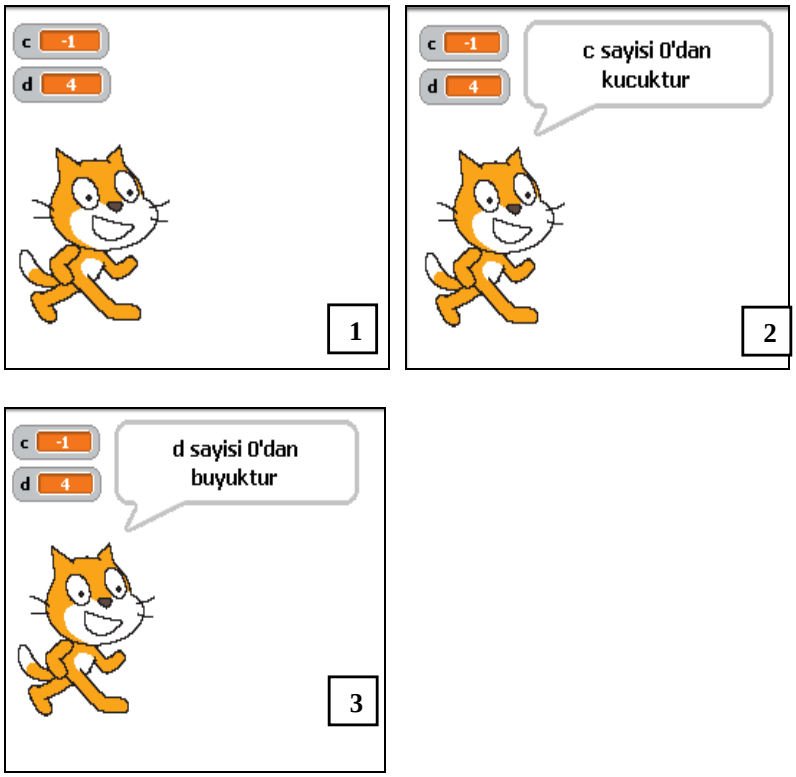
2) "c" adında bir değişken oluşturup değişkene -1 değerini atayınız.

3) "d" adında bir değişken oluşturup değişkene 4 değerini atayınız.

4) Program çalıştırıldığında c değişkeni 0'dan küçük ise karakterin "c sayisi 0'dan kucuktur" demesini sağlayın. Değilse karakter "c sayisi 0'dan buyuktur" diyecektir.

5) d değişkeni 0'dan büyük ise karakterin "d sayisi 0'dan buyuktur" demesini sağlayın. Değilse karakter "d sayisi 0'dan kucuktur" diyecektir.

6) Program çalıştırıldığında aşağıdaki gibi görünecektir.



Şekil 3 Scratch Etkinlik 3 ekran görüntüsü

3.2 Fcpro Koşul Cümleleri Dersi Çalışma Sayfası 1

Günlük hayatta bazı kararların verilmesinde mantıksal karşılaştırmalar yapılır. Örneğin, notlarınıza göre sınıfı geçip geçmediğinize karar verilirken 45'ten küçük ortalamaların "Kaldı"; 45 ve daha büyük ortalamaların "Geçti" olarak belirlenir.

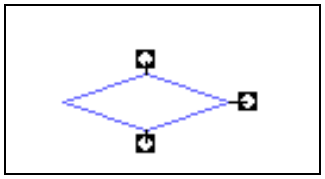
IF (Eğer): Koşula bağlı olarak hangi işlemlerin yapılacağını belirtir. Yapısı şöyledir:

```
if koşul1 then
        komutlar1
else if koşul2 then
        komutlar2
else
        komutlar3
end if
```

Koşul1 doğruysa komutlar1 çalışacaktır. Koşul1 yanlış, koşul2 doğruysa komutlar2 çalışacaktır. Koşul1 ve Koşul2 yanlışsa komutlar3 çalışacaktır.

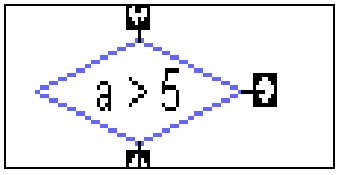
FCPRO Programında Koşul Cümleleri

FCPRO programında koşul ifadeleri için kullanılan sembol aşağıdaki şekildeki gibidir.



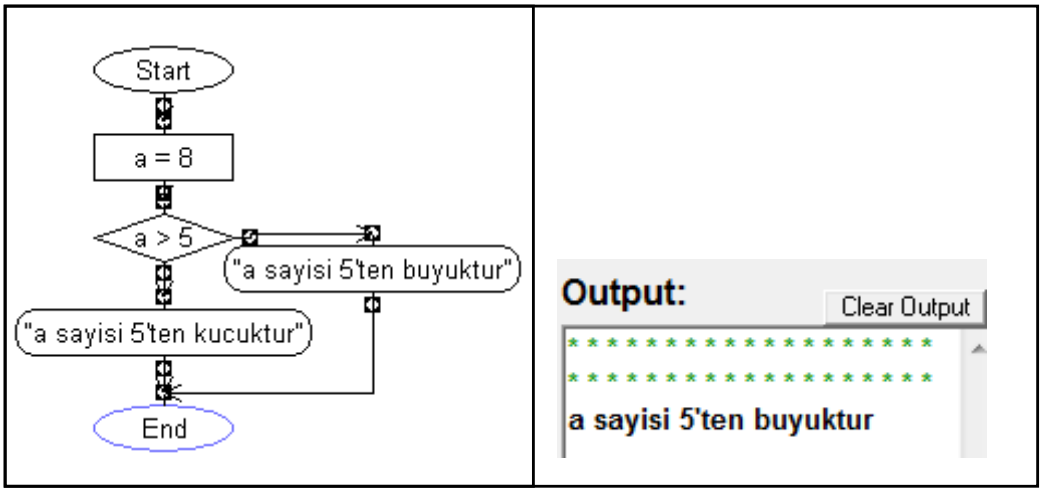
Şekil 1 FCPRO koşul komutları 1

Koşul oluşturmak için şekilde gösterildiği gibi bu sembolün içine sınanacak olan durumu yazarız.



Şekil 2 FCPRO koşul komutları 2

Üstteki şekilde öncelikle “a > 5” ifadesinin doğruluğu sınanır. Daha sonrasında koşulun doğru ya da yanlış olması durumunda yapılması istenen işlemler aşağıdaki şekilde olduğu gibi belirtilir.



Şekil 3 FCPRO koşul komutları 3

Üstteki algoritmada a değişkeninin 5’ten büyük olma durumu sınanacaktır. A değişkeni 5’ten büyük olduğu için ekrana “a sayisi 5’ten buyuktur” yazdırılacaktır.

