

**T.C.
ERCIYES ÜNİVERSİTESİ
EĞİTİM BİLİMLERİ ENSTİTÜSÜ
MATEMATİK VE FEN BİLİMLERİ ANA BİLİM DALI**

**SCRATCH KULLANIMININ MATEMATİKSEL
PROBLEM ÇÖZME BECERİLERİNE ETKİSİ**

**Hazırlayan
ELİF ÇELİK**

Yüksek Lisans Tezi

**Nisan 2024
KAYSERİ**

**T.C.
ERCIYES ÜNİVERSİTESİ
EĞİTİM BİLİMLERİ ENSTİTÜSÜ
MATEMATİK VE FEN BİLİMLERİ ANA BİLİM DALI
MATEMATİK EĞİTİMİ BİLİM DALI**

**SCRATCH KULLANIMININ MATEMATİKSEL
PROBLEM ÇÖZME BECERİLERİNE ETKİSİ**

**Hazırlayan
ELİF ÇELİK**

**Danışman
DOÇ. DR. SEVİM SEVGİ**

**Nisan 2024
KAYSERİ**

BİLİMSEL ETİĞE UYGUNLUK

Erciyes Üniversitesi Eğitim Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmada,

- Tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- Görsel, işitsel ve yazılı bütün bilgi ve sonuçları bilimsel etik kurallarına uygun olarak sunduğumu,
- Başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- Atıfta bulunduğum eserlerin bütününe kaynak olarak gösterdiğimi,
- Kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- Bu tezin herhangi bir bölümünü bu üniversitede veya başka bir üniversitede başka bir bilimsel çalışması olarak sunmadığımı

beyan ederim.

Elif Çelik

İmza

YÖNERGEYE UYGUNLUK

Scratch Kullanımının Matematiksel Problem Çözme Becerilerine Etkisi adlı Yüksek Lisans Tezi, Erciyes Üniversitesi Lisansüstü Tez Önerisi ve Tez Yazma Yönergesi'ne uygun olarak hazırlanmıştır.

Hazırlayan
Elif ÇELİK

Danışman
Doç. Dr. Sevim SEVGİ

KABUL VE ONAY

Eğitim Bilimleri Enstitüsü Müdürlüğüne,

Elif Çelik'in hazırladığı **Scratch Kullanımının Matematiksel Problem Çözme Becerilerine Etkisi** başlıklı bu çalışma jürimiz tarafından **Matematik ve Fen Bilimleri Eğitimi Ana Bilim Dalı, Matematik Eğitimi Bilim Dalında Yüksek Lisans** olarak kabul edilmiştir.

Jüri Başkanı	Dr. Öğretim Üyesi Özge GÜN	
Jüri Üyesi (Danışman)	Doç. Dr. Sevim SEVGİ	
Jüri Üyesi	Prof. Dr. Fatma BOZKURT YOUSEF	

Bu tez Erciyes Üniversitesi Lisansüstü Eğitim, Öğretim ve Sınav Yönetmeliği'nin ilgili maddeleri uyarınca yukarıdaki jüri üyeleri tarafından ... / ... / tarihinde uygun görülmüş ve Enstitü Yönetim Kurulunca ... / ... / tarihi ve sayılı karar ile kabul edilmiştir.

Prof. Dr. Hüseyin ARAK
Eğitim Bilimleri Enstitüsü Müdürü

ÖNSÖZ

Yüksek lisans eğitimim ve çalışmalarım süresince yardımlarını ve değerli zamanını esirgemeyen, tezin hazırlanması aşamasında görüş ve önerileri ile yol gösteren, bana hep destek olan sevgili danışman hocam Doç. Dr. Sevim SEVGİ'ye sonsuz teşekkürlerimi sunarım.

Tezimi inceleyen ve değerlendiren; değerli yorumları ve önerileri ile tez çalışmamı geliştirmeme yardımcı olan hocalarım Prof. Dr. Fatma BOZKURT YOUSEF'a ve Dr. Öğr. Üyesi Özge GÜN'e teşekkürlerimi sunarım.

Yüksek lisans eğitimime başladığım günden itibaren çalışmalarım beni teşvik eden, cesaretlendiren, bana her zaman destek olup hiçbir zaman yalnız bırakmayan, her daim motive eden, hayatım boyunca üzerimde en büyük emeği olan annem Ülkü ÇELİK'e, babam Nurettin ÇELİK'e ve bendeki yeri her zaman ayrı olacak olan ve her zaman destekleriyle bana güç veren kız kardeşlerim Ayşenur ÇELİK'e ve Zeynep ÇELİK'e teşekkürü bir borç bilirim.

Elif ÇELİK

Nisan 2024, KAYSERİ

ÖZ

SCRATCH KULLANIMININ MATEMATİKSEL PROBLEM ÇÖZME BECERİLERİNE ETKİSİ

Elif ÇELİK

Erciyes Üniversitesi, Eğitim Bilimleri Enstitüsü
Yüksek Lisans Tezi, Nisan 2024
Danışman: Doç. Dr. Sevim SEVGİ

Bu araştırmanın amacı, “Scratch ile hazırlanmış matematiksel problemlerin ortaokul 8. sınıf öğrencilerinin problem çözme becerileri üzerindeki etkisini incelemektir. Araştırmada karma araştırma yaklaşımlarından açıklayıcı desen esas alınmıştır. Bu doğrultuda araştırmanın nicel verileri deneysel desenlerden zayıf deneysel desen olan tek grup öntest-sontest desen kullanılarak toplanmıştır. Araştırmanın nicel verilerini tamamlamak üzere nitel veriler görüşme tekniği ile toplanmıştır. 2022-2023 eğitim öğretim yılında yürütülmüş olan araştırmanın çalışma grubu Yozgat ili Merkez ilçesinde bulunan İmam Hatip Ortaokulunun 8. sınıfında öğrenim gören sekiz öğrenciden oluşmaktadır. İlk aşamada veri toplama aracı olarak katılımcılara matematiksel problemler içeren öntest uygulanmıştır. İkinci aşamada araştırmacı tarafından geliştirilen ve matematiksel durumlar içeren Scratch programı ile oluşturulmuş kodlama etkinlikleri öğrencilerle yapılmıştır. Etkinliğin sonunda öğrencilere sontest uygulanmıştır. Daha sonra öntest ile sontest sonuçları bağımlı örneklem t-testi ile analiz edilmiştir. Son olarak 6 gönüllü öğrenci ile yarı yapılandırılmış görüşmeler yapılmıştır. Yarı yapılandırılmış görüşmelerden elde edilen veriler betimsel analiz tekniği ile analiz edilmiştir. Araştırmadan elde edilen verilere göre grubun öntest ile sontest sonuçları arasında anlamlı bir fark olmadığı görülmüştür. İlk ve ortaöğretimde kompütersel düşünmenin mevcut öğretim programına entegrasyonu ve akademik başarıya etkilerinin değerlendirilmesi konusunda stratejik bir odaklanmanın gerekliliğine yönelik tavsiyelerde bulunulmuştur.

Anahtar sözcükler: kompütasyonel düşünme, ortaokul öğrencileri, problem çözme, Scratch



ABSTRACT

THE EFFECT OF SCRATCH USE ON MATHEMATICAL PROBLEM-SOLVING SKILLS

Elif ÇELİK

Erciyes University, Graduate School of Educational Sciences

Master's Thesis, April 2024

Supervisor: Assoc. Prof. Dr. Sevim SEVGİ

The aim of the study is to investigate the effect of mathematical problems prepared with Scratch on the problem-solving skills of 8th grade middle school students. The study was based on explanatory design from mixed research approaches. Accordingly, the quantitative data of the study were collected using a one-group pretest-posttest design, which is a weak experimental design. Qualitative data were collected by interview technique to complement the quantitative data of the research. The study group of the research, which was conducted in the 2022-2023 academic year, consists of eight students studying in the 8th grade of the Imam Hatip Middle School in the Yozgat. In the first stage, a pretest containing mathematical problems was applied to the participants as a data collection tool. In the second stage, coding activities developed by the researcher and created with the Scratch program containing mathematical situations were carried out with the students. At the end of the activity, a posttest was administered to the students. Then, the results of the pretest and posttest were analyzed by dependent sample t-test. Finally, semi-structured interviews were conducted with 6 volunteer students. The data obtained from semi-structured interviews were analyzed with descriptive analysis technique. According to the data obtained from the study, there was no significant mean difference between the pretest and posttest results of the group. Recommendations were made regarding the need for a strategic focus on the integration of computational thinking into the current curriculum in primary and middle school education and the evaluation of its effects on academic achievement.

Keywords: computational thinking, middle school students, problem-solving skills, Scratch



İÇİNDEKİLER

BİLİMSEL ETİĞE UYGUNLUK.....	i
YÖNERGEYE UYGUNLUK.....	ii
KABUL VE ONAY	iii
ÖNSÖZ	iv
ÖZ.....	v
ABSTRACT.....	vii
İÇİNDEKİLER.....	ix
TABLOLAR LİSTESİ	xii
ŞEKİLLER LİSTESİ.....	xiv
SİMGELER VE KISALTMALAR LİSTESİ	xvi
BÖLÜM I.....	1
GİRİŞ	1
1.1. Problem Durumu.....	2
1.2. Araştırmanın Önemi	2
1.3. Araştırmanın Amacı ve Alt Amaçları	4
1.4. Sayıtlar ve Sınırlılıklar	5
1.5. Tanımlar.....	5
BÖLÜM II.....	7
ARAŞTIRMANIN KURAMSAL TEMELİ VE İLGİLİ ARAŞTIRMALAR.....	7
2.1. Problem Çözme	7
2.2. Kodlama Yoluyla Problem Çözme.....	13
2.3. Matematik ve Kodlamayı İlişkilendiren Çalışmalar	15
2.4. Programlama	17
2.4.1. Logo.....	17
2.4.2. Scratch	21
2.5. Kompütasyonel Düşünme	26
2.6. Scratch ve Kompütasyonel Düşünme Kavramları	33
2.6.1. Kompütasyonel Düşünme Kavramları	35
2.6.1.1. Olaylar	35
2.6.1.2. Algoritmalar ve Diziler	36
2.6.1.3. Koşullar.....	36
2.6.1.4. Döngüler.....	37

2.6.1.5. Değişkenler	37
2.6.1.6. Operatörler	38
2.6.1.7. Paralellik	39
2.6.2. <i>Kompütasyonel Düşünme Uygulamaları</i>	39
2.6.2.1. Yineleme	39
2.6.2.2. Soyutlama	40
2.6.2.3. Karıştırma	40
2.6.2.4. Hata Ayıklama ve Test Etme	40
2.6.3. <i>Kompütasyonel Düşünme Bakış Açılı</i>	41
2.6.3.1. Yeniden ifade etme	41
2.6.3.2. Bağlantı kurma	41
2.6.3.3. Sorgulama	41
2.7. FeTEMM (STEM) ve Kodlama.....	42
BÖLÜM III	46
YÖNTEM.....	46
3.1. Araştırmanın Modeli.....	46
3.2. Araştırma Grubu	47
3.3. Veri Toplama Araçları.....	47
3.3.1. <i>Problem Çözme Envanteri</i>	47
3.3.2. <i>Yarı Yapılandırılmış Görüşme Formu</i>	48
3.4. Araştırmacının Rolü.....	49
3.5. Verilerin Toplanması	50
3.5.1. <i>“Etkinlik 1: Soru-Cevap” Uygulama Süreci</i>	52
3.5.2. <i>“Etkinlik 2: Asal Mı?” Uygulama Süreci</i>	53
3.5.3. <i>“Etkinlik 3: Pozitif Tam Sayı Bölenleri” Uygulama Süreci</i>	54
3.5.4. <i>“Etkinlik 4: Aralarında Asal Mı?” Etkinliği Uygulama Süreci</i>	55
3.5.5. <i>“Etkinlik 5: EBOB ve EKOK Hesaplama Programı” Etkinliği Uygulama Süreci</i>	56
3.6. Verilerin Analizi	58
3.7. Geçerlik ve Güvenirlik	59
BÖLÜM IV.....	61
BULGULAR.....	61
4.1. Öntest ve Sontest Betimsel Analiz Sonuçlarına Yönelik Bulgular	61
4.1.1. <i>Öntest ve Sontest Frekans ve Yüzde Dağılımlarına İlişkin Bulgular</i>	62

4.1.1.2. Asal Çarpanlarına Ayırma Problemleri	65
4.1.1.3. Aralarında Asal Sayılar Problemleri	73
4.1.1.4. EBOB Problemleri	79
4.1.1.5. EKOK Problemleri	84
4.1.1.6. Pozitif Bölen Sayısı Problemi	88
4.2. Öntest ve Sontest Yordamsal Analiz Sonuçlarına Yönelik Bulgular	89
4.3. Scratch ile Tasarlanmış Matematiksel Durumlara Yönelik Öğrenci Görüşlerine İlişkin Bulgular	91
4.3.1. Scratch ile Tasarlanmış Matematiksel Durumların Olumlu Yönlerine İlişkin Bulgular ..	91
4.3.2. Scratch ile Tasarlanmış Matematiksel Durumların Olumsuz Yönlerine İlişkin Bulgular	93
4.3.3. Scratch ile Tasarlanmış Matematiksel Durumların Zorluklarına İlişkin Bulgular	95
BÖLÜM V	97
TARTIŞMA, SONUÇ VE ÖNERİLER	97
5.1. Sonuç ve Tartışma	97
5.2. Öneriler	103
KAYNAKÇA	106
EKLER	125

TABLÖLAR LİSTESİ

Tablo 1	Kompütasyonel düşünme becerisine ait kavramlar	31
Tablo 2	Kompütasyonel düşünmenin üç boyutu (Brennan ve Resnick, 2012).....	35
Tablo 3	Tek gruplu ön test- son test modeli.....	47
Tablo 4	Planlanan Scratch etkinlikleri	51
Tablo 5	Taylor'ın (2019) problem çözme süreci ve “Soru-Cevap” kodlama etkinliği	52
Tablo 6	Taylor'ın (2019) problem çözme süreci ve “Asal Mı?” kodlama etkinliği...	53
Tablo 7	Taylor'ın (2019) problem çözme süreci ve “Pozitif Tam Sayı Bölenleri” kodlama etkinliği.....	54
Tablo 8	Taylor'ın (2019) problem çözme süreci ve “Aralarında Asal Mı?” kodlama etkinliği.....	56
Tablo 9	Taylor'ın (2019) problem çözme süreci ve “EBOB ve EKOK Hesaplama Programı” kodlama etkinliği	57
Tablo 10	Öğrencilerin Scratch ile tasarlanmış matematiksel algoritmalar öntest-sontest puan değerleri.....	62
Tablo 11	Scratch kullanımının matematiksel problem çözme becerilerine etkisi öntest ve sonteste ait frekans ve yüzde dağılımları.....	63
Tablo 12	Soruya ait yüzde ve frekans dağılımları	64
Tablo 13	Asal çarpanlara ayırma içeren problemlere ait istatistiksel dağılım	65
Tablo 14	Aralarında asal sayı problemlerine ait öğrenci cevaplarının frekans ve yüzde dağılımları	73
Tablo 15	EBOB Problemlerine Ait Frekans ve Yüzde Dağılımları.....	79
Tablo 16	En Küçük Ortak Kat Problemlerine Ait Frekans ve Yüzde Dağılımları	84
Tablo 17	Pozitif bölen sayısı problemine (s5) ait frekans ve yüzde dağılımları.....	88
Tablo 18	Shapiro–Wilk normallik testi.....	90
Tablo 19	Bağımlı örneklem t-testi analiz sonuçları	90
Tablo 20	Scratch ile oluşturulan matematiksel algoritmaların olumlu yönleri.....	91
Tablo 21	Scratch ile oluşturulan matematiksel algoritmaların olumsuz yönleri.....	94
Tablo 22	Scratch ile oluşturulan matematiksel algoritmaların anlaşılmayan yönleri	95
Tablo 23	Etkinlerin tamamlanma oranları	96



ŞEKİLLER LİSTESİ

Şekil 1 Kompütasyonel düşünme ile problem çözme modeli.....	10
Şekil 2 Logo Kaplumbağa.....	19
Şekil 3 Scratch arayüzü.....	24
Şekil 4 Scratch, kompütasyonel düşünme ve problem çözme	32
Şekil 5 Karar, tekrar ve koşullar	37
Şekil 6 Değişkenler	38
Şekil 7 Operatörler	39
Şekil 8 Kodlama ve matematik ilişkisi	44
Şekil 9 Diyalog tabanlı soru cevap etkinliği içeren örnek kodlama etkinliği (Scratch sayfası)	53
Şekil 10 Asal Mıdır? adlı örnek kodlama etkinliği (Scratch sayfası)	54
Şekil 11 Pozitif tam bölen sayısı örnek kodlama etkinliği (Scratch sayfası).....	55
Şekil 12 Aralarında Asal Mı? adlı örnek kodlama etkinliği (Scratch sayfası).....	56
Şekil 13 EBOB ve EKOK hesaplama örnek kodlama etkinliği (Scratch sayfası)	58
Şekil 14 Öntest (solda) sontest (sağda) karşılaştırmalı öğrenci cevap kâğıdı.....	64
Şekil 15 Asal sayı içeren örnek öğrenci kodlama etkinliği (Scratch sayfası)	65
Şekil 16 Doğru kabul edilen cevap kâğıdı örneği	66
Şekil 17 Öğrenci kodlama etkinliği örneği (solda) ve kod blokları (sağda) (Scratch sayfası)	66
Şekil 18 Öntest (solda) ve sontest (sağda) karşılaştırmalı öğrenci cevap kâğıdı	67
Şekil 19 Öğrenci kodlama etkinliği örneği (solda) ve kod blokları (sağda) (Scratch sayfası)	68
Şekil 20 Sontest öğrenci cevap kâğıdı örneği	68
Şekil 21 Kodlama etkinliği öğrenci örneği (solda) ve etkinliğe ait kod blokları (sağda) (Scratch sayfası)	69
Şekil 22 Öntest (solda) ve sontest (sağda) öğrenci cevap kâğıdı örneği.....	69
Şekil 23 Asal sayı kodlama etkinliği öğrenci etkinliği örneği (solda) ve etkinliğe ait kod blokları (sağda) (Scratch sayfası)	70
Şekil 24 Öntest (sağda) sontest (solda) karşılaştırmalı öğrenci cevap kâğıdı örneği.	71

Şekil 25 Asal Çarpan İçeren Örnek Öğrenci Kodlaması (Scratch sayfası).....	72
Şekil 26 Öntest (solda) sontest (sağda) karşılaştırmalı öğrenci cevap kâğıdı örneği.	73
Şekil 27 Öntest (solda) sontest (sağda) karşılaştırmalı öğrenci cevabı örneği.....	74
Şekil 28 Öntest (solda) sontest (sağda) karşılaştırmalı öğrenci cevap kâğıdı örneği	75
Şekil 29 Aralarında asal örnek öğrenci kodlama etkinliği (solda) ve etkinliğe ait kod blokları (sağda) (Scratch sayfası)	76
Şekil 30 Öntest (solda) ve sontest (sağda) karşılaştırmalı öğrenci cevap kâğıdı örneği	76
Şekil 31 Öntest (solda) ve sontest (sağda) karşılaştırmalı öğrenci cevap kâğıdı örneği	77
Şekil 32 Aralarında asal sayılarla ilgili kodlama etkinliği örneği (Scratch sayfası)).	77
Şekil 33 Öntest (solda) ve sontest (sağda) karşılaştırmalı cevap kâğıdı örneği	78
Şekil 34 Aralarında asal sayılar öğrenci kodlama örneği (Scratch sayfası).....	79
Şekil 35 En büyük ortak bölen öğrenci cevap kâğıdı örneği öntest (solda) ve sontest (sağda)	80
Şekil 36 En büyük ortak bölen içeren öğrenci kodlama etkinliği örneği (Scratch sayfası)	81
Şekil 37 Öntest (solda) sontest (sağda) öğrenci cevap kâğıdı örneği.....	81
Şekil 38 En büyük ortak bölen içeren öğrenci kodlama etkinliği örneği (Scratch sayfası)	82
Şekil 39 Öntest (solda), sontest (sağda) öğrenci cevap kâğıdı örneği.....	83
Şekil 40 Öntest (solda), sontest (sağda) öğrenci cevap kâğıdı örneği.....	83
Şekil 41 Öğrenci cevap kâğıdı örneği	84
Şekil 42 Öntest (solda) sontest (sağda) öğrenci cevap kâğıdı örneği.....	85
Şekil 43 En küçük ortak kat içeren öğrenci kodlama etkinliği ve kodlama etkinliği kod blokları örneği (Scratch sayfası)	86
Şekil 44 Yanlış kabul edilen öğrenci cevap kâğıdı örneği	86
Şekil 45 Öğrenci kodlama etkinliği ve kod blokları örneği (Scratch sayfası)	87
Şekil 46 Yanlış kabul edilen öğrenci cevap kâğıdı örneği	88
Şekil 47 Öntest (solda) sontest (sağda) öğrenci cevap kâğıdı örneği.....	89
Şekil 48 Pozitif bölen bulmayı içeren öğrenci kodlama etkinliği ve kod blokları örneği (Scratch sayfası)	89

SİMGELER VE KISALTMALAR LİSTESİ

ACARA: The Australian Curriculum, Assessment and Reporting Authority
ANTLR: ANother Tool for Language Recognition
B2C3Math: Based Coding and Computational Thinking for Conceptual Mathematics
CAS: Computing at School
CSTA: Computer Science Teachers Association
CCSS: Common Core State Standards
EBOB: En Büyük Ortak Bölen
EKOK: En Küçük Ortak Kat
HGSE: Harvard Graduate School of Education
ISTE: The International Society for Technology in Education
MEB: Millî Eğitim Bakanlığı
MIT: Massachusetts Institute of Technology
NCTM: National Council of Teachers of Mathematics
OECD: Organisation for Economic Co-operation and Development
ÖDSGM: Ölçme, Değerlendirme ve Sınav Hizmetleri Genel Müdürlüğü
TTKB: Talim ve Terbiye Kurulu

BÖLÜM I

GİRİŞ

Bugünün modern toplumu bilgi toplumudur. Çeşitli sosyal değişimler bu yönleri okul hayatına da yansımakta, yaratıcı ve yetenekli bireyler yetiştirmekte, çeşitli eğitim ve öğretim yöntemleri kullanılmaktadır. Modern toplumlarda, karşılaştıkları sorunları farklı çözümler üretebilen, gelişime açık bireyler yetiştirilmesi amaçlanmaktadır. İstenen yeterliliklere ek olarak, insanlar tarafından kodlanan ve bu programlarla yönetilen bilgisayarlar, mekanik ve elektronik sistemler, modern bilgi toplumunda yaşamın vazgeçilmez bir parçası haline gelmiştir. Bilgisayar sistemleri, ülkelerdeki büyük merkezi sistemlerden akıllı telefonlara kadar pek çok yerde kullanılmaktadır. Öğretim hayatında birçok eğitimci ve araştırmacı bilgisayar desteğiyle daha etkili çalışmaktadır. Bu sayede bireylerden istenen yeterlilikleri sağlamada bilgisayarlardan, elektrik ve elektronik sistemlerden, yazılımlardan ve teknolojiyen destek alınarak istendik davranışların kazandırılması daha verimli ve kolay hale dönüştürülebilmektedir. Günümüz dünyasında sahip olunan bilgi ve donanımları teknoloji ile birleştirmek, var olan eksiklikleri gidermede yine teknolojiyi kullanmak önemli yeterlilikler arasında gösterilmekte; modern toplumlarda bireyde olması istenen özellikler arasındadır. Dijital yetkinlik olarak da adlandırılan bu kavram, günümüz toplumunda bireyler için hem bir gerekliliktir hem de bir haktır (Aksu ve Sürgevil, 2019). Çünkü modern toplumlarda bireyler sahip oldukları bilgi ve becerileri dijital yetkinlikleri ile daha verimli bir biçimde kullanabilmektedirler. Bu çerçeveden bakıldığında çağa uygun beklentilerin ve gelişimin karşılanabilmesi için öğrencilere öğretimin ilk basamaklarından itibaren bilgi ve iletişim teknolojileri eğitimleri tüm eğitim-öğretim programlarına yerleştirilmelidir. Teknoloji ile çok erken yaşta tanışan çocuklar için ilgilerini ve yeteneklerini daha verimli kullanabilecekleri ve bunun sonucunda tüketen birey pozisyonundan üreten bireye geçiş yapmalarına olanak sağlayacak Scratch gibi uygulamaların kullanımının eğitim ve öğretime ve

öğrencilerin matematiksel problem çözme becerilerine etkisi ise araştırılmaya değer bir konudur.

1.1. Problem Durumu

Scratch ile tasarlanan matematiksel problemlerin 8. sınıf öğrencilerinin problem çözme becerilerine etkisi olup olmadığını incelenmiştir. Bu amaç doğrultusunda alt problemlere cevap aranmıştır:

- i. Öğrencilerin ön test ve son test puan ortalamaları arasında anlamlı bir farklılık var mıdır?
- ii. Scratch ile programlamış matematiksel algoritmaları kodlayan öğrencilerin görüşleri nelerdir?

1.2. Araştırmanın Önemi

Millî Eğitim Bakanlığı (MEB) öğretim programlarında yapılandırmacı yaklaşımı benimsemektedir (Demirel, 2010). Yapılandırmacı öğrenme ortamları, öğrencilerin problem çözme, araştırma ve keşif yoluyla yeni bilgiler edinmelerini teşvik eder. Eğitimde, insanların günlük yaşamda ve iş hayatında işlevselliği arttırabilmeleri, problemleri çözebilmeleri ve dijital yeniliklere katkıda bulunabilmeleri için kompütasyonel düşünme yetisine sahip olmaları gerektiği kabul görmektedir (Kotsopoulos vd., 2017). Bu kabul, teknolojinin her geçen gün daha fazla hayatımızın içine girmesi ve yeni teknolojilerin anlaşılmasını da gerekli kılmasıyla daha da önem kazanmaktadır. Kompütasyonel düşünmenin bilgisayar bilimleriyle olan bariz ilgisine rağmen, akademisyenler kompütasyonel düşünmenin anaokulundan başlayarak bilgisayar bilimleri dışındaki disiplinlerde de öğretilmesi gerektiğini savunmaktadırlar (Barr ve Stephenson, 2011).

Kodlama, kompütasyonel düşünmenin en popüler uygulama alanlarından birisidir (Kotsopoulos vd., 2017). Kodlamayı geleceğin önemli becerilerinden gören araştırmalar, her öğrencinin kodların nasıl çalıştığını bilmeye hakkı olduğunu belirtmişlerdir (KOODI2016, 2014). Scratch, K-12 düzeyinde en yaygın kullanılan blok tabanlı kodlama programlarından birisidir (Marcelino vd., 2018) ve öğretim aracı olarak kullanılmakta olup (Batista vd., 2016; Hermans ve Aivaloglou, 2017) 8-16 yaş arası çocuklarda kompütasyonel düşünme gelişimini desteklemek için tasarlanmıştır

(Chowdhury vd., 2019). Scratch gibi ortamlar öğrencilerin matematiksel kavramları daha somut bir şekilde kavramasını sağlayabilir; kodlama becerilerini, sistematik, yaratıcı düşünme, problem çözme gibi becerileri geliştirebilir (Gadanidis, 2015) ve işbirlikçi öğretim ortamını destekler (Brennan ve Resnick, 2012).

Normal öğretim, bilgisayar bilimlerine katılımı artırmak ve kompütasyonel düşünme becerisini geliştirmek için mükemmel yerlerdir (Ioannidou vd., 2011) çünkü bu yıllar öğrencilerin kendi beceri ve yeteneklerine ilişkin sonuçlara ulaşmaları için kritik yıllardır (Gootman, 2007; Sears, 1995). 13 - 18 yaş arası, eğitim ve kariyer seçimleri için, çoğu gencin hangi eğitim alanını seçeceğini kesin olarak belirlediği ve bu kritik ve çoğu zaman geri dönüşü olmayan bir dönemdir (Banzato ve Tosato, 2020). Veriler, kızların genellikle fen bilimlerinin (mühendislik ve teknoloji gibi) dışındaki alanlara yöneldiklerini gösteriyor (Banzato ve Tosato, 2020). Normal öğretim programına entegre edilen kodlama eğitimi, öğretim programı dışında kalan hiçbir uygulamanın yapamayacağı bir oranla katılım oranına ulaşarak özellikle bilgisayar bilimleri alanında dezavantajlı olan kız çocukları da dahil olmak üzere tüm öğrencilerin temel bilgisayar bilimleri kavramlarını öğrenmesine, problem çözme ve analitik düşünme becerilerini geliştirmesine, öğrencilerin yaratıcılığını ve yenilikçiliğini artırmalarına ve gelecekteki kariyerlerine hazırlanmalarına imkan tanır (Ioannidou vd., 2011).

Kompütasyonel düşünme becerilerini öğretim hedefine katkıda bulunabilecek şekilde bilişim dersleri dışında kalan derslere nasıl entegre edileceği, öğrencilerin problem çözme becerisini geliştirmeye yönelik neyin, nasıl öğretileceği ve öğretilen şeyin nasıl değerlendirileceği konusunda alan yazında eksiklikler vardır (Denning, 2017). Bazı araştırmalar, Scratch içerisindeki belirli programlama içeriklerinin ve etkinliklerinin nasıl değerlendirilebileceğini kısmen göstermiştir ancak içeriklerin çok azı kompütasyonel düşünme ile bağlamsallaştırılmıştır (Fagerlund vd., 2021). Ayrıca, alan yazın Scratch ile yapılan araştırmaların çok azı programlama içermekte ve yine çok azı bilişim ve teknoloji dersi dışında kalan derslerle yapılmıştır (Elçiçek, 2020). Kodlamayı, öğrencilere sadece belirli bir programlama diliyle oyun tasarımı için öğretmenin ötesinde, kompütasyonel düşünme ve problem çözme becerilerini matematik dersine entegre ederek geliştirme yaklaşımı önemlidir. Bu sayede hem kodlamanın disiplinler arası kullanımı açısından hem de bilgisayar bilimlerinde ve programlamada yeterince temsil edilmeyen kız çocuklarının (Corbett ve Hill, 2015;

Klawe vd., 2009; Pinkard vd., 2017) çalışma grubumuzu oluřturması aısından alan yazına katkı sunmayı hedefliyoruz. Vrieler vd. (2021), kızların bilgisayar bilimiyle ilgilenmelerini teřvik eden akranların ve rol modellerinin ok nemli olduėunu; kızların teknoloji konusunda bilgili arkadař aėlarına eriřimlerinin sınırlı olabileceėini, bu yzden bu desteėin kızlar iin kritik olduėunu ifade etmektedir. Vrieler vd. (2021), ayrıca, kızların ėretmenlerinden aldıkları teřvikin, bilgisayar bilimlerine olan ilgilerini artıracadıını; topluluk desteėinin, kızların bilgisayar kimliklerini geliřtirmelerine ve farklı fırsatlarla tanışmalarına yardımcı olacağını; rol modellerinin, gen kızların z-yeterliklerini artıracadıını ve bilgisayar bilimlerine olan ilgilerini destekleyeceėini belirtmektedir. Arařtırmamız kız ocuklarının bilgisayar bilimine katılımını artırması ve onlara ėretmenleri ve akranları tarafından saėlanacak sosyal desteėi glendirmesi aısından nemlidir. GraBl (2022), kız ocuklarının programlama ėreniminde daha fazla temsil edilmesi ve ilgileri doėrultusunda desteklenmeleri, onların kendilerine olan gvenlerini artırabileceėini; Scratch gibi platformların kız ocuklarının programlama ėreniminde hem eėlenceli bir bařlangı saėlayabileceėini hem de yaratıcılıklarını ve bilgisayar bilimlerine olan ilgilerini srdrebilmelerine yardımcı olabileceėini belirtmiřtir. Bu nedenle arařtırmanın nemi, alan yazındaki belirtilen eksikliklere katkı sunmak ve Scratch ile programlanmış matematiksel durumların ortaokul 8. sınıf ėrencilerinin problem özme becerileri zerindeki etkisini ortaya koymaktır.

1.3. Arařtırmanın Amacı ve Alt Amaları

İlkėretimde bilgisayar bilimleri eėitim-ėretimin ierisinde ėretim programında nasıl ve ne řekilde yer alması gerektiėi, hangi kavramların hangi yařlarda verilmesi ve bilgisayar bilimi dıřında farklı disiplinlerde nasıl ve ne řekilde yer alması gerektiėi konusunda pek ok soruyla birlikte geliřmekte olan bir alandır. Bilgisayar bilimleri kendi doėası gereėi sahip olunması gereken bazı bilgiler ister ancak programlama bilgisi; genellikle matematik bilgisi, okuma becerisi ve genel biliřsel ykn stesinden gelme yeteneėi gibi bilgisayar bilimleri dıřı n kořul becerilerine de ihtiya duyar (Franklin vd., 2017). Chowdhury vd. (2019), kodlama konusunda acemi olan ėrenciler iin yaratıcı ve aktif ėrenme ortamının, uzmanlar tarafından nceden yazılmış kodların ėrenciler tarafından incelenmesi ve nceden var olan kodlarda kk deėiřiklikler yapılarak ėrencilerin koda yavaş yavaş yeni zellikler eklemeye

teşvik edilmesi (kompütasyonel düşünme kavramlarından olan karıştırma) ve yeni projeler oluşturmaya devam edilmesi ile oluşturulabileceğini ifade etmiştir. Amacımız, matematiksel kavramlar ve algoritmalar içeren kodlama etkinliklerini öğrencilere anlatıp daha sonra öğrencilerin kodlar üzerinde değişiklikler yapmalarına ve kendi öğrenme ortamlarını oluşturmalarına imkân sağlayarak blok tabanlı bir programlama ortamı olan Scratch'i sekizinci sınıf matematik dersine entegre etmek ve sonrasında mevcut ilköğretim programında kullanılan ölçme ve değerlendirme araçlarından olan beceri temelli sorular ile öğrencilerin problem çözme becerilerine etkisini incelenmiştir. Araştırmanın alt amaçlar arasında ayrıca uygulamadan sonra öğrencilerin görüşlerini alınmıştır.

1.4. Sayıtlar ve Sınırlılıklar

Araştırma aşağıdaki sayıtların kabul edilmesine bağlıdır:

- i. Öğrenciler araştırma sürecince Scratch öğretimindeki etkinliklerin tamamına katılım sağlamışlardır.
- ii. Öğrenciler yapılan görüşmelerde samimi cevap vermişlerdir.

Araştırma bulguları ve sonuçları, 2022-2023 eğitim öğretim yılı içerisinde Yozgat'ta yer alan ortaokul 8. sınıfta öğrenim gören sekiz öğrenci ile sınırlanmıştır. Öğrencilerin Scratch programı ile ilk kez karşılaşmakta olacağı varsayılmaktadır. Öğrencilerin problem çözme becerileri beceri temelli sorular ile belirlenmiştir. Araştırmanın sadece kız öğrencilerle yapılmış olması, tek gruplu bir yarı deneysel desen kullanılmış olması ve öntest ve sontestte aynı soruların kullanılmış olması araştırmanın sınırlılıkları arasındadır. Verilerin toplanacağı sene gerçekleşen deprem dolayısıyla bazı konuların 2022-2023 eğitim-öğretim yılı merkezi sınava yönelik sorulacak konu ve kazanımlardan çıkarılmış olması (TTKB, 2023) ve araştırma süresinin sekiz hafta ile sınırlı tutulmuş olması araştırmanın kapsamını belirli konu başlıkları ile sınırlamıştır.

1.5. Tanımlar

Scratch: Massachusetts Institute of Technology (MIT) tarafından geliştirilmiş olan görsel programlama dilidir. Programın hedef kitlesini özellikle 8-16 yaş grubu öğrenciler oluşturmaktadır. Ancak programlama ve kodlama alanıyla yeni tanışmış

yetişkinler içinde sade ve basit arayüzü ile oldukça kullanışlıdır. Bu dili kullanarak kendi sahnenizi tasarlayabilir, oyun kurabilir, çeşitli animasyonlu etkinlikler, yarışmalar vs. hazırlayabilirsiniz.

Kompütasyonel Düşünme: Problem çözme ve yeni durumlar oluşturmak için bilgisayar biliminin temel kavramlarını ve tekniklerini kullanma becerisidir. Bu beceri, algoritmalar, veri yapıları, soyutlama ve programlama gibi temel kavramları anlamayı ve problemleri bu kavramlar çerçevesinde çözme içerir.

Programlama: Bilgisayara talimatlar ve komutlar vererek bir işi yaptırmaktır. Komutlar, kelimeler ve aritmetik işlemlerden oluşur. Programlama dilleri, bu talimatları yazmak için kullanılan araçlardır. C++, Java, Python ve Scratch bu programlama dillerinden bazılarıdır.

Problem Çözme: Bir zorluğu üstesinden gelmek için problemi tanımlama, probleme yönelik çözümler üretme, çözümleri değerlendirme, çözümlerden en uygun olanını seçme ve çözümün etkililiğini değerlendirme gibi çok adımlı karmaşık bir süreci içeren üst bilişsel bir beceridir.

Boole (Boolean) İfadeleri: Matematik ve bilgisayar bilimlerinde kullanılan ve doğru (true) veya yanlış (false) olmak üzere sadece iki değere sahip olan mantıksal ifadelerdir.

BÖLÜM II

ARAŞTIRMANIN KURAMSAL TEMELİ VE İLGİLİ ARAŞTIRMALAR

2.1. Problem Çözme

Problem çözmenin öncülerinden Polya (1945) bir problemi çözmeyi, bir zorluktan çıkış yolu bulmak, bir engelin etrafından dolaşmak, hemen anlaşılamayan bir amaca ulaşmak olarak tanımlamıştır. Schoenfeld (1983), problemin ancak çözülemeyen bir kavram olduğunda problem olduğunu, onun dışında çözmesi ne kadar zor olursa olsun problem çözmeyi sadece bir egzersiz olarak ifade etmiştir. Green ve Gilhooly (2005) ise problem çözmeyi günlük hayatı anlamlı bir biçimde yapılandıran bir faaliyet olarak ifade etmişlerdir (Voskoglou ve Buckley, 2012). Programlama sürecinin öğrencilerin sistematik ve eleştirel düşünme, problem çözme gibi becerileri geliştirmelerine yardımcı olduğu düşünülmektedir (Korkmaz, 2016; Lau ve Yuen, 2011). Bir problemi tanımlamadan çözmeye kadar olan tüm süreçlerin bilgisayar ortamında gerçekleştirilmesi, programlama yoluyla problem çözme sürecini ifade etmektedir. Bu nedenle programlamayı öğrenmenin problem çözme becerisini de geliştireceği düşünülmektedir (Hwang vd., 2008; Korkmaz, 2018). Problem çözme becerisi zihinsel beceriler arasında en karmaşık olandır (Gagné, 1985) ve sorunu belirleme, bilgi toplama, olası çözümleri üretme, nihai çözümü bulma ve sorunu çözme basamaklarından oluşan yaşamsal olarak önemli ve üstbilişsel beceridir (Mayer, 1998). Problem çözme becerisi küçük yaşta öğrenilen ve okul çağlarında geliştirilen bir beceri olup (Miller ve Nunn, 2001); farklı disiplinlerle birleştirildiğinde öğrencilerin problemlerle baş etme becerilerini arttırdığı ve öğrencilerin daha iyi öğrenme sonuçları edinmişlerdir (McGehee, 2001). Ancak bunun disiplinler arası bir anlayışla farklı derslerde öğrencilerin problem çözme becerilerini nasıl geliştireceğine ve bunun nasıl değerlendirileceğine dair belirsizlikler sürmektedir.

Problem çözme için farklı birçok tanım olsa da araştırmada problem çözmenin, bir problemin üstesinden gelmek ve içinde bulunduğumuz yaşam ve koşullar hakkında zihinsel veya fiziksel araçlardan faydalanarak fikir yürütmeyi sağlayan bir faaliyet tanımını benimsiyoruz. Kompütasyonel düşünme için en fazla atıfta bulunulan özellikler genellikle soyutlama ve problem çözmedir (Ioannidou vd., 2011; Voskoglou ve Buckley, 2012). Problemi soyut bir şekilde algılama ve düşünme becerisi, daha fazla çözüm fırsatı görebilmek (Anderson vd., 1978); kompütasyonel düşünmeyi veya başka bir deyişle soyutlama yoluyla problem çözmeyi kavramak bazı durumlarda daha fazla ayrıntıyı görebilme yetisidir (Sarjoughian ve Zeigler, 1996). Öğrencilere erken yaşta verilen programlama eğitiminin onların problem çözme becerilerine, analitik ve uzamsal düşünme becerilerine ve matematiksel düşünme becerine katkı sağlamaktadır (Korkmaz, 2016).

Kompütasyonel düşünme matematiksel problemler çözmeyi, sistemler tasarlamayı ve inşa etmeyi, verileri yorumlamayı içerebilir ve bu açıdan gerek soyut gerekse mühendislik içeren somut bu araçları yaratan ya da kullanma konusunda yeterli öğrenciler yetiştirmek önemlidir (Voskoglou ve Buckley, 2012). Gelecek neslin tamamı, bilişimden büyük ölçüde etkilenen bir yaşam sürecektir ve birçoğu bilişimi içeren ya da bilişimden etkilenen alanlarda çalışacaklardır (Barr ve Stephenson, 2011). Bu nedenle kompütasyonel düşünme öğretimine ne kadar erken başlanırsa o kadar faydalı olacaktır (Magana vd., 2011). Bilgisayar bilimleri, her ne kadar bugün ve gelecek için yadsınamaz öneme ve yere sahip olsalar da öğrencilerin ilgisini çekme, odaklarının kısa sürede dağılması ve programlamayı öğrenme konusunda öğrencilerin isteksiz olması gibi dezavantajlara da sahiptir (Kazımoğlu vd., 2011). Tüm bu dezavantajlarına rağmen programlama da öncelikli olan öğrencilerin motivasyonlarını ve katılımlarını yüksek tutmak olmalıdır ve öğrencilere programlamanın sadece kodlamadan ibaret olmadığı; kompütasyonel düşünme becerilerini geliştirerek somut problemleri anlayıp problem çözme becerilerinin geliştirilerek çözüm geliştirmenin temel amaçlardan olduğu ifade edilmelidir (Voskoglou ve Buckley, 2012).

Problem çözme doğası gereği matematiğin temelini oluşturur. Kompütasyonel düşünme ise özünde mantık, yaratıcılık, algoritmik düşünme, simülasyon, modelleme gibi bilimsel metodolojileri içeren farklı becerileri geliştirir (Voskoglou ve Buckley, 2012). Kompütasyonel düşünme, kökleri matematik mühendislik, teknoloji ve bilime dayanan, gerçek anlamda 21. yüzyıl becerisi olan, programlama aracılığı ile

öğrencilere yalnızca problem çözmelerine yardımcı olan değil aynı zamanda yeni sorular üretmelerini de sağlayan ve bu sayede öğrencilere zorlukların üstesinden gelmeyi öğreten değil aynı zamanda kendi zorluklarını da yaratmalarını sağlayan olağanüstü değişiklikler ve faydalar üretmeye henüz başlamış bir beceridir (Einhorn, 2012).