3.2 Fcpro Koşul Cümleleri Dersi Çalışma Sayfası 2

Etkinlik Numarası: 3

Konu: Koşul cümleleri

Amaç: 1. Amacına uygun karşılaştırmaları yapmak için koşul cümlelerini kullanma

Bu etkinlikte sizlerle FCPRO programında iki değişken oluşturarak değişkenlere bir değer atayacağız. Değişkenlerin 0'dan büyük olması halinde ekrana "x sayısı pozitif bir sayıdır", 0'dan küçük olması halinde de "x sayısı negatif bir sayıdır" yazdıracağız. Program çalıştırıldığında ekran görüntüsünün nasıl değişeceği Şekil-4'te görüntülenmektedir. Şekli inceleyiniz.

Bunun için yapılması gereken adımlar aşağıdadır. Başarılar.

Yönerge

1) FCPRO programında şekildeki gibi c adında bir değişken oluşturarak -1 değerini atayınız.

2) d adında bir değişken oluşturarak 4 değerini atayınız.

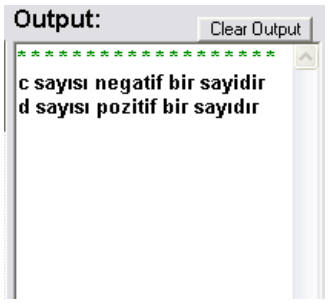
2) Koşul ile ilgili sembolü akış şemasına ekleyiniz. c < 0 koşulunun kontrolünü sağlayınız.

3) Akış şemasının c değeri 0'dan küçükse ekrana "c sayısı negatif bir sayıdır.", büyükse "c sayısı pozitif bir sayıdır." yazdıracak kodları akış şemasına ekleyiniz.

4) Koşul ile ilgili sembolü tekrar akış şemasına ekleyiniz. Bu kez d > 0 koşulunun kontrolünü sağlayınız.

5) Akış şemasının d değeri 0'dan büyükse ekrana "d sayısı pozitif bir sayıdır.", küçükse "d sayısı negatif bir sayıdır." yazdıracak kodları akış şemasına ekleyiniz.

6) Program çalıştırıldığında ekran görüntüsü aşağıdaki gibi olacaktır.



Şekil 4 FCPRO Etkinlik 3 ekran görüntüsü

Ders 4 (Döngü)

Kazanım

1.Program içerisinde tekrar eden bir grupta döngüleri kullanır.

Açıklamalar

Döngü mantığı açıklanır.

Süre: 2 ders saati

Kelime bilgisi: Döngü

ÖĞRENME – ÖĞRETME SÜRECİ

1. Dikkat Çekme

Öğrencilerden çalışma kağıdındaki Şekil-1'de verilen süs havuzu resmini incelemelerini istenir. "Resimde tekrar eden olaylar hangilerdir?" sorusunu yöneltilir. Gelen cevaplar doğrultusunda öğrencilerinize resimde havuzdaki suyun bir döngü şeklinde yukarı çıkıp tekrar aşağı indiği konusunda açıklama yapılır. Programlamada da burada olduğu gibi tekrar eden olaylar olduğunu belirterek bunlara döngü dendiği söylenir.

2. İşleniş

- Öğrencilere programlama yapılırken bu şekilde tekrar eden ifadelerin döngü komutlarının içine yazıldığını ve böylece kısaltılmış olduğunu belirtilir. Döngü kavramını ve nasıl kullanıldığı açıklanır.
- Döngü: Belirli işlemleri, belirli sayıda veya herhangi bir şart sağlanana kadar tekrarlamak amacı ile kullanılır.
- Uygulamalar

o Scratch programı kullanan öğrenciler için Scratch programında döngü oluşturmada kullanılan şekiller ve anlamları açıklanır.

4.1 Scratch ile ilgili etkinlik 7 yapılır.

o FCPRO programı kullanan öğrenciler için FCPRO programında döngü oluşturmada kullanılan şekiller ve anlamları açıklanır.

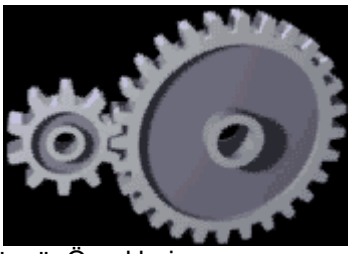
4.2 FCPRO ile ilgili etkinlik 8 yapılır.

4.1 Scratch Döngü Dersi Çalışma Sayfası 1

Algoritma: Problemin çözümünün adım adım yazılmasına algoritma denir.
Akış şeması: Algoritmanın sembollerle ifade edilmesidir.

Şekil 1'de verilen resimleri inceleyiniz.






Şekil 1 Döngü Örnekleri

"Resimlerde tekrar eden olaylar hangilerdir?

Cevap:

Döngü: Belirli işlemleri, belirli sayıda veya herhangi bir şart sağlanana kadar tekrarlamak amacı ile kullanılır.

Scratch Programında Döngü Komutları

Şekil2.Scratch programı
repeat komutu 1

Şekil 3. Scratch programı
repeat komutu 2

Şekil-2'de gösterilen repeat komutu eylemin belirli bir sayıda tekrarlanacağını gösterir. Şekil-3'te gösterilen repeat komutu ise eylemin belli bir koşul gerçekleşinceye dek süreceğini ifade eder.

4.1 Scratch Döngü Dersi Çalışma Sayfası 2

Etkinlik Numarası: 4

Konu: Döngü

Amaç: 1.Program içerisinde tekrar eden bir grupta döngüleri kullanır.

Bu etkinlikte sizlerle Scratch programındaki kedi karakterini algoritma yardımıyla 1'den 20'ye kadar olan sayıları sayacaktır. Program bittiğinde ekran görüntüsü Şekil-10'daki gibi olacaktır.

Bunun için gerekli kodları kod alanına sürüklemeniz ve algoritmayı çalıştırmanız gerekiyor. Bu iş için gerekli adımlar aşağıdadır. Başarılar.

Yönerge

1) Algoritmayı başlatmak için 

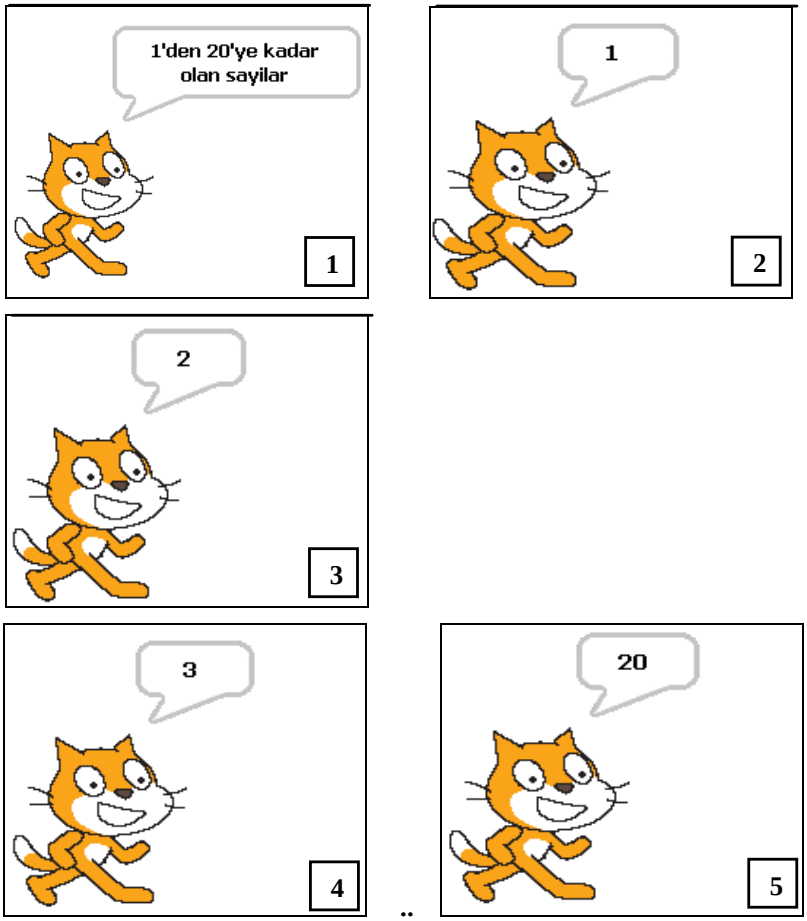
komutunu kullanınız.

2) "sayi" adinda bir değişken oluşturarak değişkene 1 değerini veriniz.

3) Bir döngü oluşturarak kedi karakterinin 1'den 20'ye kadar olan sayıları söylemesini sağlayın.

4) Sayıların döngü içerisinde 1 artırmayı unutmayın

3) Program çalıştırıldığında aşağıdaki gibi görünecektir. Karakter, şekildeki gibi 1'den 20'ye kadar olan sayıları sayacaktır.



Şekil 4 Scratch Etkinlik 4 ekran görüntüsü

4.2 Fcpro Döngü Dersi Çalışma Sayfası 1

Algoritma: Problemin çözümünün adım adım yazılmasına algoritma denir.
Akış şeması: Algoritmanın sembollerle ifade edilmesidir.

Şekil-1'de verilen resimleri inceleyiniz.




Şekil 1 Döngü Örnekleri

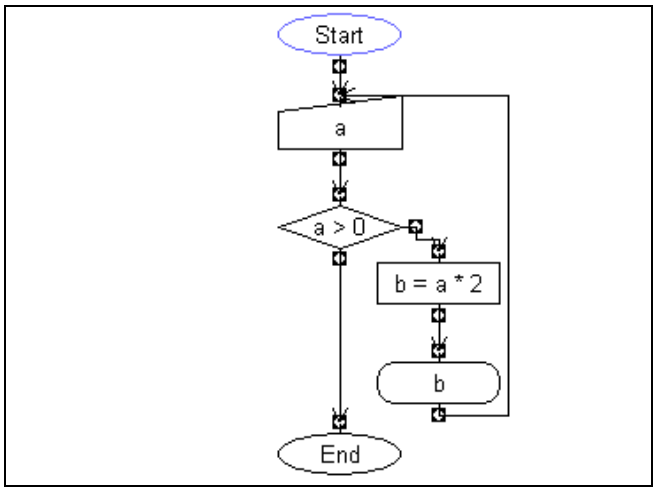
"Resimde tekrar eden olaylar hangilerdir?

Cevap:

Döngü: Belirli işlemleri, belirli sayıda veya herhangi bir şart sağlanana kadar tekrarlamak amacı ile
kullanılır.

FCPRO Programında Döngü Örneği

FCPRO programında bir döngü örneği aşağıda görülmektedir. Bu programda döngünün tekrarlanması istenen koşul oklarla gösterilerek döngünün oluşması sağlanır.



Şekil 2 FCPRO programında döngü örneği

4.2 Fcpro Döngü Dersi Çalışma Sayfası 2

Etkinlik Numarası: 4

Konu: Döngü

Amaç: 1.Program içerisinde tekrar eden bir grupta döngüleri kullanır.

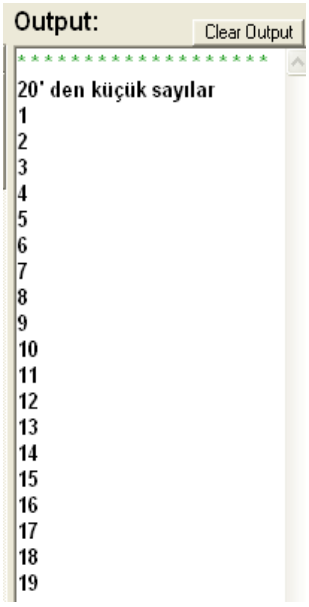
Bu etkinlikte sizlerle FCPRO programında algoritma yardımıyla algoritma ile 1'den 20'ye kadar olan sayıların yazdırılmasını sağlayacağız. Oluşturduğumuz algoritmanın ekran görüntüsü Şekil-3'teki gibi olacaktır.

Bunun için gerekli kodları kod alanına sürüklemeniz, gerekli kodları yazarak algoritmayı çalıştırmanız gerekiyor. Bu iş için gerekli adımlar aşağıdadır.

Başarılar.

Yönerge

1. FCPRO programında sayi adında bir değişken oluşturunuz. Bu değişkene 1 değerini veriniz.
2. 2. Ekrana "20'den küçük sayilar" yazdırınız.
3. Sayi değişkenini yazdırınız ve değişkeni bir arttırınız.
4. Sayi değeri 20 olana dek döngüyü sürdürünüz.
5. Program çalıştırıldığında ekran görüntüsü aşağıdaki gibi olacaktır.