Yeni teknolojiler öğrencilerin problem çözme becerilerini geliştirmede birer araç olarak kullanılabilirler. Uygun yazılımların kullanılması öğrencilerin hayal güçlerinin geliştirir; matematiksel problemlerin bu yazılımlar aracılığı ile canlandırılması öğrencilerin problemin çözümüne daha rahat ulaşmalarını sağlayabilir (Voskoglou ve Buckley, 2012). Bu araçlardan birisi olan Scratch, temel tasarım özelliklerinden dolayı bireylerin programı daha kolay öğrenmelerine yardımcı olur (Kaučič ve Asič, 2011; Resnick vd., 2009) ve programlama yabancı olan kullanıcıların algoritmik düşünme becerilerini geliştirir (Maloney vd., 2010). Scratch 21. yüzyılda öğrencilerin sahip olması gereken (Kalelioğlu vd., 2014) eleştirel düşünme, yaratıcı düşünme, sistematik akıl yürütme ve iş birlikli çalışma becerilerinin gelişmesine yardımcı olur (Maloney vd., 2010; Rodríguez-Martínez vd., 2020). Bazı araştırmalar Scratch'in, öğrencilerde matematiksel düşünmeyi, problem çözmeyi, analitik düşünmeyi geliştirdiğini göstermiştir (Calder, 2010; Kaučič ve Asič, 2011; Korkmaz, 2016; Vatansever ve Baltacı Göktalay, 2018)

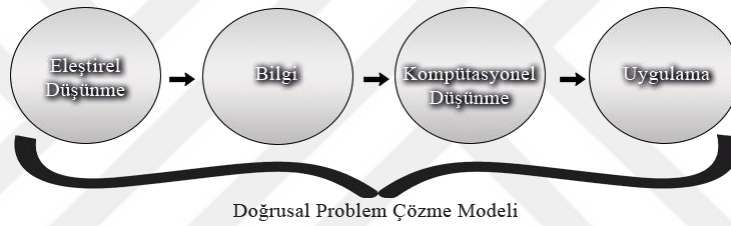
Kompütasyonel düşünme becerilerinin ne ölçüde kazanıldığına yönelik süreç içerisinde bazı araçlar geliştirilmiş ayrıca çeşitli ölçme ve değerlendirme yöntemleri kullanılmıştır. Koh vd. (2010) geliştirdiği Bilgi İşlemsel Düşünme Örüntü Analizi (CTPA) Román-González vd. (2015) geliştirdiği Dr. Scratch (Moreno-León vd., 2015) ya da Korkmaz vd. (2017) geliştirdiği Bilgi İşlemsel Düşünme Becerisi Ölçeği (CPT) gibi araçların yanı sıra öğrencilerin günlük yaşam problemlerini çözmek ve bilişim teknolojisi becerilerini ne kadar etkin kullandıklarını ölçmek için 15 çoktan seçmeli ve 8 açık uçlu sorudan oluşan bir değerlendirme aracı geliştirilmiştir (Chen vd., 2017). Ayrıca öğrencilerin kompütasyonel düşünme becerilerini ölçmek ve değerlendirmek amacıyla dijital portföy analizi (Fields vd., 2019); öğrencilerin Scratch platformunda görsel programlama dilini öğrendikten sonra ilkokul öğrencilerinin kompütasyonel kavramlara ilişkin algılarını incelemek için anket çalışmaları (Sáez-López vd., 2016); öğrencilerin kompütasyonel becerilerine yönelik bir öz-yeterlik algısı ölçeğinin güvenilirliğini ve geçerliliğini incelemek için açımlayıcı ve doğrulayıcı faktör analizi

(Glbahar vd., 2019) ve đrencilerin blok tabanlı programlama dili kullanarak video oyunları tasarlama konusundaki stratejilerini arařtırmak iin grřme (Weintrop vd., 2014) gibi yntemler kullanılmıřtır (Tang vd., 2020).

Voskoglou ve Buckley (2012), problem zme ile komptasyonel dřnme arasında kurdukları iliřkide problem durumunu belirledikten sonra var olan bilginin eleřtirel bir řekilde analiz edildikten sonra bilgi tabanından daha farklı bilgilerle var olan sorunu bilgisayar gibi dřnerek bilimsel bir řekilde problemi zer. Byle problem zme srecini komptasyonel dřnme ile gerekleřtirmiř olacađını belirmiřlerdir. řekil 1’de komptasyonel dřnme ile problem zme modeli verilmiřtir.

řekil 1

Komptasyonel dřnme ile problem zme modeli



Not. “*Problem solving and computational thinking in a learning environment*”, M. Gr. Voskoglou ve S. Buckley, 2012, *Egyptian Computer Science Journal*, 36(4), sayfa 37’den alınmıřtır.

Eleřtirel dřnme, analiz, sentez ve deđerlendirme gibi řt dzey dřnme becerilerini bir araya getirerek sonuca varmayı sađlayan daha řt dzey bir dřnme řekli olarak kabul edilir (Voskoglou ve Buckley, 2012). Her bir kavram bir ncekinin n kořulu olması sebebiyle model dođrusaldır. Problem zme model ierisinde yer almaz nk problem zme herhangi birinin kořulu deđil srecin rndr (Voskoglou ve Buckley, 2012).

Gnmzde teknolojinin derslerde etkin bir biimde kullanılması; đretmen ve đrencilerin dijital yetkinlikte teřvik edilmesi ve MEB’in bu ynde yatırımlar yapılması amalanmıřtır (Alp, 2019). Kodlama becerileri ise bu dijital yetkinliđin ierisinde nemli bir yere sahiptir. Scratch ise bu alanda đrencilerin kendilerini geliřtirmelerinde nemli aralardan birisidir (Alp, 2019). Scratch ile ilgili olarak, Brennan ve Resnick (2012) komptasyonel dřnmenin  boyutunu ne srmřtr:

kompütasyonel kavramlar, kompütasyonel uygulamalar ve kompütasyonel perspektifler. Bu boyutlar Mayer'in (1992) Logo programlama dili bilgisi ile de uyumlu olduklarından, ortaokul öğrencilerinin programlamaya nasıl yaklaştıklarını ve kompütasyonel düşünme biçimini anlamak için uygundur (Lye ve Koh, 2014).

Wing'in (2006) kompütasyonel düşünme terimini ortaya atmasından itibaren kompütasyonel düşünmeye ait tek bir tanımın yapılması konusunda tartışmalar sürmektedir. Sınırları kesin bir tanım olmamasına rağmen, Bilgisayar Bilimleri Topluluğunda, problem çözme aracı olarak soyutlama ile ilgili ortak bir terim anlayışı ortaya çıkmıştır. Soyutlama (problem temsili) ve ayrıştırma (bir problemi yeniden formüle etme) kompütasyonel düşünmenin merkezinde yer alır, çünkü bu iki analitik süreç karmaşık görevlerin yerine getirilmesine ve/veya bir çözümün keşfedilmesine yardımcı olur (Ateşkan ve Hart, 2021).

Bilgisayar Bilimleri sadece programlamayla kısıtlı olmayan, aynı zamanda bir düşünme biçimidir (Wing, 2011). Bilgisayar Bilimleri topluluğu, bu şekilde düşünmeyi öğrenmenin, kompütasyonel düşünmenin, diğer birçok disiplindeki sorunlara yeni bakış açıları ve çözümler getirebileceğini; dünya çapındaki bilgisayarlar ile kurulan ağlarının bilgiye erişmenin yanı sıra insanların ilgi alanlarını paylaşarak bağlantı kurmalarını içeren süreçte dünyanın dört bir yanındaki insanlarla özdeşleşme, ilişkileri geliştirme imkânı da sunacağına; bu kolektif oluşumlarla kimliklerimizin genişleyeceğini ve bilgi hayal gücü aracılığıyla pek çok boyut kazanacağını dile getirmektedir (Wenger, 1998). Kompütasyonel düşünme diğer disiplinlere uygulandığında öğrencilerin kendi disiplinlerinde karşılaştıkları karmaşık sorunlara benzersiz ve yenilikçi çözümler üretmek için yaratıcılıklarını kullanmalarını teşvik edebilir (Ioannidou vd., 2011). Araştırmalar, kompütasyonel düşünmenin farklı disiplinler için (özellikle fen ve matematik) sınıf eğitimine entegre edilmesine olumlu yaklaştıklarını göstermiştir (Ateşkan ve Hart, 2021).

MEB (2018) öğrencilerin matematik okuryazarlığı ve matematiksel kavramları günlük hayatta kullanabilme becerilerini geliştirmek amacıyla 8 ve 12. sınıf sonunda yapılan merkezi sınavlarda yer alan problemlerde rutin olmayan ve akıl yürütmeyi gerektiren problem türlerine yer vermeye başlamıştır. Bu sayede öğrencilerin sadece ezberden değil, problem çözme becerilerini kullanarak da matematik sorularını çözebilmelerini hedeflenmektedir (MEB, 2018). Beceri temelli sorular, öğrencilerin okuduğunu anlama, yorumlama, sonuç çıkarma, problem çözme, analiz yapma,

eleştirel düşünme ve bilimsel süreç becerileri gibi üst düzey becerilerini değerlendirmek için kullanılır (Tortop vd., 2022). Yeni nesil sorular da denilen beceri temelli sorular, öğrencilerin akıl yürütme, analiz etme, problem çözme ve değerlendirme gibi üst bilişsel becerileri kullanmalarını gerektiren soru türleridir (Kolovou vd., 2009).

Beceri temelli sorular, problem çözme, kompütasyonel düşünme ve kodlama uygulamaları birlikte daha etkili hale gelen becerilerdir. Örneğin, bir öğrencinin beceri temelli bir soruyu çözebilmesi için önce problemi tanımlaması ve çözüm için bir plan oluşturması gerekir. Bahsedilen aşamalar gerçekleştirebilmesi için problem çözme becerisine sahip olması beklenir. Bu tip sorularla bu beceriyi geliştirebilir. Öte yandan kodlama etkinlikleriyle kompütasyonel düşünme becerisini geliştiren bir öğrenci verileri sistematik hale getirip her bir parçayı algoritmalar kullanarak çözebilir, çözümü koda aktarabilir veya rutin olmayan bir problemin çözümünde kullanabilir.

Papadopoulos ve Tegos (2012), öğrencilerin bilgisayar bilimleri alanında kompütasyonel düşünme ve problem çözme becerilerinin yetersiz olduğunu belirtmişlerdir. Akıl yürütme ve problem çözme becerileri ise matematik için kritik öneme sahiptir (Leinwand vd., 2014). Bu beceriler, matematik, temel kodlama ve karmaşık problemleri çözmeye yönelik bilgisayar bilimlerindeki başarının temelini oluşturur (Psycharis ve Kallia, 2017). Kodlama yaparken öğrenciler hem mantıksal hem de soyut akıl yürütme becerilerini, problem çözme becerilerini ise oyunun algoritmasını geliştirmek için kullanırlar (Psycharis ve Kallia, 2017). Programlama sürecinde tahminlerde bulunurken ve son ürünlerini analiz ederken de mantıksal akıl yürütme devreye girer (Psycharis ve Kallia, 2017). Öğrencilerde akıl yürütme ve problem çözme becerilerini teşvik etmek için karmaşık düşünme becerilerini uygulama fırsatları yaratarak bilgisayar programlamayı öğretmek, üst düzey düşünme performansını geliştirmenin harika bir yoludur (Howard ve Howard, 2020). Eğitimciler, kodlama ve matematik yoluyla üst düzey düşünme becerilerinin kullanımının teşvik edilmesi gerektiğine inanmaktadır (Howard ve Howard, 2020). Calder (2010), kodlamanın matematiksel süreçleri ve becerileri öğrenmede motivasyon sağlayabileceğini, problem çözme, mantık ve akıl yürütme becerilerini ise geliştirdiğini belirtmiştir.

2.2. Kodlama Yoluyla Problem Çözme

Alan yazında problem çözme becerileri, okul çağındaki çocuklarda yaratıcı kodlama faaliyetlerinin önemli bir faydası olarak sıklıkla yer almaktadır fakat bugüne kadar çok az çalışma problem çözmenin disiplinler arası yaratıcı kodlama görevlerinde nasıl kullanılacağı araştırmıştır (Woo ve Falloon, 2022). Problem çözme araştırmaları büyük ölçüde Polya'nın (1945) problem çözmenin temel aşamalarını özetleyen dört adımlı modelini temel alır. Bu model, problemi anlama, planlama, uygulama ve değerlendirme olmak üzere dört aşamadan oluşmaktadır. İlk kez yayınlanmasının üzerinden seksen yıla yakın bir süre geçmesine rağmen Polya'nın çerçevesi, problem çözme anlayışımız üzerinde küresel bir etkiye sahip olmaya devam etmektedir (OECD, 2013). Bu nedenle, Polya'nın modeli, yaratıcı kodlama faaliyetlerinden kaynaklanabilecek genel problem çözme becerilerini anlamak için de uygun bir çerçeve sunmaktadır. Araştırmanın bilgisayar bilimleriyle olan doğrudan ilişkisi göz önüne alındığında, Polya'nın problem çözme ile blok tabanlı kodlamanın nasıl benimsendiğini ele almak uygundur. Taylor (2019), Polya'nın problem çözme modelinin ortaokul seviyesinde kodlama öğrenimi için uygun olabileceğini belirtmiştir. Taylor (2019), tüm programlama problemlerinin bir problem çözme düzeyi gerektirmeyeceğini; bazı problemlerin, öğrencilerin yalnızca bilinen bir algoritmayı ortak bir kod kalıbı şeklinde uygulamalarını gerektirebileceğini veya öğrencilerin basitçe koda dönüştürebilecekleri basit algoritmaya sahip olabileceğini ifade etmiştir. Taylor (2019), Polya'nın problem çözmesinde yer alan basamakların kodlamaya uyarlanırken izlenmesi gereken adımları tanımlamıştır:

1. **Problemi Anlama:** Problem net bir şekilde tanımlanır. Soru metnine dayalı olarak test senaryoları oluşturulur veya verilen test senaryolarını incelenir. Kodlamayı yapan kişi, problemi kendi sözcükleriyle yeniden ifade edebilir veya başka birine açıklayabilir.
2. **Bir Plan Geliştirme (Kodlama yapmadan problemi çözme):** Problem için gerekli problem çözme düzeyini belirlemek; verilen bir algoritmayı koda çevirmek, bilinen bir algoritma örneğini probleme uygulamak veya bir problemi çözmek için bir algoritma oluşturmak olabilir. Algoritmik bir yaklaşım oluşturmak için kullanılacak çeşitli stratejilerden biri ise; algoritmayı genelleştirilebilecek bir örüntü bulmak için çeşitli örneklerin incelenmesidir.

Karmaşık bir problem daha basit sorunlara ayrılır, ardından her basit problem ayrı ayrı çözülür. Alternatif algoritmalarla karşılaştırılır. Bir algoritma seçilir ve çözüm, koda çevrilmeye uygun olacak şekilde yazılır. Bu aşamada henüz kod yazılmaz.

3. Planı Uygulama (Çözümü koda çevirme): İkinci aşamada belirlenen çözümler adım adım koda çevrilir. Kodlayıcı bu adımda koda nasıl çevireceğini biliyorsa kendisi çevirir. Aksi halde destek alarak nasıl çalıştığını anlayan kadar programın yapıldığı araç üzerinde kodlar incelenir. Ardından yapılan kodlama, çalıştığından emin olmak için bir test girdisi üzerinde çalıştırılır.
4. Gözden Geçirme (Kodu test etme): Tüm test senaryolarının işlevselliğini sağlamak için kod çalıştırılır. Bir test senaryosu başarısız olursa, algoritmik sorunu tanımlamak ve düzeltmek için hata ayıklama kullanılır. Kodlayıcı, problemi çözmek için alternatif yaklaşımlar hakkında bilgi edinmek için çözümü başkalarıyla tartışır ve karşılaştırır.

Kodlama, karmaşık problemleri parçalara ayırıp mantıksal bir sıra halinde yeniden birleştirmeyi gerektirerek öğrencilerin problem çözme becerilerini geliştiren; basitçe teknolojinin çalışması için kullanılan dildir ve bu dili öğrenmek öğrencilere, problem çözmenin yanı sıra görevleri sıralama ve fikirlerini yaratıcı yollarla ifade etme gibi beceriler kazandırır (Siegle, 2017). Woo ve Falloon (2022), tüm kodlama problemlerinin öğrencilerin özgün bir çözüm üretmesini gerektirmeyeceğini; bazı problemlerin öğrencilerin yalnızca bilinen kod kalıplarını uygulamalarını veya bilinen bir algoritmayı koda çevirmelerini gerektireceğini, bu tür problemlerde öğrencilerin, bilinen kod kalıplarını eldeki programlama problemine uyguladıklarında örüntü tanıma yöntemini kullanabileceğini veya verilerdeki örüntüleri tanıyabileceğini, bir dizi adım adım talimat oluşturdıklarında algoritmik düşünme sürecine dâhil olacaklarını belirtmiştir. Basit problemler için algoritmalar apaçık olabilir. Bu nedenle Woo ve Falloon (2022), tüm programlama veya kodlama problemleri kompüstasyonel düşünmenin tüm aşamalarını gerektirmeyebileceğini ve öğrencilerin programlama veya kodlama problemlerini çözmek için kullanabilecekleri alternatif yöntemlerin de olabileceğini ifade etmişleridir.

Papert (1980), öğrencilerin erken yaşta problem çözme becerilerini geliştirmeleri gerektiğine inanıyordu ve bilgisayar programlamanın bu becerileri

geliştirmek için ideal bir araç olduğunu savundu. Kodlama, teknolojiye bakış açımızı kökten değiştirebilir ve bizi edilgen tüketicilerden aktif üreticilere dönüştürebilir (Morrison, 2013). Blok tabanlı programlama, yeni başlayanlar için ideal bir yöntemdir. Kolayca anlaşılır olması sebebiyle öğrenciler kodlamaya hızlı bir şekilde başlar, kullanıcı hatalarını en aza indirir ve metin tabanlı kod yazmaya geçiş için gerekli temellerin edinilmesini sağlar (Siegle, 2017). Kodlama eğitimi, çocukların hesaplamalı düşünme, yaratıcılık ve problem çözme becerilerini geliştirerek bilişsel ve sosyal gelişimlerine katkıda bulunabilir (Read ve Macfarlane, 2006). Araştırmalar, kodlama eğitimi alan öğrencilerin matematik performanslarında ve sosyal becerilerinde anlamlı bir gelişme olduğunu göstermiştir (Wong vd., 2015). Bu nedenle, kodlama eğitimi günümüz eğitim sisteminde önemli bir yer tutmaktadır. Ancak, çocuklarda kompütasyonel düşünme ve problem çözme becerilerini geliştirmede farklı teknolojik sistemlerin nasıl bir rol oynadığı hala tam olarak bilinmemektedir (Zhu vd., 2016). Bu konuda daha fazla araştırmaya ihtiyaç vardır. Farklı sistemlerin hangi bilişsel becerileri geliştirmede daha etkili olduğunu ve hangi çocukların hangi sistemlerden daha fazla faydalanabileceğini belirlemek önemlidir. Somut girdi kullanmanın robotik programlamada çocukların performansını artırdığına dair kanıtlar vardır (Sapounidis vd., 2015). Farklı girdi ve çıktı kombinasyonlarının problem çözme becerilerini nasıl etkilediği ise hala yeterince araştırılmamıştır. Bu konu, gelecekteki araştırmaların odak noktası olması gereken önemli bir alandır. Bu nedenle kodlama etkinliklerinin matematiksel problem çözme becerilerine etkisi ile alan yazına katkı sağlanmıştır.

2.3. Matematik ve Kodlamayı İlişkilendiren Çalışmalar

Kodlamayı matematikle ilişkilendiren araştırmalardan Howard ve Howard (2020), kodlamanın kökenlerinin matematikten geldiğini ve aralarındaki ilişkinin doğal olarak var olduğunu belirtir. Kodlamanın öğrencilere matematik, algoritmalar, problem çözme ve analiz gibi becerilerin pratikte nasıl kullanılabilecekleri konusunda yol gösterdiğine yönelik araştırmalar vardır (Fessakis vd., 2013; Israel vd., 2015; Jona vd., 2014). Falloon (2016), kodlamanın öğrencilerin problem çözme yeteneklerini geliştirmelerine ve üst düzey düşünme becerilerini artırmalarına yardımcı olduğunu belirtmiştir. Byrne ve Lyons (2001), matematik ve kodlama arasındaki ilişkiyi incelerken, programlama becerilerinin matematiksel problem çözme becerileri ile

benzerlikler taşıdığını belirtmektedir. Calder (2010), kodlamanın matematiksel süreçleri ve becerileri öğrenme sürecinde motivasyon sağlayabileceğini belirtmiştir. Calder (2010), Scratch kullanarak kodlamanın öğrencilerin çeşitli matematiksel kavramların gelişimine yardımcı olduğunu; problem çözme, mantık ve akıl yürütme gibi becerilerin daha da geliştirilmesine katkıda bulunduğunu belirtmiştir.

Kodlama öğrenmenin birçok öğrenci için zorlayıcı olabileceğine yönelik araştırmalardan. Ben-Ari (1998), kodlama öğrenirken, öğrenciler yeni kelimeler, kavramlar, programlama dilleri ve tamamen yeni bir yaklaşım tarzıyla karşılaşmalarının öğrenciler için zorlayıcı olabileceğinin belirtmiştir. Malan ve Leitner (2007), kodlamanın yeni bir dil öğrenmek gibi olduğunu ve tamamen farklı bir dil yapısının olduğunu öne sürer. Döngüler, değişkenler, algoritmalar, Boole ifadeleri ve koşullu mantık gibi kavramlar, kodlamayı yeni öğrenen öğrenciler için genellikle yabancıdır (Malan ve Leitner, 2007). Kodlama konusunda bir diğer engel ise çoğu öğrenci, teknolojiyi günlük yaşamlarında sıklıkla kullanırken, teknolojiyi verimli bir şekilde kullanma becerisine sahip olamayabilir (Howard ve Howard, 2020). Kafai ve Burke (2013), öğrencilerin teknolojiyi kullanabilme yeteneklerine sahip olabileceklerini, ancak bu beceriyi her zaman yaratıcılık, eleştirel düşünme veya üretkenlik gibi amaçlar için kullanamayabileceklerini belirtmektedir. Korkmaz (2016) ise Scratch aracılığı ile öğrencilere verilen temel programlama eğitimin ilerleyen dönemlerde somut sonuçları üzerine araştırmanın mevcut olmadığını ifade etmiştir.

Howard ve Howard (2020), kodlamanın eleştirel düşünme ve yaratıcılık gibi becerileri teşvik ettiğini; muhakeme, mantık ve çıkarım becerilerini kullanma eğiliminde olan bilişsel bir süreç olduğunu belirtmiştir. Korkmaz (2016), kodlamanın sadece programlama için var olduğuna yönelik görüşün aksine, kavramsal ve tasarımla ilgili problem çözme becerilerini de vurguladığını ifade eder. Chao ve Cheng (2000), Piaget'nin, tümdengelimsel akıl yürütme ergenlik döneminde gelişen bilişsel olarak ileri bir beceri olduğunu; özellikle ergenler standart mantığa karşılık gelen eksiksiz bir zihinsel mantığı bu dönemde edindiğini ifade ettiğini aktarmışlardır. Howard ve Howard (2020), öğrencilerin bu nedenle, kodlamada mantık ve akıl yürütme becerilerini uygulamaya koymakta zorlanabileceklerini belirtmiştir. Altun vd. (2016), öğrencilerin kısa sürede dikkatlerinin dağılması ve kod yazarken genellikle çok adımlı etkinliklerle çalışmak zorunda kalmaları sebebiyle zorlanabileceklerini ifade etmişlerdir. Howard ve Howard (2020), matematik ve kodlama birbirine yakın ama

birbirine değmeyen iki alan gibi görülebileceğini ancak kompütasyonel düşünme kavramının bu iki alan arasındaki köprü görevi göreceğini hem matematik hem de kodlama becerilerinin bir arada öğretilmesinin, güçlü bir matematik ve kodlama programının geliştirilmesine yardımcı olabileceğini belirtmiştir.

Scratch ile yapılan kodlama etkinlerinin değerlendirilmesinde çoktan seçmeli test yöntemi kullanan araştırmalardan Ericson ve McKlin (2012), 4.-12. sınıflar üzerinde Scratch yazılımını kullanmayı da içeren kodlama etkinliklerinde öğrencilerinin öğrenmelerini ölçmek amacıyla çoktan seçmeli test kullanmışlardır. Öğrencilerin öğrenmelerini ölçerken kullandıkları 10 soruluk çoktan seçmeli testte verdikleri problem durumları ile öğrencilerin eğitim boyunca kullandıkları kod blokları arasında ve ilişki kurmaları beklenmiş ve bu ilişki soru soru incelenmiştir. Franklin vd. (2017), Scratch kullanarak öğrenmeyi ölçmek amacıyla test yöntemini kullanmışlardır.

2.4. Programlama

Programlama temelde insanlara uygun dilden bilgisayara uygun dile çeviri yapma sürecidir (Blackwell, 2002). Programlamanın kendisi sadece kodlamadan daha fazlasıdır (Lye ve Koh, 2014). Programlama, kodlamaya göre çok daha karmaşık ve bilişsel süreçleri içerir (Blackwell, 2002). Programlama dilleri, belirli amaçlar için hazırlanmış özel kelime ve sembollerden oluşur. Bu bağlamda bir dizi komut denilebilir. Bir bilgisayarın yapması gerekeni yapmasını sağlayan programlar (yani yazılımlar), yazılı komutlar sayesinde ortaya çıkar. Bilgisayarlar ve teknoloji için oluşturulmuş tonlarca programlama dili vardır. Her dil, uygun komutları oluşturmak ve yürütmek için kendi deyimlerini ve yazım kurallarını kullanır. Bir programlama dilinin belirli komutlarını yazma işlemine kodlama veya programlama denir ve nihai ürüne program veya uygulama denir (Çubukluöz, 2019).

2.4.1. Logo

Kişisel bilgisayarlar ilk kez 1970'lerin sonlarında ve 1980'lerin başlarında okullarda yer edinmeye başladığında, Seymour Papert, bilgisayarın çocuklara ve gençlere yardım etme gücünden faydalanmak amacıyla çocuklara programlamayı öğretmek için Logo programlama dilini yazdı (Howard ve Howard, 2020). Güney Afrika doğumlu bir matematikçi ve bilgisayar bilimcisi, aynı zamanda Cenevre

Üniversitesi'nde Jean Piaget ile beraber çalışmış ve yapay zekanın (Artificial Intelligence - AI) öncülerinden kabul edilen Papert (1980), bilgisayar teknolojilerinin çocukların neyi ve nasıl yarattığını büyük ölçüde genişletebileceğini, çocukların bilgisayarlarla animasyonlar, simülasyonlar ve etkileşimli oyunlar gibi hareket eden, etkileşimde bulunan ve zaman içinde değişen şeyler yaratabileceğini ve bu süreçte kendi zihinlerinin işleyişi de dâhil olmak üzere çevrelerindeki dünyadaki dinamik sistemlerin işleyişi hakkında yeni içgörüler edinebileceklerini belirtmiştir. Ek olarak bilgisayarlar, çocukların yaratımlarını daha önce hiç yapamayacakları şekilde değiştirmelerine, çoğaltmalarına, belgelemelerine ve paylaşımlarına olanak tanıyarak yaratıcı süreci keşfetmeleri ve anlamaları için yeni yollar sağlar (Papert, 2020). Bilgisayarı yalnızca bir problem çözme aracı olarak değil, aynı zamanda ifade aracı olarak gören Papert (1980), programlamayı öğrenmenin yazmayı öğrenmeye benzediğini, çocuklara fikirlerini düzenlemenin ve ifade etmek için yeni yollar sağladığını ve bu yolla matematik fobisi oluşmadan önemli matematiksel fikirler edinebileceğini belirtmiştir.

Günümüzde bilgisayarların ve teknolojinin eğitim-öğretimin hemen her yönünü kapsamı oldukça yaygın bir durumdur. Ancak 1960'ların sonunda, bilgisayarların çocukların düşünme biçimlerini etkileyen bir ayrıştırıcı görevi görebileceği ve tetikleyici olabileceği fikri radikal bir fikirdi. Öğretim teknolojisi 1960'larda doğdu. İlk olarak bilgisayarlar okullarda yaygınlaşmaya başladı ve bu sayede çocukların erken zamanda bilgisayarlara erişimi mümkün kılındı (Bull vd., 2020). Bilgisayarı kompütasyonel bir nesne olarak düşünme ve kompütasyonel düşünme teriminin kullanımı, öğretim teknolojileri alanının kökenlerine kadar inebilir. Programlama kavramının ilköğretim düzeyine girişi Logo ile olmuştur (Franklin vd., 2017).

Matematikçi, bilgisayar bilimci ve aynı zamanda bir eğitimci olan, yapay zekânın öncülerinden kabul edilen Seymour Papert, çocuklar için tasarladığı Logo programlama dili hakkında araştırmalarıyla tanınır. Witwatersrand ve Cambridge Üniversitelerinde yaptığı çalışmalarla doktora derecesini alan Papert daha sonra Massachusetts Institute of Technology (MIT) Marvin Minski ile Yapay Zekâ Laboratuvarı'nın koordinatörlüğünü yapmadan önce Cenevre Üniversitesi'nde İsviçreli psikolog Jean Piaget ile çalıştı. Böylece Papert, bilgisayarların çocukların

düşünmesini kolaylaştırabilecek yolları düşünmek için iyi bir donanıma sahiptir (Bull vd., 2020).

Papert, 1960’larda Massachusetts Institute of Technology (MIT) profesörü olarak çalışmalarına Logo programlama dili üzerinde başladı. MIT’te çalıştığı ekip ile LISP programlama dilini geliştirdi ve kolay öğrenilebilecek basit sözdizimi ve etkileşimli bir ortam üzerine odaklandı. Amaç, çocukların gelecekteki eğitim-öğretim ve kariyerleri için değerli olan kompüstasyonel düşünme becerilerini geliştirmelerini sağlamaktı. Papert’in çabaları, çocukların bilgisayara, programlama yoluyla öğreterek öğrenebilecekleri inancına dayanıyordu. Bu inancında yalnız değildi. Seçkin bilgisayar bilimcisi Knuth (1974), benzer şekilde şöyle yazmıştı: “Bir insanın bir şeyi birine anlatana kadar gerçekten anlamadığı sıkça söylenir. Aslında insan, bir şeyi, bir bilgisayara öğretilene kadar gerçekten anlamamıştır.” (s. 327).

Şekil 2’de verilen Logo kaplumbağası, çocukların dili öğrenmelerini daha eğlenceli ve etkileşimli hale getirmek için geliştirilen küçük bir robottur. Kullanıcılar, Logo programlama dilinde kaplumbağaya ileri git veya sola dön gibi komutlar verebiliyor ve kaplumbağayı bu komutları hareket ettirebiliyorlardı. Kaplumbağa robotu, öğrencilerin programlama kavramlarını ve mantıksal düşünmeyi uygulamalı ve etkileşimli bir şekilde öğrenmelerine yardımcı olmak için öncelikle bir eğitim-öğretim aracı olarak kullanıldı. Kaplumbağa robotu ve Logo programlama dili okullarda yaygın olarak kullanılmaya başlandı ve öğretim teknolojileri alanında etkili oldu (Bull vd., 2020). Papert’in kaplumbağa robotu ve Logo programlama araştırmaları, çocukların etkileşimli, keşfedici öğrenme yoluyla karmaşık problem çözme becerilerini nasıl öğrenebileceklerini gösterdiği için yapay zekâ ve bilişsel bilim alanlarında da önemli kabul edildi (Bull vd., 2020).

Şekil 2

Logo Kaplumbağa



Not. *Mindstorms: children, computers, and powerful ideas* (ön kapak), S. Papert, 1980, Basic Books, Inc., Publishers Yayınevi’nden alınmıştır.

Papert'in Logo için ilk tanımlamaları, dil ve matematik gibi konuları keşfetmek için bilgisayar kullanımının, onu kolaylaştırmak için özel olarak tasarlanmış bilgisayar dilinin geliştirilmesini gerektirdiğine olan inancının ürünü olmuştur (Bull vd., 2020). Papert (1980) çocukların yetişkinlere yönelik programlama dillerini kullanmasını, "Küçük kardeşlere verilen giysiler gibi, asla tam olmaz." (s. 53) diyerek çocukların kendilerine uygun olmayan yetişkin kıyafetlerini kullanmasına benzetmiştir. Papert'in Logo ve öğretim teknolojisi araştırmaları, öğretim teknolojisi ve bilgisayar bilimleri alanlarında önemli bir etkiye sahiptir. Fikirleri ve yöntemleri, öğretim teknolojisinin geliştirilmesine ve bilgisayar bilimlerinin çocuklar için öğretimine devam ediyor. Ayrıca, sınıflarda teknolojinin kullanımını savunuyor ve çocukların en erken dönemlerde bilgisayarlara ve programlama dillerine erişimlerinin olması gerektiğine inanıyordu. Bu sayede çocukların erken yaşta sorun çözme ve eleştirel düşünme becerilerini geliştirmelerine yardımcı olacağını belirtmiştir.

Logo aynı zamanda kompütasyonel düşünmenin ilk uygulama alanı olup, ilkökul seviyesinde öğretim amaçlı kullanılan ilk programlama dilidir (Feurzeig vd., 2011). Papert (1980), Logo'nun öğrencilerin anlayışlarını geliştirmelerine yardımcı olacağını düşünmüştür. Papert (1988), Logo'yu "matematik ve yazı hakkında konuşma ve düşünme şeklinizi ve bunlar arasındaki ilişkiyi, öğrenme hakkında konuşma şeklinizi ve hatta okuldaki insanlar arasındaki ilişkileri değiştirmeye yardımcı olmak için tasarlanmış bir araç" (s. 7) olarak değerlendirmiştir. Logo'nun kullanımından kaynaklanan olası pedagojik avantajları gösteren geniş araştırma yelpazesinin varlığı vurgulanmasına rağmen (Bar-On, 1986; Howe vd., 1982; Layman ve Hall, 1988) çoğu sistematik araştırma yalnızca lise öğrencileriyle yapılmıştır (Rodríguez-Martínez vd., 2020). Buitrago Flórez vd (Buitrago Flórez vd., 2017), kompütasyonel düşünme ile ilgili becerilerin sadece ortaokul ve lise seviyelerinde değil, ilkökul seviyelerinden itibaren öğretilmesi gerektiğini ve öğrencilerin bilişsel gelişimlerini daha erken bir yaşta başlatmak için ilkökul seviyelerinde öğretilmesi gerektiğini belirtmektedirler.

Papert'in öncü çalışması o dönemde kalıcı olamamış ve yaygın olarak benimsenmemiştir (Pierce, 2013) ancak Logo'nun ortaya çıkmasından sonra, kodlama etkinliklerinin ortaöğretimde öğretilmesi amacıyla, Logo'ya benzeyen birçok programlama dili ortaya çıkmıştır. İlkokul seviyesinde matematik öğretimine ilişkin olarak, programlama etkinliklerinin matematik öğrenimi üzerindeki olası etkisini incelemek için farklı uygulamalar kullanılmıştır. Örneğin, Logo'ya benzeyen bir

programlama ortamı, okul öncesi öğrencilerle problem çözme stratejileri ve sayılar ve geometri ile ilgili farklı matematik becerileri çalışmak için başarıyla kullanılmıştır (Fessakis vd., 2013). Geliştirilen bir başka programlama aracı, öğrencilerle matematik fikirleri üzerinde çalışmak için etkinliği kanıtlanmış ToonTalk'tır (Kahn vd., 2011). Bu ortamlar arasında, sistematik, yaratıcı düşünme ve problem çözme gibi becerileri geliştirmek ve işbirlikli öğretim ortamını destekleyerek öğrencilerin öğrenmelerine yardımcı olmak için tasarlanmış en popüler programlama dili Scratch'tir (Brennan ve Resnick, 2012).

2.4.2. Scratch

Scratch, K-12 düzeyinde en yaygın kullanılan blok tabanlı kodlama programlarından biridir (Marcelino vd., 2018). Scratch, programlama diliyle oluşturulan görsel bir tasarım dilidir (Erol, 2016). MIT MediaLab, bilgisayar bilimi ve programlama çerçevesinde 2007'de, sekiz ile on altı yaş arasındaki çocukların kod yazmayı öğrenmelerine yardımcı olmayı amaçlayan Scratch web sitesini başlattılar. Programın amacı, programlamaya yeni başlayan öğrencilere programlamayı sevdirmektir. Yeni başlayanlar için kullanıcı dostu bir arayüz (başlangıçta Squeak, ancak daha sonra Javascript) ile başlar ve kolay giriş sağlaması açısından Logo'nun Kaplumbağa Konuşmasına benzer bir kodlamaya başlangıç ekranı vardır. Scratch, dijital (bir hareketli grafik) veya fiziksel bir nesnenin programlanmasına izin verir. Kullanıcılar yarattıkları oyun ve dijital hikâyeleri paylaşabilmektedirler (Erol, 2016). 2019 itibarıyla websitesinde (scratch.mit.edu/statistics) 44 milyonun üzerinde proje paylaşılmıştır ve bunların %43'ü Amerika Birleşik Devletleri'ndeki kullanıcılardandır (Howard ve Howard, 2022). Scratch, arama motoru sonuçlarına göre görsel blok programlama dilleri listesinin başında gelir (Zhang ve Nouri, 2019). Yurt dışında yapılan araştırmalar Scratch programının kullanımının öğrencilerin problem çözme becerilerine olumlu yönde etkilediği bulunmuştur (Alp, 2019).

Çevrimiçi ve çevrimdışı olarak erişilebilir olan Scratch, sadece programlamayı öğretmek için değil, aynı zamanda çocukların problem çözme gibi becerileri kazanmalarına yardımcı olmak için de kullanılabilir (Erol ve Çırak, 2022). Dijital okuryazarlığı ve matematiksel akıcılığı artırmak için öğrenciler, karakterler yaratarak ve canlandırarak bilgisayar kodlamasının ayrıntılarını elektronik olarak bir araya getirirler (Amador ve Soule, 2015). Scratch hesapları, kullanıcılara Scratch

programları oluşturma ve çalıştırma, programları kendi çevrimiçi klasörlerine kaydetme ve programları Scratch topluluğuyla paylaşma olanağı sağlar. Kullanıcı herhangi bir paylaşılan Scratch programını kopyalayabilir ve yeniden düzenleyebilir. Paylaşılan projelerinin kaç kez görüntülendiğini veya yeniden düzenlendiğini takip edebilir. Bu unsur öğrenciler için motive edici olabilir. Scratch, programlamayı paylaşma, tartışma ve programları değiştirme fırsatlarına izin veren işbirlikçi, sosyal bir etkinlik olarak konumlandırıran, paylaşma güçlü bir vurgu yapan geniş bir çevrimiçi topluluğa sahiptir.

Programlama, özünde bir tasarım ve üretim süreci olup, problem çözme deneyimi için bir ortam oluşturabilir çünkü programlama süreci, problemin belirlenmesi, çözümün geliştirilmesi, çözümün denetlenmesi gibi problem çözme sürecinin basamaklarını içeren zihinsel bir süreci; belirli bir plan dâhilinde ilerletir ve süreç tesadüfi değildir (Resnick vd., 2009). Wing (2006) kompütasyonel düşünmeyi problemleri çözmeye, sistemleri tasarlamaya ve insan davranışını anlamaya yönelik bir yaklaşım benimseyebilen bir beceri olarak tanımlamıştır. Bu tanım, hesaplama için temel olan kavramlardan yararlandığı bilgisayar bilimleri ve eğitim alanında büyük bir etkiye sahiptir ve eğitim alanında pek çok ülkede benimsenmiştir (Bo vd., 2017). Ancak, Scratch programının kompütasyonel düşünme becerilerini ne düzeyde geliştirdiğini değerlendirmek kolay değildir, çünkü blok tabanlı bir programlama dilidir ve kullanıcılar bu dili kullanarak görsel blokları bir araya getirerek programlar oluştururken bir grafik kullanıcı arayüzü (GUI) içinde (yani, bir bilgisayar ekranı üzerinde) geliştirirler ve çalıştırırlar. Araştırmacılar, bu konuda çeşitli araştırmalar yapmış ve süreç içerisinde bazı programlar geliştirmişlerdir. Bu araçlardan bir tanesi Scratch programlarının kompütasyonel düşünme becerilerini ne düzeyde geliştirdiğini otomatik olarak değerlendirmek için geliştirilmiş bir web aracı olan Dr. Scratch'tir (Moreno-León vd., 2015). Dr. Scratch, Scratch program dosyalarını okumak ve yazmak için kullanılan bir Python modülü olan Kurt kullanmaktadır ve Scratch JSON dosyalarını Python nesnelerine serileştirebilen Python yerleşik modülü Cpickle'a dayanmaktadır (Chang vd., 2018). Zhong Chang vd. (2018), uygulamada, Dr. Scratch'in Cpickle'a dayalı eski altyapısı, yüksek başarısızlık oranına ve düşük verimliliğe neden olduğunu iddia etmiş ve kendileri Dr. Scratch'teki bu kusurları çözmek için statik bir analiz aracı olan (Another Tool for Language Recognition) ANTLR'ye dayalı bir tür Scratch Analiz Aracı (SAT) tasarlamışlardır ancak bu aracın

da, tasarım, hata ayıklama veya yeniden karıştırma becerileri gibi bazı temel kompütasyonel düşünme yeterliliklerinin bir programın statik kodlarını analiz ederek ölçememek gibi sınırlılıkları olduğunu belirtmişlerdir.

Scratch'i kullanıcıları için kişiselleştirilebilir anlamlı bir öğrenme aracı olarak gören Resnick vd. (2009), önemli matematiksel kavramların öğrenilmesinde hazır bir ortam sunduğunu belirtmiştir. Scratch, esnek ve görsel yapısı ile programlama engellerini ortadan kaldırır. Kullanıcılarına ileri düzey animasyon ve oyunlar yaratmaları için fırsatlar sunar. Geçici sahneler, gerçek dünyadaki tiyatro sahnelerine benzetilebilir. Scratch, yetenekli öğrencilere hayal güçlerini sahnede kullanma fırsatı verir. Ayrıca yazılım, öğrencilere koordinat sistemleri gibi matematiksel kavramları da öğretir. Scratch yalnızca teknik becerileri öğretmekle kalmaz, aynı zamanda öğrencileri iş birliği yapmaya da teşvik eder. <http://scratch.mit.edu> öğrenciler için ve <http://scratchd.media.mit.edu> eğitimciler için tasarlanmıştır. <http://scratch.mit.edu/> sitesi, kullanıcılar arasında yapıcı geri bildirim oluşturmak için iyi bir temel sağlar.

Scratch arayüzü, sahne adı verilen merkezi bir alandan ve bu alanda kullanıcının oluşturduğu animasyonların veya oyunların gösterildiği bir bölümden oluşur. Sahne alanının sol tarafında blokların bulunduğu bir blok paleti yer alır ve bu palet farklı renklerdeki bloklardan oluşur. Her bir blok, farklı bir kodlama işlevini temsil eder. Blok paletinin üst kısmında farklı kategorilerde sıralanmış bir dizi sekme yer alır. Arayüzün alt kısmında farklı özellikleri ve değişkenleri düzenlemek için farklı sekmeler vardır. Bu sekmeler arasında, projenin görünümü, sesi, sahne geçişleri ve diğer özelliklerin kontrol edilebileceği alanlar bulunur. Ayrıca, kullanıcının projenin kodunu düzenleyebileceği bir metin editörü de yine bu kısımda yer alır.

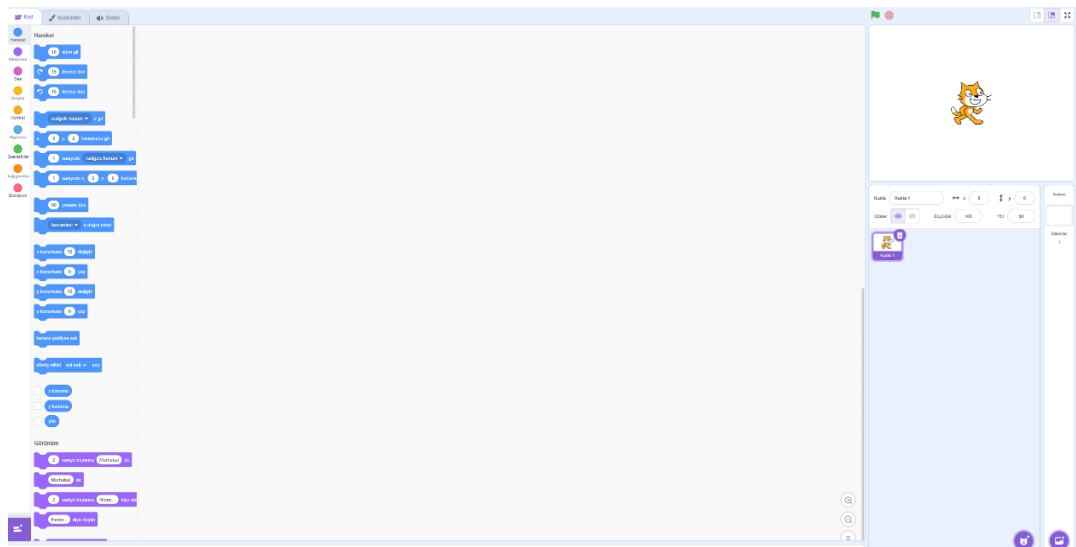
Scratch programının arayüzünde ayrıca projeleri çalıştırmak, duraklatmak ve yeniden başlatmak için bir dizi düğme ve kontrol bulunur. Bu düğmeler, projenin kodunda yapılan değişiklikleri kaydetmek, projeyi paylaşmak ve diğer işlevleri gerçekleştirmek gibi pek çok amaç için kullanılabilir. Scratch ara yüzü Türkçe dilini desteklemektedir. Scratch ekranında oluştur sekmesine tıkladıktan sonra Şekil 3'te verilen kodlama yapacağımız alan gelir. Ortada yer alan büyük boşluk sol tarafta yer alan kodlama bloklarının sürüklenerek bırakılacağı ve böylece sağ tarafta bulunan sahnedeki kuklanın kurgulanacağı yerdir. Ekranın sağ alt tarafında kodlayacağımız kuklanın yer aldığı kısımdır. Burada kuklamızı silebilir ya da sahneye farklı kuklalar ekleyebiliriz. Yine bu kısımda yer alan göz simgesine tıklayarak kuklanın sahnede

görünüp görünmeyeceğini, büyüklüğünü, x ve y koordinatlarını kullanarak kuklanın sahnedeki yerini belirleyebilir, adını değiştirebiliriz. Ekranın sol üst köşesinde kod sekmesinin yanında yer alan kostümler sekmesinden seçilen kuklanın görünümünü, ses kısmından ise arka plansa çalacak sesleri seçebiliriz. Bunlar programın tabanında yer alan hazır seslerden seçilebildiği gibi kullanıcı tarafından oluşturulan sesler de eklenebilir ya da kullanıcı istediği farklı bir sesi yükleyebilir.