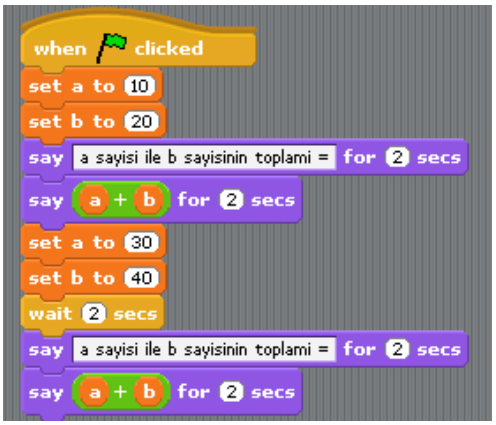


Şekil 3 FCPRO Etkinlik 4 ekran görüntüsü

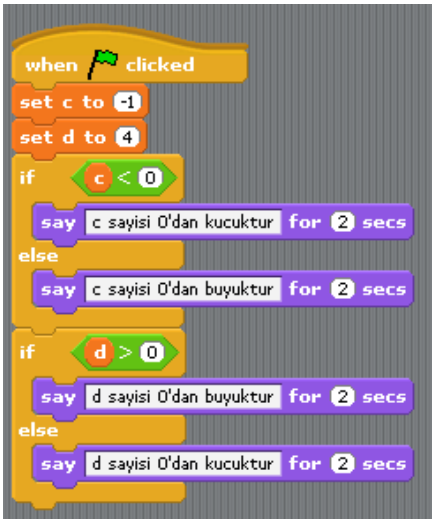
5. SCRATCH ETKİNLİKLERİ

1

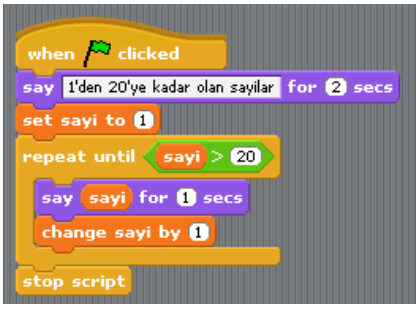
2



3

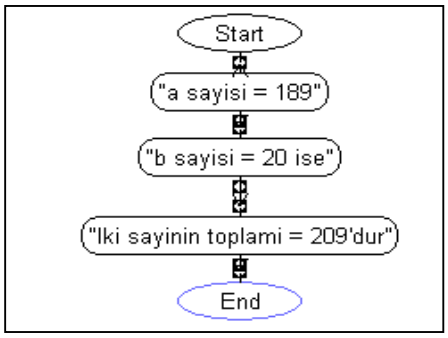


4

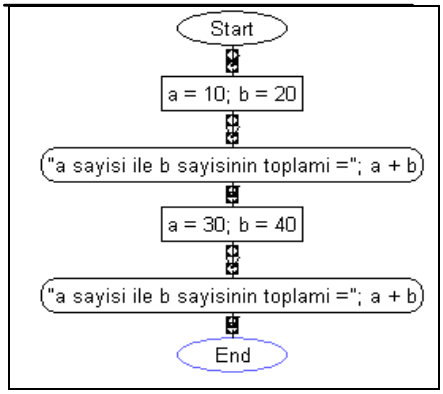


FCPRO ETKİNLİKLERİ

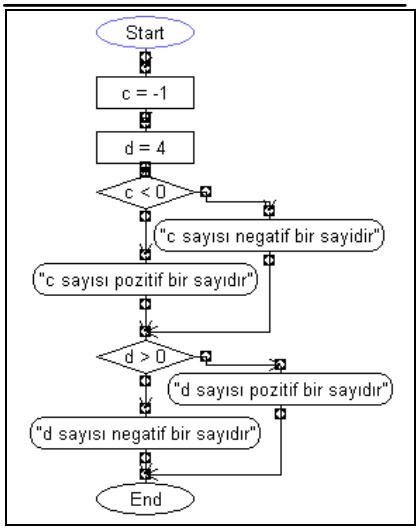
1



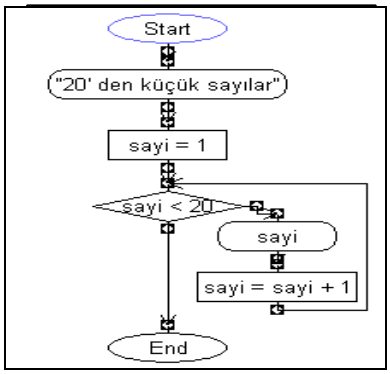
2



3



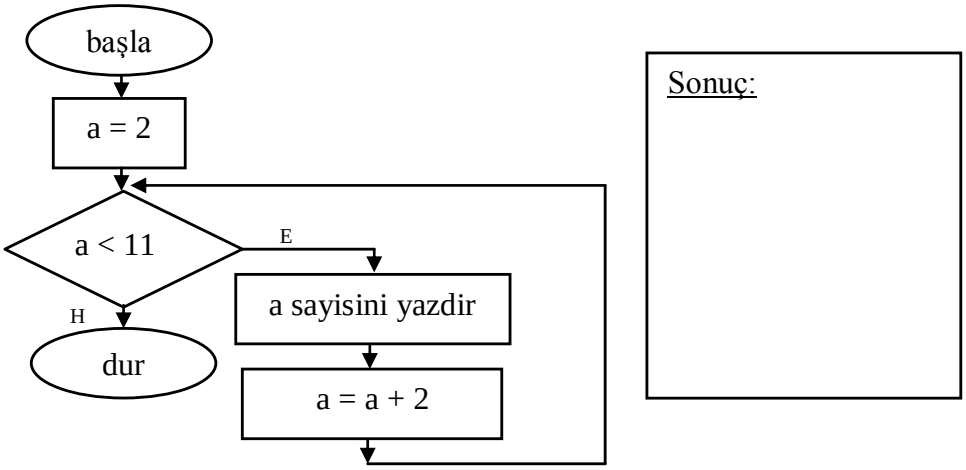
4



EK 3 SON TEST

1. Değişken, döngü ve koşul kavramlarını bir örnekle açıklayınız.

2. Aşağıdaki algoritmanın döndüreceği sonucu yanına yazınız.



3. Ekmek almaya gidiyorsunuz. Bunun için gerekli olan adımlar aşağıdaki gibidir. Adımları doğru biçimde sıralamak için aşağıdaki boşluklara numara veriniz.

__ Kasaya git.
__ Bakkala gittiğin yoldan eve gel.
__ Parayı öde.
__ Cüzdanını yanına al.
__ Bir ekmek al.
__ Bakkala git.

4. Bilgisayarınızdaki hoparlörden ses gelmiyor. Bununla ilgili olabilecek problemlerden birini seçiniz ve çözüm yollarını adımlar halinde sıralayınız.

5. Aşağıdaki soruları cevaplayınız ve cevaplarınızın veri türlerini yazın.

Soru	Cevap	Veri Türü
Yaşın	15	Sayı
Arkadaşının adı nedir?		
En sevdiğin TV programı nedir?		
Burcun nedir?		
Doğum tarihin nedir?		

6. Arkadaşınız havanın nasıl olduğunu soruyor. Siz de bugünkü hava durumunu izleyecek eğer hava sıcaklığı 0°'den küçükse "hava çok soğuk" eğer hava sıcaklığı 0 ile 10 derece arasında ise "hava soğuk", eğer hava sıcaklığı 10 ile 25 derece arasında ise "hava sıcaklığı normal" hava sıcaklığı 25 derecenin üzerinde ise "hava çok sıcak" yazacaksınız. Bununla ilgili adımları bir akış şeması halinde sıralayınız.

7. Aşağıdaki algoritmayı x'e 1'den 10'a kadar olan değer vererek uyguladığınızda sonuç ne olacaktır?

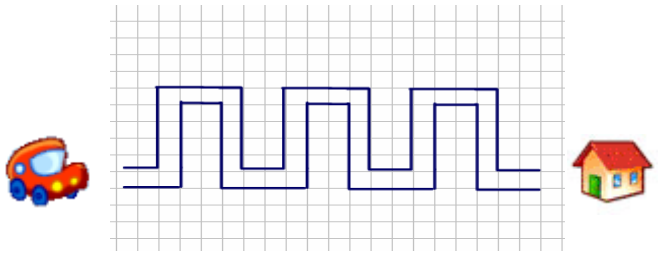
```
X= 1'den 10'a kadar tekrarla
{
        Eğer x = 3 veya x = 6 ise
                x sayısını 3 ile çarp
                sonucu yaz.
        Eğer x = 2 veya x = 4 ise
                x sayısını 2 ile çarp
sonucu yaz.
        }
```

8. Aşağıdaki ifadelerin yanlarına doğru ise (D) yanlış ise (Y) yazınız.

a) 14 / (2 + 3) * 5 = 14 (…)

b) Eğer x = 2 ise
(x * 2 > x + 2) ve (2 / 2 < 2 * 0) (…)

9. Aşağıda yolunu kaybetmiş bir sürücünün eve gitmek için izlemesi gereken yol verilmiştir. Buna göre sürücünün izleyeceği adımları en kısa biçimde sıralayınız.



10. Aşağıdaki algoritma doğru çalışır mı? Neden?

```
Adım1:  Başla
Adım2:  a = 1
Adım3:  eğer (a > 3)
        {
Adım4:      yazdır "a sayısı 3'ten küçüktür."
Adım5:      Adım8'e git.
        }
Adım6:  değilse
        {
Adım7:      Adım1'e git
        }
Adım8:  Dur
```

11. Ayşe'nin İngilizce not ortalaması puan cinsinden a olsun. Ayşe'nin İngilizce puanı 50'den küçükse "Puanınız 50'nin altında", puanı 50'den büyükse "Puanınız 50'nin üstünde" yazdıran bir algoritmanın akış şemasını çiziniz.

12. Suyun belli sıcaklıklardaki halleri aşağıda verilmiştir. Suyun sıcaklığı B olsun. B değeri;

a. 0'dan küçük ise "katı",
b. 0 ile 100□° arasında ise "sıvı",
c. 100º' den büyük ise "gaz"

yazdıran bir algoritmayı koşul cümlesi kullanarak oluşturunuz.

13. Koşul cümlesi kullanarak öğrencilerin notu girildiğinde;

d. Not = 1 ise "başarısız",
e. Not = 2 ise "geçer",
f. Not = 3 ise "orta",
g. Not = 4 ise "iyi",
h. Not = 5 ise "pekiyi" yazdıran bir programın algoritmasını yazınız.

14. Döngü kullanarak 0'dan 10'a kadar olan sayıları yazdırınız.

15. Bir aracın hızı a olsun. Bir döngü kullanarak aracın hızını 20 km artırarak hızın 60 km olmasını sağlayınız.

16. Döngü kullanarak 20 kez "okulumu çok seviyorum" yazdıran bir algoritmanın akış şemasını çiziniz.

EK 4 UYGULAMA SINAVI

1. Değerleri 5 ve 20 olan iki değişkeni çarpmak için gerekli olan adımları Scratch / Fcpro kullanarak sıralayıp bir algoritma oluşturunuz.

2. İki tane sayı türünde değişken oluşturup değişkenlere değer veriniz. Bu değişkenlerin toplanmasını ve sonucun ekranda görüntülenmesini sağlayın.

3. Önce sayi1 adında bir değişken oluşturarak değişkene 20 değerini atayınız. Daha sonra sayi2 adında bir değişken oluşturarak değişkene 30 değerini atayınız. Daha sonra ekrana "sayi1 sayisi ile sayi2 sayisinin çarpimi =" yazınız. Operatörleri kullanarak algoritmanızın çarpım işlemini yapmasını sağlayınız. Çarpım sonucunu ekrana yazdırınız.

4. Öncelikle sayi1 adında bir değişken oluşturarak değişkene 15 değerini atayınız. Algoritmanızın sayi1 değişkeninin 0'dan büyük olma durumunu koşul cümlesi kullanarak sınamasını sağlayınız. Eğer sayi1 değişkeni 0'dan büyükse ekrana "sayi1 sayisi pozitif bir sayidir", değilse " sayi1 sayisi negatif bir sayidir" yazdırınız.

5. Ekrana "1'den 10'a kadar olan cift sayilar" cümlesini yazdırınız. Daha sonra döngü kullanarak 1'den 10'a kadar olan çift sayıları yazdırınız.