Şekil 3'te gösterilen arayüzde sol sütunda kod blokları, ortada kodlamanın yapılacağı alan ve sağ tarafta kodlamayı yaptığımız sahne vardır. O sahneye ait kukla yer almaktadır. Öğrencilere arayüzde yer alan kod bloklarının işlevleri, sahne ve kukla seçimi gibi temel kodlama dinamikleri bu arayüz üzerinde gösterilmiştir. Kodlama yapılacak olan ortadaki büyük alanın sağ alt köşesinde yer alan büyüteç imleçleri ile kodlama blokları yakınlaştırılıp uzaklaştırılabilir. Bu, kodlama yapan kullanıcının kodlamayı yaptığı cihaza bağlı olarak kullanabileceği bir erişilebilirlik özelliğidir. Sahnenin arka planında yer alan dekor ise ekranın en sağ alt köşesinde yer alan arka plan seçme özelliğinden faydalanılarak değiştirilebilir. Dekor olarak Scratch programında yer alan hazır dekorlar seçilebileceği gibi kullanıcı tarafından çizilerek oluşturulabilir. Sağ tarafta yer alan bölmede sahnenin hemen üzerinde yer alan yeşil bayrağa tıklandığında oluşturulan sahne oynatılır, yanında yer alan kırmızı buton ile sahne durdurulur.

Şekil 3

Scratch arayüzü



Scratch programlama ortamında kullanılabilecek blokları içeren Scratch blok kütüphanesi, öğrencilerin daha karmaşık programlar yazmalarına olanak sağlayan bir

araçtır. Kütüphane, programlama konseptleri, kontrol yapıları, operatörler, değişkenler, işlevler ve daha birçok konuda blokları içermektedir. Öğrenciler, blokları sürükleyip bırakarak kolayca kendi programlarını oluşturabilir; arzu ettikleri sahneleri tasarlayabilirler. Örneğin, hareket bloklarında yer alan blok kodları ile kuklaların hareket yönleri, biçimleri bu hareketleri ne kadar süreyle yapacakları gibi pek çok hareket çeşidi kodlanır. Yönelme kod blokları kullanılırken x eksenini üzerinde yapılacak kodlamanın pozitif değerleri sağa, negatif değerleri ise sola yönelmeyi temsil ederken benzer şekilde y eksenini üzerinde yapılacak olan kodlamanın pozitif değerleri yukarı, negatif değerleri ise aşağı yönelmeyi temsil edecek şekilde tasarlanmıştır. Görünüm bloklarında yer alan kod blokları ile kukla için konuşma baloncukları oluşturulabilir. Yine bu bölümde yer alan kostümler ile sahnenin akışına göre kuklanın görünümü değiştirilebilir. Blok kütüphanesinde yer alan ses blokları ile kodlanan sahne için sesler eklenebileceği gibi belirli anlar için özel efektler de eklenebilir. Seslerin seviyesi, zamanlaması vs. gibi pek çok işlev de yine bu blokların içerisinde yer almaktadır. Sahnenin başlatılması ve kodların tetiklenmesi için gerekli bloklar olay blokları arasında yer alır. Tasarlanan sahne için gerekli bekleme, tekrarlama vb. işlevler için kullanılırlar. Kodlar kontrol blokları içerisinde yer alır. Bloklar arasında yer alan koşul blokları, belirli şartları sağlanması ya da sağlanmaması durumunda kuklanın yapması ya da yapmaması gerekenleri tanımlamaya olanak sağlar. Kütüphanede yer alan algılama blokları farklı işlevlere sahiptir. Örneğin, altıgen biçiminde tasarlanmış olan algılama blokları doğru veya yanlış şeklinde değerler üretir. Bu değerler koşul blokları ile örneğin “eğer mavi renkte ise şunları yap” şeklinde tanımlanmış bir koşul kodunun mavi renk şartını sağlayıp sağlamadığını denetler. Bu denetimi yaparken istenen koşullar sağlanıyorsa doğru diyerek belirtilen kodların çalışmasını; istenen şartların karşılanmaması durumunda ise yine isteğe bağlı farklı kodların tetiklenmesini sağlar. Oval şekilde tasarlanmış algılama kod blokları ise sayısal veya sözcüksel değerler üretir. Örneğin, ses düzeyini görmek için burada yer alan ilgili kod bloğuna tıkladığınızda ses seviyesini sayısal olarak gösterir. Kütüphanede yer alan operatörler blok kodlarında toplama, çıkarma, çarpma ya da bölme gibi dört işlem içeren ve değer üreten bloklar olduğu gibi doğru ya da yanlış şeklinde değerler üreten bloklar da mevcuttur. Bunlar kontrol bloklarında yer alan karar bloklarıyla beraber kullanılırlar. Bu blokların arasında yer alan mod kod bloğu bir sayının bir sayıya bölümünden kalanı verir. Yuvarlama bloğu virgüllü sayıları aşağı ya da yukarı yuvarlar. Yine bu kısımda yer alan fonksiyon kod bloğu içerisinde örneğin

karekök alma, trigonometrik fonksiyonlar gibi pek çok matematiksel işlem de yer almaktadır. Blok kütüphanesinde yer alan değişkenler içerisinde pek çok değeri tutabilen yapılardır. Bu işlev liste oluşturma ile kullanılabilir. Kütüphanede yer alan blokların kısmında ise özel girdiler eklenebilir.

2.5. Kompütasyonel Düşünme

Teknolojinin hayatımızdaki yeri her geçen gün artarken, yeni teknolojileri anlayabilme ihtiyacı da bu yaklaşımın önemini ve gerekliliğini her geçen gün daha da artırmaktadır. Kompütasyonel düşünme, dijitalleşen dünyada bilgisayar bilimlerini kullanarak öğrencilerin problem çözme becerilerine katkı sunabilir. Eğitimde, insanların günlük yaşamda ve iş hayatında işlevselliği arttırabilmeleri, problemleri çözebilmeleri ve dijital yeniliklere katkıda bulunabilmeleri için kompütasyonel düşünme yetisine sahip olmaları gerektiği kabul görmektedir (Kotsopoulos vd., 2017). Bu ihtiyaç, teknolojinin her geçen gün daha fazla hayatımızın içine girmesi ve yeni teknolojilerin anlaşılmasını da gerekli kılmasıyla daha da önem kazanmaktadır. Kompütasyonel düşünme, temelde problem çözme, sistem tasarlama ve insan davranışını anlamaya yönelik bir yaklaşımdır ve bilgi işlemin temel kavramlarını kullanır. Bu bakış açısı, bilgisayar biliminin ilkelerini bir çerçeve olarak kullanarak algoritmik düşünmeyi de içerir. Bu çerçevede problemler, soyutlama (bir sorunu farklı ayrıntı düzeylerinde görme), algoritmik düşünme (görevleri küçük parçalara bölme ve birbirleri ile bağlama), ayrıştırma (bir sorunu çözmek bir dizi küçük sorunu çözme) ve örüntü (yeni bir sorunu daha önce karşılaşılan sorunlarla ilişkilendirme) olarak tanımlanabilir. Kompütasyonel düşünme tanımını alan yazında farklı şekillerde görebiliriz. Wing (2006), kompütasyonel düşünmeyi bilgisayar biliminin temel kavramlarından yararlanarak problemleri çözmek, sistemleri tasarlamak ve insan davranışlarını anlamak olarak tanımlamıştır. Aho (2012), yeni sorunlarda çözüm üretmek için her zaman uygun modellere sahip olmadığımızı, bu durumlarda, kompütasyonel düşünmenin uygun yeni hesaplama modelleri icat etmeyi içeren araştırma faaliyeti olduğunu belirtmiştir. Fagerlund vd. (2021) kompütasyonel düşünmeyi, hesaplamalı dünyayı anlamak ve disiplinler içinde ve arasında çok boyutlu problem çözme becerilerini kullanmak için gerekli olan entelektüel temeli somutlaştıran bir terim olarak tanımlamaktadır. Uluslararası Eğitimde Teknoloji Derneği (ISTE) , kompütasyonel düşünme becerisini veri toplama, analiz etme,

soyutlama, algoritma tasarımı ve bilgi işlem kullanarak öğrencilerin hem bilgisayar bilimi kavramlarını hem de akademik disiplinlerini geliştirecek uygulamaları öğrenme ve etkinliklere entegre etme olarak tanımlarken, Ulusal Araştırma Konseyi (NRC) , matematik ve bilişimsel düşünme, fiziksel değişkenleri ve bunlar arasındaki ilişkileri temsil etmek için bilgisayar araçlarının kullanımını içeren bir düşünme biçimi olarak tanımlar. Genel olarak kompütasyonel düşünme, temel bilgisayar bilimi kavramlarının yanı sıra programlama becerilerini de kapsayan daha kapsamlı bir terimdir (Chowdhury vd., 2019). Tüm bu tanımlamalara karşın kompütasyonel düşünmenin sağlam çekirdek bir tanımı yoktur (Grover ve Pea, 2018). Ortak görüş ise bilgisayar gücünün kullanıldığı yönündedir (Denning ve Tedre, 2019). Sınırları ne olursa olsun kompütasyonel düşünme gelecek nesillerin geliştirmesi gereken bir 21. yüzyıl becerisidir (Zhang & Nouri, 2019). Bu inanç uluslararası düzeyde kabul görmüştür. Artan sayıda eğitim-öğretim sistemi son yıllarda kompütasyonel düşünmeyi zorunlu eğitim-öğretime bütünleştirmiştir. Kompütasyonel düşünme, günümüzün teknoloji odaklı dünyasında bireylerin ve toplumların ilerlemesi için kritik öneme sahip bir beceri haline gelmiştir (Kotsopoulos vd., 2017). Kompütasyonel düşünmenin bilgisayar bilimleriyle olan bariz ilgisine rağmen, akademisyenler kompütasyonel düşünmenin anaokulundan başlayarak bilgisayar bilimleri dışındaki disiplinlerde de öğretilmesi gerektiğini savunmaktadır (Barr ve Stephenson, 2011).

Tüm dünyada birleşik ve net bir isme sahip diğer köklü konuların (matematik ve tarih gibi) aksine, farklı ülkeler bu yeni konu için farklı terimler benimsemiştir. Örneğin, Avustralya’da ICT yeteneği ve İsveç’te dijital yeterlilik adlarıyla anılan ve İngilizce’de computational thinking olarak geçen kompütasyonel düşünme, alan yazında farklı kaynaklarda farklı biçimlerde dilimize çevrilerek kullanılmıştır. Bunlardan kompütasyonel düşünme (Şahiner ve Kert, 2016), bilişimsel düşünme (Özkeş, 2016), bilgisayarca düşünme (Korkmaz vd., 2015), hesaplamalı düşünme (Özçınar, 2017), bilgisayarlı düşünme ifadeleridir (Elçiçek, 2020).

Kompütasyonel düşünme, karmaşık bir sorunu daha kolay anlaşılır ve çözülebilir daha küçük parçalara bölerek ve mantıksal akıl yürütme kullanarak çözüm bulma yoludur. Hesaplama temel ilkelerini anlamak ve bu ilkeleri kullanarak yeni sistemler ve araçlar tasarlamak ve geliştirmeyi içerir. Kompütasyonel düşünmenin önemli yönlerinden biri, sorunları daha küçük ve yönetilebilir parçalara bölmektir. Bu süreç, ayrıştırma olarak bilinir ve sorunun daha kolay anlaşılıp çözülmesine olanak

tanır. Örneğin, sorun yeni bir bilgisayar programı oluşturmak ise, ilk adım sorunu girdi, işleme ve çıktı gibi daha küçük parçalara bölmek olacaktır. Kompütasyonel düşünmenin diğer önemli bir yönü, sorunun anahtar özelliklerini belirleme ve ilgisiz ayrıntıları görmezden gelme olarak adlandırılan soyutlama sürecini içermesidir. Bu, sorunun esasen önemli yönlerine odaklanarak daha verimli bir çözüm bulmak için olanak tanır. Örneğin, yeni bir bilgisayar programı tasarlarken, donanım veya işletim sistemi özel ayrıntılarını görmezden gelmek ve programın temel işlevselliğine odaklanmak önemlidir.

Mantıksal akıl yürütme de kompütasyonel düşünmenin önemli bir parçasıdır. Bu süreç mantıksal argümanlar ve sistematik yöntemler kullanarak çözüm bulma yolunu içerir. Algoritmalar, matematiksel modeller ve diğer mantıksal araçlar kullanarak bir sorunu çözmeyi içerebilir. Örneğin, bir problemi çözerken, mantıksal akıl yürütme sorunu çözmek için en verimli ve etkili yolu bulmak için kullanılabilir. Papert'in kompütasyonel düşünme vizyonu iki temel fikre dayanır (Bull vd., 2020):

1. Bilgisayarı programlama eylemi, matematik ve dil sanatları gibi konuları öğrenme sürecini kolaylaştırabilir.
2. Dili ve matematiği bu şekilde keşfetmek ve kolaylaştırmak için ise özel olarak tasarlanmış bir bilgisayar dilinin geliştirilmesi gerekir.

Kompütasyonel düşünmenin kullanışlı bir çerçeve oluşturması için, bileşenlerini sınırlandırmak gereklidir. Kompütasyonel düşünme, programlamadan daha karmaşıktır, ancak programlama, kompütasyonel düşünmenin kullanımını gerektirir ve genellikle ona ulaşmak için kullanılır. Kompütasyonel düşünmenin temel kavramları ve programlamanın mantık-yapıları, temel sıralar, döngüler, yinelemeler, koşullar, fonksiyonlar ve değişkenler gibi farklı hesaplama kavramları olarak sınıflandırılabilir (Papert, 1980). Kompütasyonel düşünme alanının henüz olgunlaşmamış olması, bu düşüncme biçiminin eğitim-öğretim alanları içerisinde incelenmesinde bir belirsizliğe yol açtığından öğretmenler genellikle kompütasyonel düşünme ile aşina değildir ve bu nedenle öğretim programı ile kompütasyonel düşünme arasındaki bağlantıları bulamazlar (Rodríguez-Martínez vd., 2020). Bu gerçek, kompütasyonel düşünmenin öğretim süreçlerine nasıl entegre edileceği konusunda biraz uzlaşma sağlasa da farklı vizyonlar, plansız eylemler ve politikalar, kompütasyonel düşünmenin gelişimini son derece zorlaştırmaktadır. Ayrıca,

kompütasyonel düşünmenin standart bir tanımının olmaması, ölçümlerin her araştırma parçasında büyük ölçüde değişmesine yol açmakta ve sonuçların uygunluğunu ve karşılaştırılabilirliğini oldukça zorlaştırmaktadır. Kompütasyonel düşünme, hedefi, problemleri etkili ve yaratıcı bir şekilde çözmek olan beş bilişsel süreçle sınıflandırılabilir (Shute vd., 2017):

- Problemi yeniden yapılandırma: Problemi çözülebilir ve tanıdık hale getirerek yeniden yapılandırmak.
- Yinelemeli yapılandırma: Önceki bilgiler temel alınarak sistemi aşamalı olarak oluşturmak.
- Problemi parçalara ayırma: Yönetilebilir parçalara ayırmak.
- Soyutlama: Karmaşık problemlerin veya sistemlerin temel yönlerini modellemek.
- Sistematik test: Çözümler üretmek için amaçlı adımlar atmak.

Soyutlama, kompütasyonel düşünmenin altında yatan ana ögedir (Wing, 2006). İnsanlar karmaşık sistemlerden ilgisiz verileri atıp ilgili bilgileri çıkararak farklı temsiller arasında ortaklıklar bulmak ve desenler oluşturmak için soyutlama yaparlar (Wing, 2010). Soyutlama katmanlara sahiptir, bu nedenle her katmanı tanımlamak ve arasındaki ilişkileri açıklamak gerekir. Bu ilişkiler;

- a. Her katmanda soyutlama,
- b. Tümünde soyutlama,
- c. Katmanlar arasındaki bağlantıyı içerir (Shute vd., 2017).

Ulusal Araştırma Konseyi'nin kapsamlı bir raporuna göre, kompütasyonel düşünme, tüm alanlarda temel ve evrensel olan beş unsurdan oluşur (Marcia Linn, 2010):

1. Hipotez testi,
2. Veri yönetimi,
3. Paralellik,
4. Soyutlama,
5. Hata ayıklama.

Herhangi bir alandaki karmaşık bir problemi çözerken, sistem nasıl çalıştığını anlamak için sistemli bir şekilde hipotez üretmek ve test etmek gereklidir. Tüm olasılıkları test etmek imkânsız olduğundan, test edilecek doğru parametreleri seçmek önemlidir (Shute vd., 2017). Veri yönetimi, çeşitli kaynaklardan veri toplama, veri desenlerini işleme ve verileri anlamlı bir şekilde temsil etmeyi içerir. Paralellik, birden fazla kaynaktan veya boyuttan aynı anda bilgi işlemeyi ifade eder. Soyutlama, karmaşık bir problemin/sistemin çalışmasını modelleme üzerine odaklanır. Son olarak, hata ayıklama, belirli modeller oluşturduktan sonra hataları bulmayı ve düzeltmeyi içerir (Shute vd., 2017). Matematiksel düşünme, denklemler ve fonksiyonlar gibi matematik problemlerini çözmek için matematik becerilerinin uygulanmasını içeren, matematik hakkındaki inançlar, problem çözme süreçleri ve çözümler için gerekçelerden oluşan bir süreçtir (Weintrop vd., 2016). Kompütasyonel düşünme ve matematiksel düşünme arasındaki temel ortaklık, problem çözme süreçleridir (Wing, 2006).

Wing (2006), kompütasyonel düşünme yaklaşımlarının tüm disiplinlerde temel haline geleceğini ve hesaplama teknolojilerindeki ilerlemelerin araştırmacılara yeni problem çözme stratejileri tasarlama ve sanal ve gerçek dünyada yeni çözümler test etme imkânı sağlayacağını belirtir. Fizik ve mühendislik gibi belirli alanlar hesaplama yöntemleriyle uzun süredir disiplinler arası iş birliği içerisinde. Ancak biyoloji ve kimya gibi disiplinlerde, gerçek dünya problemleri genellikle stokastik ve doğrusal olmayan yapıya sahip olduğundan, deterministik sistemler bu disiplinlerde daha az kullanışlıdır ve doğrusal olmayan teriminin neredeyse pratikte çözümsüz ile eş anlamlı olması demektir (Weintrop vd., 2016). Ancak doğa doğal olarak doğrusal olmayandır ve kaotik davranışlarla karakterize edilebilir. Hem matematiksel hem de fiziksel olarak, deterministik/doğrusal sistemler ise istisnaları oluşturur. Son yıllarda, doğrusal olmayan olguların matematiksel ve simülasyon modelleri kullanılarak araştırılmasıyla birlikte, hesaplama yöntemleri ile kapsamı genişleyen bir yelpaze oluşmuştur. Bilimsel alanlarda daha güçlü bilgisayarlar, yeni analitik yöntemler ve yüksek ayrıntılı hesaplama modellerinin geliştirilmesi, çeşitli bileşenlerin ve mekanizmaların dâhil edilebileceği yaklaşımların yeniden canlanmasına yol açtı. Bu nedenle, öğrencilerin hesaplama yöntemleri ve teknikleri konusunda eğitilmeleri, tüm bu gelişmelerle ayak uydurabilmeleri ve yaşanan gelişmeleri anlamlandırabilmeleri için gereklidir (Shute vd., 2017). Çeşitli kuruluşların kompütasyonel düşünce biçimini temsil ettiğini ifade

ettiği kavramlar Tablo 1’de verilmiştir (Denning, 2017). Bu kavramlar her kuruluş tarafından farklı şekillerde listelenmiştir.

Uluslararası Eğitimde Teknoloji Derneği (ISTE), Bilgisayar Bilimleri Öğretmenleri Derneği (CSTA) ile işbirliği içinde kompütasyonel düşünme özelliklerinin tanımlayıcı özellikleri aşağıda verilmiştir (ISTE, 2023).

Tablo 1

Kompütasyonel düşünme becerisine ait kavramlar

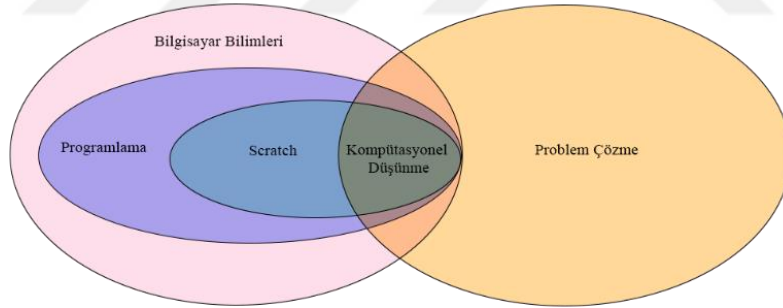
Öğrenciler için ISTE Standartları	Tanım	Alt kavramlar	Açıklama
Kompütasyonel Düşünme	Öğrenciler, çözümleri geliştirmek ve test etmek için teknolojik yöntemlerin gücünden yararlanacak şekilde sorunları anlamak ve çözmek için stratejiler geliştirir ve kullanırlar.	Problemi Tanımlama	Öğrenciler veri toplar veya ilgili veri kümelerini tanımlar. Bunları analiz etmek için dijital araçları kullanır; problem çözmeyi ve karar vermeyi kolaylaştırmak için verileri çeşitli şekillerde temsil eder.
		Ayrıştırma	Öğrenciler problemleri parçalarına ayırır, önemli bilgileri çıkarır ve karmaşık sistemleri anlamak veya problem çözmeyi kolaylaştırmak için modeller geliştirir.
		Algoritmik Düşünme	Öğrenciler otomasyonun nasıl çalıştığını anlar ve otomatik çözümler oluşturmak ve test etmek için bir dizi adım geliştirmek amacıyla algoritmik düşünceyi kullanır.

Denning (2017), kompütasyonel düşünmenin her çocuğu giderek dijitalleşen bir dünyada yaşama daha iyi hazırlayacağı ve kompütasyonel düşüncenin tüm alanlarda problem çözme becerilerine katkı sunacağını ifade etmiştir. Howard ve Howard (2020), matematiğin ve kodlamanın birbirine yakın ama birbirine değmeyen

iki beceri gibi görünebileceğini ancak kompütasyonel düşünmenin bu iki alan arasında bir köprü olduğunu belirtmişlerdir. Howard ve Howard (2020) ayrıca hem matematiğin hem de kodlamanın birlikte öğretilmesinin, güçlü bir matematik ve kodlama programının geliştirilmesine yardımcı olacağını belirtmişlerdir. Sorunlar hakkında daha soyut bir şekilde düşünme becerisi, öğrencilerin daha fazla çözüm fırsatı görebilmeleri için gereklidir. Bu soyut şekilde ya da kompütasyonel olarak düşünmek, genellikle öğrencinin var olan şemasının dışında kalan çözümlere ulaşmasının bir yoludur (Ioannidou vd., 2011). Kompütasyonel düşünme çoğunlukla soyutlama yoluyla problem çözmeye yönelik bilgisayar bilimleri yaklaşımı olarak anılsa da kompütasyonel düşünmenin diğer disiplinlerde kullanımı bir soruyu veya güçlüğü soyut bir şekilde kavramsallaştırılmasına destek olabilir. Öğrenciler kompütasyonel düşünmeyi ne kadar sık kullanırlarsa, çeşitli alanlardaki karmaşık sorunlara alternatif, benzersiz ve yaratıcı çözümler bulma konusunda o kadar gelişeceklerdir (Bennett vd., 2011) Şekil 4’te kompütasyonel düşünme, Scratch ve problem çözme arasındaki örnek bir ilişki gösterilmiştir.

Şekil 4

Scratch, kompütasyonel düşünme ve problem çözme



Not. “A Systematic Review of Learning Computational Thinking Through Scratch in K-9”, L. Zhang ve J. Nouri, 2019, *Computers & Education*, 141(103607), sayfa 2’den alınmıştır.

Yapılan pek çok tanım dikkate alındığında kompütasyonel düşünme için kesin ana hatlar çizmek zor. K-12 Computer Science Framework (K12CS2, 2024) çizdiği çerçevede kompütasyonel düşünme, tek bir beceriden ziyade birden fazla işi yapma becerisidir. Wing’in (2006) kompütasyonel düşünme hareketi olan sadece bilgisayar bilimlerinde uzmanlaşanların değil, herkesin bir bilgisayar bilimcisi gibi düşünmekten fayda sağlayabileceği vizyonundan yola çıkılarak bu becerinin problem çözme becerilerine entegre edilip edilemeyeceği araştırılmak istemiştir. Öğretimin büyük bir kısmının öğrencilerin bilgi edinmesi etrafında şekillendiği göz önüne alındığında,

komputasyonel düşünme becerilerinin gelişimine dikkatle bakmak yeni ve zorlu bir fikirdir. Birçok araştırma komputasyonel düşünmenin temel kavramlarını ve uygulamalarını daha genel olarak programlama, bilgi işlem veya bilgisayar bilimleri ile sınırlandırmış olmasına rağmen, komputasyonel düşünme kendine net sınırlar bulmakta zorlanan bir terimdir. Komputasyonel düşünmeye ilişkin özgü görüşler daha ziyade programlama merkezli olabilir ancak bu potansiyel olarak önemli alanları ihmal etme riski vardır. Buna karşılık, genel görüşler matematik gibi diğer yetkinlik alanlarıyla örtüşebileceğini söylemektedir (Fagerlund vd., 2021). Wing'in (2006) komputasyonel düşünme için, bilgisayar biliminin temel kavramlarından yararlanarak problem çözme, sistem tasarlama ve insan davranışlarını anlama sürecine yaptığı vurgudan yola çıkarak komputasyonel düşünmenin yalnızca bilgisayar bilimleriyle sınırlı kalmaması gerektiği; okuma, yazma ve aritmetik amaçlı analitik düşünme ile ilgilenen herkes için bir beceri olarak görülmesi gerektiği söylenebilir (Ateşkan ve Hart, 2021). Bu bağlamda, diğer disiplinlerde komputasyonel düşünmenin amacı öğrencilere nasıl düşüneceklerini öğretmek ve problemlerini çözmek, yaratmak ve verimli bir şekilde keşfedilebilecek yeni soruları ortaya çıkarmak için hesaplamayı nasıl kullanacaklarını anlamak olmalıdır (Ioannidou vd., 2011).

2.6. Scratch ve Komputasyonel Düşünme Kavramları

Komputasyonel düşünmenin en yaygın kullanım alanlarından birisi kodlamadır ve çocukların tamamını kodlama sürecine dâhil etmemek, gelecekte kodlama alanında inşa edilecek olan yapılara katılımın potansiyeline zarar verebilir (Kotsopoulos vd., 2017). Kodlamayı sadece programlamanın bir parçası olarak değerlendirmek yanlış olur. Çünkü Papert (1980), komputasyonel düşünmenin diğer düşünme becerilerini geliştirmenin iyi bir yolu olduğunu savunur. Papert'ın (1980) komputasyonel düşünme tanımı, bilgi işlemsel fikirler ve bilgisayar okuryazarlığı ile uygun problemleri çözmeyi içerir. Wing (2006), komputasyonel düşünmeyi çözümlerin bir formda temsil edilmesi için problemlerin ve çözümlerinin formüle edilmesinde yer alan düşünce süreçleri olarak tanımlar. Programlama dili soyut ifadelerden oluşur. Bu dil öğrencilerin programlama sürecini gerçek durumlarla bağdaşmayan bir olgu olarak anlamlandırmalarına yol açabilir (Korkmaz, 2016). Scratch ve Logo gibi ortamlar ise programlamanın soyut halini öğrenciler için daha izlenebilir ve somut hale dönüştüren araçlardır (Korkmaz, 2016).

Kodlama disiplinler arası yaklaşımla farklı derslerde de kullanılabilir. Örneğin öğrenciler, önceden hazırlanmış programları inceleyerek matematiğin kodlamada nasıl kullanıldığını keşfedebilirler. Bu programlar, matematiğin en belirgin olduğu kod satırlarına odaklanarak öğrencilerin matematiğin kodlama ile nasıl entegre olduğunu daha iyi anlamalarını sağlayabilir. Öğrenciler değişkenler ve matematik formülleri üzerinde yoğunlaşarak ve kodu ayarlayarak yaptıkları değişikliklerin sonuçlarını anında gözlemleyerek kodlamanın matematiği deneyimlemenin, temsil etmenin ve araştırmanın yeni bir yolu olduğunu keşfedebilirler. Bu sayede, matematiksel kavramları daha somut bir şekilde kavrayabilir ve kodlama becerilerini geliştirirken matematiksel düşüncelerini de pekiştirebilirler (Gadanidis, 2015). Pedagojik bir bakış açısıyla, “eğer” sorusuna odaklanmak, varsayımında bulunma, problem çözme ve genelleme için bir bağlam sağlar (Kotsopoulos vd., 2017).

Bazı araştırmalarda öğrencilerin birbirleriyle işbirliği yapmalarını, birbirlerine ilham vermelerini ve işbirlikli öğrenmeden bazı eğitsel faydalar elde etmelerini sağlamak için Web 2.0 teknolojisini kullanan sistemler mevcuttur ancak süreçte hangi becerilerin kazanıldığı net değildir (Ioannidou vd., 2011). Kodlama yoluyla programlama yaparken potansiyel olarak öğrenilen kompütasyonel düşünme (Yalçinkaya vd., 2018), Scratch gibi araçlarla kullanılan kod yapıları yüksek seviyeli kompütasyonel düşünce yerine temel düzeyde programlama yeterliliğini temsil ettiği iddia edilebilir. Scratch gibi araçları kullanırken, öğrencilerin katıldıkları programlama etkinlikleri, kompütasyonel düşünce ile ilgili becerileri ve anlama alanlarını farklı şekillerde geliştirebilirken, Scratch'teki pratik programlama olanaklarının, temel kavram ve uygulamalarında tanımlandığı gibi kompütasyonel düşünce ile ilişkili çeşitli beceri ve anlayış alanlarını nasıl temsil edebileceğini ve geliştirebileceğini sistematik ve kapsamlı bir şekilde alan yazında araştırılmamıştır (Fagerlund vd., 2021). Bu nedenle Scratch'i ve blok kütüphanesinin işlevlerini kompütasyonel düşünme kavramları açısından değerlendirmek doğru olacaktır. Brennan ve Resnick (2012), kompütasyonel düşünmeyi üç boyutta ele alır: kompütasyonel düşünme kavramları, Kompütasyonel düşünme uygulamaları ve kompütasyonel düşünme perspektifleri. Kompütasyonel düşünme kavramları, algoritmalar ve veri yapıları gibi temel bilgisayar bilimi prensiplerini içerir. Kompütasyonel düşünme uygulamaları, bu kavramların problem çözme ve proje geliştirmede nasıl kullanıldığını vurgularken, kompütasyonel düşünme perspektifleri

ise bireylerin ve toplumların teknoloji ve bilgi işlemi nasıl algıladığını ve bunlarla nasıl etkileşimde bulunduğunu inceler.

Tablo 2

Kompütasyonel düşünmenin üç boyutu (Brennan ve Resnick, 2012)

Kavram	Tanım	İçerik
<i>Kompütasyonel Düşünme Kavramları</i>	Programcının kullandığı kavramlar	Olaylar, algoritmalar ve diziler, koşullar, döngüler, değişkenler, operatörler, paralellik
<i>Kompütasyonel Düşünme Uygulamaları</i>	Programlama sürecinde ortaya çıkan problem çözme uygulamaları	Yineleme, soyutlama, modülerleştirme, yeniden kullanma, karıştırma, hata ayıklama
<i>Kompütasyonel Düşünme Bakış Açıları</i>	Öğrencilerin kendilerini, başkalarıyla ilişkilerini ve çevrelerindeki teknolojik dünyayı anlamaları	Yeniden ifade etme, bağlantı kurma, sorgulama

2.6.1. Kompütasyonel Düşünme Kavramları

2.6.1.1. Olaylar

Olay temelli kodlama, programın belirli tetikleyicilere nasıl tepki vereceğini tanımlama yöntemidir ve programlamada matematikten daha önemli bir kavram olarak kabul edilir (Howard ve Howard, 2020). Olay temelli kodlama, programın işleyişi belirli olayların gerçekleşmesine bağlı olan bir programlama yaklaşımıdır (ICS, 2023). Öğrencilerin basit olayları kullanabildiğini, ancak karmaşık olay kullanımını nadir kullandıklarını gösteriyor (Howard ve Howard, 2020). Akış şemaları ve bağımsız programlama, olay işleme becerilerini geliştirmeye yardımcı olabilecek yöntemler arasındadır. Bu yaklaşımda bir programın çalışması belirli olayların gerçekleşmesine bağlıdır. Örneğin, kullanıcının fareyi hareket ettirmesi, pencerenin yeniden boyutlandırılmasını tetikleyebilir ve program, pencere boyutunu kullanıcının tercihi göre değiştirerek yeniden boyutlandırabilir. Bu yaklaşım genellikle etkinlik yönetimi için kullanılır ve program, olayları algılayabilir ve gereken eylemleri yapabilir.

Scratch olay temelli bir kodlama programıdır (Akyol, 2022). Olay blok kodlarında yer alan kod blokları ile kuklanın belirli davranışları sergilemesi ya da

bağımsız farklı olayların gerçekleşmesi tetiklenir. Örneğin, yeşil bayrağa tıklandığında müziğin başlanması isteniyorsa olay bloklarından bayrağa tıklandığında başlat kod bloğu kodlama ekranına sürüklenip bırakılır, ardından gerçekleşmesi istenen müziğin başlaması olayı ise ses bloklarından alınarak bu kod bloğunun altına sürüklenip bırakılır. Bu sayede müziğin başlaması olayı bayrağa tıklandığında tetiklenmiş olur. Olay blokları sadece bayrak ile sınırlı değildir. Klavye üzerinde ya da dokunmatik bir ekranda yer alan belirli tuşlara basılarak da başlatılabilir.

2.6.1.2. Algoritmalar ve Diziler

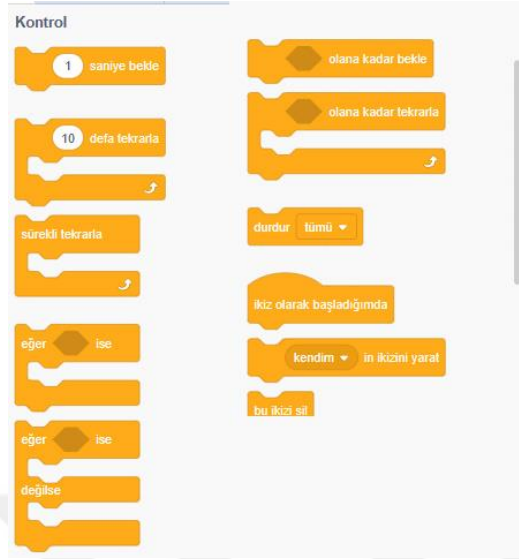
Algoritmalar, kuklaların hareketlerini, sesleri ve grafikleri kontrol etmek için kullanılan bir dizi talimattır. Algoritmalar, tekrar eden görevleri otomatikleştirmek ve karmaşık programlar oluşturmak için önemlidir. Diziler ise birden fazla değeri tek bir değişkende saklamak için kullanılır. Bilgisayar programları, algoritmaları işlemek için idealdir. Veriler de kodlanmış algoritmalar ile işlenir. Bu, matematik alıştırmalarındaki algoritmik hesaplamalara benzer. Scratch kullanıcıları, algoritmik hesaplamalarda platformda çalıştıktan sonra daha iyi performans gösterdikleri görmüştür (Howard ve Howard, 2020). Tamamlanmış bir programı izleyerek, Scratch kullanıcıları gerekli dizileri yeniden oluşturabilir. Bu, metin tabanlı dillerde sözdizimi deneyimi sağlamasa da, kavramsal düzeyde sıralamayı anlamayı gösterir (Howard ve Howard, 2020).

2.6.1.3. Koşullar

Koşul yapıları, programların belirli koşulları kontrol etmesine ve buna göre farklı işlemler gerçekleştirmesine olanak tanır. Koşullu ifadeler, programın önceden belirlenmiş bir komut dosyasından daha fazlasını yapmasını ve yorumlanabilir sonuçlara dayalı kararlar vermesini sağlar (Howard ve Howard, 2020). Koşullu ifadelerin kullanımı, programlamada geniş çapta araştırılmış ve Scratch gibi ortamlarda program akışını kontrol etmede etkili olduğu gösterilmiştir (Howard ve Howard, 2020). Scratch içerisinde yer alan koşul yapıları programlama yaparken karmaşık koşulların yönetimini kolaylaştırır ve kodlamanın daha esnek ve özelleştirilebilir olmasını sağlar. Şekil 5'te gösterilmiş olan Scratch programında bulunan karar, tekrar ve koşul yapıları gösterilmiştir.

Şekil 5

Karar, tekrar ve koşullar



2.6.1.4. Döngüler

Döngüler, programlamada tekrarlanan görevleri otomatikleştirmek için kullanılır. Kullanıcılar “tekrar” bloğunu ve “sor” bloğunu kullanarak döngü oluşturabilir. Bu, programcıya esneklik sağlar, ancak hatalara da yol açabilir. Sonsuz döngüler yaygın bir hata kaynağıdır ve farklı döngü türleri kafa karışıklığına neden olabilir ancak blok tabanlı programlama bunu metin tabanlı dillere kıyasla daha kolay hale getirir (Zhang ve Nouri, 2019). Matematikteki döngüler toplama (Σ) ile benzerlik gösterebilir (Howard ve Howard, 2020).

2.6.1.5. Değişkenler

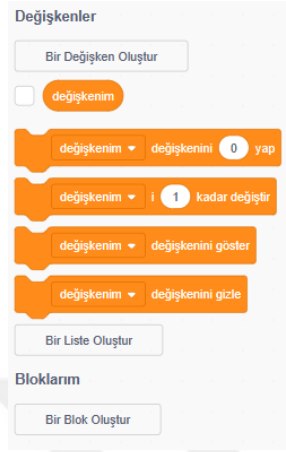
Programlamada, en temel kavram değişkenlerdir (Howard ve Howard, 2020). Zhang ve Nouri (2019), öğrencilerin değişkenleri entegre etme konusunda yeterince bilgi sahibi olmadıklarını ve bu konuda zorluk çektiklerini belirtmişlerdir. Matematik öğretiminde öğrencilerin temel programlama kavramlarını daha iyi anlamaları için değişkenlere daha fazla önem verilmesi gerekir (Howard ve Howard, 2020).

Howard ve Howard (2020), altı yaşındaki çocukların bile değişkenler açısından düşünme yeteneğine sahip olduğunu ve bu kavramın daha erken yaşta anlatılmasının öğrenme sürecini kolaylaştırabileceği belirtir. Scratch'te bulunan değişkenlerden olan listeler, verileri depolamak için kullanabileceğiniz bir yapısal öğelerdir ve benzer veri tiplerinin koleksiyonlarını içerir. Bu verilerin eklenmesi, çıkarılması ve değiştirilmesi

gibi işlemler yapılmasına izin verir. Şekil 6’da gösterilen değişken kod blokları, farklı amaçlar için kullanılan önemli yapısal öğelerdir.

Şekil 6

Değişkenler

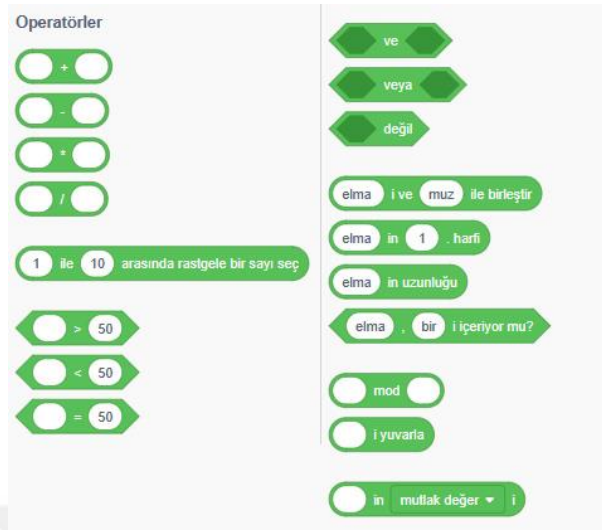


2.6.1.6. Operatörler

Matematiksel operatörler, programlamada da kullanılan temel kavramlardır (Howard ve Howard, 2020). Toplama, çıkarma, çarpma ve bölme gibi matematik işlemleri, programlama dillerinde de benzer şekilde kullanılır. Ayrıca operatörler, karar verme ve veri tabanı aramalarında da kullanılır. Zhang ve Nouri (2019), mantık operatörlerinin kullanımına yönelik bazı öğrencilerin bu operatörleri iyi anlarken, bazılarının ise önemli bilgi eksiklikleri olduğunu belirtmişlerdir. Bu durum, programlamada karar verme kodu yazmada zorluklara yol açabilir. Bu nedenle, matematiksel operatörlerin programlamada nasıl kullanıldığının öğrencilere açık ve net bir şekilde öğretilmesi önemlidir. Şekil 7’de Scratch programında yer alan operatör dizisi verilmiştir.

Şekil 7

Operatörler



2.6.1.7. Paralellik

Bilgisayar programlarında paralellik, birden fazla görevin aynı anda gerçekleşmesidir. Bu, daha verimli programlar ve daha zengin kullanıcı deneyimleri sağlar. Paralellik, birden fazla eylemi aynı kontrol bloğuna yerleştirmek veya blok dizilerini ayrı yığınlara sürüklemek suretiyle oluşturulabilir. Paralellik kavramını anlamak kolay olsa da, kodlamada uygulamak daha zordur (Howard ve Howard, 2020). Akış şemaları ve görsel örnekler, öğrencilerin bu kavramı öğrenmesine yardımcı olabilir.

2.6.2. Kompütasyonel Düşünme Uygulamaları

2.6.2.1. Yineleme

Program yazmak, doğrusal olmayan bir süreçtir. Kodlama, yinelemeli bir süreç içerir ve gerektiğinde planlar değiştirilebilmelidir. Bu uygulama, programlama öğreniminin önemli bir parçasıdır ancak öğrenciler tarafından genellikle göz ardı edilir (Howard ve Howard, 2020). Zhang ve Nouri (2019), öğrencilerin Scratch tabanlı projeler üzerinde çalışırken yinelemeye çoğunlukla başvurmasalar da aşamalı olarak kodlama kararlılığı kazandıklarını belirtmişlerdir.

2.6.2.2. Soyutlama

Soyutlama ve modülerleştirme, programlamada önemli iki kavramdır. Soyutlama, karmaşık problemleri daha basit parçalara ayırma yeteneğidir (Webb ve Rosson, 2013). Modülerleştirme ise bu parçaları bir araya getirerek daha karmaşık programlar oluşturma yeteneğidir (Zhang ve Nouri, 2019). Somut ve soyut düşünmenin eşiğinde olduğu varsayılan öğrencilerle yapılan araştırmalarda, Scratch kullanırken soyutlamayı kavramada zorluklar yaşandığı görülmüştür (Howard ve Howard, 2020). Her iki beceri de programlamada daha karmaşık problemleri çözmek için gereklidir.

2.6.2.3. Karıştırma

Yeni başlayan bir kullanıcı ya da uzman birisi fark etmeksizin tüm programcılar, çevrimiçi paylaşılan kod topluluklarından faydalanabilir. Scratch platformunda paylaşılan projeler, sistem üzerinde depolanır. Bu depolar, herkese açık olarak paylaşılan kod modüllerini indirip, özel ihtiyaçlarınıza göre uyarlamaya imkân tanıyarak tekerleği yeniden icat etme ihtiyacını ortadan kaldırır (Howard ve Howard, 2020). Scratch ortamında, diğer kullanıcılar tarafından paylaşılan kodun yeniden kullanılması ve yeniden karıştırılması oldukça yaygındır (Zhang ve Nouri, 2019). Bu durum diğer platformlara geçiş yapmak isteyen ve çevrimiçi veri havuzlarını kullanan öğrencilerin işini kolaylaştıracaktır.

2.6.2.4. Hata Ayıklama ve Test Etme

Yinelemeli programlama sürecinin önemli bir parçası olan test etme, öğrencilerin kodlarının beklendiği gibi çalışıp çalışmadığını değerlendirmelerini sağlar. Planlama, kodlama ve modülerleştirme gibi aşamalardan sonra, öğrenciler projelerini test etmek ve geri bildirim almak isterler. Bu geri bildirim kodun doğruluğuna dayanır ve öğrencilerin hatalarını düzeltmeleri ve gelişmeleri için bir fırsat sunar. Scratch araştırmaları, test etmenin ve hata ayıklamanın en sık gözlemlenen uygulama olduğunu göstermiştir (Howard ve Howard, 2020). Hata ayıklama anlayışı zamanla gelişmesi sebebiyle programlamanın erken yaşlarda başlatılması bu becerilerin geliştirilmesi için daha fazla zaman sağlar ve ortak matematik standartları olan Common Core State Standards'ta (CCSS) öngörülen azim becerisine katkıda bulunur (Howard ve Howard, 2020).