EK 5 MOTİVASYON ÖLÇEĞİ

Aşağıdaki soruları dikkatle okuyup durumunuza uygun seçeneğin önündeki parantez (Kutucuğun) içine (x) işareti koyunuz. Bu cümlelerin tek doğru cevabı yoktur. Duygu ve düşüncelerinize uygun olan cevap sizin için doğrudur. Önemli olan sizin duygu, düşünce ve davranışlarınızdır. Bu nedenle arkadaşlarınızın cevapları ile ilgilenmeyiniz. Rakamların anlamları aşağıda belirtilmiştir

SIRA	MADDE	5. Tamamen Katılıyorum	4. Katılıyorum	3. Kararsızım	2. Katılmıyorum	1. Hiç Katılmıyorum
1.	Sınıfta kendimi yalnız hissediyorum					
2.	Sınıf içinde öğrendiklerim hakkında mantıklı bir değerlendirme yapabilirim					
3.	Derste sorulara cevap vermekten çekiniyorum					
4.	Derste öğrendiklerim ile gerçek hayat arasında ilişki kuramıyorum					
5.	Diğer öğrencilerin öğrenmesine yardım etmekten hoşlanıyorum					
6.	Sınıfta öğrendiklerim beni heyecanlandırmıyor					
7.	Sınıftaki tartışmalara hiç çekinmeden katılıyorum					
8.	Sınıfta dersle ilgili yapılan etkinlikleri yeterli bulmuyorum					
9.	Dersteki etkinlikler, derse aktif olarak katılmamı sağlıyor					
10.	Sınıfta düşüncelerimi açıkça ifade edebilecek kadar kendimi güvende hissetmiyorum					
11.	Benim için övgü ve onaylama önemlidir					
12.	Sınıfta hata yaptığımda,hatalı davranışımı fark ederek düzeltiyorum					
13.	Derste bizi mutlu edecek etkinliklere yer verilmiyor					
14.	Sınıf atmosferi ders için elverişli olduğunu düşünüyorum					
15.	Derste araç-gereçleri etkili olarak kullanabiliyorum					
16.	Dersin hedeflerini yeterli bulmuyorum					
17.	Derste yapılan tartışmalara katılmaktan hoşlanıyorum					
18.	Arkadaşlarım bana karşı genelde olumlu düşüncelere sahip olduklarını düşünmüyorum					
19.	Eleştirilere açık biri olduğumu düşünüyorum					
20.	Dersteki etkinliklere sıkça katılıyorum					
21.	Sınıf içerisindeki öğretmen ve öğrenci arasında ki bilgi akışı yeterli değil					
22.	Dersten daha fazla yararlanmak için değişik bilgi kaynaklarından yararlanmam					
23.	Derse yeterince motive olduğuma inanıyorum					
24.	Bilgi için öğretmenimle rahatlıkla iletişim kurabiliyorum					
25.	Öğretmenin benim hakkımdaki düşüncelerini önemsemiyorum					
26.	Derste öğrendiklerim ile gerçek hayat arsında ilişki kurabiliyorum					
27.	Değişik ortamlarda ve şekillerde ders yapmaktan zevk alıyorum					
28.	Derste başarılı olmam ve bundan dolayı takdir edilmem hoşuma gidiyor					
29.	Derste bir etkinliği gerçekleştirdiğimde mutlu oluyorum					
30.	Kendimle barışık bir insan olduğumu düşünmüyorum					

EK 6 UYGULAMA SINAVI KONTROL LİSTESİ

1. Değerleri 5 ve 20 olan iki değişkeni çarpmak için gerekli olan adımları programı kullanarak sıralayıp bir algoritma oluşturunuz.

   a. Gerekli olan işlemleri doğru biçimde belirler
   b. İşlemleri doğru sıralar
   c. Değişkenleri doğru kullanır
   d. Algoritma doğru sonucu döndürür

   20

2. İki tane sayı türünde değişken oluşturup değişkenlere değer veriniz. Bu değişkenlerin toplanmasını ve sonucun ekranda görüntülenmesini sağlayın.

   a. Gerekli olan işlemleri doğru biçimde belirler
   b. Değişken türünü doğru belirler
   c. Değişkenleri doğru kullanır
   d. Algoritma doğru sonucu döndürür

   20

3. Önce sayi1 adında bir değişken oluşturarak değişkene 20 değerini atayınız. Daha sonra sayi2 adında bir değişken oluşturarak değişkene 30 değerini atayınız. Daha sonra ekrana"sayi1 sayisi ile sayi2 sayisinin çarpimi =" yazınız. Operatörleri kullanarak algoritmanızın çarpım işlemini yapmasını sağlayınız. Çarpım sonucunu ekrana yazdırınız.

   a. Gerekli olan işlemleri doğru biçimde belirler
   b. Değişkenlere soruda istenilen ismi verir
   c. Değişkenlere soruda istenilen değeri verir
   d. Operatörleri kullanılarak verilen işlemleri doğru biçimde yapar
   e. Algoritma doğru sonucu döndürür

   20

4. Öncelikle sayi1 adında bir değişken oluşturarak değişkene 15 değerini veriniz. Algoritmanızın sayi1 değişkeninin 0'dan büyük olma durumunu koşul cümlesi kullanarak sınamasını sağlayınız. Eğer sayi1 değişkeni0'dan büyükse ekrana "sayi1 sayisi pozitif bir sayidir", değilse "sayi1 sayisi negatif bir sayidir" yazdırınız.

a. Gerekli olan işlemleri doğru biçimde belirler
b. Değişkenlere soruda istenen ismi verir
c. Değişkenlere soruda istenilen değeri verir
d. Koşul cümlesini oluşturur
e. Koşul cümlesinin istenen koşulu sınamasını sağlar
f. Algoritmanın koşul sonucuna göre istenen işlemi yapmasını sağlar
g. Algoritma doğru sonucu döndürür

20

5. Ekrana "1'den 10'a kadar olan cift sayilar" yazdırınız. Döngü kullanarak 1'den 10'a kadar olan çift sayıları yazdırınız.

a. Gerekli olan işlemleri doğru biçimde belirler
b. Değişkenleri doğru kullanır
c. Döngü cümlesini oluşturur
d. Döngünün hangi koşulda ya da kaç defa devam edeceğini belirler (Döngünün ne zaman sonlanacağını belirler)
e. Döngü içerisinde yapılacak işlemleri doğru belirler
f. Algoritma doğru sonucu döndürür.

20

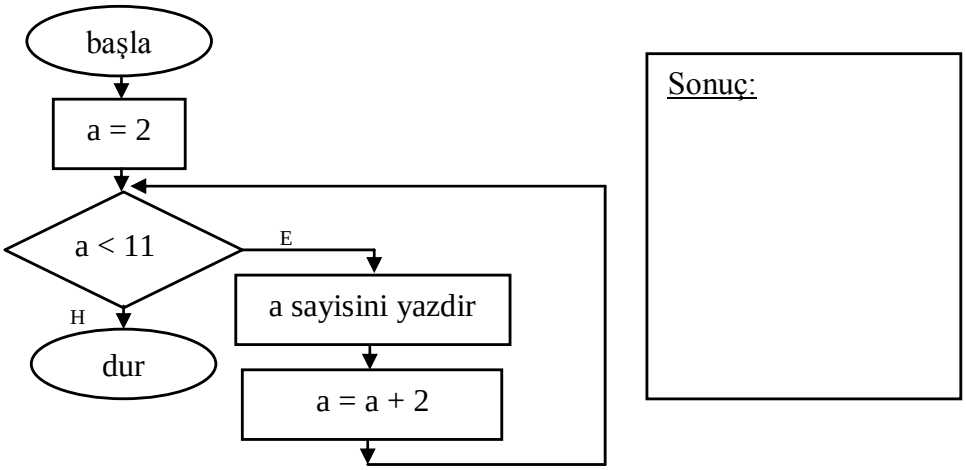
EK 7 SON TEST DEĞERLENDİRME ÖLÇEĞİ

1) Değişken, döngü ve koşul kavramlarını bir örnekle açıklayınız. (5 puan)

Hedef davranış: Algoritma ile ilgili temel kavramları bilir ve bunları bilgi verici bir biçimde ifade edebilir.

a. Değişken kavramının doğru olarak açıklar.
b. Koşul kavramının doğru olarak açıklar.
c. Döngü kavramının doğru olarak açıklar.
d. Kavramlarla ilgili örnek verir.