2.6.3. Kompütasyonel Düşünme Bakış Açıları

2.6.3.1. Yeniden ifade etme

Brennan ve Resnick (2012), teknoloji kullanımının öğrenmek için önemli olduğunu ancak yalnızca fareyle tıklamanın, sayfalarda gezmenin ve sohbet etmenin, bir kompütasyonel düşünme becerisini geliştirmek için yeterli olmadığını belirtmişleridir. Kompütasyonel düşünme becerisi, tüketilmesi gereken bir şeyden öte; hesaplama, tasarım ve kendini ifade için kullanabilecekleri bir araçtır. Programlama yaparken bilgisayarın takip edebileceği bir dizi talimat oluşturmak ve programlamanın bir yaratım aracı olduğunun farkına varmaktır (HGSE, 2024).

2.6.3.2. Bağlantı kurma

Brennan ve Resnick (2012), bağlantı kurmayı birlikte yaratmanın değeri ve başkaları için yaratmanın değeri olarak tanımlar. Scratch kullanıcıları birlikte yaratmanın, ister çevrimiçi forumlarda sorularının cevaplarını bulmak (örneğin, projede belirli bir hatayı düzeltme yardımı almak), ister başkalarının kodlarını incelemek ve yeniden düzenlemek (örneğin, üzerine inşa edebilecekleri yeni kaydırmalı bir proje temeli bulmak), isterse ortak stüdyolar veya yapım şirketleri oluşturarak kasıtlı ortaklıklar ve işbirlikleri kurmak olsun, tek başına yapabileceklerinden daha fazlasını başarmalarına imkânı verir (Brennan ve Resnick, 2012). Başkaları için yaratmak ise, bir proje yaratmak, blok kodlar geliştirmek, başkalarını eğitmek, simülasyonlar oluşturmak, başkalarının faydalanması için formüller geliştirmek veya yanıtlanması için anketler oluşturmak gibi şekillerle gerçekleştirilebilen süreci ifade eder.

2.6.3.3. Sorgulama

Teknoloji hayatımızın her alanında ancak çoğu insan bunları anlamıyor ya da onları etkileme gücüne sahip olmadığını düşünüyor (Bandura, 2001). Sorgulamaya dayalı kompütasyonel bakış açısıyla, öğrencilerin çevrelerindeki teknolojiler ile bu teknolojileri anlamlandırma yetenekleri geliştirilmelidir. Öğrenciler teknoloji hakkında ve teknolojiyle birlikte soru sorma gücüne sahip olmalıdır. Yaşadığı dünya hakkında eleştirel bakış açısıyla kompütasyonel düşünme biçimlerini kullanarak soru sormalıdır (Brennan ve Resnick, 2012).

2.7. FeTEMM (STEM) ve Kodlama

Dört temel disiplin olan bilim, teknoloji, matematik ve mühendisliğin bir araya getirildiği öğretim modeli FeTEMM (STEM) olarak tanımlanır (Şen, 2023). Programlama, gerçek hayat problemlerini çözmemizi bilimsel simülasyonlar yaratmamızı ve FeTEMM (STEM) alanlarında yetkinliklerimizi geliştirmemizi sağlayan bir araçtır (Basawapatna vd., 2011). Kodlama ise FeTEMM (STEM) içerisinde yer alan uygulamalardan birisidir. Kodlama genellikle programlama kavramıyla sıklıkla birbirlerinin yerine kullanılabilmektedirler. Kodlama kelimesinin daha fazla tercih edilmesinin sebebi ise kelimenin içerisinde yatan bir koda dair gizem ve bu kodu kırma ile gelen başarı duygusunun varlığıdır (Duncan vd., 2014). Geleneksel olarak kodlama yalnızca programlama sürecinin son aşamasına, yani tasarlanmış bir programı programlama ifadelerine çevirmek ve bunları bir bilgisayara yazmaktır (Duncan vd., 2014). Uluslararası Eğitimde Teknoloji Topluluğu'nun (ISTE) yayınladığı raporda kompütasyonel düşünme 21. yüzyılda öğrencilerin sahip olması gereken beceriler arasında sayılmaktadır. Bu beceri bilgisayar birimlerinin araç olarak kullanıldığı bir problem çözme sürecidir (Şen vd., 2021). Bireylere problem çözme sürecinde teknoloji okur-yazarlığını da geliştirdiği, teknolojinin imkânlarından ve gücünden faydalandığı geniş bir alan sunar. Avrupa Komisyonu (2018), öğrenciler, eğitim personeli ve öğrenim sağlayıcılar için öğrenmeyi geliştirmek ve dijital yeterliliklerin gelişimini desteklemek amacıyla kodlama gibi dijital teknolojileri kullanmasını tavsiye etmekte, problem çözme, eleştirel düşünme, iş birliği yapma yeteneği, yaratıcılık, öz düzenleme gibi becerilerin yanında kompütasyonel düşünmenin de hızla değişen toplumumuzda her zamankinden daha önemli olduğunu vurgulamaktadır. FeTEMM (STEM) yenilikleri, dünya çapında ekonomilerin büyümesinde ve gelişmesinde önemli faktörlerden biridir. Code org, Code Club, CoderDojo ve Learn to Code, Scratch gibi pek çok FeTEMM (STEM) girişimi ise bilgisayar bilimi (Computer Science- CS) programının uygulanması kapsamında artmaktadır. Bu girişimler kapsamında, CS eğitimi genellikle temel kodlama, bilgisayar programlama ve kompütasyonel düşünme olarak kabul edilen temel kavramları içerir (Howard ve Howard, 2020). Kodlama terimi ise ilköğretim sınıflarında bilgisayar bilimi ve bilgisayar programlama ile ilgili etkinliklere atıfta bulunmak için daha sık kullanılmaktadır. Bilgi işlem, kodlama, komut dosyası oluşturma, programlama veya kompütasyonel düşünme olarak uygulanan yirmi birinci

yüzyıl öğretimindeki en önemli bileşenlerindendir (Rich vd., 2019). Pek çok hükümet, eğitimcilerin öğrencilerin okula başladıkları ilk yıllardan itibaren kodlamayı öğretmelerini teşvik ediyor (Rich vd., 2019). Avustralya, yeni dijital teknolojiler standardının (ACARA, 2023) bir parçası olarak kompütasyonel düşünme biçimini geliştirmek için programlamayı ilkokul ikinci sınıfta dâhil etmiştir. Bilgi ve İletişim Teknolojisini (ICT) öğrencilerin sahip olması gereken temel yeterlilikler arasında göstererek kullanılan dijital yazılımların ve teknolojilerin matematik dersindeki temel kavramların öğrenilmesinde öğrencilere kolaylık sağlayacağını belirtmiştir.

Kodlamayı okul ortamında öğrencilerle buluşturmanın bilgisayar bilimleri açısından dezavantajlı grupların bu alanla tanışması ve var olan potansiyellerin ortaya çıkarılması açısından önemlidir. Bilgisayar bilimleri eğitime ilişkin uluslararası araştırmalar, bilgi ve teknoloji sektörlerinde kadınların endişe verici bir şekilde yetersiz temsil edildiğini belirtmekte ve toplumsal cinsiyet etkisine ilişkin rahatsız edici sonuçlar ortaya koymaktadır (Tatnall, 2020). Bilgisayar kullanımı ve bilgisayar ile ilgili faaliyetler yakın geçmişte erkek alanı olarak görülmekte ve kadınlara yönelik muameledeki küçük iyileşmelere rağmen bugün hala devam etmektedir (Pechtelidis vd., 2015). Bu durum, çeşitli faktörlere bağlı olabilir. Toplumsal cinsiyet rolleriyle karşı karşıya kalmaları, bu alanlarda rol model eksikliği veya eğitim sisteminin kendisi kızların STEM alanlarında ilerlemesini engelleyebilir (Tatnall, 2020). Okul ortamında yapılacak olan bu tarz kodlama etkinlikleri kız çocukları gibi dezavantajlı grupların STEM alanında gelişim göstermelerine yardımcı olabilir.

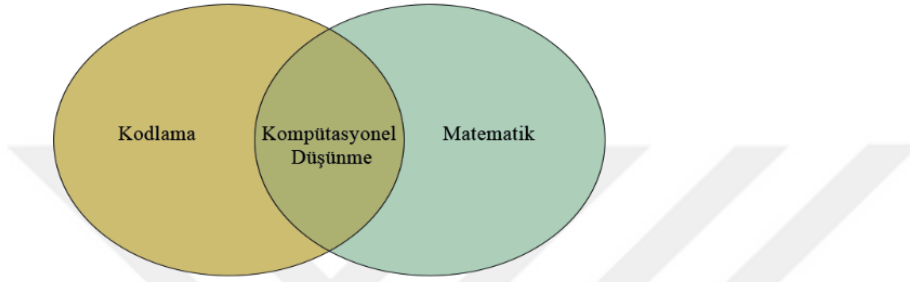
Kodlamayı matematiği kolaylaştırmak için kullanma fikri, Papert (1980) tarafından ortaya atılmıştır ve günümüzde de geçerliliğini korumaktadır (Gleasant ve Kim, 2020). Based Coding and Computational Thinking for Conceptual Mathematics (B2C3Math), matematik ve kodlama kavramlarının örtüşmesine odaklanan bir yaklaşımdır ve hem matematik hem de kodlama öğrenimini destekler (Gleasant ve Kim, 2020). B2C3Math'te öğretmen adaylarına kodlama platformlarının sürekli değişebileceği ve değişeceği, sürükle ve bırak kodlama öğretim programlarının sınırlamaları ve bir kodlama-matematik etkinliğinin nasıl geliştirileceği ve matematik öğreniminin kodlama süreciyle nasıl eş zamanlı olarak gerçekleştirilebileceği öğretilir.

B2C3Math, öğretmen adaylarının kodlama ve matematiği birleştirerek öğrencilerin her iki alanda da becerilerini geliştirmelerine yardımcı olacak şekilde tasarlanmış bir yaklaşımdır (Gleasant ve Kim, 2020). Matematik öğretiminde dijital

deneyimler kullanmak, sürükle ve bırak özelliğiyle öğrencilere kompütasyonel düşünme ve kodlama kavramlarını keyifli bir şekilde öğretme potansiyeline sahiptir. Bu tür programlar, bilgisayar bilimleri eğitiminin yanı sıra temel ilköğretim matematiği öğretiminde de kullanılabilir ve problem çözme ve anlama becerilerini geliştirmeye yardımcı olabilir (Gleasant ve Kim, 2020). Matematik ve kodlama arasındaki ilişki Şekil 8’de verilmiştir.

Şekil 8

Kodlama ve matematik ilişkisi



Not. *Coding + Math: Strengthen K–5 Math Skills With Computer Science* (s.103), Nicol R. Howard ve Keith E. Howard, 2020, International Society for Technology in Education’den alınmıştır.

Blok tabanlı programlama dilleri, bilişimsel düşünme kavramlarını içerir ve farklı disiplinlere kolayca aktarılabilir (Howard ve Howard, 2020). Programlama ve kompütasyonel düşünme öğrencilerin matematik konusunda daha derin bir anlayışa sahip olmalarına yardımcı olmak için kullanılabilir (Gleasant ve Kim, 2020). Bir formülü ezberlemek yerine neden bu şekilde çalıştığını öğrenebilirler. Gleasant ve Kim (2020), öğrencilerin bir diziye oluşturan olayları anlamasının matematiksel kalıpları anlamalarına yardımcı olacağını belirtmektedir. Gleasant ve Kim (2020), kodlamanın matematik dersine entegre edilirken programlama ve matematiği ayrı ayrı öğretmek yerine öğrencilerin matematiği programlama yoluyla öğrenmelerini tavsiye etmişlerdir.

Bilim, teknoloji, mühendislik ve matematik FETEMM (STEM) alanındaki yenilikler, dünya çapında ekonomik kalkınma için önemli bir rol oynamaktadır (Howard ve Howard, 2020). Kodlama, bilgisayar bilimlerine ve programlamaya temel giriş olarak kullanılır. Bu nedenle kodlamanın temel tanımlarını ve uygulamalarını anlamak, bilgisayar bilimlerinin matematiğe uygun şekilde entegrasyonu için önemlidir. Araştırmalar, kodlamanın öğrencileri motive ettiğini göstermektedir (Erol ve Çırak, 2022; Genç ve Karakuş, 2013; Howard ve Howard, 2020). Programlama ve

bilişimsel düşünmenin ilköğretim çağındaki öğrencilerin başarısı üzerindeki etkisi ise daha az bilinmektedir (Howard ve Howard, 2020). Kodlama programlarının daha fazla öğrenciye ulaşması için uluslararası araştırmalar yapılmaktadır. FETEMM (STEM) alanlarında daha nitelikli ve sayıda öğrenci yetiştirilmesine yardımcı olabilir. Artan dijitalleşme ile bilgisayar bilimleri, programlama ve yazılım mühendisliği alanlarına olan ihtiyaç da artmaktadır. Bu nedenle eğitimciler bilgisayar bilimlerini derslerine entegre etmeye hazır olmaları önemlidir. Bu aşamada, blok tabanlı programlama ideal bir başlangıç noktasıdır (Howard ve Howard, 2020).



BÖLÜM III

YÖNTEM

3.1. Araştırmanın Modeli

Nitel ve nicel yöntemler birlikte kullanıldığı karma desenlerden biri olan açıklayıcı desen (explanatory design) kullanılmıştır. Bu desene göre araştırmacılar öncelikle nicel verileri toplar ve analiz ederler ve ardından bu verileri tamamlamak ve rafine edebilmek için nitel verileri toplarlar (Büyüköztürk, 2023). Bu doğrultuda araştırmanın nicel verileri deneysel desenlerden zayıf deneysel desen olan tek grup öntest-sontest desen kullanılarak toplanmıştır. Nicel verilerini tamamlamak üzere nitel veriler görüşme tekniği kullanılarak toplanmıştır. Zayıf deneysel desenlerden olan tek gruplu öntest-sontest modeli kullanıldı. Zayıf deneysel desenlerde seçkisiz atama olmadığı gibi grup eşleştirmesi yapılmamıştır. Tek grup öntest-sontest modelinde seçkisiz atama rastgele seçilmiş bir gruba bağımsız değişkenin uygulanmasından önce öntest ve sontest yapılmasıdır (Yıldırım ve Şimşek, 2021). Bu bağlamda rastgele veya amaçlı örnekleme yöntemleriyle seçilen bir gruba tek grup öntest sontest modeli uygulanır ve bağımsız değişken hem deney öncesi hem de deney sonrası ölçümlerle test edilir. Tek gruba ait öntest ve sontest değerleri arasındaki farkın anlamlılığı ölçülür (Büyüköztürk vd., 2008). Standart bir t-testinde verilerin normal dağılım göstermesi beklenir ancak çarpıklık ve basıklık aşırı düzeyde olmadığı müddetçe normal olmayan dağılımın sorun olmadığı görülmektedir (Ross ve Willson, 2018). Tablo 3'te verildiği gibi araştırmada 8. sınıfta eğitim alan 8 öğrenciye öncelikle öntest uygulandı. Sontest kodlama eğitiminin tamamlanmasından sonra tekrar uygulandı. Tek gruplu öntest sontest modelinin uygulanmasından sonra gönüllü 6 öğrenci ile görüşme yapıldı.

Tablo 3*Tek gruplu ön test- son test modeli*

Grup	Ön Test	Uygulama	Son Test
Tek Grup	Öntest Envanteri	Kodlama Eğitimi	Sontest Envanteri

3.2. Araştırma Grubu

Araştırma grubunu Yozgat iline bağlı bir imam hatip ortaokulunda öğrenim görmekte olan sekizinci sınıf düzeyinde sekiz kız öğrenci oluşturmuştur. Kız imam-hatip olduğundan sadece kız öğrenciler okumaktadır. Okulda mevcut durumdaki tüm öğrencilere ulaşılmıştır. Seçkisiz olmayan örnekleme yöntemlerinden tabakalı amaçsal yöntemi kullanıldı. Bu yöntem, belirli alt grupları tanımlamaya, betimlemeye ve karşılaştırmaya olanak tanır ve rasgele seçim yerine, kolay ulaşılabilen ve uç durumdaki deneklerle çalışma imkânı sağlar (Büyüköztürk, 2023). Mahallede eğitim gören 8. sınıf öğrencilerinin %20'sine ulaşılmıştır. Araştırma grubu, soyut düşünme ve problem çözme becerileri gelişme aşamasında olan 8. sınıf öğrencilerinden seçilmiştir. Bu seçim, Scratch ile yapılan kodlama ve problem çözmeye yönelik bir etkinliğin becerilerin gelişmesine katkıda bulunabileceği varsayımına dayanmaktadır.

3.3. Veri Toplama Araçları

Sekizinci sınıfların problem çözme becerilerini belirlemek amacıyla öntest ve sontest olarak uygulanan problem çözme envanteri Ek 2'de verilmiştir. Araştırma sonucunda ayrıca öğrencilerle görüşme yapılmıştır. Görüşmeler 6 gönüllü öğrenci ile gerçekleştirildi. Görüşme soruları hazırlanırken uzman görüşleri alındı.

3.3.1. Problem Çözme Envanteri

Öntest ve sontestte kullanılan sorular Millî Eğitim Bakanlığı'nın Ölçme, Değerlendirme ve Sınav Hizmetleri Genel Müdürlüğü'nce (ÖDSGM, 2023) hazırlanan ve 2022-2023 yılında yayınlanan 8. Sınıf Matematik Çalışma Kitabı soruları temel alınarak araştırmacı tarafından uyarlanmıştır. Araştırmacının soruları Ölçme, Değerlendirme ve Sınav Hizmetleri Genel Müdürlüğü'nden (ÖDSGM, 2023) almasının sebepleri aşağıda sıralanmıştır:

- Geçerlilik ve Güvenilirlik: Ölçme, Değerlendirme ve Sınav Hizmetleri Genel Müdürlüğü'nce (ÖDSGM) tarafından hazırlanan testler, geniş bir öğrenci örneklemini üzerinde uygulanmış ve kapsamlı analiz edilmiştir. Testlerin geçerliliği ve güvenilirliği yüksektir. Araştırmacı, ÖDSGM'den aldığı soruları kullanarak, bulgularının daha sağlam ve genellenebilir olmasını sağlamayı amaçlamıştır.
- Standardizasyon: ÖDSGM tarafından hazırlanan testler, tüm Türkiye'deki öğrenciler için aynı standartlarda hazırlanır ve uygulanır. Farklı okullardan ve coğrafi bölgelerden gelen öğrencilerin performanslarını karşılaştırmak mümkün olur. Araştırmacı, farklı gruplar arasındaki farkları daha net bir şekilde ortaya koymayı amaçlamıştır.
- Zamandan Kazanç: ÖDSGM tarafından hazırlanan testler sayesinde araştırmacıların sıfırdan test hazırlama ve uygulama sürecine harcadıkları zamanı azaltmayı amaçlamıştır.
- Eşitlik: ÖDSGM tarafından hazırlanan testler, tüm öğrenciler için eşit şartlarda uygulanır. Herhangi bir öğrencinin dezavantajlı duruma düşmesinin önüne geçilmesi amaçlanmıştır.

3.3.2. Yarı Yapılandırılmış Görüşme Formu

Katılımcıların blok tabanlı kodlama etkinliğine yönelik görüşlerini belirlemek için yarı yapılandırılmış görüşme formu kullanılmıştır. Görüşme, önceden planlanmış, önemli bir hedefe yönelik olarak gerçekleştirilen, soru sorma ve yanıtlama odaklı, karşılıklı etkileşim içeren bir iletişim sürecidir (Karataş, 2015). Yarı yapılandırılmış görüşme tekniğinin kullanılmasının nedeni, önceden belirlenmiş bir formata dayanarak yapılması ve araştırmacıya daha düzenli ve karşılaştırılabilir bilgiler sunmasına (Yıldırım ve Şimşek, 2021) imkân tanınması sebebiyle esneklik sağlamasıdır.

Görüşme formu, öğrencilerin kodlama etkinliğine yönelik görüşlerini belirlemek amacıyla iki alan uzmanının görüşü alınarak araştırmacı tarafından hazırlanmıştır. Görüşme soruları iki defa hazırlanmış, iki uzman görüşüne sunulmuş sorulara son hâli verilmiştir. Söz konusu görüşmeler, araştırmacı tarafından yapılmış ve aşağıdaki sorular sorulmuştur:

1. Aldığınız kodlama eğitimi hakkında ne düşünüyorsunuz? Daha önce kodlama deneyiminiz oldu mu?
2. Aldığınız kodlama eğitiminin önemi konusunda ne düşünüyorsunuz? Önemliyse hangi açıdan önemli olduğunu açıklar mısınız?
3. Kodlama yaparken karşılaştığınız güçlükler nelerdir?
Alt Soru: Okulunuzda kodlama eğitimi alabileceğiniz bir kodlama atölyeniz var mı?
4. Kodlama becerisine sahip olmak sizin eğitim hayatınızı ya da gelecekte meslek seçiminizi nasıl etkileyebilir?
Alt Soru: Günümüzde kodlamayı her öğrencinin sahip olması gereken önemli beceri olarak görüyor musunuz? Neden?
5. Matematik dersi ile kodlama arasında bir ilişki olduğunu düşünüyor musunuz? Eğer cevap evet ise “Bu ilişkiyi açıklar mısınız?
6. Aldığınız kodlama eğitimi matematik dersine yönelik ilginizi etkiledi mi? Nedenini ayrıntılı açıklayınız.
7. Aldığınız kodlama eğitiminin farklı derslerde kullanılabileceğini düşünüyor musunuz? Nedenini örnekler vererek açıklayınız.

3.4. Araştırmacının Rolü

Araştırmada araştırmacının rolü aşağıda verilen başlıklarla özetlenebilir:

1. Öntest Uygulamak: MEB kaynaklı 20 soruluk bir matematik öntesti kullanarak öğrencilerin başlangıçtaki matematik becerilerini ölçmüştür.
2. Kodlama Etkinlikleri Yürütmek: 8 hafta boyunca Scratch platformunu kullanarak kodlama etkinlikleri düzenlemiştir. Etkinlikler hazır kod blokları şeklinde verilmiş ve öğrencilerin ufak değişikliklerle kendi özgün ürünlerini ortaya koymaları sağlanmıştır. Etkinlikler, öntestte sorulan beceri temelli sorulara ait konular ve o konuların matematiksel formalizasyonu ile ilişkilendirilmiştir.

3. Sontest Uygulamak: Kodlama etkinlikleri tamamlandıktan sonra, öğrencilere aynı öntesti tekrar uygulayarak kodlama eğitiminin matematik becerileri üzerindeki etkisini ölçmeye çalışmıştır.
4. Verileri Analiz Etmek: Öntest ve sontest sonuçlarını karşılaştırarak kodlama eğitiminin matematik becerilerini geliştirmede anlamlı bir fark yaratıp yaratmadığını istatistiksel yöntemlerle (t testi) analiz etmiştir.
5. Görüşme Yapmak: Kodlama etkinliklerine katılan öğrencilerle görüşmeler yaparak süreç hakkındaki fikirlerini ve kodlama eğitiminin onlara kattıklarını değerlendirmiştir.

3.5. Verilerin Toplanması

Sekizinci sınıfların problem çözme becerilerini belirlemek amacıyla öntest uygulanmış, ardından Scratch üzerinden kodlama eğitimi verilmiştir. Bu eğitimde Tablo 4'te verilen etkinlikler uygulanmıştır. Etkinlikler öğrencilerin yaş gruplarına göre belirlenmiş ve ders içeriği ve kazanımlara uygun olarak tasarlanmıştır. Aynı zamanda, günlük hayatta uygulanabilecek problem durumları dikkate alınarak hazırlanmıştır. Yapılacak uygulamanın işlenişinde öğrencilere Scratch tanıtılmış ve bulunan kod bloklarının işlevleri anlatılmıştır.

1. Öğrencilere kod bloklarının işlevleri ve örnek kodlama etkinlikleri sunulmuştur.
2. Amaca uygun etkinlikleri oluşturmaları istenmiştir.
3. Kodlama etkinliklerinden sonra öğrencilerle fikirleri alınmak üzere görüşme yapılmıştır.

Katılımcıların gerçek isimleri gizlenerek K1, K2 vb. kod isimleri kullanılmıştır. Araştırma grubuna dâhil olan öğrencilerle yapılan görüşmelerde, katılımcıların görüşlerini serbestçe ifade etmişlerdir. Samimi bir sohbet ortamında geçtiği ve katılımcı öğrencilerin rahat oldukları gözlemlenmiştir.

Scratch programlamaya giriş kavramlarını göstermek için kukla denilen yapılarla örnek programlardan oluşan bir sahne ve bu sahnede kurgulanacak kodlama temelli senaryolar geliştirilmiştir. Bu senaryolar, öğrencilerin kendi projeleri için

potansiyel başlangıç noktaları olarak hizmet etmesi amacıyla tasarlanmıştır. İlk derslerde yeni bir projenin nasıl oluşturulacağı, karakterlerin nasıl ekleneceği ve değiştirileceği, karakterlerin ekranda nasıl hareket ettirileceği ve karakterlerin nasıl konuşulacağı ele alınmıştır. Öğrencilere mevcut kazanımlara uygun çeşitli etkinlikler sunulmuştur. Etkinlikler kazanım sırasına göre değil kodlama için kullanılacak kod bloklarının karmaşıklığına göre sıralanmıştır. Kodlama vurgusu ve tanıtılan matematiksel kavramlar da dâhil olmak üzere derslerin genel ilerleyişi ve etkinliklere ait kazanım numaraları (MEB, 2023) Tablo 4’te özetlenmiştir. Etkinliklerden önce Scratch programı öğrencilere tanıtılmıştır. Öğretmen hesabı üzerinden öğrenci hesapları oluşturularak programın arayüzü öğrencilere gösterilmiştir. Arayüzde bulunan sekmeler, görev çubukları ve alanların ne olduğundan bahsedilmiştir (Şekil 3). Blok kütüphanesi, karar ve tekrar yapıları, liste ve fonksiyon kullanımı gibi yapılar anlatılmıştır.

Tablo 4

Planlanan Scratch etkinlikleri

Etkinlik No	Etkinlik Adı	Amaç	Konulara Ait Kazanım No
1	Soru-Cevap	Kuklalar arası diyaloglarda kısa soru cevap etkinlikleri kullanılır. Öğrenci çarpanlar ve katlar konusuyla ilgili belirlediği soruları yarattığı kuklalar aracılığı ile sahneye yerleştirir (Şekil 10).	M.8.1.1.1. M.8.1.1.2. M.8.1.1.3.
2	Asal Mı?	Scratch kod blokları ile sayının asal sayı olup olmadığını tespit eden bir program kodlar (Şekil 11).	M.8.1.1.1. alt kazanım
3	Pozitif Tam Sayı Bölenleri	Scratch kod blokları ile pozitif bir tam sayının pozitif bölenlerini bulacağı bir program kodlar (Şekil 12).	M.8.1.1.1.
4	Aralarında Asal Mı?	Scratch kod blokları ile iki sayının aralarında asal olup olmadığını tespit eden bir program kodlar (Şekil 13).	M.8.1.1.3.
5	EBOB ve EKOK Hesaplama Programı	Scratch kod bloklarıyla programa girilen iki sayının EBOB ve EKOK’unu hesaplayan bir program yazar (Şekil 14).	M.8.1.1.2.

3.5.1. “Etkinlik 1: Soru-Cevap” Uygulama Süreci

Tablo 5’te Taylor’un (2019) programla ile problem çözme becerilerinin yapılacak etkinlik ile eşleştirilmiş hali verilmiştir. Yapılan etkinlikte öğrencilere değişken atama ve koşul kullanımı hazır kod blokları olarak öğrencilere verilmiştir. Kalan kısımlar öğrenciler tarafından tasarlanmıştır.

Tablo 5

Taylor’ın (2019) problem çözme süreci ve “Soru-Cevap” kodlama etkinliği

Problem çözme süreci (Polya 1945; Taylor, 2019)	“Etkinlik 1: Soru-Cevap” Kodlama Etkinliği
Problemi Tanımlama	Öğrenci soracağı soruları belirler. Sorular kazanım tablosunda yer alan kazanımlarla ilişkilidir. Öğrenci soracağı soruların cevabını bilir. Öğrenci seçeceği dekoru, sesi, kuklaları belirler.
Bir Plan Tasarlamak ve Uygulamak (Problem çözme seviyesine bağlı olacak şekilde kod kalıplarını uygular, kod dizilerini kullanır)	Kullanılacak kod blokları belirlenir. Değişken ataması yapar. Öğrenci “Soru sor”, “Yanıt” kod bloklarını kullanır. Alacağı cevaplara doğru veya yanlış olacak şekilde dönüt sağlaması için “Koşul” bloğunu kullanır.
Değerlendirme	Öğrenci “başlat” butonuna basar. Kodların çalışıp çalışmadığını kontrol eder.

Şekil 9’da yer alan karşılıklı soru cevap örnek kodlama etkinliğinde öğrencinin etkinliği birden fazla seçenekle kurgulama imkânı vardır. Soruları hazırlayacak olan öğrencilere konu başlıkları verilmiştir. Sorular hakkında adedi biçimi vs. gibi kısıtlamalara gidilmemiştir. Öğrenciler burada kodlamayı yaparken farklı soru cevap stillerinden faydalanmışlardır.

Şekil 9

Diyalog tabanlı soru cevap etkinliği içeren örnek kodlama etkinliği (Scratch sayfası)



3.5.2. “Etkinlik 2: Asal Mı?” Uygulama Süreci

Tablo 6’da Taylor’un (2019) programla ile problem çözme becerilerinin yapılacak etkinlik ile eşleştirilmiş hali verilmiştir. Yapılan etkinlikte öğrencilere değişken atama ve mod kullanımı hazır kod blokları olarak öğrencilere verilmiştir. Kalan kısımlar öğrenciler tarafından tasarlanmıştır.

Tablo 6

Taylor’ın (2019) problem çözme süreci ve “Asal Mı?” kodlama etkinliği

Problem çözme süreci (Polya 1945; Taylor, 2019)	“Etkinlik 2: Asal Mı?” Kodlama Etkinliği
Problemi Tanımlama	Öğrenci bir sayının asal olup olmadığını bilir. Öğrenci seçeceği dekoru, sesi, kuklaları belirler.
Bir Plan Tasarlamak ve Uygulamak (Problem çözme seviyesine bağlı olacak şekilde kod kalıplarını uygular, kod dizilerini kullanır)	Kullanılacak kod blokları belirlenir. “Soru sor” kod bloğu kullanır. Değişken ataması yapar. “Mod” bloğu ile sayının varsa bölenlerini kontrol ettirir. 1’i ve kendisini almaması için koşul bloklarını kullanır. Bölüne çıkmazsa programa “yanıt” kod bloğunu kullanarak sayıyı “asal” olarak okutur.
Değerlendirme	Öğrenci “başlat” butonuna basar. Kodların çalışıp çalışmadığını kontrol eder.

Şekil 10’da verilen örnek kodlama etkinliğinde öğrenciler tarafından soru cevap şeklinde ilerleyen bir sahne kurgulandı. Etkinlikte amaç, üçüncü bir kullanıcı

tarafından sisteme girilecek herhangi bir sayının asal olup olmadığını programın tespit etmesidir. Problemi tanımlama aşamasında öğrenciler kâğıt üzerinde bir sayının asal olup olmadığını önce kendileri belirleyebilir. Kodlama ise plan tasarlama ve uygulama aşamasında yapılır.

Şekil 10

Asal Mıdır? adlı örnek kodlama etkinliği (Scratch sayfası)



3.5.3. “Etkinlik 3: Pozitif Tam Sayı Bölenleri” Uygulama Süreci

Bir sayının bölenlerini bulmak için öğrencilerden aşağıdaki adımları izlenmesi istenir Tablo 7’de Taylor’un (2019) programla ilgili problem çözme becerilerinin yapılacak etkinlik ile eşleştirilmiş hali verilmiştir. Yapılan etkinlikte öğrencilere değişken atama ve liste kullanımı hazır kod blokları olarak öğrencilere verilmiştir. Kalan kısımlar öğrenciler tarafından tasarlanmıştır.

Tablo 7

Taylor’ın (2019) problem çözme süreci ve “Pozitif Tam Sayı Bölenleri” kodlama etkinliği

Problem çözme süreci (Polya 1945; Taylor, 2019)	“Etkinlik 3: Pozitif Tam Sayı Bölenleri” Kodlama Etkinliği
Problemi Tanımlama	Öğrenci bir sayının pozitif tam sayı bölenlerini hesaplamayı bilir. Öğrenci seçeceği dekoru, sesi, kuklaları belirler.
Bir Plan Tasarlamak ve Uygulamak (Problem çözme seviyesine bağlı olacak şekilde kod kalıplarını uygular, kod dizilerini kullanır)	Kullanılacak kod blokları belirlenir. “Soru sor” kod bloğu kullanır. Değişken ataması yapar. Programı mod bloğu ile sayının yarısına kadar olan tüm sayıları bölerek kontrol ettirir Kalanın sıfır olduğu sayıları “değişken” blokları ile ayırır; “liste” blokları ile listeler. “Liste” bloğunu programa “yanıt” kod bloğunu kullanarak sayının “pozitif bölenleri” olarak okutur.
Değerlendirme	Öğrenci “başlat” butonuna basar. Kodların çalışıp çalışmadığını kontrol eder.

Şekil 11’de bir sayının pozitif bölenlerinin bulunacağı örnek bir kodlama etkinliğinde program kullanıcı tarafında girilen bir sayısının pozitif bölenlerini kullanıcıya sıralayacaktır. Programı kodlayan öğrenci problemi tanımlama aşamasında bir sayının pozitif bölenlerinin kâğıt üzerinde nasıl bulunacağını bilir. Kodlama, planı tasarlama ve uygulama aşamasındadır.

Şekil 11

Pozitif tam bölen sayısı örnek kodlama etkinliği (Scratch sayfası)



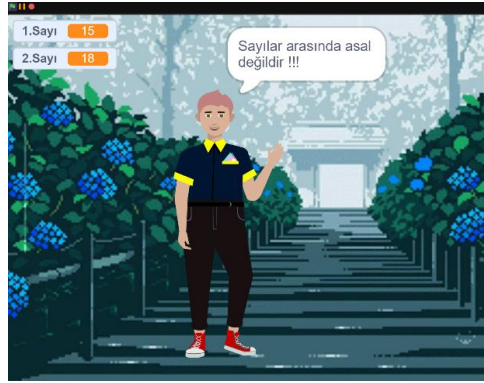
3.5.4. “Etkinlik 4: Aralarında Asal Mı?” Etkinliği Uygulama Süreci

Tablo 8’de Taylor’un (2019) programla ile problem çözme becerilerinin yapılacak etkinlik ile eşleştirilmiş hali verilmiştir. Yapılan etkinlikte öğrencilere değişken atama hazır kod blokları olarak öğrencilere verilmiştir. Kalan kısımlar öğrenciler tarafından tasarlanmıştır.

Şekil 12’de verilen iki sayının aralarında asal olup olmadığını bulan örnek bir kodlama etkinliğinde kodlanan program bu iki sayının aralarında asal olup olmadığını tespit eder. Problemi tanımlama aşamasında öğrenciler iki sayının aralarında asal olup olmadığını kâğıt üzerinde hesaplayabilir. Kodlama, plan oluşturma ve uygulama aşamasındadır.

Şekil 12

Aralarında Asal Mı? adlı örnek kodlama etkinliği (Scratch sayfası)



Tablo 8

Taylor'ın (2019) problem çözme süreci ve “Aralarında Asal Mı?” kodlama etkinliği

Problem çözme süreci (Polya 1945; Taylor, 2019)	“Etkinlik 4: Aralarında Asal Mı?” Kodlama Etkinliği
Problemi Tanımlama	Öğrenci iki sayının aralarında asal olup olmadığını bilir. Öğrenci seçeceği dekoru, sesi, kuklaları belirler.
Bir Plan Tasarlamak ve Uygulamak (Problem çözme seviyesine bağlı olacak şekilde kod kalıplarını uygular, kod dizilerini kullanır)	Kullanılacak kod blokları belirlenir. “Soru sor” ve “Yanıt” kod bloğu kullanır. Değişken ataması yapar. Programı “Yanıt” kod bloğu ile aldığı iki sayıdan küçük olanını belirlemek için “operatör” kod bloklarını kullanır. “Mod” bloğu ile sayıları küçük olan sayıya kadar olan tüm sayıları böldürerek ortak bir böleni olup olmadığını kontrol ettirir. Ortak bölen tespit edilememesi durumunda programa “yanıt” kod bloğunu kullanarak sayıları “aralarında asaldır” olarak okutur.
Değerlendirme	Öğrenci “Başlat” butonuna basar. Kodların çalışıp çalışmadığını kontrol eder.

3.5.5. “Etkinlik 5: EBOB ve EKOK Hesaplama Programı” Etkinliği Uygulama Süreci

Tablo 9’da Taylor’un (2019) programla ilgili problem çözme becerilerinin yapılacak etkinlik ile eşleştirilmiş hali verilmiştir. Yapılan etkinlikte öğrencilere değişken atama hazır kod blokları olarak öğrencilere verilmiştir. Kalan kısımlar öğrenciler tarafından tasarlanmıştır.

Tablo 9

Taylor'ın (2019) problem çözme süreci ve “EBOB ve EKOK Hesaplama Programı” kodlama etkinliği

Problem çözme süreci (Polya 1945; Taylor, 2019)	“Etkinlik 5: EBOB ve EKOK Hesaplama Programı” Kodlama Etkinliği
Problemi Tanımlama	Öğrenci iki sayının EBOB’unu ve EKOK’unu hesaplamayı bilir. Öğrenci seçeceği dekoru, sesi, kuklaları belirler.
Bir Plan Tasarlamak ve Uygulamak (Problem çözme seviyesine bağlı olacak şekilde kod kalıplarını uygular, kod dizilerini kullanır)	Kullanılacak kod blokları belirlenir. “Soru sor” kod bloğu kullanır. Değişken ataması yapar. Programa “mod” bloğu ile k değişkeninin sisteme girilen her iki sayıdan da kalanının sıfır olacak şekilde sayılara bölerek kontrol ettirir. Kalanın sıfır olduğu sayıları “değişken” blokları ile ayırır; şartı sağlayan sayılar “koşul” blokları ile ayırır. İstisnai durumlar için yine “koşul” bloklarını kullanır. Programa “yanıt” kod bloğunu kullanarak sayının EBOB ve EKOK değerlerini okutur.
Değerlendirme	Öğrenci “başlat” butonuna basar. Kodların çalışıp çalışmadığını kontrol eder.

Şekil 13’te verilen örnek kodlama etkinliğinde EBOB ve EKOK hesaplaması yapan bir program kodlanmıştır. Problemi tanımlama aşamasında öğrenci iki sayının EBOB ve EKOK değeri kâğıt üzerinde hesaplayabilir. Kodlama, plan oluşturma ve uygulama aşamasındadır.

Şekil 13

EBOB ve EKOK hesaplama örnek kodlama etkinliği (Scratch sayfası)



Yeni kavramları tanıtmak için, her oturum ders kavramları ve buna uygun kod blokları hakkında tartışma ile başladı ve ardından öğrenci için programlama etkinliği başlatıldı. Tartışma süresi boyunca, öğretmen kod bloklarının temel özelliklerini açıkladı ve işlevleri; etkinlik içerisindeki kullanımları hakkında ipucu verdi. Öğrenciler, programlama sürecinde öğretmenle birlikte programlamaya ve kendi fikirlerini geliştirmeye, sorular sormaya ve diğer öğrencilerle sohbet etmeye teşvik edildi.

Öntest-sontest sürecinin tamamlanmasının ardından öğrencilerle görüşme yapıldı. Katılımcılar gönüllülük esasına göre belirlendi. Görüşmeler, yarı yapılandırılmış bir formatta gerçekleştirildi. Görüşmelerin başlangıcında, katılımcılara araştırmanın amacı, süreci ve gizlilik prensipleri hakkında bilgi verildi. Katılımcılardan araştırmaya ilişkin deneyimlerini, görüşlerini ve önerilerini yarı yapılandırılmış görüşme formunda yer alan sorular doğrultusunda paylaşımları istendi. Görüşmeler sırasında, katılımcıların ifadeleri not alındı ve gerekli olduğunda sorular soruldu.

3.6. Verilerin Analizi

Sekizinci sınıfların problem çözme becerilerini belirlemek amacıyla ilk olarak öğrencilere Ek 2’de verilen öntest uygulanmıştır. Kodlama etkinlikleri tamamlandıktan sonra öğrencilere sontest uygulanmıştır. Toplanan veriler istatistik programında analize hazır hale getirilmiştir. Katılımcıların uygulamadan önceki öntest sonuçları ile sontest sonuçları arasında anlamlı bir fark olup olmadığını belirlemek için

bağımlı örneklem t testi kullanılmıştır. Veriler istatistik programı ile analiz edilmiştir. İstatistik programı kullanılarak, öğrencilerin öntest ve sontest puanları temel alınarak frekans (f), yüzde (%), ortalama ($\bar{X}$) ve standart sapma (ss) değerleri hesaplanmıştır. Verilerin dağılımının normallik analizinde örneklem sayısının az olmasından dolayı Shapiro-Wilk (S-W) testleri kullanılmıştır. Araştırma bulgularının normal dağılım göstermesi sonucunda, öntest ve sontest ortalamaları arasındaki farkın anlamlı olup olmadığı bağımlı gruplar için t testi ile incelenmiştir.

Yarı yapılandırılmış görüşmelerden elde edilen verilerin analizinde ise betimsel içerik analiz tekniği kullanılmıştır. Betimsel analizin ana hedefi, toplanan verileri anlamlandırabilecek kavramlara ve ilişkilere ulaşmaktır (Şeker, 2020). Görüşmelerden elde edilen nitel verilerin yorumlanmasında betimsel analiz ve içerik analizi teknikleri kullanılmıştır. Betimsel analiz, çeşitli veri toplama yöntemleriyle elde edilen bilgilerin belirli temalara göre özetlenmesini ve yorumlanmasını içeren nitel veri analiz türüdür (Gültepe, 2018). Analizde, görüşme yapılan katılımcıların ifadeleri doğrudan alıntılarla aktarılmış, bulguları düzenlenmiş ve yorumlanmış bir şekilde sunulmuştur. İçerik analizinde ise toplanan verileri açıklayabilecek kavramlara ve bu kavramlara bağlı ilişkilere ulaşmak hedeflenir. Betimsel analizde özetlenip yorumlanan veriler, içerik analizi sürecinde daha detaylı bir şekilde işlenir; bu sayede betimsel bir yaklaşımla fark edilemeyen kavramlar ve temalar ortaya çıkabilir. Bu nedenle, toplanan verilerin önce kavramlaştırılması, ardından mantıklı bir düzenleme ve temalara göre sınıflandırılması ile içerik analizi yapılır (Yıldırım ve Şimşek, 2021).

3.7. Geçerlik ve Güvenirlik

ÖDSGM'den uyarlanmış aynı beceri temelli soruların öntest ve sontestte kullanılması, testin iç geçerliliğini sağlaması açısından önemlidir. Dış geçerlilik, kodlama etkinliklerinin öğrencilerin matematik performansı ile ilişkilendirilmiş; yapılan etkinliklerin matematik becerilerini geliştirdiği varsayılarak dış geçerliliğin sağlanmıştır. Araştırmada tutarlılığın, aynı testin iki farklı zamanda uygulanmasıyla sağlandığı varsayılmıştır. Testin geçerlik ve güvenirliliği için tehdit oluşturabilecek bazı unsurlar ve bu unsurlara karşı araştırmacının aldığı önlemler aşağıda verilmiştir:

1. Dış Tehditler:

- a. Testin Güvenliğine Yönelik Saldırıları: Test sırasında, dış kökenli kötü niyetli kişiler veya gruplar, testin güvenliğini tehlikeye atabilir.

Öğrencilerin, sınav sorularını ele geçirdiği veya cevapları önceden elde etmesi gibi bir durum yaşanmamıştır.

- b. Sınav Sürecine Müdahale: Dış tehditler, sınav sırasında teknik sorunlar oluşturarak öğrencilerin testi normal bir şekilde tamamlamalarını engelleyebileceği bir durum yaşanmamıştır.