2) Aşağıdaki algoritmanın döndüreceği sonucu yanına yazınız. (5 puan)



Hedef davranış: Verilen bir algoritmayı takip ederek döndüreceği sonucu tahmin eder.

a. Akışı doğru takip ederek işlemleri doğru sırada yapar.
b. Algoritmada yer alan hesaplamaları doğru biçimde yapar.
c. Algoritmada değişken değerlerindeki değişimi gözlemler.
d. Algoritmanın döndüreceği sonucu doğru biçimde yazar.

3) Ekmek almaya gidiyorsunuz. Bunun için gerekli olan adımlar aşağıdaki gibidir. Adımları doğru biçimde sıralamak için aşağıdaki boşluklara numara veriniz (5 puan).

__ Kasaya git.

__ Bakkala gittiğin yoldan eve gel.
__ Parayı öde.
__ Cüzdanını yanına al.
__ Bir ekmek al.
__ Bakkala git.

Hedef davranış: Bir problemin çözümlenmesi için gerekli olan işlem adımlarını doğru biçimde sıralayabilir.

a. Mantıksal akışı doğru biçimde oluşturur.
b. Sıralamayı doğru yapar.

4) Bilgisayarınızdaki hoparlörden ses gelmiyor. Bununla ilgili olabilecek problemlerden birini seçiniz ve çözüm yollarını adımlar halinde sıralayınız. (5 puan)

Hedef davranış: Bir problemi küçük problemlere ayırarak çözüm yolu geliştirme

a. Problemin nedenini bulur ve doğru biçimde ifade eder.
b. Mantıklı bir çözüm yolu bulur.
c. Çözümü küçük adımlara böler.
d. Adımları doğru biçimde sıralar.

5) Aşağıdaki soruları cevaplayınız ve cevaplarınızın veri türlerini yazın. (5 puan)

Soru	Cevap	Veri Türü
Yaşın	15	Sayı
Arkadaşının adı nedir?		
En sevdiğin TV programı nedir?		
Burcun nedir?		
Doğum tarihin nedir?		

Hedef davranış: Değişkenlerin veri türlerini doğru biçimde belirleyebilir.

a. Cevapları yazarak, yazdığı cevap doğrultusunda veri türünü belirler.

6) Arkadaşınız havanın nasıl olduğunu soruyor. Siz de bugünkü hava durumunu izleyecek eğer hava sıcaklığı 0°'den küçükse "hava çok soğuk" eğer hava sıcaklığı 0 ile 10 derece arasında ise "hava soğuk", eğer hava sıcaklığı 10 ile 25 derece arasında ise "hava sıcaklığı normal" hava sıcaklığı 25 derecenin üzerinde ise "hava çok sıcak" yazacaksınız. Bununla ilgili adımları bir akış şeması halinde sıralayınız. (5 puan)

Hedef davranış: Belli bir koşulda belli eylemlerin yapıldığı bir olayın algoritmasını oluşturabilir.

a. Koşul cümlesini doğru kullanır.
   i. Koşul ifadesini oluşturur.
   ii. Koşul içinde yer alması gereken mantıksal sınamayı doğru biçimde yapar.
   iii. Koşul doğrultusunda algoritmanın gerekli işlemlerin yapabilmesi için gerekli ifadeyi oluşturur.
b. Problemi çözmek için gerekli adımları akış şeması kullanarak ifade eder.
c. Akış şemasının problemde istenen sonucu döndürmesini sağlar.

7) Aşağıdaki algoritmayı x'e 1'den 10'a kadar olan değer vererek uyguladığınızda sonuç ne olacaktır? (5 puan)

X= 1'den 10'a kadar tekrarla
{
Eğer x = 3 veya x = 6 ise
x sayısını 3 ile çarp

sonucu yaz.

Eğer x = 2 veya x = 4 ise

x sayısını 2 ile çarp

sonucu yaz.

}

Hedef davranış: Verilen bir algoritmada yer alan değişkenlere değer vererek gerekli işlemleri yapar.

a. Algoritmayı takip ederek yapılacak işlem sırasını doğru belirler.
b. Problemin çözümü için algoritmada verilen hesaplamaları yapar.
c. Algoritma çalıştırıldığında değişkenlerdeki değerlerin değişimini doğru biçimde ifade eder.
d. Algoritmanın problemde istenen sonucu döndürmesini sağlar.

8) Aşağıdaki ifadelerin yanlarına doğru ise (D) yanlış ise (Y) yazınız. (5 puan)

a) 14 / (2 + 3) * 5 = 14 (…)

b) Eğer x = 2 ise

(x * 2 > x + 2) ve (2 / 2 < 2 * 0) (…)

Hedef davranış: Operatörleri kullanarak verilen işlemleri yapabilir.

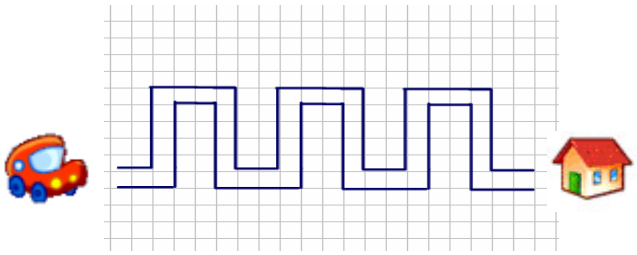
a şıkkı için;

a. Gerekli hesaplamaları yapar.
b. İfadenin doğruluğunu belirler.

b şıkkı için;

a. Değişkene soruda istenen değeri verir.
b. Gerekli hesaplamaları ve karşılaştırmaları yapar.
c. İfadenin doğruluğunu belirler.

9) Aşağıda yolunu kaybetmiş bir sürücünün eve gitmek için izlemesi gereken yol verilmiştir. Buna göre sürücünün izleyeceği adımları en kısa biçimde sıralayınız. (5 puan)



Hedef davranış: Verilen bir problemin çözülmesi için belli bir süre ya da sayıca tekrar edilen bir olayı küçük adımlar halinde döngü kullanarak algoritmaya dönüştürebilir.

a. Mantıksal akışı doğru biçimde oluşturur.
b. Döngü cümlesini doğru biçimde kullanır.
    i. Döngü ifadesini oluşturur.
    ii. Döngü içerisinde yapılacak işlemleri doğru belirler.
    iii. Döngünün hangi koşulda durdurulacağını doğru biçimde ifade eder.
c. Algoritmanın istenen sonucu döndürmesini sağlar.

10) Aşağıdaki algoritma doğru çalışır mı? Neden? (5 puan)

```
Adım1:  Başla
Adım2:  a = 1
Adım3:  eğer (a > 3)
        {
Adım4:      yazdır "a sayısı 3'ten küçüktür."
Adım5:      Adım8'e git.
        }
Adım6:  değilse
        {
```

Adım7: Adım1'e git

}

Adım8: Dur

Hedef davranış: Verilen bir algoritmayı inceleyerek algoritma ile ilgili sorgulama yapabilir.

a. Algoritmayı takip ederek doğru cevabı yazar.
b. Nedeni algoritmanın sonlanmamasına dayalı olarak açıklar.

11) Ayşe'nin İngilizce not ortalaması puan cinsinden a olsun. Ayşe'nin İngilizce puanı 50'den küçükse "Puanınız 50'nin altında", puanı 50'den büyükse "Puanınız 50'nin üstünde" yazdıran bir algoritmanın akış şemasını çiziniz. (9 puan)

Hedef davranış: Belli bir koşulda belli eylemlerin yapıldığı bir olayın algoritmasını oluşturur.

a. Problemi çözerken değişken kullanır.
b. Koşul cümlesini doğru kullanır.
    i. Koşul ifadesini oluşturur.
    ii. Koşul içinde yer alması gereken mantıksal sınamayı doğru biçimde yapar.
    iii. Koşul doğrultusunda algoritmanın gerekli işlemlerin yapabilmesi için gerekli ifadeyi oluşturur.
c. Problemi çözmek için gerekli adımları akış şeması kullanarak ifade eder.
d. Akış şemasının problemde istenen sonucu döndürmesini sağlar.

12) Suyun belli sıcaklıklardaki halleri aşağıda verilmiştir. Suyun sıcaklığı B olsun. B değeri;

a. 0'dan küçük ise "katı",
b. 0 ile 100º arasında ise "sıvı",
c. 100º' den büyük ise "gaz"

yazdıran bir algoritmayı koşul cümlesi kullanarak oluşturunuz. (8 puan)

Hedef davranış: Belli bir koşulda belli eylemlerin yapıldığı bir olayın algoritmasını oluşturur.

a. Problemi çözerken değişken kullanır.
b. Koşul cümlesini doğru kullanır.
    i. Koşul ifadesini oluşturur.
    ii. Koşul içinde yer alması gereken mantıksal sınamayı doğru biçimde yapar.
    iii. Koşul doğrultusunda algoritmanın gerekli işlemlerin yapabilmesi için gerekli ifadeyi oluşturur.
c. Algoritmanın doğru sonucu döndürmesini sağlar.

13) Koşul cümlesi kullanarak öğrencilerin notu girildiğinde;

a. Not = 1 ise "başarısız",
b. Not = 2 ise "geçer",
c. Not = 3 ise "orta",
d. Not = 4 ise "iyi",
e. Not = 5 ise "pekiyi" yazdıran bir programın algoritmasını yazınız. (8 puan)

Hedef davranış: Belli bir koşulda belli eylemlerin yapıldığı bir olayın algoritmasını oluşturur.

a. Problemi çözerken değişken kullanır.
b. Koşul cümlesini doğru kullanır.
    i. Koşul ifadesini oluşturur.
    ii. Koşul içinde yer alması gereken mantıksal sınamayı doğru biçimde yapar.
    iii. Koşul doğrultusunda algoritmanın gerekli işlemlerin yapabilmesi için gerekli ifadeyi oluşturur.
c. Algoritmanın doğru sonucu döndürmesini sağlar.

14) Döngü kullanarak 0'dan 10'a kadar olan sayıları yazdırınız. (8 puan)

Hedef davranış: Verilen bir problemin çözülmesi için belli bir süre ya da sayıca tekrar edilen bir olayı küçük adımlar halinde döngü kullanarak algoritmaya dönüştürebilir.

a. Mantıksal akışı doğru biçimde oluşturur.
b. Döngü cümlesini doğru biçimde kullanır.
    i. Döngü ifadesini oluşturur.
    ii. Döngü içerisinde yapılacak işlemleri doğru belirler.
    iii. Döngünün hangi koşulda durdurulacağını doğru biçimde ifade eder.
c. Algoritmanın istenen sonucu döndürmesini sağlar.

15) Bir aracın hızı a olsun. Bir döngü kullanarak aracın hızını 20 km artırarak hızın 60 km olmasını sağlayınız. (8 puan)

Hedef davranış: Verilen bir problemin çözülmesi için belli bir süre ya da sayıca tekrar edilen bir olayı küçük adımlar halinde döngü kullanarak algoritmaya dönüştürebilir.

a. Mantıksal akışı doğru biçimde oluşturur.
b. Döngü cümlesini doğru biçimde kullanır.
    i. Döngü ifadesini oluşturur.
    ii. Döngü içerisinde yapılacak işlemleri doğru belirler.
    iii. Döngünün hangi koşulda durdurulacağını doğru biçimde ifade eder.
c. Algoritmanın istenen sonucu döndürmesini sağlar.

16) Döngü kullanarak 20 kez "okulumu çok seviyorum" yazdıran bir algoritmanın akış şemasını çiziniz. (9 puan)

Hedef davranış: Verilen bir problemin çözülmesi için belli bir süre ya da sayıca tekrar edilen bir olayı küçük adımlar halinde döngü kullanarak algoritmaya dönüştürebilir.

a. Mantıksal akışı doğru biçimde oluşturur.
b. Döngü cümlesini doğru biçimde kullanır.
    i. Döngü ifadesini oluşturur.
    ii. Döngü içerisinde yapılacak işlemleri doğru belirler.
    iii. Döngünün hangi koşulda durdurulacağını doğru biçimde ifade eder.
c. Akış şeması çizer.
d. Algoritmanın istenen sonucu döndürmesini sağlar.