2. İç Tehditler:

- a. Cevapları Hatırlama: Öğrencilerin, testin birinci ve ikinci uygulaması arasında cevapları hatırlama riski vardır. Özellikle kısa bir süre içinde tekrar edilen testlerde, öğrencilerin cevapları hatırlama olasılığı artabilir. Bu tehdit için araştırmacı sontestte bazı sorularda şıkların yerlerini değiştirmiş ve ek olarak öğrencilerin sorulara yaklaşımlarını inceleyerek yer yer öğrenci cevapları paylaşılmıştır.
- b. Tehdit Edici Davranışlar: Testin güvenilirliğini etkileyebilecek öğrenci davranışları da iç tehditler arasında yer alır. Öğrencilerin kopya çekme girişimleri veya hile yapma girişimleri olmamıştır.

BÖLÜM IV

BULGULAR

Bu bölümde araştırmanın bulguları öğrencilerin cevapları ve yapılan kodlama etkinlikleri olarak ele alınmıştır. Öğrencilerin yaptıkları kodlama etkinlikleri ve öntest-sontest cevapları karşılaştırılmıştır. Öğrencilerin uygulama öncesinde seviyelerini belirlemek amacıyla 20 soruluk öntest uygulanmıştır. Değerlendirme öğrencilerin doğru sayıları üzerinden yapılmıştır. Öğrencilerin en çok zorlandıkları ve en kolay yanıt verdikleri sorular tespit etmek amacıyla testte yer alan her bir soruyla ilgili dağılımlar verilmiştir. Son olarak sorulan sorular ile yapılan kodlama etkinlikleri arasındaki beceriyi inceleyebilmek adına sorular gruplara ayrılmıştır. Verilen cevaplar ile öğrencilerin kodlama esnasında yaptıkları etkinlikler karşılaştırmalı olarak incelenmiş ve öğrencilerin zorlandıkları yerler tespit edilmiştir. Öğrencilerin yaptıkları kodlama etkinlikleri arasından araştırma için önemli görülenlere yer verilmiştir. Ayrıca öğrencilerle yapılan görüşmelerde hataların kaynaklarına ve öğrencilerin yaygın olarak yaşadıkları güçlüklerle de yer verilmiştir.

4.1. Öntest ve Sontest Betimsel Analiz Sonuçlarına Yönelik Bulgular

Öğrencilerin Scratch ile tasarlanmış matematiksel algoritmalar öntest ve sontestinden aldıkları toplam puanların ortalaması, en düşük ve en yüksek puan değerleri Tablo 10'da verilmiştir. Öğrencilerin her bir doğru yanıtları için belirlenen puan değeri 1'dir. Testte alınabilecek en düşük puan 0, en yüksek puan ise 20'dir. 0 puan alan öğrenci sayısı öntestte ve sontestte hiç yokken 20 alan öğrenci de bulunmamaktadır. Öğrencilerin öntestte aldıkları puanların ortalaması 8,5 iken sontestte 9,5 olmuştur. Öntestte minimum puan alan öğrencinin puanı 3 iken maksimum puanı alan öğrencinin puanı ise öntestte 16'dır. Sontestte ise minimum puan 4 olurken maksimum değer 18 puan olmuştur. Öntestte 0-5 puan aralığında 3 öğrenci, 6-10 puan aralığında 2 öğrenci, 11-15 puan aralığında 3 öğrenci ve 16-20 aralığında 0 öğrenci bulunurken sontestte bu dağılım 0-5 puan aralığında 3 öğrenci, 6-

10 puan aralığında 2 öğrenci, 11-15 puan aralığında 2 öğrenci ve 16-20 aralığında 1 öğrenci olmuştur. Yapılan değerlendirmede öntest ile sontest arasındaki farkı negatif olan iki öğrenci vardır.

Tablo 10

Öğrencilerin Scratch ile tasarlanmış matematiksel algoritmalar öntest- sontest puan değerleri

Test	$\bar{X}$	Min	Max	Mod	Çarpıklık	Basıklık	Toplam
Öntest	8,5	3	16	3	,145	-1,816	68
Sontest	9,75	4	18	5	,406	-,983	78

4.1.1. Öntest ve Sontest Frekans ve Yüzde Dağılımlarına İlişkin Bulgular

Öğrencilerin testte yer alan sorulara yönelik vermiş oldukları cevaplara ait dağılımlar ve Tablo 11’de uygulanan öntest ve sontest ait frekans ve yüzde değerleri verilmiştir. Öğrencilerin tamamı her soruyu yanıtlamış ve herhangi bir soruyu boş bırakmamışlardır. Öğrencilerin tamamının doğru yaptığı soru bulunmamakla birlikte hepsinin yanlış yaptığı iki soru olduğu görülmektedir. EBOB bulunmasının ve ilgili problemin çözülmesinin istendiği bu iki soruda öğrencilerin verileri ilişkilendirmekte zorlandıkları görülmüştür. 6. Soru öğrenciler tarafından en kolay cevaplandırılan olmuştur. Uygulanan testte EBOB ve EKOK doğrudan cevap olarak istendiği sorularda öğrenciler daha başarılı olurken EBOB veya EKOK çok adımlı bir problemlerin çözümü için gerekli bir adım olarak alındığı daha karmaşık problemlerde ise zorlandıkları görülmüştür.

Tablo 11

Scratch kullanımının matematiksel problem çözme becerilerine etkisi öntest ve sontest ait frekans ve yüzde dağılımları

	Öntest				Sontest			
	Yanlış		Doğru		Yanlış		Doğru	
	f	%	f	%	f	%	f	%
S1	3	37,5	5	62,5	3	37,5	5	62,5
S2	2	25	6	75	2	25	6	75
S3	3	37,5	5	62,5	2	25	6	75
S4	3	37,5	5	62,5	3	37,5	5	62,5
S5	6	75	2	25	5	62,5	3	37,5
S6	1	12,5	7	87,5	2	25	6	75
S7	5	62,5	3	37,5	2	25	6	75
S8	7	87,5	1	12,5	5	62,5	3	37,5
S9	3	37,5	5	62,5	4	50	4	50
S10	3	37,5	5	62,5	3	37,5	5	62,5
S11	6	75	2	25	5	62,5	3	37,5
S12	5	62,5	3	37,5	7	87,5	1	12,5
S13	5	62,5	3	37,5	3	37,5	5	62,5
S14	5	62,5	3	37,5	6	75	2	25
S15	3	37,5	5	62,5	5	62,5	3	37,5
S16	6	75	2	25	6	75	2	25
S17	8	100	0	0	7	87,5	1	12,5
S18	8	100	0	0	5	62,5	3	37,5
S19	4	50	4	50	4	50	4	50
S20	5	62,5	3	37,5	5	62,5	3	37,5

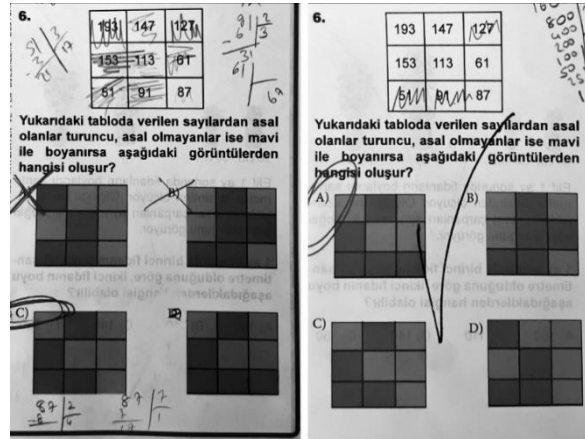
4.1.1.1. Asal Sayı Problemleri

Öntest sontest asal sayı içeren 6. soruya öğrencilerin verdiği cevaplara ait istatistiksel dağılımlara ait frekans ve yüzde dağılımları Tablo 12’de verilmiştir. Öntestte soruyu yalnızca bir öğrenci doğru cevap veremezken sontestte bu sayının ikiye çıktığı görülmüştür. Hem öntestte hem de sontestte katılımcılarının çoğunun doğru cevap verdiği görülmektedir.

Tablo 12*Soruya ait yüzde ve frekans dağılımları*

Öntest				Sontest			
Yanlış		Doğru		Yanlış		Doğru	
f	%	f	%	f	%	f	%
1	12,5	7	87,5	2	25	6	75

Öğrencilerden asal olan sayıları belirtilen renge boyadıklarında ortaya çıkacak durumu bulmaları istenmiştir. Öğrencilerden bölünebilme kurallarını kullanarak verilen büyük bir sayının asal olup olmadığını tespit etmeleri beklenmiştir. Öntestte soruya doğru cevap veren iki öğrencinin sontestte soruya doğru cevap verememişlerdir. Öntestte soruya yanlış cevap veren bir öğrencinin ise sontestte soruya doğru cevap vermiştir. Öntestte soruya doğru cevap verirken sontestte soruya yanlış cevap veren öğrencilerin asal sayılarla ilgili kodlama etkinliklerini yapmamışlardır. Öntestte soruya yanlış cevap verirken sontestte soruya doğru cevap veren öğrencinin öntest sontest karşılaştırmalı cevap kâğıdı örneği şekil 14'te verilmiştir. Öğrenciye ait asal sayılarla ilgili kodlama etkinliği ise şekil 15'te gösterilmiştir.

Şekil 14*Öntest (solda) sontest (sağda) karşılaştırmalı öğrenci cevap kâğıdı*

Öğrencinin öntestinde (solda) 87'yi asal olarak aldığı görülmektedir. Kalan boyamaların ise tamamı doğrudur. Sontestte bu sayıyı asal olarak almamış ve soruya doğru cevap vermiştir.

Şekil 15

Asal sayı içeren örnek öğrenci kodlama etkinliği (Scratch sayfası)



4.1.1.2. Asal Çarpanlarına Ayırma Problemleri

Asal çarpanlara ayırma içeren 2, 3, 7, 11 ve 16. sorulara öğrencilerin verdiği cevaplara ait öntest ve sontest frekans ve yüzdelere Tablo 13'te verilmiştir. Öntestte 5 sorudan yalnızca ikisinde öğrencilerin yarıdan fazlası başarılı olurken sontestte bu oran 5 soruda 3'e çıkmıştır. 2 ve 16. sorular için başarı durumu değişmezken 3 ve 11. sorularda başarı yüzdesi artmış, en fazla artış ise 7. soruda olmuştur. Önteste göre başarı durumunun azaldığı bir soru olmamıştır.

Tablo 13

Asal çarpanlara ayırma içeren problemlere ait istatistiksel dağılım

Soru	Öntest				Sontest			
	Yanlış		Doğru		Yanlış		Doğru	
	f	%	f	%	f	%	f	%
S2	2	25	6	75	2	25	6	75
S3	3	37,5	5	62,5	2	25	6	75
S7	5	62,5	3	37,5	2	25	6	75
S11	6	75	2	25	5	62,5	3	37,5
S16	6	75	2	25	6	75	2	25

Şekil 16'da verilen problemde öğrencilerden sayıları asal çarpanlarına ayırabilmesi beklenmiştir. Bitkilerin boylarının asal çarpanlarının aynı olduğunu belirtmiş istenen bitkinin boyunun aynı asal çarpanlara sahip olan sayının bulunması istenmiştir. İki öğrenci hariç hepsi doğru cevap vermiştir. Doğru cevap veren

öğrenciler asal çarpanlara ayırmayı doğru bir şekilde yaparken yanlış cevap veren öğrenciler ise soru hakkında fikir yürütemedikleri buna rağmen rastgele bir cevabı işaretledikleri görülmüştür. Soruya öntestte yanlış cevap veren iki öğrencinin kodlama etkinliği sonucunda uygulanan sontestte aynı soruya yine yanlış cevap verdikleri görülmüştür. Yanlış cevap veren öğrencilerin asal çarpanlarını bulmayı içeren kodlama etkinliğine hiç başlamadıkları görülmektedir. Şekil 16’da öntest ve sontestte soruya doğru cevap vermiş bir öğrencinin cevap kâğıdı ve Şekil 17’de asal çarpan içeren etkinlik örneği ve etkinliğe ait kodlama blokları verilmiştir.

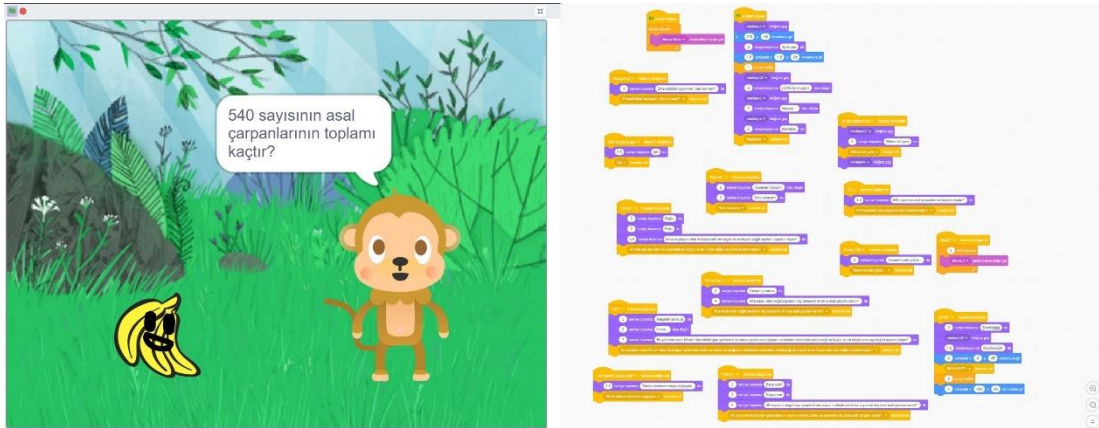
Şekil 16

Doğru kabul edilen cevap kâğıdı örneği



Şekil 17

Öğrenci kodlama etkinliği örneği (solda) ve kod blokları (sağda (Scratch sayfası)

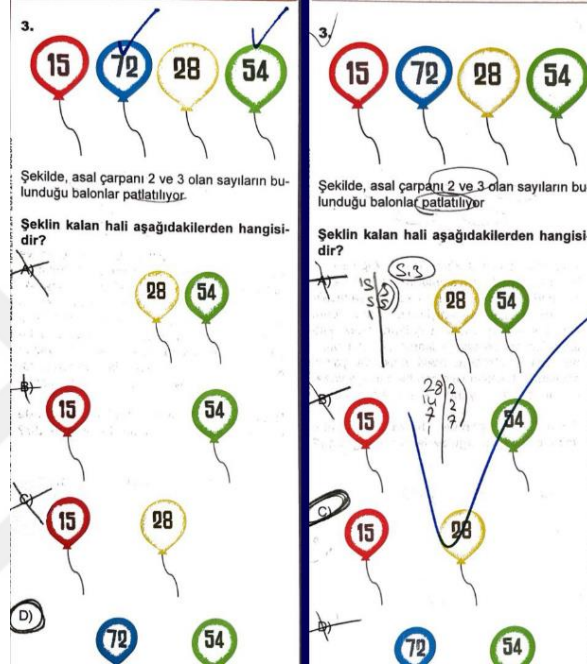


Şekil 18’de asal çarpan bulmayı içeren üçüncü soru ve öğrenciye ait öntest ve sontest cevapları karşılaştırmalı olarak sunulmuştur. Verilen problemde asal çarpanları

verilen sayıların bulunması sonucunda dışarıda kalan sayılar sorulmuştur. Soruya öntestte beş öğrenci doğru cevap verirken sontestte altı öğrenci doğru cevap vermiştir. Öntestte soruya yanlış cevap veren öğrencilerle sontestte soruya yanlış cevap veren öğrencilerin aynı olduğu görülmüştür. Öğrencinin öntestte soruya yönelik herhangi bir fikir yürütemeyen öğrencinin sontestte asal çarpanlarına ayırmaya çalıştığı görülmüyor.

Şekil 18

Öntest (solda) ve sontest (sağda) karşılaştırmalı öğrenci cevap kâğıdı

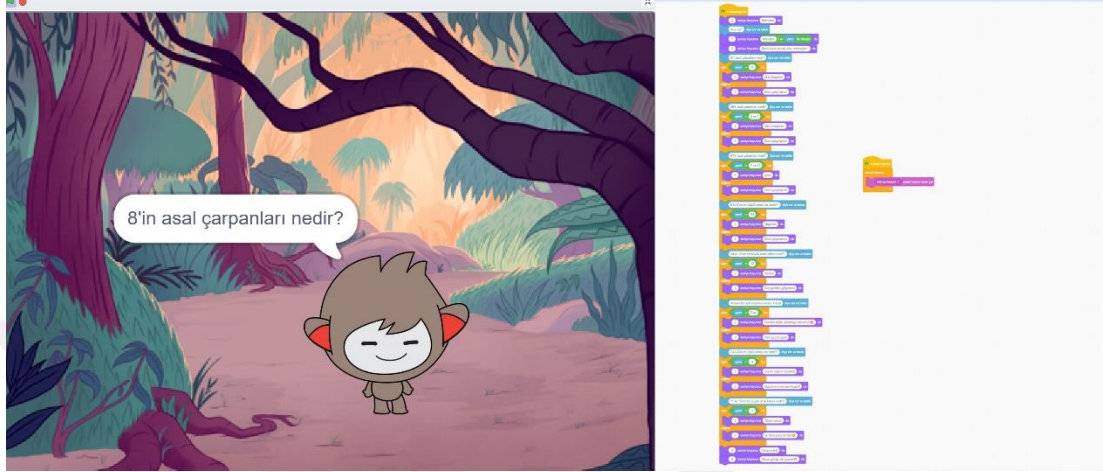


Öğrenciye ait örnek kodlama etkinliği Şekil 19’da verilmiştir. Sontestte doğru cevap veren öğrencinin asal çarpanlara ayırma ile ilgili yapılan soru-cevap etkinliğini yapmadığı ancak bir sayının asal olup olmadığının bulunduğu kodlama etkinliğini tamamladığı görülmektedir. Verilen etkinliğin kod blokları incelendiğinde sahnenin öğrenci tarafından doğru kurgulandığı söylenebilir.

Kodlamayı tamamlayan öğrenci etkinliği ve etkinliğe ait kod blokları örneği Şekil 21’de gösterilmiştir.

Şekil 21

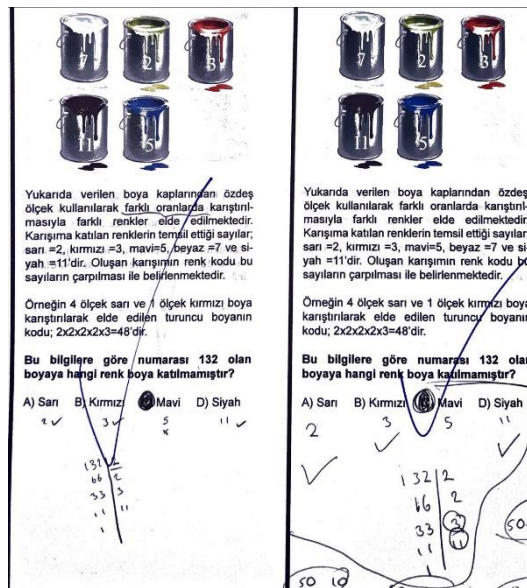
Kodlama etkinliği öğrenci örneği (solda) ve etkinliğe ait kod blokları (sağda) (Scratch sayfası)



Şekil 22’de hem öntestte hem de sontestte soruya doğru cevap veren öğrencinin öntest sontest karşılaştırmalı cevap kâğıdı verilmiştir. Öğrencinin soruya her iki testte de benzer şekilde yaklaştığı görülmektedir. Şekil 23’te yaptığı kodlama etkinliği ve bu etkinliğe ait kod blokları verilmiştir. Verilen kodlama etkinliğinde öğrencinin sahneyi doğru kurguladığı, asal çarpan içeren soruları ise 2 adet ile sınırlı tuttuğu, sorulardan birinin cevabını ise yanlış aldığı görülmektedir.

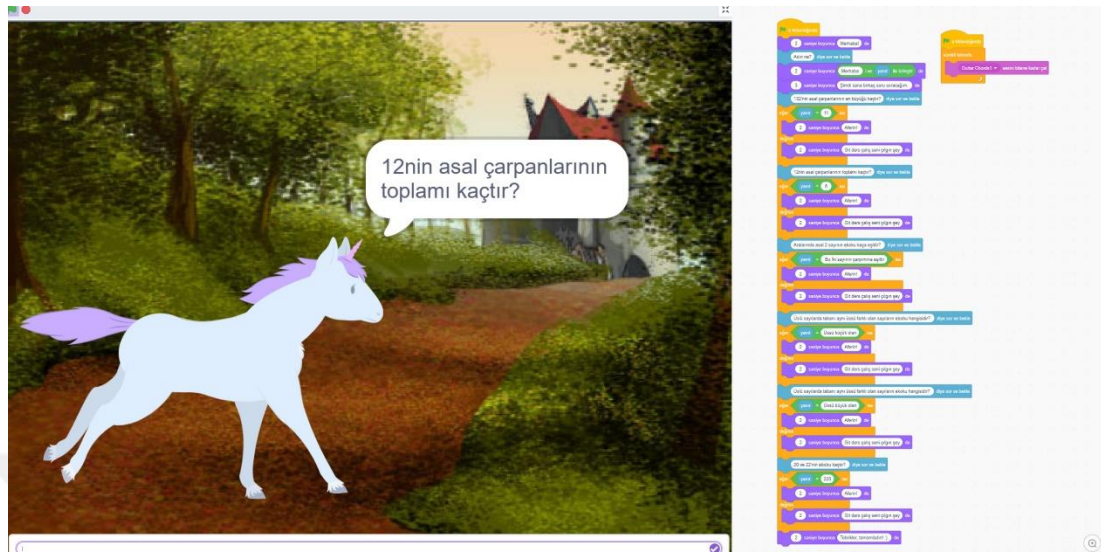
Şekil 22

Öntest (solda) ve sontest (sağda) öğrenci cevap kâğıdı örneği



Şekil 23

Asal sayı kodlama etkinliği öğrenci etkinliği örneği (solda) ve etkinliğe ait kod blokları (sağda) (Scratch sayfası)



Bir başka asal çarpan problemi Şekil 24'te gösterilmiştir. Verilen problemde öğrenciden şekil içerisinde yazan sayıların asal çarpanlarını bulması, bulunduğu asal çarpanları da şıklarda verilen sayıların asal çarpanları ile eşleştirmesi beklenmiştir. Verilen probleme öntestte yalnızca iki öğrenci doğru cevap verirken sontestte bu sayının üçe çıktığı görülmektedir. Öntestte soruya doğru cevap veren iki öğrenci sontestte de yine soruya doğru cevap vermişlerdir. Öntestte soruya doğru cevap veremeyip sontestte soruya doğru cevap veren öğrencinin öntest sontest karşılaştırmalı cevapları Şekil 24'te verilmiştir. Öntestte şekil içerisinde verilen sayıları asal çarpanlarına ayıran öğrencinin bulunduğu asal çarpanları şıklarda verilen sayılarla ilişkilendiremediği ve daha fazla devam edemediği görülmektedir. Sontestte ise sayıları aynı şekilde asal çarpanlarına ayıran öğrencinin bu kez şıklarda verilen sayıları da asal çarpanlarına ayırmıştır. Bu sayıların asal çarpanları ile verilen sayıların asal çarpanlarını eşleştirmeye çalıştığı görülmektedir.

Şekil 24

Öntest (sağda) sontest (solda) karşılaştırmalı öğrenci cevap kâğıdı örneği

11.

Şekil - I

Şekil - II

Şekil - III

Şekil - IV

Berra, şekilde yazan her sayının asal çarpanlarını bulup etrafındaki üçgenlere yazıyor ve ilk üç şeklin her birinden, birbirinden farklı asal sayılar alarak Şekil IV'e yazıyor ve asal çarpanları bu sayılar olan bir sayı oluşturuyor.

Berra'nın elde ettiği bu sayı aşağıdakilerden hangisi olabilir?

A) 204 B) 147 C) 385 D) 693

11.

Şekil - I

Şekil - II

Şekil - III

Şekil - IV

Berra, şekilde yazan her sayının asal çarpanlarını bulup etrafındaki üçgenlere yazıyor ve ilk üç şeklin her birinden, birbirinden farklı asal sayılar alarak Şekil IV'e yazıyor ve asal çarpanları bu sayılar olan bir sayı oluşturuyor.

Berra'nın elde ettiği bu sayı aşağıdakilerden hangisi olabilir?

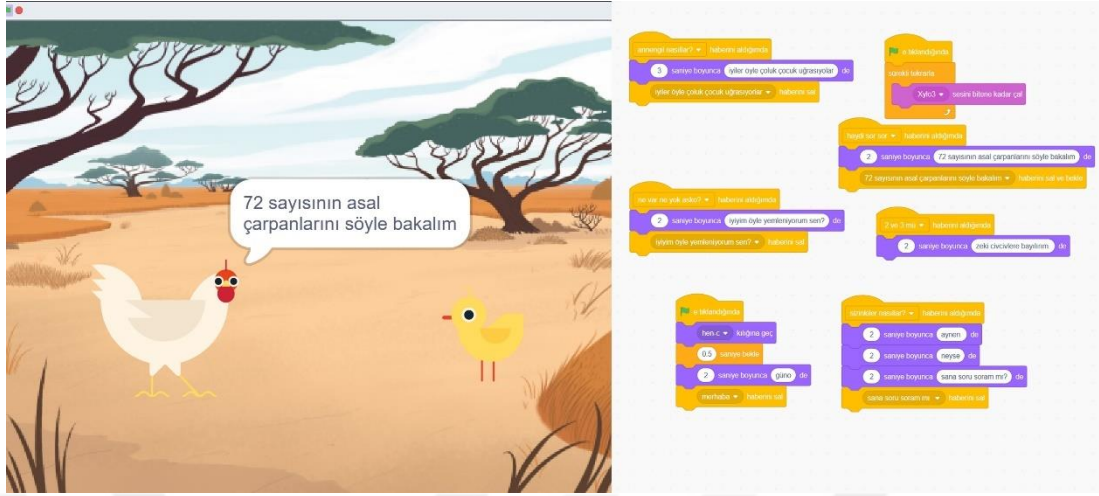
A) 204 B) 147 C) 385 D) 693

Handwritten calculations for prime factorization are visible on the right side of the page, showing the prime factors of 168, 165, and 156, and the selection of prime factors 2, 3, and 5 to form the number 30.

Öğrencinin asal çarpan içeren kodlama etkinliği ve etkinliğe ait kod blokları Şekil 25'te sunulmuştur. Verilen kodlama etkinliğinde öğrencinin sahneyi doğru kurguladığı ancak yapılan etkinliğin yarım bırakıldığı görülmektedir. Verilen etkinlikte asal çarpan içermesi gereken soru miktarının yalnızca bir tane ile sınırlı kaldığını görülmektedir. Bunun yanı sıra sorulan soruya uygun bir kod bloğunu da koymadığı görülmektedir. Verilen etkinlik aynı zamanda soru-cevap etkinliğini de içermektedir. Öğrenci verilen etkinliği görünüm bloklarıyla değil olay bloklarıyla kodladığı görülmektedir.

Şekil 25

Asal Çarpan İçeren Örnek Öğrenci Kodlaması (Scratch sayfası)



Şekil 26'da asal çarpan içeren problem örneğinde öğrenciden belirli sayıları asal çarpanlarına ayırması değil asal çarpanları verilmiş olan sayılar bazı özel durumlarla sınırlandırılması durumunda belirtilen şartları taşıyan sayıları bulması istenmiştir. Bu soruda öğrenci belirtilen asal çarpanların katlarını değil kuvvetlerini bulmalıdır. Örneğin birinci torbada 2'nin kuvvetleri olan iki basamaklı sayıları; 2. torbada ise sadece 3'ün kuvvetleri veya 2'nin ve 3'ün kuvvetlerinin çarpımı olan iki basamaklı sayıları bulması; 3. torbada ise 5'in kuvveti ya da 2'nin, 3'ün ve 5'in kuvvetlerinin birer kombinasyonu olan iki basamaklı sayıları bulabilmesi beklenmektedir. Öntestte öğrencilerden yalnızca ikisi soruya doğru cevap verebilirken sontestte de bu sayının değişmediği görülmektedir. Öntestte ve sontestte soruya doğru cevap verenlerin aynı öğrenciler olduğu; öntestte soruyu çözemeyen öğrencilerin sontestte de soruyu çözemedikleri görülmektedir. Şekil 26'da soruya hem öntestte hem de sontestte çözemeyen bir öğrencinin öntest sontest karşılaştırmalı cevapları verilmiştir. Öntestte sorunun üçüncü maddesi için bir çözüm geliştirebilen öğrencinin sontestte bu çözüm yöntemini tekrar kullanmadığı, soruyu boş bırakıp geçtiği görülmektedir. Soruya yanlış cevap veren diğer öğrencilerin hem öntestte hem de sontestte soruya yönelik herhangi bir çözüm yöntemi geliştiremedikleri ya boş bıraktıkları ya da rastgele bir şık işaretledikleri görülmüştür. Soruya yanlış cevap veren öğrencilerden ikisinin asal çarpanlara ayırma ile ilgili kodlama etkinliklerini hiç yapmazlarken kalan dört öğrenci etkinliklerini tamamlamışlardır. Soruya doğru cevap veren her iki öğrenci de kodlama etkinliklerini tamamlamışlardır.

Şekil 26

Öntest (solda) sontest (sağda) karşılaştırmalı öğrenci cevap kâğıdı örneği

16.



Asal çarpanı sadece 2 olan iki basamaklı sayılar torbası



En büyük asal çarpanı 3 olan iki basamaklı sayılar torbası



En büyük asal çarpanı 5 olan iki basamaklı sayılar torbası

Yukarıda üç torba verilmiştir. Bu torbalarda bulunan sayılarla ilgili bilgiler torbaların altında yazmaktadır.

Birinci torbadan A, ikinci torbadan B, üçüncü torbadan C sayısı çekildiğine göre;

I. $A + B$ en az 41'dir.
 II. $A - B$ en çok 52'dir.
 III. C nin alabileceği en büyük ve en küçük değerlerin toplamı 100'dür.

İfadelerinden hangileri doğru olur?

A) I, II, III B) II ve III C) I ve II D) Yalnız III

I. $A + B$ en az 41'dir.
 II. $A - B$ en çok 52'dir.
 III. C nin alabileceği en büyük ve en küçük değerlerin toplamı 100'dür.

İfadelerinden hangileri doğru olur?

A) I, II, III B) II ve III C) I ve II D) Yalnız III

Handwritten calculations:

```

72  24  24
36  12  24
18   6  36
6    2  24
3    1

24  24  24
12  12  12
6   6   6
2   2   2
1   1   1

```

4.1.1.3. Aralarında Asal Sayılar Problemleri

Tablo 14'te aralarında asal sayı içeren problemlere katılımcıların vermiş oldukları cevaplara ait frekans ve yüzde dağılımları gösterilmiştir. Öntestte katılımcıların yarısından fazlası 9, 10 ve 19.'uncu sorulara doğru cevap verebilirken 20. soruda ise yarısından fazlasının sorulara yanlış cevap verdiği ya da hiç cevap veremediği görülmektedir. Sontestte ise 9 ve 19. sorulara doğru cevap veren katılımcı sayısında bir azalma söz konusu olurken 10. soruda bu oranın değişmediği; 20. soruda ise soruya doğru cevap veren katılımcıların sayısında artış olduğu ancak yine de yarıdan fazlasının soruyu hala çözemediği görülmektedir.

Tablo 14

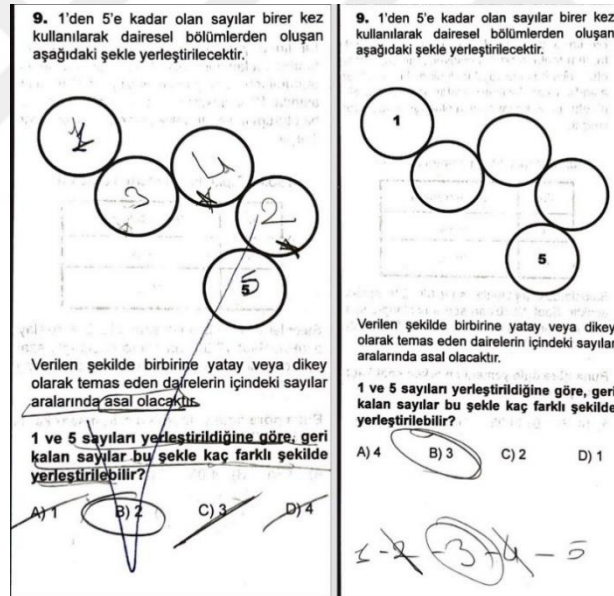
Aralarında asal sayı problemlerine ait öğrenci cevaplarının frekans ve yüzde dağılımları

Soru	Öntest				Sontest			
	Yanlış		Doğru		Yanlış		Doğru	
	f	%	f	%	f	%	f	%
S9	3	37,5	5	62,5	4	50	4	50
S10	3	37,5	5	62,5	3	37,5	5	62,5
S19	4	50	4	50	4	50	4	50
S20	5	62,5	3	37,5	5	62,5	3	37,5

Şekil 27’de asal çarpanlara ayırma içeren problem ve öntestte soruya doğru cevap verip sontestte doğru cevap veremeyen öğrencinin öntest sontest karşılaştırmalı cevap kâğıdı verilmiştir. Problemden öğrencilerden 1’den 5’e kadar olan sayıların aralarında asal olacak şekilde sıralamaları istenmiştir. Verilen problemi öntestte 5 öğrenci doğru cevaplarırken bu sayının sontestte 4’e düştüğü görülmüştür. Soruya öntestte doğru cevap veren 4 öğrenci sontestte de soruya doğru cevap vermiştir. 1 öğrenci ise sontestte yanlış cevap vermiştir. Şekil 27’de verilen öğrenci cevabında öğrencinin öntestte soruya doğru cevap vermesine rağmen sıralamayı doğru yapmadığı görülmektedir. Sontestte verdiği cevabı incelediğimizde ise cevap yanlış olmasına rağmen yapılan sıralamanın doğru olduğu görülmektedir. Ancak öğrencinin kaç farklı sıralama yapılabilir sorusunu tek bir sıralamada dairenin içerisine gelebilecek sayının ne olabileceği şeklinde algıladığı da görülmektedir.

Şekil 27

Öntest (solda) sontest (sağda) karşılaştırmalı öğrenci cevabı örneği

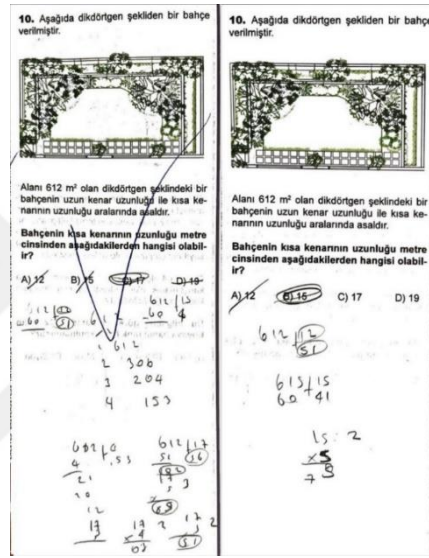


Şekil 28’de aralarında asal sayıların sorulduğu başka bir soru örneği ve soruyu öntestte doğru cevaplarırken sontestte yanlış yapan bir öğrenciye ait örnek öğrenci çözümü verilmiştir. Verilen soruda öğrencilerden alan bilgisi de istenmiş olup uzun kenarın kısa kenar ile aralarında asal olduğu belirtilmiştir. Alanı asal çarpanlarına ayırması ve çarpanları birbirleriyle bir kısa bir uzun kenar olacak şekilde eşleştirmesi beklenmiştir. Katılımcı soruya alternatif bir çözüm olarak şıklardan da gidebilir. Seçtiği şıktaki sayıyı verilen alana bölerek diğer kenarı bulabilir ve böylece şıkta verilen sayı ile kendi bulduğu ikinci kenarı karşılaştırarak sayıların aralarında asal olup

olmadığına bakarak sonuca ulaşabilir. Soruya öntestte öğrencilerin yarısından fazlası doğru cevap verirken sontestte de bu oranın değişmediği görülmektedir. Öntestte soruya doğru cevap veren öğrencilerden ikisinin sontestte soruya doğru cevap veremezken öntestte soruya doğru cevap veremeyen iki öğrencinin ise sontestte soruya doğru cevap verdikleri görülmektedir. Şekil 28’de öğrencinin soruyu şıklardan giderek çözdüğü görülmektedir. Öğrencinin soruya öntestte aralarında asal kavramına dikkat ederek yaklaşırken sontestte bu detayı gözünden kaçırdığı görülmektedir.

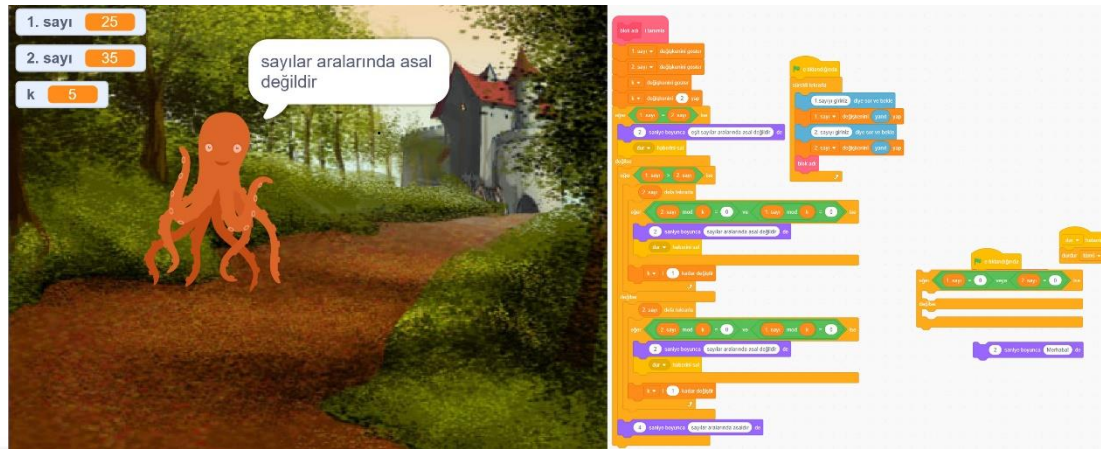
Şekil 28

Öntest (solda) sontest (sağda) karşılaştırmalı öğrenci cevap kâğıdı örneği



Öntestte soruya doğru cevap verirken sontestte yanlış cevap veren öğrencinin aralarında asal sayılarla ilgili yaptığı kodlama etkinliği ve etkinliğe ait kod blokları Şekil 29’da verilmiştir. Verilen kodlama etkinliğinde öğrencinin sahneyi doğru kurguladığı görülmektedir. Etkinlikte boşta kalan bazı kod bloklarının sahnedan atılmadığı görülmüyor.

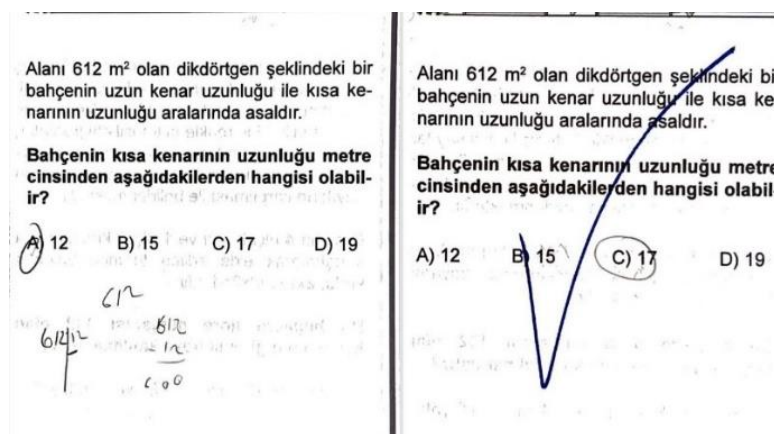
Aralarında asal örnek öğrenci kodlama etkinliği (solda) ve etkinliğe ait kod blokları (sağda) (Scratch sayfası)



Şekil 30’da ise öntestte soruya yanlış cevap verirken sontestte soruya doğru cevap veren öğrencinin cevap kâğıdı örneği verilmiştir. Soruda bir dikdörtgenin alanı verilmiş ve bu alana ait olabilecek kısa ve uzun kenarların aralarında asal olması istenmiştir. Verilen şekilde öğrencinin öntestte soruya şıklarda verilen sayıya bölmesi gerektiğini düşündüğü ancak işlemi devam ettirmediği; sontestte ise soruya herhangi bir çözüm sunmadığı ancak buna rağmen soruyu doğru yanıtladığı görülmektedir. Ayrıca öntestte soruya yanlış cevap verip sontestte soruyu doğru cevaplandıran öğrencilerin her ikisinin de kodlama etkinliklerini tamamlamadıkları görülmektedir.

Şekil 30

Öntest (solda) ve sontest (sağda) karşılaştırmalı öğrenci cevap kâğıdı örneği

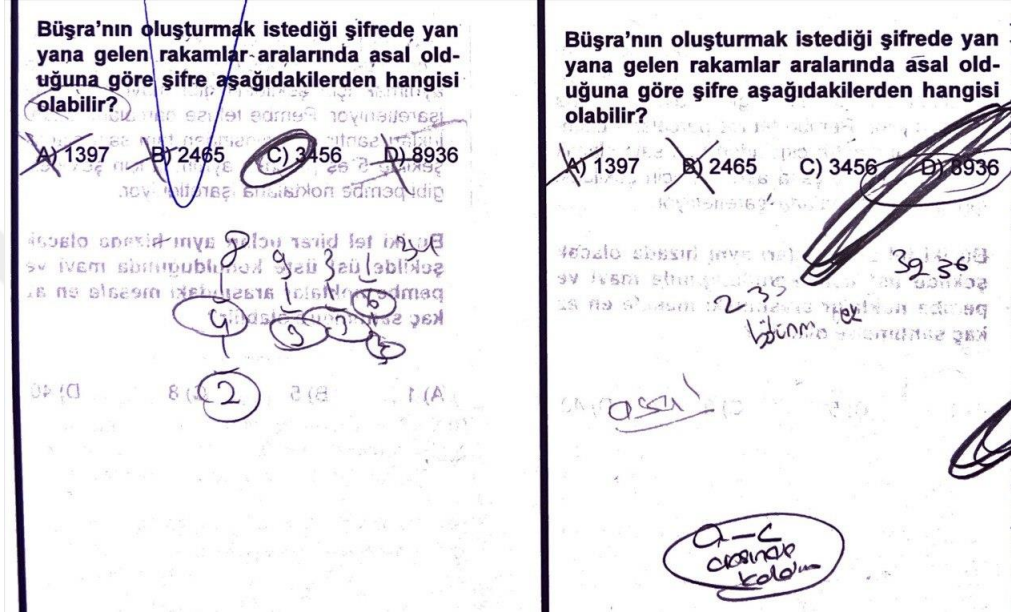


Şekil 31’de öntestte soruya doğru cevap verirken sontestte soruya yanlış cevap veren bir öğrencinin cevap kâğıdı örneği verilmiştir. Soruda öğrencilerden yan yana gelen sayıların aralarında asal olması şartıyla 4 basamaklı bir şifre oluşturmaları istenmiştir. Soruya öntestte doğru cevap veren öğrencinin son şıkta verilen sayının

basamaklarında yer alan her bir rakamın asal çarpanını bularak soruya doğru cevap verirken sontestte iki şık arasında kaldığını belirten bir ifadeyle yanlış şıkkı seçtiği görülmektedir. Soruya öntestte öğrencilerin yarıdan fazlası doğru cevap verirken sontestte bu oranın düştüğü görülmüştür. Öğrenciye ait kodlama etkinliği ve kod blokları örneği ise Şekil 32’de gösterilmiştir.

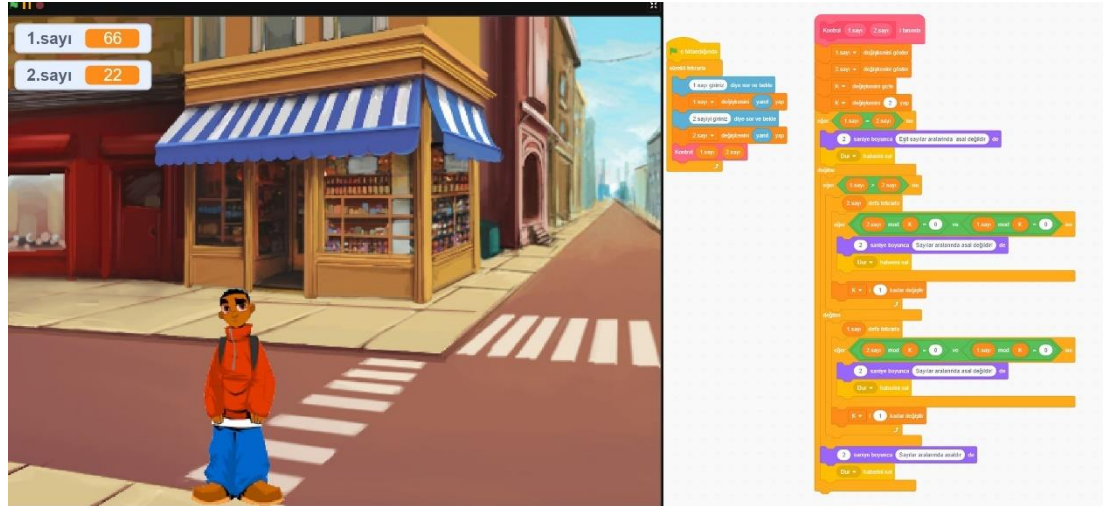
Şekil 31

Öntest (solda) ve sontest (sağda) karşılaştırmalı öğrenci cevap kâğıdı örneği



Şekil 32

Aralarında asal sayılarla ilgili kodlama etkinliği örneği (Scratch sayfası))

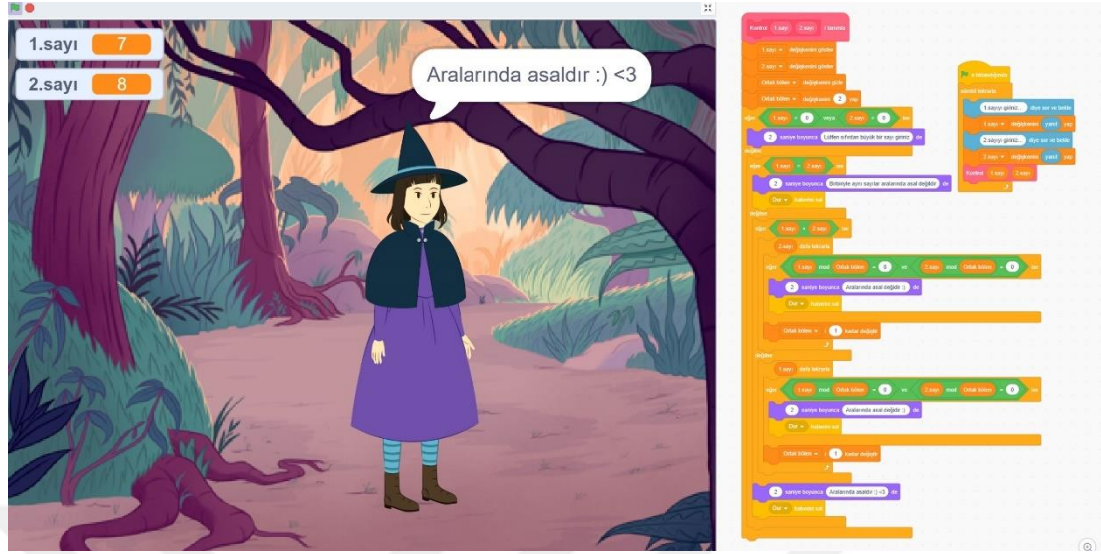


Şekil 33’te ise öğrencilerin yarısından fazlasının yanlış cevapladığı bir aralarında asal sayı problemi ve öntest (solda) ve sontest (sağda) karşılaştırmalı cevap kâğıdı örneği gösterilmiştir. Öğrencilerden problemde verilen kartları bir kırmızı ve bir mavi kartı aralarında asal olacak biçimde eşleştirmeleri beklenmiştir. Eşleştirme

Öntest (solda) ve sontest (sağda) karşılaştırmalı cevap kâğıdı örneği

Şekil 34

Aralarında asal sayılar öğrenci kodlama örneği (Scratch sayfası)



4.1.1.4. EBOB Problemleri

Tablo 15'te EBOB problemlerine ait frekans ve yüzde dağılımlarına ait veriler sunulmuştur. EBOB problemlerine öğrencilerin hem öntestte hem de sontestte doğru cevap verme oranlarının düşük olduğu görülmektedir. 12, 14 ve 15. Sorularda öğrencilerin doğru cevap verme oranlarının sontestte düştüğü görülmektedir. 13. Soruda öntestte katılımcıların yarısından fazlası soruya yanlış cevap verirken sontestte bu oranın tersine dönerek yarısından fazlasının doğru cevap verdiği görülmektedir.

Tablo 15

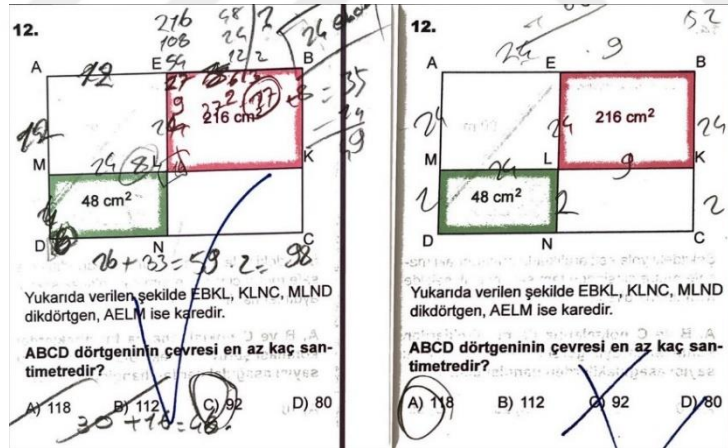
EBOB Problemlerine Ait Frekans ve Yüzde Dağılımları

	Öntest				Sontest			
	Yanlış		Doğru		Yanlış		Doğru	
	f	%	f	%	f	%	f	%
S12	5	62,5	3	37,5	7	87,5	1	12,5
S13	5	62,5	3	37,5	3	37,5	5	62,5
S14	5	62,5	3	37,5	6	75	2	25
S15	3	37,5	5	62,5	5	62,5	3	37,5
S17	8	100	0	0	7	87,5	1	12,5

Şekil 35'te öntestte öğrencilerin yalnızca üç tanesinin doğru cevap verebildiği, sontestte ise bu oranın daha da düşerek doğru cevaplandıran öğrenci sayısının sadece bir ile sınırlı kaldığı bir EBOB problem örneği ve soruya öntestte doğru cevap verirken sontestte yanlış cevap veren bir öğrencinin öntest ve sontest karşılaştırmalı cevap kâğıdı verilmiştir. Bir öğrenci soruyu hem öntestte hem de sontestte doğru cevaplarırken kalan iki öğrencinin soruyu öntestte doğru cevaplarırken sontestte yanlış cevap verdikleri görülmüştür. Problemden alanı verilen dikdörtgensel bölgelerden her iki bölgenin de birer kenar uzunluklarının aynı olması sebebiyle ortak bölenlerinin bulunması beklenmiştir. Çevreyi en az istediği için uzun ve kısa kenar uzunluklarının olabildiğince birbirine yakın sayılardan seçilmesi beklenmiştir. Bu tip sorularda ortak bölenlerin en büyüğü bazen doğrudan cevap olabileceken bazen bu sayının kendisi değil bölenleri de koşulu sağlayan sayılar arasında yer alabilmektedir. Soruda yer alan en az ibaresi de bunun bir örneğidir. Verilen alanların EBOB cevap olarak şıklarda vardır ancak ortak bölenlerin en büyüğünün bölenlerinden birisi de cevaplar da bulunmaktadır. En az şartını sağlayan hangi sayı ise koşullara göre o seçilmelidir.

Şekil 35

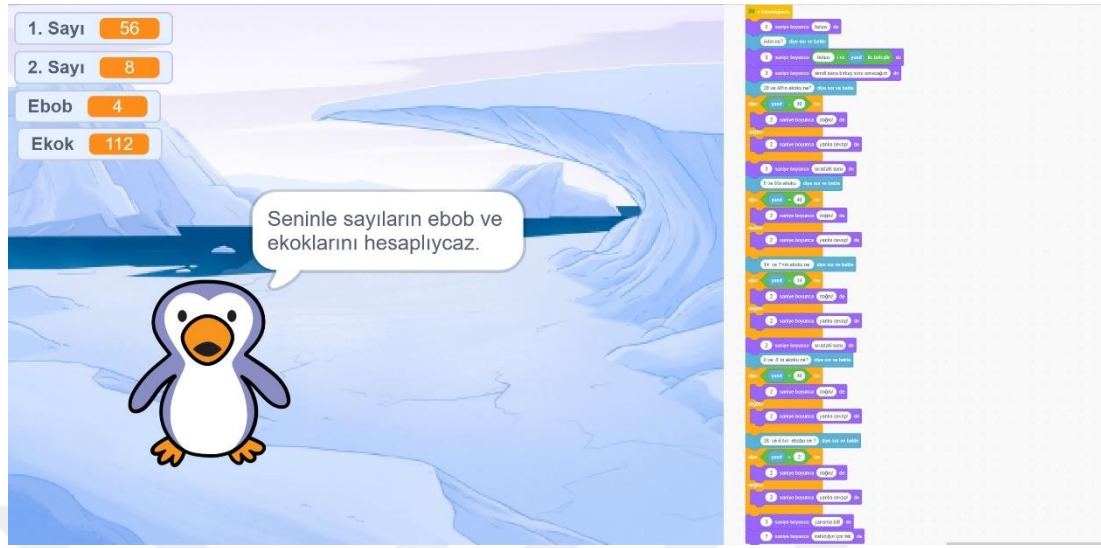
En büyük ortak bölen öğrenci cevap kâğıdı örneği öntest (solda) ve sontest (sağda)



Şekil 35'te öğrenci öntestte ortak olan kenar için EBOB bir bölenini ve diğer kenar uzunluklarını doğru seçerken sontestte ortak olan kenar için doğrudan EBOB seçme eğiliminde olduğu görülmektedir. Şekil 37'de ise öğrencinin EBOB ile ilgili yaptığı kodlama etkinliği ve etkinliğe ait kod blokları gösterilmiştir. Gösterilen etkinlikte öğrencinin sahneyi doğru kurguladığı ancak EBOB ile alakalı soruyu yalnızca 1 tane ile sınırlı tuttuğu görülmektedir. Etkinlik ayrıca etkinliği çalıştıran kullanıcı ile soru-cevap tarzı içeriği ve EBOB sorusu yanında EKOK ile alakalı birkaç soru da içermektedir.

Şekil 36

En büyük ortak bölen içeren öğrenci kodlama etkinliği örneği (Scratch sayfası)



Şekil 37’de öntestte soruya yanlış cevap verip sontestte doğru cevaplandıran öğrencilerden birine ait bir cevap kâğıdı örneği verilmiştir. Soruyu öntestte doğru yanıtlayan üç öğrencinin sontestte de soruya yine doğru cevap verdiği görülmüştür. Sontestte ise beş öğrenci soruyu doğru cevaplandırmıştır. Problemde öğrencilerden işaretlenmiş sayı ile EBOB’u 5 olan sayıları tespit etmesi beklenmektedir.

Şekil 37

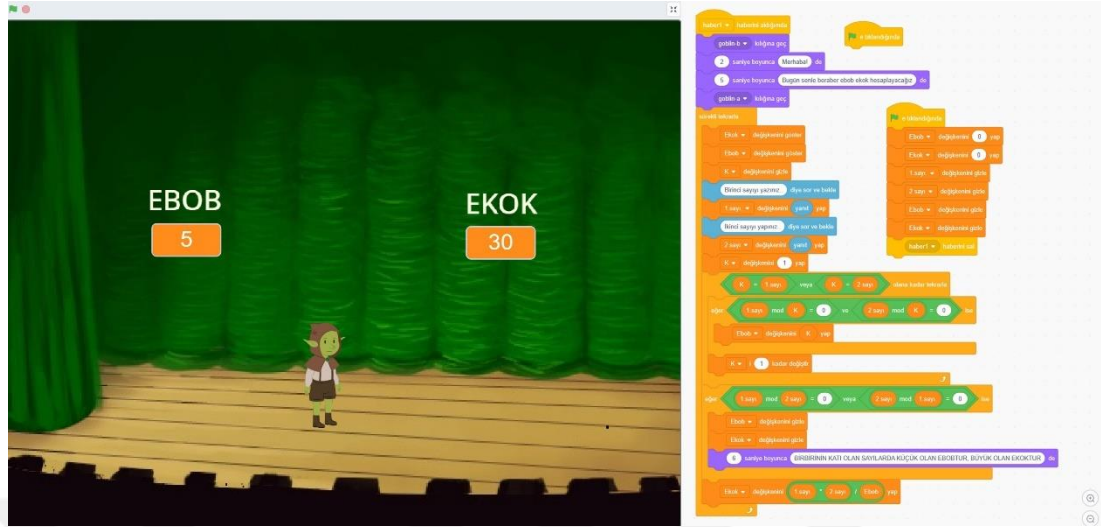
Öntest (solda) sontest (sağda) öğrenci cevap kâğıdı örneği



Şekil 37’de verilen cevap kâğıdı örneğinde öğrencinin öntestte 25 sayısını asal çarpanlarına ayırdığı fakat bunu takvimde verilen sayılarla bağdaştıramadığı ve soruya rastgele cevap verdiği görülmektedir. Sontestte ise ortak böleni 5 olan günleri işaretleyerek cumartesi gününü boşa bıraktığı görülmektedir. EBOB içeren örnek kodlama etkinliği ve bu etkinliğe ait kod blokları örneği ise Şekil 38’de verilmiştir. Yapılan kodlama etkinliğinde öğrencinin sahnesini doğru kurguladığı görülmektedir. Etkinlik EBOB’un yanı sıra EKOK’ta içermektedir.

Şekil 38

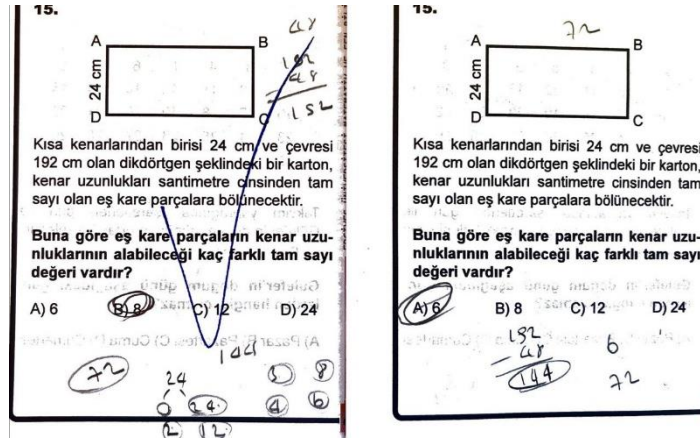
En büyük ortak bölen içeren öğrenci kodlama etkinliği örneği (Scratch sayfası)



Şekil 39’da verilen problemde uzunlukları verilen iki yolun arasına eşit aralıklarla dikilecek direklerin sayısının neler olabileceğini bulmaları ve buna karşılık dışarıda kalan şikkı işaretlemeleri istenmiştir. Dikdörtgeni kare parçalara ayırması ve bunu kaç farklı uzunlukta kare ile yapabileceğini hesaplanması istenmiştir. Öntestte beş öğrenci soruyu doğru cevaplarırken sayının sontestte üçe düştüğü görülmektedir. İki öğrenci soruyu hem öntestte hem de sontestte doğru cevaplandırırken üç öğrenci öntestte doğru yaptığı sontestte ise yanlış cevaplandırırdığı; bir öğrencinin ise öntestte yanlış yaparken sontestte doğru yaptığı görülmüştür. Sontestte soruya yanlış cevaplar veren üç öğrenciden ikisinin cevap kâğıtlarında soruya ilişkin herhangi bir çözüm görülmemiştir. Öntestte öğrencilerin üçü soruya doğru cevap verebilirken sontestte bu sayının ikiye düştüğü görülmektedir. Öntestte doğru cevap veren öğrencilerin ikisi sontestte de soruyu doğru yanıtlarken bir öğrenci sontestte soruya doğru cevap verememiştir. Sontestte yanlış cevap veren öğrencinin ne öntestte ne de sontestte soruya ilişkin herhangi bir çözüm yaklaşımında bulunmamıştır. EBOB ile ilgili kodlama etkinliğini tamamlamadığı görülmüştür. Şekil 39’da öntest ve sontestte farklı cevaplar veren bir öğrenciye ait cevap kâğıdı örneğinde öğrencinin verilmeyen uzun kenarı bularak hem öntestte hem de sontestte uzun ve kısa kenarların ortak bölenlerinden birini doğrudan cevap olarak işaretlediği görülmektedir.

Şekil 39

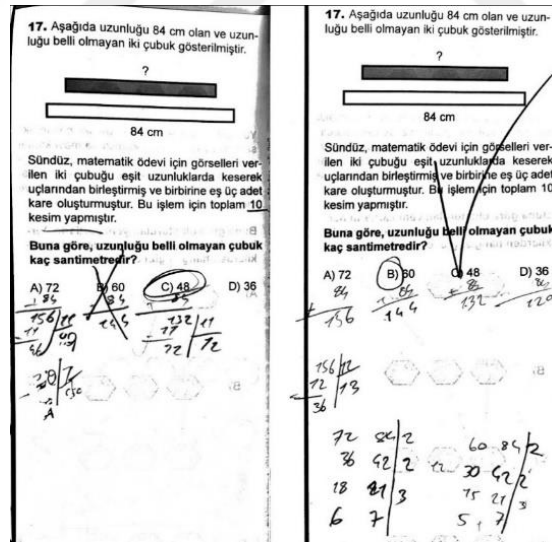
Öntest (solda), sontest (sağda) öğrenci cevap kâğıdı örneği



Şekil 40'ta öğrenci öntest, sontest karşılaştırmalı cevap kâğıdı örneği içeren bir EBOB problemi verilmiştir. Soruya öntestte öğrencilerin tamamının hatalı yaklaşırken sontestte bir öğrenci soruyu doğru yanıtladığı görülmüştür. Problemde katılımcılardan uzunlukları farklı iki çubuğu eşit uzunlukta ayrıldığında yapılan kesim sayısından yola çıkarak uzunluğu belli olmayan çubuğun uzunluğunu bulmaları istenmiştir.

Şekil 40

Öntest (solda), sontest (sağda) öğrenci cevap kâğıdı örneği

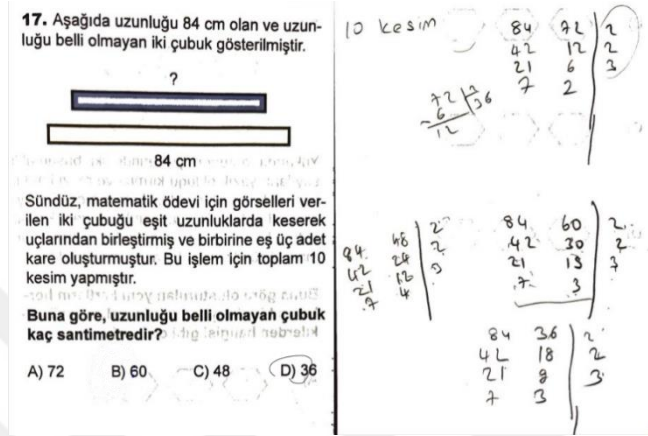


Şekil 40'ta verilen cevap kâğıdı örneğinde öğrencinin öntestte soruya uzunluğu verilen çubuğu şıklarda verilen uzunluklarla toplayarak yaklaştığı ancak çözüme giden bir yol bulamadığı görülmektedir. Sontestte ise tıpkı öntestte olduğu gibi şıklarda verilen uzunları uzunluğu belli olan çubuğun uzunlukla topladığı görülmektedir. Buna ek olarak şıklarda verilen sayılarla uzunluğu belli olan çubuğun uzunluğu ile asal çarpanlarına ayırdığı da görülmektedir. Şıklarda verilen sayıları deneyerek çözüme

ulaşmaya çalışan öğrencinin b şıkkından sonraki sayıları denemeyi bıraktığı görülmüyor. Şekil 41’de verilen bir başka cevap kâğıdı örneğinde ise öğrencinin şıklarda verilen sayılarla uzunluğu verilen çubuğun EBOB’unu bulmaya çalıştığı ancak bunu kesim sayısı ve oluşan parça sayısı ile ilişkilendiremediği görülmektedir.

Şekil 41

Öğrenci cevap kâğıdı örneği



4.1.1.5. EKOK Problemleri

EKOK problemlerine ait frekans ve yüzde değerleri Tablo 16’da gösterilmiştir. Tablo 16’da yer alan verilere göre öntestte öğrencilerin yarısından fazlası 1 ve 4. sorulara doğru yanıt vermişlerdir. 8 ve 18. sorularda ise öğrencilerin çoğunun sorulara doğru cevap veremedikleri görülmektedir.

Tablo 16

En Küçük Ortak Kat Problemlerine Ait Frekans ve Yüzde Dağılımları

Soru	Öntest				Sontest			
	Yanlış		Doğru		Yanlış		Doğru	
	f	%	f	%	f	%	f	%
S1	3	37,5	5	62,5	3	37,5	5	62,5
S4	3	37,5	5	62,5	3	37,5	5	62,5
S8	7	87,5	1	12,5	5	62,5	3	37,5
S18	8	100	0	0	5	62,5	3	37,5

Şekil 42’de verilen ve aynı zamanda öntest sontest karşılaştırmalı öğrenci cevap kâğıdını da içeren problem örneğinde ise konuşma balonlarının içerisinde

verilen sayı çiftlerinin ortak katlarının bulunması ve bulunan sonuçların kendi aralarında bir kez daha EKOK hesaplanması istenmiştir. Problemi doğru yanıtlayan öğrenci sayısında hem öntestte hem de sontestte bir değişiklik olmamasına rağmen bir öğrenci öntestte soruyu doğru cevaplandırmıştır. Ancak sontestte yanlış cevap vermiş; bir öğrenci ise öntestte yanlış cevaplandırırken sontestte soruya doğru cevap vermiştir.

Şekil 42

Öntest (solda) sontest (sağda) öğrenci cevap kâğıdı örneği

Yukarıda verilen şekilde Türkü, Alperen ve Hasret, söyledikleri sayıların ortak katlarını önlerinde bulunan sarı halkalara yazacakları bir oyuna başlamışlardır. Daha sonra sarı halkalarda yazan sayıların ortak katlarını ise her iki sarı halkanın önünde bulunan mavi halkalara, mavi halkarda yazan sayıların ortak katını ise pembe halkaya yazacakları şekilde oyunu devam ettirmişlerdir.

Buna göre pembe halka içerisinde yazılacak sayı aşağıdakilerden hangisidir?

A) 30 B) 60 C) 90 D) 120

193 =
167 =
124 =
51 = 312

Yukarıda verilen şekilde Türkü, Alperen ve Hasret, söyledikleri sayıların ortak katlarını önlerinde bulunan sarı halkalara yazacakları bir oyuna başlamışlardır. Daha sonra sarı halkalarda yazan sayıların ortak katlarını ise her iki sarı halkanın önünde bulunan mavi halkalara, mavi halkarda yazan sayıların ortak katını ise pembe halkaya yazacakları şekilde oyunu devam ettirmişlerdir.

Buna göre pembe halka içerisinde yazılacak sayı aşağıdakilerden hangisidir?

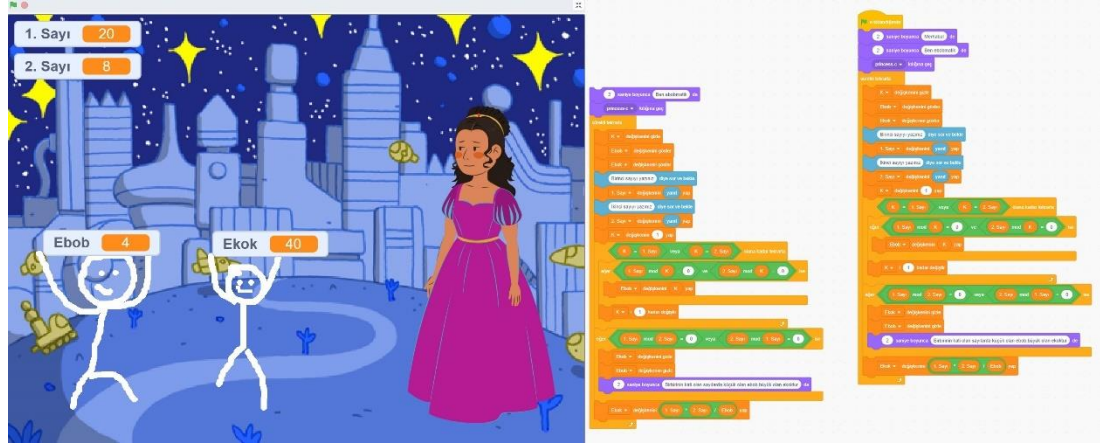
A) 120 B) 90 C) 60 D) 30

8

Şekil 42’de verilen cevap kâğıdı örneğinde öntestte soruya cevap veremeyen öğrencinin sontestte soruya doğru cevap verdiği ancak sorunun çözümüne dair net bir ifade bulunmadığı görülmektedir. Öğrenciye ait kodlama etkinliği ve bu etkinliğe ait kod blokları Şekil 43’te gösterilmiştir. Verilen öğrenciye ait kodlama etkinliğinde sahnenin birden fazla kurgulandığı ancak başlat kod bloğunun yalnızca birisine atanması sebebiyle sahnenin oynatılmasında hataya sebep olmaması sebebiyle doğru çalıştığı görülmektedir.

Şekil 43

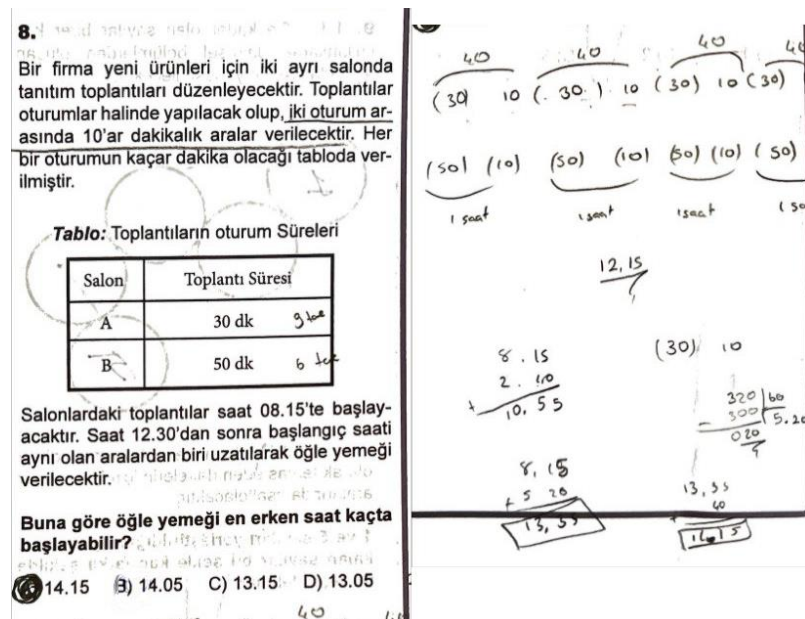
En küçük ortak kat içeren öğrenci kodlama etkinliği ve kodlama etkinliği kod blokları örneği (Scratch sayfası)



Şekil 44’te öğrencilerin yine büyük çoğunluğunun hatalı cevap verdiği EKOK problem örneği görülmektedir. Probleme öntestte yalnızca bir öğrenci doğru yanıt verebilirken sontestte üç öğrenci soruya doğru yanıt vermişlerdir. Öğrencilerden öntestte soruya doğru yanıt veren bir öğrencinin sontestte soruya yanlış cevap verdiği görülmüştür. Sontestte soruya doğru cevap veren öğrencilerin tamamı öntestte soruya yanlış cevap veren öğrencilerdir.

Şekil 44

Yanlış kabul edilen öğrenci cevap kâğıdı örneği

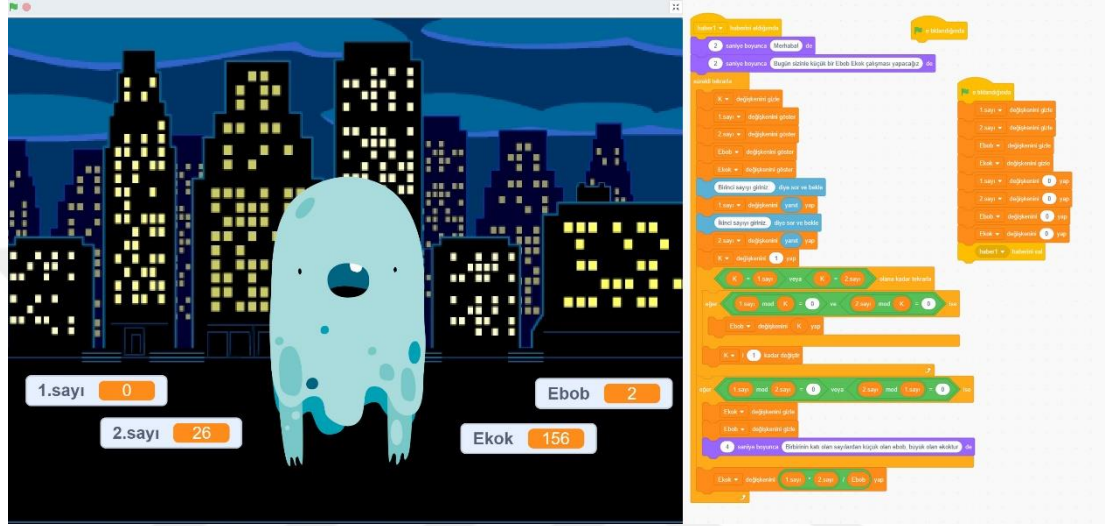


Şekil 44’te verilen öntest sönest karşılaştırmalı öğrenci cevap kâğıdı örneğinde öğrencinin soruda toplantı sürelerinin ortak katlarının her iki toplantı salonunun da ara verdiğiinde beraber denk gelecekleri zaman aralığı olacağını görebildiği ancak soruda

verilen “Saat 12:30’dan sonra başlangıç saati aynı olan aralardan biri uzatılarak öğle yemeği verilecektir.” ifadesini gözden kaçırdığı tahmin edilmektedir. Öğrenciye ait EKOK içeren kodlama etkinliği ve etkinliğe ait kod blokları örneği Şekil 45’te verilmiştir. Verilen etkinlikte öğrencinin sahneyi doğru kurguladığı görülmüştür.

Şekil 45

Öğrenci kodlama etkinliği ve kod blokları örneği (Scratch sayfası)

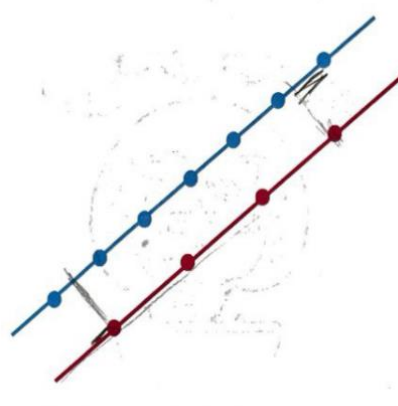


Tüm öğrencilerin öntestte hatalı yaklaştığı soru Şekil 46’da verilmiştir. Sontestte 3 öğrencinin soruya doğru cevap verdiği görülmektedir. Soruda eşit uzunlukta iki telin farklı sayıda parçalara ayrıldığı ve oluşan parçaların birbirine uzaklığının cm cinsinden alabileceği en az değerin bulunması istenmiştir. Öğrencinin oluşan parça sayılarının ortak katını alarak telin toplam uzunluğu hakkında bir yorum yapabilse de noktalar arasındaki mesafenin en kısa olduğu yeri sadece ilk noktalar ile kısıtlı tutarak doğru cevaba ulaşamadığı görülmektedir. Noktaların birbirine en çok yaklaştığı yer mavi telden üçüncü, pembe telden ise ikinci noktadır. İlk noktaları temel alarak 3 cm sonucuna varan öğrencinin 3 cm’e en yakın cevap olan 5 cm cevap olarak alma yoluna gittiği görülmüştür.

Şekil 46

Yanlış kabul edilen öğrenci cevap kâğıdı örneği

18. Aşağıda eşit uzunluğa sahip mavi ve pembe renkli iki tel verilmiştir.



Mavi tel, parçaların uzunlukları santimetre cins-

Bu iki tel birer uçları aynı hizada olacak şekilde üst üste konulduğunda mavi ve pembe noktalar arasındaki mesafe en az kaç santimetre olabilir?

A) 1 B) 5 C) 8 D) 40

emin değilim

40 5 80 10
8 30 176 46

20 8 120 15
20 15

25

4.1.1.6. Pozitif Bölen Sayısı Problemi

Tablo 17’de pozitif bölen sayısına ait frekans ve yüzde değerleri verilmiştir. Öntestte problemi iki öğrenci doğru cevaplarırken sontestte üç öğrenci doğru yanıtlamıştır. Beşinci soruya doğru cevap veren öğrencilerin bir tanesi soruya hem öntestte hem de sontestte doğru cevap verirken iki öğrenci öntestte soruya yanlış cevap verip sontestte soruyu doğru yanıtladığı; bir öğrencinin ise soruyu öntestte doğru yaparken sontestte yanlış cevapladığı görülmüştür.

Tablo 17

Pozitif bölen sayısı problemine (s5) ait frekans ve yüzde dağılımları

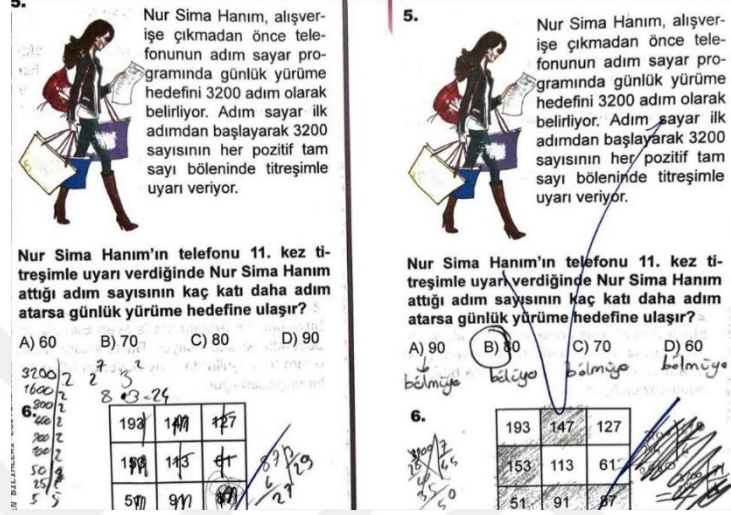
Öntest				Sontest			
Yanlış		Doğru		Yanlış		Doğru	
f	%	f	%	f	%	f	%
6	75	2	25	5	62,5	3	37,5

Şekil 47’de pozitif bölen sayısı hesaplamayı içeren problem örneği gösterilmiştir. Problemden öğrencilerden 3200 sayısının pozitif bölenini bulması beklenmiştir. Şekil 47’de verilen öğrenci cevap kâğıdı örneğini incelediğimizde öğrencinin öntestte 3200 sayısını asal çarpanlarına ayırarak sonuca ulaşmaya çalıştığı görülmektedir. Sontestte ise sayıyı asal çarpanlarına ayırmak yerine şıklardan gittiği ve şıklarda verilen sayıların 3200 bir böleni olup olmadığını kontrol ederek sonuca

ulaştığı görülüyor. Öğrenciye ait pozitif bölen sayısı içeren örnek bir kodlama etkinliği ve kodlama etkinliğine ait kod blokları Şekil 48’de verilmiştir. Öğrencinin sahneyi doğru kurguladığı görülmektedir.

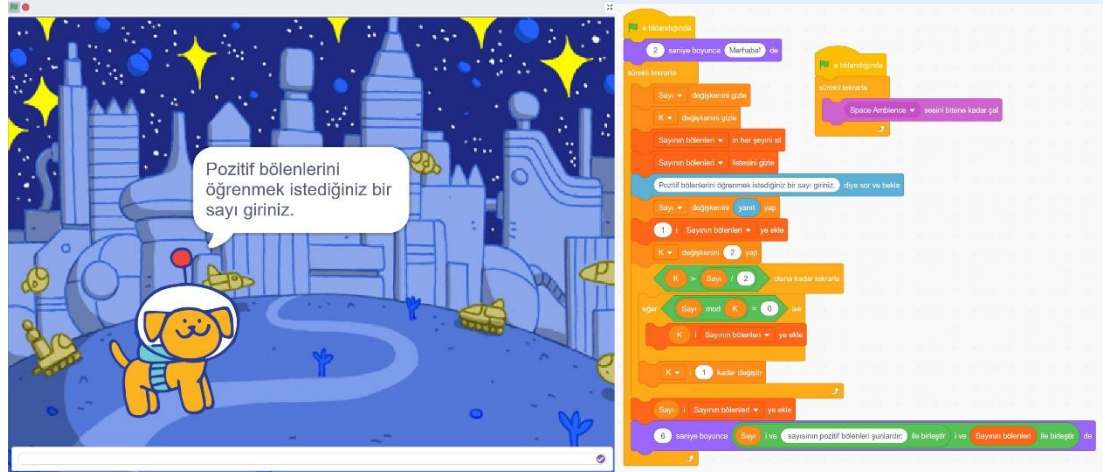
Şekil 47

Öntest (solda) sontest (sağda) öğrenci cevap kâğıdı örneği



Şekil 48

Pozitif bölen bulmayı içeren öğrenci kodlama etkinliği ve kod blokları örneği (Scratch sayfası)



4.2. Öntest ve Sontest Yordamsal Analiz Sonuçlarına Yönelik Bulgular

Veriler analiz edilirken ölççeklerin normalliklerini test etmek için 50’den küçük olan örneklemelere Shapiro-Wilk normallik test uygulanmıştır (Büyüköztürk, 2023). Shapiro-Wilk normallik testinin yanı sıra çarpıklık katsayısını ve basıklık katsayısını sırasıyla, çarpıklığın ve basıklığın standart hatasına böldüğümüzde, çıkan değerler -1,96 ile +1,96 arasında kalıyorsa, dağılımı normal kabul edilebilir (Büyüköztürk,

2023). Verilerin normal dağılımı ile bu sayıltı sağladıktan sonra öntest sontest sonuçları analiz edilirken bağımlı örneklem t testi uygulanmıştır. Tablo 18’de Shapiro-Wilk normallik değerleri verilmiştir. Örneklem 8 kişiden oluştuğu için Shapiro-Wilk testi dikkate alınmıştır. Öntest ($p=0,293>0,05$) ve sontest ($p=0,515>0,05$) bulunmuş ve verilerin normal dağılım gösterdiği görülmüştür. Öntest W 0,901 ve sontest W 0,930 değeri 1’e yakın olduğu için veriler normal dağılım göstermektedir. Öntest ve sontest arasındaki korelasyon değeri 0,808 ve $p= 0,015<0,05$ olduğundan sayıltı sağlanmıştır.

Tablo 18

Shapiro–Wilk normallik testi

Test	Shapiro-Wilk		
	İstatistik	sd	p
Öntest	,901	8	,293
Sontest	,930	8	,515

Tablo 19’da bağımlı örneklem t-testi analiz sonuçları ve katılımcıların öntest ile sontesti arasındaki ilişkiye yönelik ortalama değerleri verilmiştir. Öntest ve sontest ortalama puanlarını karşılaştıran bağımlı örneklem t-testinde öntest ve sontest ortalamaları arasında istatistiksel olarak anlamlı bir ilişki olmadığını göstermektedir [$t_{(7)} = -1,063$; $p>0,05$]. Eşleştirilmiş farklar tablosu, t-değeri -1,063, sd = 7 ve $p=0,323$ olarak göstermektedir. $p > 0,05$, değeri için anlamlılık olmayacağından dolayı öntest ve sontest puanları arasındaki ortalamalar arasında istatistiksel olarak anlamlı bir fark olmadığını söyleyebiliriz. Örneklem küçük olmasından dolayı bağımlı örneklem t-testi uygulanabilmesi için normalliğin yeterli bir referans olup olmadığına yönelik tartışmalar nedeniyle (StackExchange, 2024) Wilcoxon test sonuçları da ayrıca Ek 4’te paylaşılmıştır.

Tablo 19

Bağımlı örneklem t-testi analiz sonuçları

Test	$\bar{X}$	ss	se	t	sd	p
Öntest	8,5	5,58058	1,97303	-1,063	7	0,323
Sontest	9,75	5,00714	1,77029			

4.3. Scratch ile Tasarlanmış Matematiksel Durumlara Yönelik Öğrenci Görüşlerine İlişkin Bulgular

4.3.1. Scratch ile Tasarlanmış Matematiksel Durumların Olumlu Yönlerine İlişkin Bulgular

Scratch ile matematiksel algoritmalar tasarlayan öğrencilerin yaptıkları kodlama etkinliklerinin olumlu yönlerinin neler olduğuna dair görüşlerini içeren temalar ve bu temalara ait kodlar Tablo 20’de sunulmuştur. Görüşmelere 6 öğrenci katılmıştır. Görüşmeler gönüllü öğrencilerle yapılmıştır.

Tablo 20

Scratch ile oluşturulan matematiksel algoritmaların olumlu yönleri

Tema	Kodlar (Alt Temalar)	Öğrenci	f	%
Yararlılık	Faydalı	K1, K6	2	33
	Konu eksiklerini giderme	K3	1	16
	Eğlenceli	K2, K4	2	33
	Gelecek için önemli görme	K1, K2, K5, K6	4	66
	Olumlu tutum	K1, K2, K3, K4, K5, K6	6	100
Gelişim	Derse yönelik ilgiyi arttırması	K5	1	16
	Başarma duygusu	K1, K2, K5	2	50
	Mantık ve muhakemeyi geliştirmesi	K6	1	16
	Deneyim kazanma	K4	1	16
	Özgüven artışı	K1	1	16
	Kodlama hakkında farkındalık	K4, K5	2	33
Meslek Seçimi	Yazılım Mühendisliği	K1, K6	2	33
	Matematik Öğretmenliği	K2	1	16
	Etkilemez	K3, K4, K5	3	50

Tablo 20'nin (devamı)

Disiplinler Arası	Kodlamanın farklı derslerde kullanımı	K2, K3, K4, K5	3	50
-------------------	---------------------------------------	----------------	---	----

Öğrencilerin görüşlerine göre Scratch ile tasarlanmış matematiksel algoritmaların öğrenciler açısından olumlu yönde tutum geliştirdikleri, etkinliği faydalı buldukları, derse yönelik ilgilerinin arttırdığı, kodlama yaparken eğlendikleri, başarıma duygularını geliştirdiği, kodlamayı geleceğin mesleklerine ait gördükleri, konu eksiklerini gidermede ve konuları pekiştirmede yardımcı olduğu, kodlama hakkında deneyim kazandıkları, yapılan etkinliğin kodlama konusunda özgüvenlerini arttırdığı ve kodlama hakkında farkındalık yarattığı yönünde görüşler ortaya çıkmaktadır. Ayrıca kodlamanın farklı derslerde de kullanımı hakkında derslerin sıkıcılığını gidereceğini belirtmişlerdir. Öğrencilerin konu hakkındaki görüşlerini içeren doğrudan alıntılar aşağıda verilmiştir.

K1: Kodlama eğitimi güzeldi. Zaten büyüyünce yazılımcı olmayı düşünüyorum yani bursu kazanırsam kendi paramı biriktirip bilgisayar almayı düşünüyorum kendi paramı biriktirip, bilgisayar alıp öğrenmeyi düşünüyorum. Demek ki yapabiliyormuşum hissi uyandırdı.

K2: Gayet eğlenceliydi. Bir şeyleri eğlenerek öğrenmek gayet güzeldi. Hem daha akılda kalıcı oldu. (Kodlamayı her öğrencinin sahip olması gereken bir beceri olarak görüyor musun?) evet görüyorum çünkü gelecekte daha teknolojik bir döneme gireceğiz. Matematik dersindeki başarımızı da etkiledi. Matematik önceden zor geliyordu. Şu anda biraz daha eğlenceli. (Kodlamanın farklı derslerde de kullanımı hakkında) Evet. Sıkıcı geçen derslerde kullanmak daha eğlenceli olabilir. Ders daha çok ilgi çekirdi.

K3: Çok eğlenceliydi. Eğitici oldu. Konuları pekiştirmemi sağladı. (Kodlamanın meslek seçimine ve eğitim hayatına etkisi hakkında) Ne diyeceğimi bilemiyorum. İleride öğretmen olursam bilgi sahibi olmuş olurum. Matematik öğretmeni olmak istiyorum. (Aldığınız kodlama eğitimi matematik dersine yönelik ilginizi etkilemesi hakkında) Dersi zaten seviyordum daha keyifli geçti. (Kodlamanın farklı derslerde de kullanımı hakkında) İsterim. İşe yarar. O dersleri de daha keyifli hale getirir.

K4: *Güzel bir etkinlik. Kodlama eğitimi önemli çünkü daha eğlenceli bir şekilde öğreniyoruz. (Kodlamanın meslek seçimine ve eğitim hayatına etkisi hakkında) Olabilir. Meslek tercihim etkileyeceğini düşünmüyorum ama eğitim hayatımı etkiler. Mesela lisede kodlama yaparsak bilinçli bir şekilde olurum.*

K5: *Olumlu anlamda bir etkisi olduğunu düşünüyorum. Biraz olsun zorlanmalar oldu ama. (Kodlama eğitiminin önemi hakkında) Yani önemli aslında. Bilgimizi arttırdı. (Aldığınız kodlama eğitimi matematik dersine yönelik ilgiyi etkilemesi hakkında) Zaten matematik dersini seviyorum ama başarıım düşük. Matematik başarıyı bir tık arttırdı. Derse sevgimi arttırdı. (Kodlamanın farklı derslerde de kullanımı hakkında) Kullanılmalı. İnkılap dersinde kullanılabilir. O dersi pek sevmiyorum. O dersi bana sevdirebilir.*

K6: *(kodlama eğitiminin) faydalı olduğunu düşünüyorum çünkü belirli mantık çerçevesinde cevap veriyorum bu da mantık yeteneğimizin pratiğini geliştiriyor. Mesela sizin verdiğiniz testte soruları daha bir benimseyerek yaptığımı fark ettim. Hisler bakımında da eğlenceliydi. Ben meslek açısından yazılım mühendisliği de düşünüyordum. O yüzden en azından birkaç bir şeyle katkısı olacağını düşünüyorum. (Kodlamayı her öğrencinin sahip olması gereken bir beceri olarak görüp görmemesi hakkında) Bu bir yetenekse her öğrenciden mükemmel başarıyı beklemek doğru değil elbette. Ama en azından eğitiminin verilmesi gerektiğini düşünüyorum. En azından deneyimlemeli. Belki çok yeteneklidir. Yeteneğinin keşfedilmesi açısından önemli olabilir. (Kodlama ile matematik arasındaki ilişki hakkında) Yani matematik formüllere dayanıyor. Kodlama da öyle bence. (Kodlamanın derse yönelik etkisine etkisi hakkında) Olumlu etkiledi. Kodlamadan sonra sorulardaki ne demek istediğini ve işlemler arasındaki bağlantıyı daha hızlı yaptığımı fark ettim. İşlemler arasındaki bağlantıyı daha hızlı kurduğum için özgüvenimi arttırdı.*

4.3.2. Scratch ile Tasarlanmış Matematiksel Durumların Olumsuz Yönlerine İlişkin Bulgular

Scratch ile matematiksel algoritmalar tasarlayan öğrencilerin yaptıkları kodlama etkinliklerinin olumlu yönlerinin neler olduğuna dair görüşlerini içeren temalar ve bu temalara ait kodlar Tablo 21’de verildiği gibi öğrenci görüşlerine göre Scratch ile Tasarlanan Matematiksel Algoritmalar uygulamasında internet kesintisi ve

uygulamanın kendisiyle alakalı yaşadıkları zorluklar aksayan yönler olarak belirtilmiştir. Öğrencilerin büyük bir kısmı ise herhangi bir zorluk yaşamadıklarını belirtmişlerdir. Ayrıca kodlamanın farklı disiplinlerde de kullanımı hakkında kararsız kalmışlar ve nasıl kullanılacağını kestiremediklerinden olumsuz cevap vermişlerdir. Öğrencilerin konu hakkındaki görüşlerini içeren doğrudan alıntılar aşağıda verilmiştir.

Tablo 21

Scratch ile oluşturulan matematiksel algoritmaların olumsuz yönleri

Tema	Kodlar (Alt Temalar)	Öğrenci	f	%
Alt yapı yetersizliği	İnternet kesintisi	K5	1	33
	Bilgi eksikliği	K2	1	33
Olumsuz Yön Yok		K1, K3, K5, K6	4	66
Disiplinler Arası	Kodlamanın farklı derslerde kullanımı	K1, K6	3	50

K1: (Kodlama yaparken karşılaştığı güçlükler hakkında) Güçlük yok. Basit. (Kodlamanın farklı derslerde de kullanımı hakkında) Yani özellikle kodlama yapmaya gerek yok.

K2: Kodlama yaparken bazen ne yapmam gerektiğini anlayamıyorum. Düşündüğüm şeyi kodlamaya nasıl döneceğimi bilemediğim için zorlanıyorum.

K3: (Kodlama yaparken karşılaştığı güçlükler hakkında) Yoktu. Bazı aşamaları sadece zordu. Kodları yerleştirme bazen zordu.

K4: (Kodlama yaparken karşılaştığı güçlükler hakkında) Şöyle, bazen kodlama yapıyoruz ve kaydetmiyor.

K5: (Kodlama yaparken karşılaştığı güçlükler hakkında) Çok zorlanmadım. Telefonda yapamadık. İnternette kesilmeler oldu. Kodlamalara baştan başlamak zorunda kaldık. Strese girdik.

K6: (Kodlama yaparken karşılaştığı güçlükler hakkında) Öyle çok aman aman güçlük yaşadığımı düşünmüyorum. (Kodlamanın farklı derslerde de kullanımı hakkında) Kodlamanın geliştirdiği şeylerle çok ilgisi olmayan derslere eklenebileceğini düşünmüyorum. Daha çok mantık, muhakeme, beceri gerektiren derslere yoğunlaştırabilir. Matematikte daha fazla mantık ve

muhakeme uyguladığınız için, işlemler arası bağlantıyı, matematiğe daha fazla etkisi olduğunu düşünüyorum.

4.3.3. Scratch ile Tasarlanmış Matematiksel Durumların Zorluklarına İlişkin Bulgular

Scratch ile matematiksel algoritmalar tasarlayan öğrencilerin yaptıkları kodlama etkinliklerinin anlaşılmayan yönlerinin neler olduğuna dair görüşlerini içeren temalar ve temalara ait kodlar Tablo 22’de sunulmuştur. Öğrencilerin tamamı “Kodlama atölyeniz var mı?” sorusuna “kodlama atölyesi” kavramını bilmedikleri için yanıt verememişlerdir. Ayrıca kodlamanın farklı disiplinlerde de kullanımı hakkında kararsız kalmışlar ve nasıl kullanılacağını kestiremedikleri olumsuz cevap vermişlerdir.

Tablo 22

Scratch ile oluşturulan matematiksel algoritmaların anlaşılmayan yönleri

Tema	Kodlar (Alt Temalar)	f	%
Kavram Eksikliği	Kodlama Atölyesi	6	100

Öğrencilerle yapılan görüşmeler sonucunda öğrencilerin büyük bir kısmının yapılan etkinlikleri eğlenceli bulduğu ve edindikleri kodlama bilgilerinin eğitim hayatlarına olumlu anlamda katkı sunabileceğine yönelik inanç geliştirdiği görülmektedir. Öğrencilerin bir kısmı kod bloklarının işlevlerini tam anlayamadıkları ya da projelerini kaydetme gibi olumsuzluklar yaşasalar da etkinlikleri yaparken çoğunlukla güçlük yaşamadıklarını belirtmişlerdir. Öğrencilerin tamamı etkinliklere katılsa da her projenin öğrenciler tarafından ya yarım bırakılarak ya da hiç başlamaksızın tamamlanmadığı görülmüştür. Öğrencilerin etkinlikleri tamamlama yüzdeleri Tablo 23’te gösterilmiştir.

Tablo 23*Etkinliklerin tamamlanma oranları*

Etkinliği Adı	Tamamlama	%
Soru- Cevap	4	50
Asal Mı?	6	75
Pozitif Tam Sayı Bölenleri	8	100
Aralarında Asal Mı?	5	62,5
EBOB ve EKOK Hesaplama Programı	6	75

Tablo 23'te verilenlere göre öğrencilerin tamamının pozitif bölen sayısı kodlama etkinliğini tamamladığı, en az tamamlanma oranının ise soru cevap etkinliği olduğu görülmektedir. Bu durumun soru-cevap etkinliğinin yapılan ilk kodlama etkinliği olmasına bağlı öğrencilerin kod bloklarını ve uygulamanın arayüzü ile yeni tanışmış olmalarına bağlı yaşadıkları kaydetme vb. problemlerden kaynaklanabileceği düşünülmektedir. Ayrıca bazı projelerin öğrenciler tarafından yarım bırakılmış olması öğrenci tarafından etkinliğin tamamlanmış olmasını zannetme gibi bir durumdan da kaynaklanmış olabileceği düşünülmektedir. Hiç başlanmayan etkinliklerin ise öğrenci tarafından hiç anlaşılamamış veya başlaması için gerekli motivasyonun sağlanamamış olması gibi sebeplerden kaynaklanıyor olabileceği düşünülmektedir.

BÖLÜM V

TARTIŞMA, SONUÇ VE ÖNERİLER

5.1. Sonuç ve Tartışma

Taylor'un (2019) programlamaya özgü problem çözme stratejisi ile bağlam sallastırdığımız kodlama etkinliklerinin Polya (1945) tarafından önerilen genel aşamalarla uyumlu olmadığını göstermektedir. Öğrenciler problemi tanımlama aşamasında çoğu zaman sadece dekorlar ve kuklalarla ilgilenmişler; K1 ve K2 hariç çoğunlukla bir plan tasarlamamışlardır. Bunun yerine çoğunlukla kodun çalışıp çalışmamasıyla ilgilenmişler ancak kodları geliştirmekle pek ilgilenmemişlerdir. Öğrenciler çoğunlukla (K1 hariç) yalnızca etkinliklerde sorulacak soruları üretme düzeyinde faaliyet göstermişlerdir. K1, yaptığı etkinliklerin bazılarında “haber sal” bloğu ile daha dallanmış ve bağımsız algoritmalar geliştirirken kalan öğrenciler yukarıdan aşağı doğrusal tasarımlar kullanmayı tercih etmişlerdir. K1'in LGS sınavında Fen Lisesi'ne yerleşmiş olması dikkat çekicidir çünkü bu sonuç K1'in daha farklı ve özgün tasarımlar ortaya koymasının Pea ve Kurland (1984) derin alan bilgisinin problem çözme stratejilerini etkilediği; öğrencilerin programlamadaki yeteneklerinin ve bunu yeni durumlara aktarma kabiliyetlerinin sahip oldukları beceri türleriyle yakından ilişkili olduğu bulgusu ile uyumludur. Bu bulgular, öğrencilerde kodlama süreçlerinin, daha derin ve bilişsel açıdan karmaşık problem çözme becerilerini geliştirmekten ziyade, önceden belirlenmiş talimatları takip etme becerisini ön plana çıkardığını göstermektedir. Benzer şekilde, bazı aşamalarda öğrencilere hazır olarak verilen kod bloklarının ve bunun bir sonucu olarak kodu kopyalama ve yapıştırma işleminin nihai ürünlerde gözle görülür bir olumsuz etkiye yol açmamasına rağmen, bu yöntemin öğrencilerin kodlarını ne kadar anladıklarını sorgulatmaktadır. Öğrencilerin uyguladıkları problem çözme stratejilerinin ürettikleri farklı alanlara her zaman verimli bir şekilde yansımadağı görülmektedir. Birçok proje başarılı ve tutarlı kodlamalar olarak nitelendirilmiştir fakat yine pek çoğu kod

kopyalama gibi düşük seviyeli stratejilerle oluşturulmuştur. Bu nedenle, kodlama görevlerinde, en etkileyici veya görsel açıdan çekici projelerin bile, teknik yeterlilik veya sistematik kompütasyonel düşünme ve problem çözme becerisi gerektirdiğini varsaymak hata olacaktır. Öğrencilerin kodlama davranışlarını matematiksel problem çözme açısından inceleyerek, kodlama yaparken bir problemi tanımlama, uygulama ve değerlendirme ile problem çözme seviyelerinin nasıl ilişkili olduğunu araştırdığımız sonuçlar, kodlama projelerinde kod kopyalama faaliyetlerinin baskın olduğunu gösterdi. Öğrencilerin kodları alıntılarla ve karıştırarak ilerlemesi bir kodlama faaliyeti olsa da eleştirel ve yaratıcı düşünme, soyutlama veya problem ayrıştırma gibi kompütasyonel düşünme becerilerini kullanmaya teşvik etmez. Bu durum, öğrencilerin kompütasyonel düşünmede gelişme gösterememesinin sebebini açıklayabilir. Kodlama görevlerinin problem çözmeyi nasıl destekleyebileceğini göstermek için seçilen Scratch projelerinin geçerliliği de sorgulanabilir.

Öntest ve sontest karşılaştırmalı sonuçlarına göre öğrencilerin öntest ve sontest sonuçları arasında anlamlı bir fark olmadığı görülmektedir. Denning'in (2017), kompütasyonel düşünme becerisini de içeren beceri kavramının, zaman içinde pratikle kazanılan bir yetenek olduğunu; olgu veya bilgi bilgisi olmadığını, bilgi birikimlerinin test edilmesinin öğrencilerin kompütasyonel düşünme yetkinliklerini ya da duyarlılıklarını test etmede yeterli olmadığını ve bu sebeple, testlerde iyi puan alan bir öğrencinin yine de beceriksiz veya duyarsız bir algoritma tasarımcısı olması mümkün olacağı sonucuyla örtüşmektedir. Kompütasyonel düşünmenin tanımının hala net bir şekilde ortaya konulamaması, kompütasyonel düşünme becerisinin nasıl ölçüleceğinin belirsizliği ve kompütasyonel düşünmenin herkes için faydalı olup olmadığı, kompütasyonel düşünmenin problem çözme becerilerine katkısı, kompütasyonel düşünme biçimini tanımlayan kavramlar konusundaki belirsizlikler nedeniyle etkililiği sorgulanmaktadır (Denning, 2017). Okullara bilişimsel bilginin katkılarından ve başarılarından yana bir eleştiri söz konusu değildir ancak bilgisayar bilimlerinin gereğinden fazla anlam yükleyip, karşılanamayacak beklentiler yaratıp; öğretmenlerin tam olarak ne öğretmeleri gerektiği ve bu öğretileri nasıl değerlendirmeleri gerektiği konusundaki belirsizlikler kompütasyonel düşünmeyi kazandırmaya yönelik çabaları baltalamaktadır (Denning ve Tedre, 2019). Kompütasyonel düşünmenin öğretim ve değerlendirme sürecindeki belirsizliklere rağmen bilgisayar bilimleri olgunlaştıkça, kompütasyonel düşünce başarılı bir şekilde diğer bilimlerle bütünleşmiş ve çoğu bilim

dalı artık kendi alanlarındaki bilgi süreçlerini değerlendirmeye başlamışlardır (Easton, 2006).

Sekiz hafta boyunca verilen kodlama eğitiminde öğrencilerin zaman zaman zorlanması, hatta yer yer bazı blokların ve bu bloklara ait işleyişlerinin ve bütündeki işlevlerinin öğrenciler tarafından anlaşılmakta zorlanması, kodlama araçlarının sağladığı olanaklara rağmen, kodlamanın doğası gereği bilişsel olarak karmaşık süreçleri içermesi nedeniyle algoritmaların kendiliğinden ortaya çıkamayabileceği görülmektedir. Bu nedenle yeni bir bilginin çeşitli proje tabanlı etkinlikler aracılığıyla öğrenciye sunulduğunda beklenen etkinin zaman içinde ortaya çıkacağı pedagojik açıdan popüler bir yaklaşımdır (Fagerlund vd., 2021). Franklin vd. (2017), öğrencilerin bir kavramı genellikle bir yılda tamamen öğrenemediğini, tıpkı ilkokulda üç yıl boyunca sayıları toplamayı öğrenmeleri gibi, bilgisayar konularını da her yıl daha karmaşık ve derinlemesine maruz kalmaları gerektiğini ve öğrencilerin öğrendiği temel bilgisayar kavramlarının birkaç sınıf boyunca öğrencilerde nasıl geliştiğinin takip edilmesi gerektiğini vurgulamışlardır. Sonuçlarda anlamlı bir fark görülmemesinin nedeni bu bağlamda öğrencilerin daha önce hiç kodlama etkinlikleri yapmamaları, Scratch ile ilk kez tanışmaları, daha önce programlama yapmamış olmaları ve bunun neticesinde kompütasyonel düşünme becerisini tam olarak kazanamamış oldukları söylenebilir. Bu becerinin gelişmemiş olmasıyla beraber geliştirmesini beklediğimiz ve üst bilişsel bir beceri olan problem çözme becerilerine bir katkı sunmadığı görülmektedir. Öğrencilerin sekiz haftalık kısa uygulama sürecinde algoritma veya akış şemalarının nasıl oluşturulacağını anlamamış olabilecekleri düşünülmektedir. Ayrıca sonuçlar arasında anlamlı bir fark olmamasının bir nedeni de grubun küçük olmasından kaynaklandığı düşünülmektedir.

Öğrencilerin bilişimsel yeteneklerinin ölçülmesi ve kompütasyonel düşünme becerilerinin değerlendirilmesi ise yine bu kapsamda değerlendirilebilir. Kompütasyonel düşünme, İngiliz Bilgisayar Topluluğu (CAS) tarafından bir beceri olarak görülüyor (Csizmadia vd., 2015). Ancak bu beceriyi neyin oluşturduğu konusunda bir fikir birliği yok ve mevcut değerlendirme yöntemleri de güvenilirmez göstergeler içeriyor (Denning, 2017).

Scratch ile yapılan kodlama etkinlerinin problem çözme becerileri üzerine etkisini inceleyen Oluk ve Saltan (2015), 6. sınıf öğrencileri ile Scratch kullanımlarının problem çözme becerileri üzerine yaptıkları altı haftalık çalışmada öğrenme çıktıları

açısından deney ve kontrol grubu arasında anlamlı bir fark bulamamışlardır (Oluk ve Saltan, 2015). Kalelioğlu vd. (2014), Scratch'in problem çözme becerilerini önemli ölçüde etkilemediğini belirtmişlerdir. Voskoglou ve Buckley (2012), lisansüstü öğrencilerinin kompütasyonel düşünme ile problem çözme becerileri arasında anlamlı bir fark bulunup bulunmadığına araştırmada deney grubuna, öğrencilere matematiksel yazılımların öğretilmesinden sonra, bilgisayar destekli ortamlarda ilgili matematik konularını göstermişlerdir. Dönem sonunda yapılan final sınavında iki grup arasında anlamlı bir fark bulunamamıştır. İlgili konuların programlama ile gösterilmesi ve bu yöntemle problem çözme becerilerinin gelişiminin belirlenmesi açısından araştırmamıza benzerlik göstermektedir. Burada en önemli fark kullanılan programlama araçları ve elbette katılımcıların yaşları arasındaki farktır. Ayrıca Voskoglou ve Buckley (2012), sonuçlarını değerlendirirken final sınavında öğrencilere konularla alakalı teorik soruların yanında matematiksel modelleme içeren gerçek hayat problemleri de sormuşlardır. Teorik sorularda iki grup arasında anlamlı fark bulamayan Voskoglou ve Buckley (2012), gerçek hayat problemlerinde deney grubunun cevaplarını önemli ölçüde daha iyi bulmuşlardır. Araştırmamızın sonucu Voskoglou ve Buckley'in (2012) sonuçlarının teorik kısmını desteklemektedir. Burada öğrencilerin yaşından, eğitim durumlarından ve kullanılan programla aracından bağımsız olarak, matematik derslerine programlamanın bütünleşmiş olması bunun ne derecede başarılı olduğunu ölçülmesi konusunda alan yazındaki araştırmalara yetersizdir.

Öğrencilerin Scratch üzerinde yaptıkları kodlama etkinliklerinde kukla seçmek, kuklanın görünümünü değiştirme, sahne arka planı oluşturmak, tek bir kukla için diyaloglar oluşturmak, sahne arka planı için müzik, ses vb. efektler eklemek, olay başına sadece bir eylem bloğu kullanmak gibi görev setlerini yerine getirirken neredeyse hiç zorlanmadıkları görülmüştür. Öğrencilerin örneğin, bir sayının bölenlerini hesaplarken kullandıkları operatörlerin işlevleri, birden fazla işi bir araya getirmek için kullandıkları kontrol blokları ve farklı işleri tek bir eylemde bir araya getirmek için kullandıkları olay bloklarının öğrenciler için oldukça zorlayıcı olduğu görülmüştür.

Bu sonuç, Franklin vd. (2017) için bir sonuç olan yoğun matematiksel hesaplamalar içeren etkinliklerin istenmeyen zorluklara yol açabileceğine ve bazı öğrenciler için temel bilgisayar bilimi kavramlarıyla etkileşime girme konusunda potansiyel olarak bir engel oluşturabileceğine; öğrencilerin kesin matematiksel

hesaplamalar kullanmaları gerektiğinde, yaşı daha küçük olan öğrencilerin diğer öğrencilerden daha kötü performans gösterdiğine; arzu edilen bilgisayar bilimi içeriğine istenmeyen bir engel getirebileceğinden ötürü etkinlikler için gereken harici matematiği en aza indirmeye çalışmak gerektiğine yönelik bulgularıyla örtüşüyor.

Kompütasyonel düşünme becerilerinin öğretim programına dahil edilmesinin etkinliğini değerlendirilebilmesi için kompütasyonel düşünme ölçekleri kullanılmalıdır (Román-González vd., 2017) ancak kompütasyonel düşünme becerisinin, çağdaş dünya için giderek daha fazla kabul gören temel bir yetkinlik olmasına rağmen, akademik performansla olan ilişkisi yeterince anlaşılamamıştır (Doleck vd., 2017). Kompütasyonel düşünmenin değerlendirilmesi (Brennan ve Resnick, 2012), kompütasyonel düşünmede programlamanın rolü (Lu ve Fletcher, 2009), öğrencilerin kompütasyonel becerilerinin geliştirilmesi (Atmatzidou ve Demetriadis, 2016), kompütasyonel düşünme becerisini geliştirmek için geliştirilen araçlar (Grover ve Pea, 2013), kompütasyonel düşünmenin öğretim programında yer edinmesi (Lye ve Koh, 2014) gibi kompütasyonel düşünme ile ilgili yapılan tüm bu araştırmalara rağmen kompütasyonel düşünme ile akademik performans arasındaki ilişkiyi inceleyen sınırlı sayıda araştırma bulunmaktadır (Doleck vd., 2017). Araştırmamızın alan yazına bu anlamda katkı sunduğunu düşünmekteyiz.

Kiss ve Akri (2017), geleneksel öğretim stratejilerinin kodlama ve problem çözme becerileri için gerekli kavramsal çerçeveyi geliştirmek için uygun olmadığı iddia etmişlerdir (Kiss ve Arki, 2017). Doleck vd. (2017) ise işbirlikli öğrenme dışında kompütasyonel düşünme ile akademik performans arasında bir ilişki bulamadıklarını, işbirliğinin ise akademik performansla negatif ilişkili olduğu belirtmişlerdir (Doleck vd., 2017). Bu sonuç, Scratch ile geliştirilen matematiksel algoritmaların öğrencilerin akademik performansları yani problem çözme becerileri üzerine etkisi olmaması açısından benzerdir. Kompütasyonel düşünme becerisi ile akademik performans arasında ilişkinin olmaması sadece kompütasyonel becerinin ölçülmesindeki belirsizliklerin değil, aynı zamanda mevcut öğretim programının problem çözme becerilerinin açıkça öğretilip öğretilmediği konusunda da soru işaretleri doğurmaktadır (Doleck vd., 2017). 21. Yüzyıl becerilerinden olan kompütasyonel düşünme becerileri ile mevcut öğretim programının daha uyumlu hale getirilmesi gerekmektedir.

Öğrencilerin çoğu Scratch ile programlama yapmaya ve bunu matematik dersinde kullanmaya yönelik olumlu tutum sergilemişlerdir. Korkmaz (2016), Scratch tabanlı eğitim etkinliklerinin, geleneksel yöntemlere kıyasla, öğrencilerin programlama eğitimine yönelik tutumları üzerinde herhangi bir etkisi olmadığı bulmuştur. Genç ve Karakuş'un (2011) Scratch kullanımına yönelik öğrencilerin geliştirdiği tutumun olumlu olduğu sonuçları ile (Erol ve Çırak, 2022; Genç ve Karakuş, 2013) örtüşmekte; Korkmaz'ın (2016) bulgularıyla örtüşmemektedir. Ayrıca araştırmamızda öğrencilerin kodlama eğitimini ayrı bir ders olarak almanın gereksiz olduğunu belirtmişlerdir. Bu ifade Korkmaz'ın (2016) öğrencilerin programlama eğitimini gereksiz bulmaları ile kısmen örtüşmektedir.

Öğrencilerin öğrenim hayatlarını ve ilerleyen zamanda meslek seçimlerine yönelik algılarının nasıl etkilendiği sorusu üzerine kesin konuşamadıkları, pozitif bir etki hissettikleri ancak bu etkinin eğitim-öğretim hayatlarına nasıl etki edeceğini ve meslek seçimlerini ise büyük çoğunluğun etkilemeyeceği yönünde cevaplar aldık. Ioannidou vd. (2011), ITEST programı kapsamında Ulusal Bilim Vakfı tarafından finanse edilen iDREAMS projesi¹ (Integrative Design-based Reform-oriented Educational Approach for Motivating Students), Ölçeklenebilir Oyun Tasarımı adı verilen bir yaklaşımla ortaokul düzeyinde bilgisayar bilimlerine ilgiyi artırarak kompüsyonel düşünmenin iş gücü üzerindeki potansiyel etkiyi nasıl etkilediğine dair araştırmalarında çoğu öğrencinin ve özellikle ortaokul seviyesindeki kızların programlamaya ve bilgisayar bilimlerinin programlamadan ibaret olduğu inancına yönelik bakış açısının, bilgisayar bilimlerine sınırlı ilgi duymasının ana nedenlerinden biri olduğunu ifade etmiştir. Öğrencilerin meslek seçimini etkilemeyeceği yönündeki görüşleri Ioannidou vd. (2011) bu yöndeki görüşlerini destekleyebilir.

Repenning (2006), ölçeklenebilir oyun tasarımlarının öğrencilerde başarıma hissi ve bilgisayar bilimleri ve FeTEMM (STEM) içeriğine geçiş yapmalarına olanak tanıdığını ifade etmiştir. Öğrencilerin bazılarında etkinlikleri tamamladıktan sonra yapabildiklerini gördüklerini ve bu durumun onları gelecekte yazılım mühendisi olma isteklerini arttırdığına yönelik bulgularımızla örtüşmektedir. Bakır (2023), deney grubunda Scratch'in algoritmik, yaratıcılık ve kritik düşünme becerilerinde fark oluştururken deney ve kontrol grubu arasında işbirlikli ve problem çözme becerileri açısından anlamlı bir fark bulunmadığını ifade etmiştir. Bu sonuç problem çözme

becerileri üzerinde anlamlı bir fark oluşturmaması açısından bizim bulgularımızla örtüşmektedir.

1990'lardan beri bilgisayar bilimleri eğitiminde öğrenci motivasyonunu artırmak için yapılan araştırmalar başarıya ulaşmıştır (Werner vd., 2012). Fakat bu araştırmaların öğrenci başarısına etkisi bilinmemektedir (Lye ve Koh, 2014). Karşılaşılan problemlerden biri sınıf içi öğrenmeyi ölçmek için uygun araç eksikliğidir (Lye ve Koh, 2014). Mevcut araçlar, örneğin dereceli puanlama anahtarları, öğrenmeyi ölçmek için kullanılabilir. Fakat bu araçlar çok zaman alıcıdır ve öğrenme ilerlemesini takip etme ve öğrencilere ve öğretmenlere sürekli geri bildirim sağlama konusunda sınırlı işlevselliğe sahiptir (Lye ve Koh, 2014). Öğrencilerle yaptığımız görüşmelerimiz ve yapılan öntest ve sonteste yönelik bulgularımız bu ifadeleri desteklemektedir. Scratch ile kodlanan matematiksel algoritmaların öğrencilerin derse yönelik motivasyonlarını ve ilgilerinin arttırdığını söyleyebiliriz ancak mevcut ölçme ve değerlendirme araçlarının, öğrencilerin neyi ne ölçüde başardıklarını ölçme noktasında doğru yargılara varmada yetersiz kaldığı ortadadır. Bu durum, öğrencilerin öğrenme potansiyellerini tam olarak gerçekleştirmelerini engelleyebilir.

5.2. Öneriler

Dersler ve yapılan etkinlikler için biçimlendirici değerlendirme yöntemleri araştırılmış ve alan yazında bazı yöntemler geliştirilmiş olsa da öğrencilerin Scratch gibi programlama süreçlerinde edindikleri kompütasyonel düşünme becerilerinin düzeyini değerlendirmenin pedagojik olarak anlamlı yolları üzerinde daha fazla araştırmaya ihtiyaç vardır. Onaylanmış değerlendirme çerçeveleri, potansiyel olarak öğrencilerin programlama yaparken yararlanabilecekleri otomatik biçimlendirici öğrenme sistemleri geliştirilmelidir. Odak noktası etkinlik, öğrenme ortamı veya programla olan az sayıda araştırmanın olduğu çoğunluğunun bilgisayar ve öğretim teknolojileri alanında yer aldığı, kompütasyonel düşünme becerisinin geliştirilmesine yönelik alan yazında oldukça az sayıda araştırma olduğu görülmektedir (Elçiçek, 2020). Kompütasyonel düşünme becerisinin bilgisayar ve teknolojiye yararlanarak problem çözme becerisi olmasından (Gretter ve Yadav, 2016) yola çıkarak farklı alanlarda yapılacak araştırmaların kompütasyonel düşünme becerisinin ve dolayısıyla öğrencilerin problem çözme becerilerinin geliştirilmesine önemli katkılar sunacaktır.

Öğrencilerin problem çözme becerilerinin Scratch üzerinden daha uzun sürede geliştirilmesine yardımcı olacak etkinlikler tasarlanabilir. Öğrencilerin kodlama ve kompütasyonel düşünme becerilerini geliştirebilmek için bilgisayarlar, tabletler ve telefonlar gibi cihazlara ve bir robotik sınıfı veya bilgisayar laboratuvarı ihtiyaç duyulmaktadır. Araştırmacılara uygun koşullarda öğrenme ortamları oluşturmaları önerilmektedir. İlk ve orta öğretimde algoritmik ve kompütasyonel düşünmenin mevcut öğretim programına entegrasyonu ve akademik başarıya etkilerinin değerlendirilmesi için stratejik bir odaklanmaya ihtiyaç vardır.

Örnekleminin sadece bir ortaokuldan alınmış olması gibi sınırlılıklar bulguların genellenebilirliği konusunda doğal bir endişe yaratmaktadır. Bu nedenle, benzer araştırmaların farklı bağlamlarda (örneğin, farklı okul seviyeleri ve öğrenci branşları) yapılması, incelenen ilişkilerin değişkenliğini daha iyi anlamamıza yardımcı olacağı düşünülmektedir. Araştırmada öntest not ortalamasını kontrol değişkeni olarak kullanıldı. Gelecekteki araştırmalarda diğer belirgin kontrol değişkenlerini de hesaba katılabilir. Ayrıca, mevcut bulgular araştırma grubunun az ve çalışmanın kesitsel olmasından ötürü nedensel çıkarımlara izin vermemektedir.

Kompütasyonel düşünme becerilerinin öğretimi ile akademik performans arasındaki bazı nedensel bağlantıları araştıran bazı deneysel çalışmalar yapılmıştır (Lockwood ve Mooney, 2017), ancak bu araştırma alanı henüz emekleme aşamasındadır ve bu nedenle kompütasyonel düşünme becerileri ile akademik performans arasındaki ilişkinin sınırlarını belirlemek için daha fazla araştırmaya ihtiyaç vardır. Öğrencilerin kodlama deneyimlerinin, işbirlikli öğrenme, eleştirel düşünme, algoritmik düşünme ve yaratıcı düşünme becerileri gibi kompütasyonel düşünmenin diğer bileşenlerine olan etkileri kontrol edilmemiştir ve bu nedenle bulgular potansiyel olarak etkilenmiş olabilir.

Scratch ile yapılan kodlama etkinlikleri ve kompütasyonel düşünmenin alt basamağı olan problem çözme becerisinin öğretim programında yer alan mevcut ölçme ve değerlendirme araçları kullanılarak ölçülmesi ile sınırlıdır. Bu kısıtlı kapsam, göze çarpan diğer kompütasyonel düşünme becerilerinin araştırmada göz ardı edilmiş olma ve farklı ölçme teknikleri ile daha verimli ölçülmesi ihtimallerini açık bırakmaktadır. Bu alanda yapılacak olan araştırmalar kompütasyonel düşünme becerisini geliştirmeye yönelik araştırmaların, öğrencilerin öğrenme çıktıları üzerinde herhangi bir etkisi olup olmadığını belirlemeye odaklanabilir.

Her ne kadar blok tabanlı programlama, matematik öğretim programında zorunlu bir unsur olmasa da B2C3Math gibi yaklaşımlar matematik öğrenimini tamamlayıcı ve etkili bir şekilde destekleyebilir. Araştırmalar, B2C3Math'in tüm matematik alanlarına uygulanmasının gerekli olmadığını, ancak belirli alanlarda faydalı olabileceğini gösteriyor. Bu da B2C3Math'in öğretmen eğitimine bütünleşmiş edilmesi için bir gerekçe oluşturuyor. Öğretmen adayları ile yapılan araştırmalar (Ateşkan ve Hart, 2021; Gleasman ve Kim, 2020) öğretmenlerin, programlamanın matematiği öğretmek için kullanılabilecek etkili bir yöntem olduğuna inanıyor. Teknolojinin ve bilgisayarlı düşünmenin matematiği farklı şekilde öğretme ve öğrenmenin bir yolu olarak görülmesi, kodlama temelli matematik öğretiminin potansiyelini ve önemini vurguluyor.

Ayrıca çalışmanın daha sağlıklı sonuç verebilmesi için ileride yapılacak çalışmalarda müdahalenin etkisini izole etmek için deney ve kontrol grubu içeren karşılaştırma yapılabilir. Rastgele atama ve öntest- sontest tasarımından farklı bir tasarım kullanılabilir. Yakınsak ve ıraksak geçerliliği değerlendirmek için matematik problem çözme becerilerini ölçen başka bir test eklenebilir. Daha güvenilir sonuçlar için daha fazla katılımcı dâhil edilebilir. Sorular hazırlanırken alanında uzman kişilerin yardımıyla test soruları hazırlanabilir.

KAYNAKÇA

- ACARA. (2023). Australian Curriculum. 2023, 10 Haziran tarihinde <https://www.australiancurriculum.edu.au/download/> adresinden erişilmiştir.
- Aho, A. V. (2012). Computation and computational thinking. *The computer journal*, 55(7), 832-835. <https://doi.org/10.1093/comjnl/bxs074>
- Aksu, S. G. ve Sürgevil, O. (2019). Dijital Çağın Yetkinlikleri: Çalışanlar, İnsan Kaynakları Uzmanları Ve Yöneticiler Çerçevesinden Bakış. *Business in The Digital Age Dergisi*, 2(2), 54-68.
- Akyol, A. (2022). *Scratch 3.0 ile Kodlama ve Yapay Zeka Uygulamaları*. 2023, Haziran 20 tarihinde <https://www.udemy.com/course/scratch-3-0-ile-kodlama-ve-yapay-zeka-uygulamalar/> adresinden erişilmiştir.
- Alp, G. (2019). *Scratch programı ile web destekli işbirlikli öğrenme yönteminin ilkokul 5. sınıf öğrencilerinin kavramsal anlama düzeylerine ve eleştirel düşünme becerilerine etkisi* (Tez no. 543479) [Yüksek Lisans, Bursa Uludağ Üniversitesi].
- Altun, M., Hazar, M. ve Hazar, Z. (2016). Investigation of the Effects of Brain Teasers on Attention Spans of Pre-School Children. *International Journal of Environmental and Science Education*, 11(15), 8112-8119.
- Amador, J. M. ve Soule, T. (2015). Girls build excitement for math from Scratch. *Mathematics Teaching in the Middle School*, 20(7), 408-415.
- Anderson, R. C., Spiro, R. J. ve Anderson, M. C. (1978). Schemata as scaffolding for the representation of information in connected discourse. *American Educational Research Journal*, 15(3), 433-440. <https://doi.org/10.3102/0002831201500343>
- Ateşkan, A. ve Hart, D. O. (2021). Demystifying computational thinking for teacher candidates: A case study on Turkish secondary school pre-service teachers.

Education and Information Technologies, 26(5), 6383-6399.
<https://doi.org/10.1007/s10639-021-10626-9>

Atmatzidou, S. ve Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, 75, 661-670.
<https://doi.org/10.1016/j.robot.2015.10.008>

Bakır, E. E. (2023). *Blok Tabanlı Scratch Eğitimi ve Uygulamalarının Okul Öncesi Öğretmen Adaylarının Bilimsel Yaratıcılık ve Bilgi İşlemsel Düşünme Becerilerine Etkisinin İncelenmesi* (Tez no. 805462) [Yüksek Lisans, Kastamonu Üniversitesi].

Bandura, A. (2001). Social Cognitive Theory: An Agentic Perspective. *Annual Review of Psychology*, 52(1), 1-26. <https://doi.org/10.1146/annurev.psych.52.1.1>

Banzato, M. ve Tosato, P. (2020). Gender Difference in Self-Efficacy for Computational Thinking in K-12. In A. Tatnall (Ed.), *Encyclopedia of Education and Information Technologies* (pp. 827-846). Springer.
https://doi.org/10.1007/978-3-030-10576-1_300037

Bar-On, E. (1986). A programming approach to mathematics. *Computers & Education*, 10(4), 393-401. [https://doi.org/10.1016/0360-1315\(86\)90015-1](https://doi.org/10.1016/0360-1315(86)90015-1)

Barr, V. ve Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *Acm Inroads*, 2(1), 48-54. <https://doi.org/10.1145/1929887.1929905>

Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C. ve Marshall, K. S. (2011). Recognizing Computational Thinking Patterns. In T. J. Cortina, *SIGCSE '11: Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 245-250). The 42nd ACM Technical Symposium on Computer Science Education. Dallas, ABD. <https://doi.org/10.1145/1953163.1953241>

Batista, E. J. S., de Castro, C. P. C. ve de Castro, A. A. (2016). Development of learning objects on scratch: Training of teachers of information and communication technologies. *Latin American Conference on Learning Objects*

and Technology (pp. 1-5). IEEE. XI Latin American Conference on Learning Objects and Technology (LACLO). San Carlos, Kosta Rika.
<https://doi.org/10.1109/LACLO.2016.7751768>

Ben-Ari, M. (1998). Constructivism in computer science education. *ACM SIGCSE Bulletin*, 30(1), 257-261. <https://doi.org/10.1145/274790.274308>

Bennett, V. E., Koh, K. H. ve Repenning, A. (2011). CS education re-kindles creativity in public schools. In G. Rößling, *ITiCSE '11: Annual Conference on Innovation and Technology in Computer Science Education* (pp. 183-187). ACM. Proceedings of the 16th annual joint conference on Innovation and technology in computer science education. Darmstadt, Almanya.
<https://doi.org/10.1145/1999747.1999800>

Blackwell, A. F. (2002). What is programming? *Proceedings of the 13th Annual Workshop of the Psychology of Programming Interest Group, (PPIG)* (pp. 204-218). 14th Workshop of the Psychology of Programming Interest Group. Londra, Birleşik Krallık.

Bo, L., Xiaobin, Z. ve Ning, W. (2017). Teaching evaluation for computational-thinking-oriented elementary computer courses. In J. Lui, *Association for Computing Machinery* (pp. 1-6). ACM. Proceedings of the ACM Turing 50th Celebration Conference Shanghai, Çin.
<https://doi.org/10.1145/3063955.3063967>

Brennan, K. ve Resnick, M. (2012). *New Frameworks for Studying and Assessing the Development of Computational Thinking* [White paper]. Harvard.
<http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf>

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S. ve Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research*, 87(4), 834-860. <https://doi.org/10.3102/0034654317710096>

Bull, G., Garofalo, J. ve Hguyen, N. R. (2020). Thinking about computational thinking: Origins of computational thinking in educational computing. *Journal*

of Digital Learning in Teacher Education, 36(1), 6-18.
<https://doi.org/10.1080/21532974.2019.1694381>

Büyüköztürk, Ş. (2023). *Sosyal Bilimler için Veri Analizi El Kitabı İstatistik, Araştırma Deseni SPSS Uygulamaları ve Yorum*. Pegem Akademi.

Büyüköztürk, Ş., Kılıç-Çakmak, E., Akgün, Ö., Karadeniz, Ş. ve Demirel, F. (2008). *Eğitimde Bilimsel Araştırma Yöntemleri*. Pegem Akademi
<https://doi.org/10.14527/9789944919289>

Byrne, P. ve Lyons, G. (2001). The effect of student attributes on success in programming. In S. Fincher, *Innovation and Technology in Computer Science Education* (pp. 49-52). SIGCSE. ITICSE01: The 6th Annual Conference on Innovation and Technology in Computer Science Education. Canterbury, Birleşik Krallık. <https://doi.org/10.1145/377435.377467>

Calder, N. (2010). Using Scratch: An integrated problem-solving approach to mathematical thinking. *Australian Primary Mathematics Classroom*, 15(4), 9-14.

Chang, Z., Sun, Y., Wu, T.-Y. ve Guizani, M. (2018). Scratch analysis Tool (SAT): A Modern Scratch Project Analysis Tool Based on ANTLR to Assess Computational Thinking Skills. *Wireless Communications & Mobile Computing Conference (IWCMC)* (pp. 950-955). IEEE. 14th International Wireless Communications & Mobile Computing Conference (IWCMC). Limassol, Kıbrıs. <https://doi.org/10.1109/IWCMC.2018.8450296>

Chao, S.-J. ve Cheng, P. W. (2000). The emergence of inferential rules. *Cognitive Development*, 15(1), 39-62. [https://doi.org/10.1016/s0885-2014\(00\)00018-6](https://doi.org/10.1016/s0885-2014(00)00018-6)

Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X. ve Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers & Education*, 109, 162-175. <https://doi.org/10.1016/j.compedu.2017.03.001>

Chowdhury, B., Johri, A., Kafura, D. ve Lohani, V. (2019). Be Constructive: Learning Computational Thinking Using Scratch Online Community. In (pp. 49-60).

Springer International Publishing. https://doi.org/10.1007/978-3-030-35758-0_5

Corbett, C. ve Hill, C. (2015). *Solving the Equation: The Variables for Women's Success in Engineering and Computing*. American Association of University Women.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C. ve Woollard, J. (2015). *Computational thinking-A guide for teachers* [White paper]. UISR. <http://eprints.soton.ac.uk/id/eprint/424545>

Çubukluöz, Ö. (2019). *6. sınıf öğrencilerinin matematik dersindeki öğrenme zorluklarının Scratch programıyla tasarlanan matematiksel oyunlarla giderilmesi: Bir eylem araştırması* (Tez no. 551033) [Yüksek Lisans, Bartın Üniversitesi]. Ulusal Tez Merkezi.

Demirel, Ö. (2010). *Öğretim İlke ve Yöntemleri Öğretme Sanatı* (26 ed.). Pegem Akademi.

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, 60(6), 33-39. <https://doi.org/10.1145/2998438>

Denning, P. J. ve Tedre, M. (2019). *Computational Thinking*. Mit Press.

Doleck, T., Bazalais, P., Lemay, D. J., Saxena, A. ve Basnet, R. B. (2017). Algorithmic thinking, cooperativity, creativity, critical thinking, and problem solving: exploring the relationship between computational thinking skills and academic performance. *Journal of Computers in Education*, 4, 355-369. <https://doi.org/10.1007/s40692-017-0090-9>

Duncan, C., Bell, T. ve Tanimoto, S. (2014). Should your 8-year-old learn coding? In C. Schulte, *WiPSCE '14: Workshop in Primary and Secondary Computing Education* (pp. 60-69). ACM. Proceedings of the 9th Workshop in Primary and Secondary Computing Education. Berlin, Almanya. <https://doi.org/10.1145/2670757.2670774>

- Easton, T. A. (2006). Beyond the algorithmization of the sciences. *Communications of the ACM*, 49(5), 31-33. <https://doi.org/10.1145/1125944.1125967>
- Einhorn, S. (2012). *Micro-Worlds, Computational Thinking, and 21st Century Learning* [White paper]. LCSI. <http://www.microworlds.com/support/files/lcsi-computational-thinking.pdf>
- Elçiçek, M. (2020). Türkiye’de bilgi işlemsel düşünme becerisi ile ilgili yapılmış lisansüstü tezlerin tematik ve yöntemsel eğimleri. *Sakarya University Journal of Education*, 10(3), 485-506. <https://doi.org/10.19126/suje.720618>
- Ericson, B. ve McKlin, T. (2012). Effective and sustainable computing summer camps. In L. S. King, *Symposium on Computer Science Education* (pp. 289-294). ACM. The 43rd ACM Technical Symposium on Computer Science Education. Raleigh North Carolina, ABD. <https://doi.org/10.1145/2157136.2157223>
- Erol, O. (2016). *Scratch ile Programlama Öğretiminin Bilişim Teknolojileri Öğretmen Adaylarının Motivasyon ve Başarılarına etkisi* (Tez no. 395186) [Doktora, Anadolu Üniversitesi]. Ulusal Tez Merkezi.
- Erol, O. ve Çırak, N. S. (2022). The effect of a programming tool scratch on the problem-solving skills of middle school students. *Education and Information Technologies*, 27(3), 4065-4086. <https://doi.org/10.1007/s10639-021-10776-w>
- EUR-Lex. (2018). *Council Recommendation On Key Competences For Lifelong Learning*. 2023, Ekim 19 tarihinde https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.C_.2018.189.01.0001.01.ENG&toc=OJ:C:2018:189:TOC adresinden erişilmiştir.
- Fagerlund, J., Häkkinen, P., Vesisenaho, M. ve Viiri, J. (2021). Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education*, 29(1), 12-28. <https://doi.org/10.1002/cae.22255>
- Falloon, G. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. On the iPad. *Journal of Computer Assisted Learning*, 32(6), 576-593. <https://doi.org/10.1111/jcal.12155>

- Fessakis, G., Gouli, E. ve Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, 63, 87-97. <https://doi.org/10.1016/j.compedu.2012.11.016>
- Feurzeig, W., Papert, S. A. ve Lawler, B. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments*, 19(5), 487-501. <https://doi.org/10.1080/10494820903520040>
- Fields, D. A., Lui, D. ve Kafai, Y. B. (2019). Teaching Computational Thinking with Electronic Textiles: Modeling Iterative Practices and Supporting Personal Projects in Exploring Computer Science. In (pp. 279-294). Springer Singapore. https://doi.org/10.1007/978-981-13-6528-7_16
- Franklin, D., Skifstad, G., Rolock, R., Mehrotra, I., Ding, V., Hansen, A., Weintrop, D. ve Harlow, D. (2017). Using upper-elementary student performance to understand conceptual sequencing in a blocks-based curriculum. In M. E. Caspersen, *Special Interest Group on Computer Science Education* (pp. 231-236). ACM. The 48th ACM Technical Symposium on Computer Science Education. Washington, ABD. <https://doi.org/10.1145/3017680.3017760>
- Gadanidis, G. (2015). Coding as a Trojan Horse for mathematics education reform. *Journal of Computers in Mathematics and Science Teaching*, 34(2), 155-173. <https://www.learntechlib.org/primary/p/147419/>.
- Gagné, R. M. (1985). *The Conditions of Learning and Theory of Instruction* (4 ed.). Holt, Rinehart and Winston.
- Genç, Z. ve Karakuş, S. (2013). Tasarımla Öğrenme: Eğitsel Bilgisayar Oyunları Tasarımında Scratch Kullanımı In Z. Genç, *International Computer & Instructional Technologies Symposium* (pp. 22-24). 5th International Computer & Instructional Technologies Symposium. Elazığ, Türkiye.
- Gleasant, C. ve Kim, C. (2020). Pre-service teacher's use of block-based programming and computational thinking to teach elementary mathematics. *Digital Experiences in Mathematics Education*, 6, 52-90. <https://doi.org/10.1007/s40751-019-00056-1>

- Gootman, E. (2007). For teachers, middle school is test of wills. *New York Times*, D11. <https://www.nytimes.com/2007/03/17/education/17middle.html>
- Graßl, I. (2022). Diversity in programming education: help underrepresented groups learn programming. In M. B. Dwyer, *International Conference on Software Engineering* (pp. 272–274). ACM. 44th International Conference on Software Engineering. Pittsburgh, Pennsylvania. <https://doi.org/10.1145/3510454.3517062>
- Gretter, S. ve Yadav, A. (2016). Computational Thinking and Media & Information Literacy: An Integrated Approach to Teaching Twenty-First Century Skills. *TechTrends*, 60(5), 510-516. <https://doi.org/10.1007/s11528-016-0098-4>
- Grover, S. ve Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher*, 42(1), 38-43. <https://doi.org/10.3102/0013189X1246305>
- Grover, S. ve Pea, R. (2018). Computational Thinking: A Competency Whose Time Has Come. In *Computer science education: Perspectives on teaching and learning in school* (Vol. 19, pp. 19-38). Bloomsbury.
- Gülbahar, Y., Kert, S. B. ve Kalelioğlu, F. (2019). The self-efficacy perception scale for computational thinking skill: Validity and reliability study. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 10(1), 01-29.
- Gültepe, A. (2018). Kodlama Öğretimi Yapan Bilişim Teknolojileri Öğretmenleri Gözüyle Öğrenciler Kodluyor. *Uluslararası Liderlik Eğitimi Dergisi*, 2(2), 50-60.
- Hermans, F. ve Aivaloglou, E. (2017). Teaching Software Engineering Principles to K-12 Students: A MOOC on Scratch. In S. Uchitel, *International Conference on Software Engineering (ICSE)* (pp. 13-22). IEEE. 39th International Conference on Software Engineering: Software Engineering Education and Training Track Buenos Aires, Arjantin. <https://doi.org/10.1109/ICSE-SEET.2017.13>

- HGSE. (2024). *Computational Thinking With Scratch; Developing Fluency With Computational Concepts, Practices, and Perspectives*. Harvard Graduate School of Education. 2023, Mart 4 tarihinde <https://scratched.gse.harvard.edu/ct/defining.html> adresinden erişilmiştir.
- Howard, N. R. ve Howard, K. E. (2020). *Coding+ Math: Strengthen K–5 Math Skills With Computer Science*. International Society for Technology in Education.
- Howe, J. A., Ross, P., Johnson, K., Plane, F. ve Inglis, R. (1982). Teaching mathematics through programming in the classroom. *Computers & Education*, 6(1), 85-91. [https://doi.org/10.1016/0360-1315\(82\)90016-1](https://doi.org/10.1016/0360-1315(82)90016-1)
- Hwang, W.-Y., Wang, C.-Y., Hwang, G.-J., Huang, Y.-M. ve Huang, S. (2008). A web-based programming learning environment to support cognitive development. *Interacting with Computers*, 20(6), 524-534. <https://doi.org/10.1016/j.intcom.2008.07.002>
- ICS. (2023). *Event-driven programming*. Isaac Computer Science. 2023, Haziran 5 tarihinde https://isaaccomputerscience.org/topics/event_driven_programming?examBoard=all&stage=all adresinden erişilmiştir.
- Ioannidou, A., Bennett, V., Repenning, A., Koh, K. H. ve Basawapatna, A. (2011). *Computational Thinking Patterns* [White paper]. AERA. <https://files.eric.ed.gov/fulltext/ED520742.pdf>
- Israel, M., Wherfel, Q. M., Pearson, J., Shehab, S. ve Tapia, T. (2015). Empowering K–12 Students With Disabilities to Learn Computational Thinking and Computer Programming. *TEACHING Exceptional Children*, 48(1), 45-53. <https://doi.org/10.1177/0040059915594790>
- ISTE. (2023). *Iste Computational Thinking Competencies*. 2023, Kasım 16 tarihinde <https://www.iste.org/standards/iste-standards-for-computational-thinking> adresinden erişilmiştir.
- Jona, K., Wilensky, U., Trouille, L., Horn, M., Orton, K., Weintrop, D. ve Beheshti, E. (2014). *Embedding Computational Thinking in Science, Technology,*

Engineering, and Math (CT-STEM). In Future Directions In Computer Science Education Summit Meeting, Orlando, Florida.
<http://ccl.sesp.northwestern.edu/papers/2014/OrtonKaiNorthwestern-1.pdf>

K12CS2. (2024). *K-12 Computer Science Framework 2023*, Kasım 5 tarihinde <https://k12cs.org/> adresinden erişilmiştir.

Kafai, Y. B. ve Burke, Q. (2013). Computer Programming Goes Back to School. *Phi Delta Kappan*, 95(1), 61-65. <https://doi.org/10.1177/003172171309500111>

Kahn, K., Sendova, E., Sacristán, A. I. ve Noss, R. (2011). Young students exploring cardinality by constructing infinite processes. *Technology, Knowledge and Learning*, 16, 3-34. <https://doi.org/10.1007/s10758-011-9175-0>

Kalelioğlu, F., Gülbahar, Y., Akçay, S. ve Doğan, D. (2014). Curriculum Integration Ideas for Improving the Computational Thinking Skills of Learners through Programming via Scratch. In Y. Gülbahar, *Informatics in Schools: Situation, Evolution and Perspectives (ISSEP)* (pp. 101-112). Local Proceedings of the 7th international Conference İstanbul, Türkiye.

Karataş, Z. (2015). Sosyal Bilimlerde Nitel Araştırma Yöntemleri. *Manevi temelli sosyal hizmet araştırmaları dergisi*, 1(1), 62-80.

Kaučič, B. ve Asič, T. (2011). Improving introductory programming with Scratch? In P. Biljanović, *Croatian Society Mipro* (pp. 1095-1100). IEEE. 34th International Convention on Information and Communication Technology, Electronics and Microelectronics. Opatija, Hırvatistan.

Kazımoğlu, Ç., Kiernan, M., Bacon, L. ve MacKinnon, L. (2011). Understanding computational thinking before programming: Developing guidelines for the design of games to learn introductory programming through game-play. *International Journal of Game-Based Learning (IJGBL)*, 1(3), 30-52.

Kiss, G. ve Arki, Z. (2017). The influence of game-based programming education on the algorithmic thinking. *Procedia-Social and Behavioral Sciences*, 237, 613-617.

- Klawe, M., Whitney, T. ve Simard, C. (2009). Women in computing: Take 2. *Communications of the ACM - Inspiring Women in Computing*, 52(2), 68-76. <https://doi.org/10.1145/1461928.1461947>
- Knuth, D. E. (1974). Computer science and its relation to mathematics. *The American Mathematical Monthly*, 81(4), 323-343.
- Kolovou, A., Van Den Heuvel-Panhuizen, M. ve Bakker, A. (2009). Non-Routine Problem Solving Tasks in Primary School Mathematics Textbooks – A Needle in a Haystack. *Mediterranean Journal for Research in Mathematics Education*, 8(2), 31-68.
- KOODI2016. (2014). *Koodi2016*. Juhani ve Linda. 2023, Mart 3 tarihinde <https://koodi2016.fi/> adresinden erişilmiştir.
- Korkmaz, Ö. (2016). The Effect of Scratch-Based Game Activities on Students' Attitudes, Self-Efficacy and Academic Achievement. *I.J. Modern Education and Computer Science*, 8(1), 16-23. <https://doi.org/10.5815/ijmecs.2016.01.03>
- Korkmaz, Ö. (2018). The effect of scratch-and lego mindstorms Ev3-Based programming activities on academic achievement, problem-solving skills and logical-mathematical thinking skills of students. *MOJES: Malaysian Online Journal of Educational Sciences*, 4(3), 73-88.
- Korkmaz, Ö., Çakır, R. ve Özden, M. Y. (2015). Bilgisayarca Düşünme Beceri Düzeyleri Ölçeğinin (BDBD) Ortaokul Düzeyine Uyarlanması. *Gazi Eğitim Bilimleri Dergisi*, 1(2), 143-162.
- Kotsopoulos, D., Floyd, L., Khan, S., Namukasa, I. K., Somanath, S., Weber, J. ve Yiu, C. (2017). A Pedagogical Framework for Computational Thinking. *Digital Experiences in Mathematics Education*, 3(2), 154-171. <https://doi.org/10.1007/s40751-017-0031-2>
- Lau, W. W. ve Yuen, A. H. (2011). Modelling programming performance: Beyond the influence of learner characteristics. *Computers & Education*, 57(1), 1202-1213.

- Layman, J. ve Hall, W. (1988). Logo: A cause for concern. *Computers & Education*, 12(1), 107-112.
- Leinwand, S., Bowling, D. J. B. ve Huinker, D. (2014). *Principles to Actions: Ensuring Mathematical Success for All*. National Council of Teachers of Mathematics.
- Lockwood, J. ve Mooney, A. (2017). *Computational Thinking in Education: Where does it fit? A Systematic Literary Review* [White paper]. MU. <https://arxiv.org/pdf/1703.07659>
- Lu, J. J. ve Fletcher, G. H. (2009). Thinking about computational thinking. In S. Fitzgerald, *Special Interest Group on Computer Science Education* (pp. 260-264). The 40th ACM Technical Symposium on Computer Science Education. Chattanooga, ABD. <https://doi.org/10.1145/1508865.1508959>
- Lye, S. Y. ve Koh, J. H. L. (2014). Review on Teaching and Learning of Computational Thinking Through Programming: What is next for K-12? *Computers in human behavior*, 41, 51-61. <https://doi.org/10.1016/j.chb.2014.09.012>
- Magana, A. J., Marepalli, P. ve Clark, J. V. (2011). Work in Progress-Integrating Computational And Engineering Thinking Through Online Design And Simulation Of Multidisciplinary Systems. *Information and Communications Technology in Engineering Education* (pp. S3C-1-S3C-2). IEEE. 2011 Frontiers in Education Conference (FIE). Rapid City, ABD. <https://doi.org/10.1109/FIE.2011.6143083>
- Malan, D. J. ve Leitner, H. H. (2007). Scratch for budding computer scientists. *ACM SIGCSE Bulletin*, 39(1), 223-227. <https://doi.org/10.1145/1227504.1227388>
- Maloney, J., Resnick, M., Rusk, N., Silverman, B. ve Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, 10(4), 1-15.
- Marcelino, M. J., Pessoa, T., Vieira, C., Salvador, T. ve Mendes, A. J. (2018). Learning computational thinking and scratch at distance. *Computers in human behavior*, 80, 470-477.

- Marcia Linn, A. V. A., M. Brian Blake, Robert Constable, Yasmin B. Kafai, Janet L. Kolodner, Lawrence Snyder, Uri Wilensky. (2010). *Report of a Workshop on the Scope and Nature of Computational Thinking*. National Academies Press.
- Mayer, R. E. (1998). Cognitive, metacognitive, and motivational aspects of problem solving. *Instructional science*, 26(1), 49-63.
- McGehee, J. J. (2001). Developing interdisciplinary units: A strategy based on problem solving. *School Science and Mathematics*, 101(7), 380-389.
- MEB. (2018). *Güçlü Yarınlar için 2023 Eğitim Vizyonu* [White paper]. MEB. https://www.gmka.gov.tr/dokumanlar/yayinlar/2023_E%C4%9Fitim%20Vizyonu.pdf
- MEB. (2023). *Matematik Dersi Öğretim Programı* [White paper]. MEB. <https://mufredat.meb.gov.tr/ProgramDetay.aspx?PID=329>
- Miller, M. ve Nunn, G. D. (2001). Using Group Discussions To Improve Social Problem-Solving And Learning. *Education*, 121(3), 470.
- Moreno-León, J., Robles, G. ve Román-González, M. (2015). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *RED. Revista de Educación a Distancia* (46), 1-23.
- Morrison, N. (2013, Haziran 30). *Teach Kids How To Code And You Give Them A Skill For Life*. Forbes. <https://www.forbes.com/sites/nickmorrison/2013/12/27/teach-kids-how-to-code-and-you-give-them-a-skill-for-life/?sh=5e355d6365d2>
- OECD. (2013). “*Problem-Solving Framework*”, in *PISA 2012 Assessment and Analytical Framework: Mathematics, Reading, Science, Problem Solving and Financial Literacy* [White paper]. O. Publishing. <https://doi.org/10.1787/9789264190511-en>
- Oluk, A. ve Saltan, F. (2015). Effects of using the scratch program in 6th grade information technologies courses on algorithm development and problem solving skills. *Participatory educational research*, 2(5), 10-20.

ÖDSGM. (2023). *8.Sınıf Matematik Çalışma Kitabı*. Ölçme, Değerlendirme ve Sınav Hizmetleri Genel Müdürlüğü. 2024, Şubat 12 tarihinde <https://odsgm.meb.gov.tr/www/temel-egitim-kitaplari/icerik/834> adresinden erişilmiştir.

Özçınar, H. (2017). Hesaplamalı Düşünme Araştırmalarının Bibliyometrik Analizi. *Eğitim Teknolojisi Kuram ve Uygulama*, 7(2), 149-171.

Özkeş, B. (2016). *Bilişimsel düşünme temelli ders etkinliklerinin öğrencilerin eleştirel düşünme becerileri ve problem çözme becerilerine yönelik algıları üzerine etkisinin incelenmesi* (Tez no. 619695) [Yüksek Lisans, Konya Mevlana Üniversitesi]. Ulusal Tez Merkezi.

Papadopoulos, Y. ve Tegou, S. (2012). Using Microworlds to Introduce Programming to Novices. In S. Katsikas, *Panhellenic Conference on Informatics (PCI)* (pp. 180-185). Institute of Electrical and Electronics Engineers. Proceedings of the 2012 16th Panhellenic Conference on Informatics Pireus, Yunanistan. <https://doi.org/10.1109/pci.2012.18>

Papert, S. (1980). *Mindstorms: Children, Computers, And Powerful Ideas*. Basic Books.

Papert, S. (1988). A Critique of Technocentrism in Thinking About the School of the Future. In (pp. 3-18). Elsevier. <https://doi.org/10.1016/b978-0-08-036464-3.50006-5>

Papert, S. A. (2020). *Mindstorms: Children, computers, and powerful ideas*. Basic books.

Pea, R. D. ve Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, 2(2), 137-168. [https://doi.org/10.1016/0732-118x\(84\)90018-7](https://doi.org/10.1016/0732-118x(84)90018-7)

Pechtelidis, Y., Kosma, Y. ve Chronaki, A. (2015). Between a rock and a hard place: women and computer technology. *Gender and Education*, 27(2), 164-182. <https://doi.org/10.1080/09540253.2015.1008421>

- Pierce, M. (2013). Coding for middle schoolers: Next-generation programming languages for children are taking up where Logo left off and teaching young students how to code to learn. *THE Journal [Technological Horizons In Education]*, 40(5), 20.
- Pinkard, N., Erete, S., Martin, C. K. ve McKinney de Royston, M. (2017). Digital youth divas: Exploring narrative-driven curriculum to spark middle school girls' interest in computational activities. *Journal of the Learning Sciences*, 26(3), 477-516.
- Psycharis, S. ve Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional science*, 45(5), 583-602. <https://doi.org/10.1007/s11251-017-9421-5>
- Read, J. C. ve Macfarlane, S. (2006). Using The Fun Toolkit And Other Survey Methods To Gather Opinions In Child Computer Interaction. In K.-J. Räihä, *IDC06: 5th International Conference on Interaction Design and Children* (pp. 81–88). ACM. IDC '06: Proceedings of the 2006 conference on Interaction design and children. New York, ABD. <https://doi.org/10.1145/1139073.1139096>
- Repenning, A. (2006). Excuse me, I need better AI! Employing Collaborative Diffusion to make Game AI Child's Play. In A. Heirich, *Sandbox '06: An ACM Video Game Symposium 2006* (pp. 169-178). Association for Computing Machinery. Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames. Boston, Massachusetts <https://doi.org/10.1145/1183316.1183341>
- Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J. ve Silverman, B. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11), 60-67. <https://doi.org/10.1145/1592761.1592779>

- Rich, P. J., Browning, S. F., Perkins, M., Shoop, T., Yoshikawa, E. ve Belikov, O. M. (2019). Coding in K-8: International trends in teaching elementary/primary computing. *TechTrends*, 63(3), 311-329.
- Rodríguez-Martínez, J. A., González-Calero, J. A. ve Sáez-López, J. M. (2020). Computational thinking and mathematics using Scratch: an experiment with sixth-grade students. *Interactive Learning Environments*, 28(3), 316-327. <https://doi.org/10.1080/10494820.2019.1612448>
- Román-González, M., Pérez-González, J.-C. ve Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in human behavior*, 72, 678-691. <https://doi.org/10.1016/j.chb.2016.08.047>
- Ross, A. ve Willson, V. L. (2018). *Basic and advanced statistical tests: Writing results sections and creating tables and figures*. Springer.
- Sáez-López, J.-M., Román-González, M. ve Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using “Scratch” in five schools. *Computers & Education*, 97, 129-141. <https://doi.org/10.1016/j.compedu.2016.03.003>
- Sapounidis, T., Demetriadis, S. ve Stamelos, I. (2015). Evaluating children performance with graphical and tangible robot programming tools. *Personal and Ubiquitous Computing*, 19(1), 225-237. <https://doi.org/10.1007/s00779-014-0774-3>
- Sarjoughian, H. S. ve Zeigler, B. P. (1996). Abstraction mechanisms in discrete-event inductive modeling. In J. M. Charnes, *WSC90: 1990 Winter Simulation Conference* (pp. 748-755). Proceedings of the 28th conference on Winter simulation. California, ABD. <https://doi.org/10.1145/256562.256803>
- Sears, S. J. (1995). Career and educational planning in the middle level school. *NASSP Bulletin*, 79(570), 36-42.

- Shute, V. J., Sun, C. ve Asbell-Clarke, J. (2017). Demystifying Computational Thinking. *Educational research review*, 22, 142-158. <https://doi.org/10.1016/j.edurev.2017.09.003>
- Siegle, D. (2017). Technology. *Gifted Child Today*, 40(2), 117-123. <https://doi.org/10.1177/1076217517690861>
- StackExchange. (2024). *How to choose between t-test or non-parametric test e.g. Wilcoxon in small samples*. 07 Mayıs 2024 tarihinde <https://stats.stackexchange.com/questions/121852/how-to-choose-between-t-test-or-non-parametric-test-e-g-wilcoxon-in-small-sampl/123389#123389> adresinden erişilmiştir.
- Şahiner, A. ve Kert, S. (2016). Examining studies related with the concept of computational thinking between the years of 2006-2015. *European Journal of Science and Technology*, 5(9), 38-43.
- Şeker, Z. (2020). Dinleme ve konuşma becerilerine yönelik lisansüstü tezlerin anahtar kelimeleri üzerine bir inceleme: Betimsel analiz. *RumeliDE Dil ve Edebiyat Araştırmaları Dergisi*(19), 128-140.
- Şen, A. İ. (Ed.). (2023). *Okul Dışı Öğrenme Ortamları* (Vol. 3). Pegem Akademi.
- Tang, X., Yin, Y., Lin, Q., Hadad, R. ve Zhai, X. (2020). Assessing computational thinking: A systematic review of empirical studies. *Computers & Education*, 148, 103798. <https://doi.org/10.1016/j.compedu.2019.103798>
- Taylor, M. (2019). *Class Notes: Algorithmic Thinking*. 2023, Mayıs 2 tarihinde <https://www.cs.cmu.edu/~112-s23/notes/notes-algorithmic-thinking.html> adresinden erişilmiştir.
- Tortop, F., Cumali, A., Çelenli, M. ve Taşpınar-Şener, Z. (2022). LGS Sınavındaki Beceri Temelli Matematik Sorularına Yönelik Öğretmen Görüşleri. *Erciyes Journal of Education*, 6(2), 99-126. <https://doi.org/10.32433/eje.1076448>
- TTKB. (2023). *Sınavla Öğrenci Alacak Ortaöğretim Kurumlarına İlişkin Merkezi Sınava Yönelik Konu, Kazanım ve Açıklamalar* [White paper]. M. T. v. T.

Kurulu.

https://ttkb.meb.gov.tr/meb_iys_dosyalar/2023_02/13150011_2023_LGS_tum_dersler.pdf

Vatansever, Ö. ve Baltacı Göktalay, Ş. (2018). How does teaching programming through scratch affect problem-solving skills of 5th and 6th grade middle school students? *Uluslararası Avrasya Sosyal Bilimler Dergisi*, 9(33), 1778-1801.

Voskoglou, M. G. ve Buckley, S. (2012). Problem Solving And Computational Thinking In A Learning Environment. *arXiv preprint arXiv:1212.0750*. <https://doi.org/10.48550/arXiv.1212.0750>

Vrieler, T., Nylén, A. ve Cajander, Å. (2021). Computer science club for girls and boys – a survey study on gender differences. *Computer Science Education*, 31(4), 431-461. <https://doi.org/10.1080/08993408.2020.1832412>

Webb, H. ve Rosson, M. B. (2013). Using scaffolded examples to teach computational thinking concepts. In T. Camp, *SIGCSE '13: The 44th ACM Technical Symposium on Computer Science Education* (pp. 95–100). ACM. Proceeding of the 44th ACM technical symposium on Computer science education. Colorado, ABD. <https://doi.org/10.1145/2445196.2445227>

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L. ve Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of science education and technology*, 25, 127-147.

Weintrop, D., Beheshti, E., Horn, M. S., Orton, K., Trouille, L., Jona, K. ve Wilensky, U. (2014). Interactive Assessment Tools for Computational Thinking in High School STEM Classrooms. In (pp. 22-25). Springer International Publishing. https://doi.org/10.1007/978-3-319-08189-2_3

Werner, L., Campe, S. ve Denner, J. (2012). Children learning computer science concepts via Alice game-programming. In L. S. King, *Special Interest Group on Computer Science Education (SIGCSE)* (pp. 427-432). Proceedings of the 43rd ACM Technical Symposium on Computer Science Education. Raleigh North Carolina, ABD. <https://doi.org/10.1145/2157136.2157263>

- Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, 49(3), 33-35.
- Wing, J. M. (2010). *Computational Thinking: What and Why?* [White paper]. CMU. <https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf>
- Wong, K. W. G., Ching, C. C., Mark, K. P., Tang, J., Lei, C., Cheung, H. Y. ve Chui, H. L. (2015). Impact of computational thinking through coding in K-12 education: A pilot study in Hong Kong. *General Studies*, 85(88.01), 2-08.
- Woo, K. ve Falloon, G. (2022). Problem solved, but how? An exploratory study into students' problem solving processes in creative coding tasks. *Thinking Skills and Creativity*, 46, 101193. <https://doi.org/10.1016/j.tsc.2022.101193>
- Yalçinkaya, B., Dönmez, A., Aydın, F., Kayalı, N. ve Sönmez, A. (2018). İlköğretim Çocuklarının Kodlama Algısı Üzerine Empirik Bir Analiz Çalışması ve Çocuk Kütüphanelerinde Uygulanmasının Önemi. In A. K. Yıldız, *Uluslararası Çocuk Kütüphaneleri Sempozyumu Bildirileri* (pp. 126-138). 1. Uluslararası Çocuk Kütüphaneleri Sempozyumu Bildirileri. Nevşehir, Türkiye.
- Yıldırım, A. ve Şimşek, H. (2021). *Sosyal Bilimlerde Nitel Araştırma Yöntemleri* (Vol. 12). Seçkin Yayıncılık.
- Zhang, L. ve Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education*, 141, 103607.
- Zhu, K., Ma, X., Wong, G. K. W. ve Huen, J. M. H. (2016). How Different Input and Output Modalities Support Coding as a Problem-Solving Process for Children. In J. C. Read, *IDC '16: Interaction Design and Children* (pp. 238-245). ACM. Proceedings of the The 15th International Conference on Interaction Design and Children. Manchester, Birleşik Krallık. <https://doi.org/10.1145/2930674.2930697>

EK 2: Öntest Sontest Envanteri

1. Bir çuvalda 203 kg pirinç vardır.

Bu çuvala en az kaç kg daha pirinç eklenirse, çuvaldaki pirinç hem 17 kg'lık hem de 13 kg'lık poşetlere hiç artmayacak ve poşetlerin tamamı dolacak şekilde paylaştırılabilir?

- A) 15 B) 16 C) 17 D) 18

2. Bitkilerin farklı ortam ve koşullardaki büyümelerini araştırmak isteyen Elif, 30 cm boyunda iki fidan alıyor. Bu fidanların farklı ortam ve koşullarda 1 ay boyunca büyüme-lerini gözlemliyor.



Elif 1 ay sonunda fidanların boylarını santimetre cinsinden ölçüyor. Ölçtüğü bu uzunlukların, asal çarpanları aynı olan iki doğal sayı olduğunu görüyor.

1 ay sonunda birinci fidanın boyu 90 santimetre olduğuna göre, ikinci fidanın boyu aşağıdakilerden hangisi olabilir?

- A) 100 B) 110 C) 140 D) 150

3.



Şekilde, asal çarpanı 2 ve 3 olan sayıların bulunduğu balonlar patlatılıyor

Şeklin kalan hali aşağıdakilerden hangisidir?

A)



B)



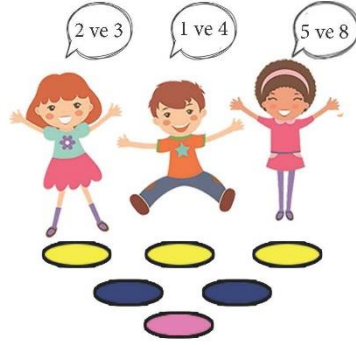
C)



D)



4.



Yukarıda verilen şekilde Türkü, Alperen ve Hasret, söyledikleri sayıların ortak katlarını önlerinde bulunan sarı halkalara yazacakları bir oyuna başlamışlardır. Daha sonra sarı halkalarda yazan sayıların ortak katlarını ise her iki sarı halkanın önünde bulunan mavi halkalara, mavi halkarda yazan sayıların ortak katını ise pembe halkaya yazacakları şekilde oyunu devam ettirmişlerdir.

Buna göre pembe halka içerisinde yazılacak sayı aşağıdakilerden hangisidir?

- A) 30 B) 60 C) 90 D) 120

5.



Nur Sima Hanım, alışverişe çıkmadan önce telefonunun adım sayar programında günlük yürüme hedefini 3200 adım olarak belirliyor. Adım sayar ilk adımdan başlayarak 3200 sayısının her pozitif tam sayı böleninde titreşimle uyarı veriyor.

Nur Sima Hanım'ın telefonu 11. kez titreşimle uyarı verdiğinde Nur Sima Hanım attığı adım sayısının kaç katı daha adım atarsa günlük yürüme hedefine ulaşır?

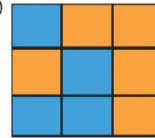
- A) 60 B) 70 C) 80 D) 90

6.

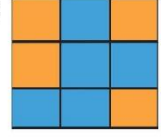
193	147	127
153	113	61
51	91	87

Yukarıdaki tabloda verilen sayılardan asal olanlar turuncu, asal olmayanlar ise mavi ile boyanırsa aşağıdaki görüntülerden hangisi oluşur?

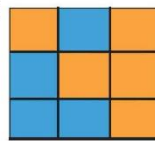
A)



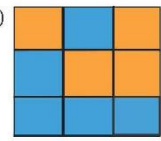
B)



C)



D)



7.



Yukarıda verilen boya kaplarından özdeş ölçek kullanılarak farklı oranlarda karıştırılmasıyla farklı renkler elde edilmektedir. Karışıma katılan renklerin temsil ettiği sayılar; sarı =2, kırmızı =3, mavi=5, beyaz =7 ve siyah =11'dir. Oluşan karışımın renk kodu bu sayıların çarpılması ile belirlenmektedir.

Örneğin 4 ölçek sarı ve 1 ölçek kırmızı boya karıştırılarak elde edilen turuncu boyanın kodu; $2 \times 2 \times 2 \times 3 = 48$ 'dir.

Bu bilgilere göre numarası 132 olan boyaya hangi renk boya katılmamıştır?

- A) Sarı B) Kırmızı C) Mavi D) Siyah

8.

Bir firma yeni ürünleri için iki ayrı salonda tanıtım toplantıları düzenleyecektir. Toplantılar oturumlar halinde yapılacak olup, iki oturum arasında 10'ar dakikalık aralar verilecektir. Her bir oturumun kaçar dakika olacağı tabloda verilmiştir.

Tablo: Toplantıların oturum Süreleri

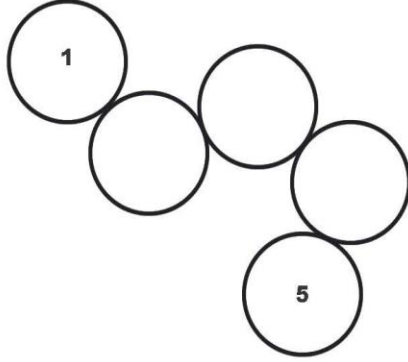
Salon	Toplantı Süresi
A	30 dk
B	50 dk

Salonlardaki toplantılar saat 08.15'te başlayacaktır. Saat 12.30'dan sonra başlangıç saati aynı olan aralardan biri uzatılarak öğle yemeği verilecektir.

Buna göre öğle yemeği en erken saat kaçta başlayabilir?

- A) 14.15 B) 14.05 C) 13.15 D) 13.05

9. 1'den 5'e kadar olan sayılar birer kez kullanılarak dairesel bölümlerden oluşan aşağıdaki şekle yerleştirilecektir.

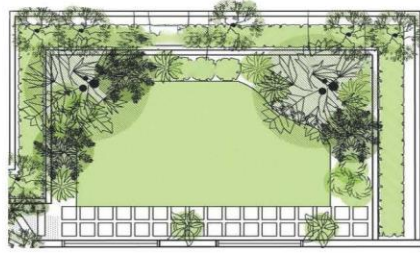


Verilen şekilde birbirine yatay veya dikey olarak temas eden dairelerin içindeki sayılar aralarında asal olacaktır.

1 ve 5 sayıları yerleştirildiğine göre, geri kalan sayılar bu şekle kaç farklı şekilde yerleştirilebilir?

- A) 1 B) 2 C) 3 D) 4

10. Aşağıda dikdörtgen şekilden bir bahçe verilmiştir.

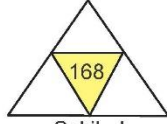


Alanı 612 m^2 olan dikdörtgen şeklindeki bir bahçenin uzun kenar uzunluğu ile kısa kenarının uzunluğu aralarında asaldır.

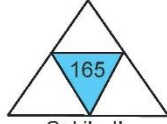
Bahçenin kısa kenarının uzunluğu metre cinsinden aşağıdakilerden hangisi olabilir?

- A) 12 B) 15 C) 17 D) 19

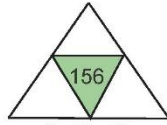
11.



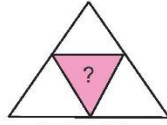
Şekil - I



Şekil - II



Şekil - III



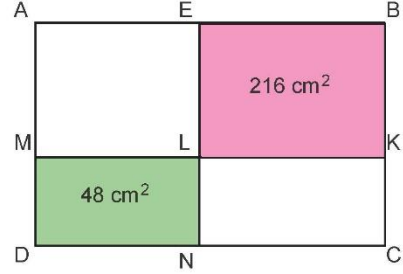
Şekil - IV

Berra, şekilde yazan her sayının asal çarpanlarını bulup etrafındaki üçgenlere yazıyor ve ilk üç şeklin her birinden, birbirinden farklı asal sayılar alarak Şekil IV'e yazıyor ve asal çarpanları bu sayılar olan bir sayı oluşturuyor.

Berra'nın elde ettiği bu sayı aşağıdakilerden hangisi olabilir?

- A) 204 B) 147 C) 385 D) 693

12.



Yukarıda verilen şekilde EBKL, KLNC, MLND dikdörtgen, AELM ise karedir.

ABCD dörtgeninin çevresi en az kaç santimetredir?

- A) 118 B) 112 C) 92 D) 80

13. Aşağıda 2010 yılının ağustos ayına ait bir takvim yaprağı görseli verilmiştir.

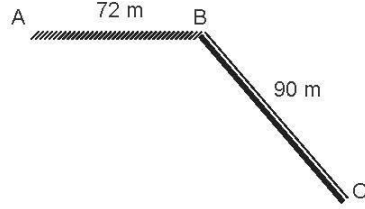
pt	sa	ça	pe	cu	ct	pz
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Takvim yaprağında işaretlenen gün ile Gülefer'in doğum gününün ortak bölenlerinin en büyüğü 5 ise

Gülefer'in doğum günü aşağıdaki günlerden hangisi olamaz?

- A) Pazar B) Pazartesi C) Cuma D) Cumartesi

14.

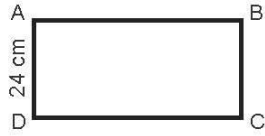


Şekildeki yola eşit aralıklarla aralarındaki mesafe metre cinsinden tam sayı olacak şekilde aydınlatma direkleri dikilecektir.

A, B ve C noktalarına da bu direklerden konması şartıyla gerekli olan toplam direk sayısı aşağıdakilerden hangisi olamaz?

- A) 10 B) 19 C) 29 D) 55

15.



Kısa kenarlarından birisi 24 cm ve çevresi 192 cm olan dikdörtgen şeklindeki bir karton, kenar uzunlukları santimetre cinsinden tam sayı olan eş kare parçalara bölünecektir.

Buna göre eş kare parçaların kenar uzunluklarının alabileceği kaç farklı tam sayı değeri vardır?

- A) 6 B) 8 C) 12 D) 24

16.



Asal çarpanı sadece 2 olan iki basamaklı sayılar torbası En büyük asal çarpanı 3 olan iki basamaklı sayılar torbası En büyük asal çarpanı 5 olan iki basamaklı sayılar torbası

Yukarıda üç torba verilmiştir. Bu torbalarda bulunan sayılarla ilgili bilgiler torbaların altında yazmaktadır.

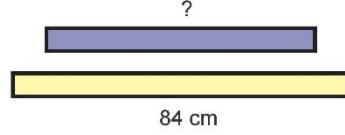
Birinci torbadan A, ikinci torbadan B, üçüncü torbadan C sayısı çekildiğine göre;

- I. $A + B$ en az 41'dir.
II. $A - B$ en çok 52'dir.
III. C'nin alabileceği en büyük ve en küçük değerlerin toplamı 100'dür.

İfadelerinden hangileri doğru olur?

- A) I, II, III B) II ve III C) I ve II D) Yalnız III

17. Aşağıda uzunluğu 84 cm olan ve uzunluğu belli olmayan iki çubuk gösterilmiştir.

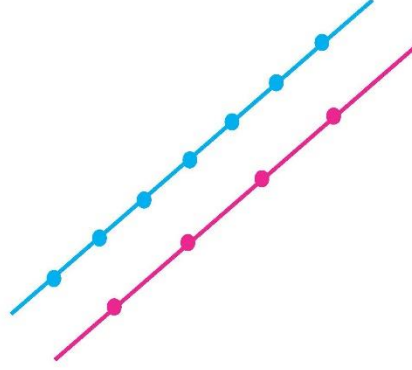


Sündüz, matematik ödevi için görselleri verilen iki çubuğu eşit uzunluklarda keserek uçlarından birleştirmiş ve birbirine eş üç adet kare oluşturmuştur. Bu işlem için toplam 10 kesim yapmıştır.

Buna göre, uzunluğu belli olmayan çubuk kaç santimetredir?

- A) 72 B) 60 C) 48 D) 36

18. Aşağıda eşit uzunluğa sahip mavi ve pembe renkli iki tel verilmiştir.



Mavi tel, parçaların uzunlukları santimetre cinsinden tam sayı olacak şekilde 8 eş parçaya ayrılmak için şekildeki gibi mavi noktalarla işaretleniyor. Pembe tel ise parçaların uzunlukları santimetre cinsinden tam sayı olacak şekilde 5 eş parçaya ayrılmak için şekildeki gibi pembe noktalarla işaretleniyor.

Bu iki tel birer uçları aynı hizada olacak şekilde üst üste konulduğunda mavi ve pembe noktalar arasındaki mesafe en az kaç santimetre olabilir?

- A) 1 B) 5 C) 8 D) 40

19. Hacker Büşra 0 hariç rakamları kullanarak dört basamaklı şifre oluşturacaktır.



Büşra'nın oluşturmak istediği şifrede yan yana gelen rakamlar aralarında asal olduğuna göre şifre aşağıdakilerden hangisi olabilir?

- A) 1397 B) 2465 C) 3456 D) 8936

20.



Yukarıda görünen yüzlerinde iki basamaklı sayıların yazılı olduğu kırmızı ve mavi kartlar verilmiştir. Farklı renkteki kartların görünmeyen yüzleri birbirine yapıştırılarak yeni kartlar elde ediliyor. Oluşan yeni kartların kırmızı ve mavi yüzündeki sayılar aralarında asaldır.

Buna göre oluşturulan yeni kartların herhangi bir yüzeyden görünüşleri aşağıdakilerden hangisi gibi olabilir?

- A) 
- B) 
- C) 
- D) 

1. Bir çuvalda 203 kg pirinç vardır.

Bu çuvala en az kaç kg daha pirinç eklenirse, çuvaldaki pirinç hem 17 kg'lık hem de 13 kg'lık poşetlere hiç artmayacak ve poşetlerin tamamı dolacak şekilde paylaştırılabilir?

- A) 15 B) 16 C) 17 D) 18

2. Bitkilerin farklı ortam ve koşullardaki büyümelerini araştırmak isteyen Elif, 30 cm boyunda iki fidan alıyor. Bu fidanların farklı ortam ve koşullarda 1 ay boyunca büyümelerini gözlemliyor.



Elif 1 ay sonunda fidanların boylarını santimetre cinsinden ölçüyor. Ölçtüğü bu uzunlukların, asal çarpanları aynı olan iki doğal sayı olduğunu görüyor.

1 ay sonunda birinci fidanın boyu 90 santimetre olduğuna göre, ikinci fidanın boyu aşağıdakilerden hangisi olabilir?

- A) 100 B) 110 C) 140 D) 150

3.



Şekilde, asal çarpanı 2 ve 3 olan sayıların bulunduğu balonlar patlatılıyor

Şeklin kalan hali aşağıdakilerden hangisidir?

A)



B)



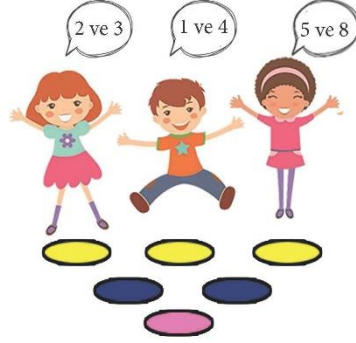
C)



D)



4.



Yukarıda verilen şekilde Türkü, Alperen ve Hasret, söyledikleri sayıların ortak katlarını önlerinde bulunan sarı halkalara yazacakları bir oyuna başlamışlardır. Daha sonra sarı halkalarda yazan sayıların ortak katlarını ise her iki sarı halkanın önünde bulunan mavi halkalara, mavi halkarda yazan sayıların ortak katını ise pembe halkaya yazacakları şekilde oyunu devam ettirmişlerdir.

Buna göre pembe halka içerisinde yazılacak sayı aşağıdakilerden hangisidir?

- A) 120 B) 90 C) 60 D) 30

5.



Nur Sima Hanım, alışverişe çıkmadan önce telefonunun adım sayar programında günlük yürüme hedefini 3200 adım olarak belirliyor. Adım sayar ilk adımdan başlayarak 3200 sayısının her pozitif tam sayı böleninde titreşimle uyarı veriyor.

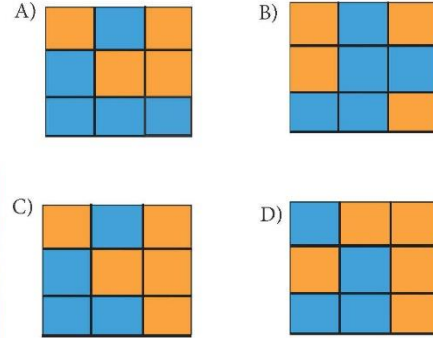
Nur Sima Hanım'ın telefonu 11. kez titreşimle uyarı verdiğinde Nur Sima Hanım attığı adım sayısının kaç katı daha adım atarsa günlük yürüme hedefine ulaşır?

- A) 90 B) 80 C) 70 D) 60

6.

193	147	127
153	113	61
51	91	87

Yukarıdaki tabloda verilen sayılardan asal olanlar turuncu, asal olmayanlar ise mavi ile boyanırsa aşağıdaki görüntülerden hangisi oluşur?



7.



Yukarıda verilen boya kaplarından özdeş ölçek kullanılarak farklı oranlarda karıştırılmasıyla farklı renkler elde edilmektedir. Karışıma katılan renklerin temsil ettiği sayılar; sarı =2, kırmızı =3, mavi=5, beyaz =7 ve siyah =11'dir. Oluşan karışımın renk kodu bu sayıların çarpılması ile belirlenmektedir.

Örneğin 4 ölçek sarı ve 1 ölçek kırmızı boya karıştırılarak elde edilen turuncu boyanın kodu; $2 \times 2 \times 2 \times 2 \times 3 = 48$ 'dir.

Bu bilgilere göre numarası 132 olan boyaya hangi renk boya katılmamıştır?

- A) Sarı B) Kırmızı C) Mavi D) Siyah

8.

Bir firma yeni ürünleri için iki ayrı salonda tanıtım toplantıları düzenleyecektir. Toplantılar oturumlar halinde yapılacak olup, iki oturum arasında 10'ar dakikalık aralar verilecektir. Her bir oturumun kaçar dakika olacağı tabloda verilmiştir.

Tablo: Toplantıların oturum Süreleri

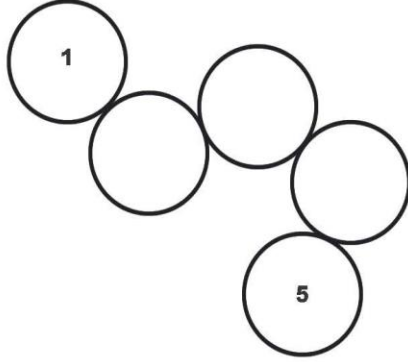
Salon	Toplantı Süresi
A	30 dk
B	50 dk

Salonlardaki toplantılar saat 08.15'te başlayacaktır. Saat 12.30'dan sonra başlangıç saati aynı olan aralardan biri uzatılarak öğle yemeği verilecektir.

Buna göre öğle yemeği en erken saat kaçta başlayabilir?

- A) 14.15 B) 14.05 C) 13.15 D) 13.05

9. 1'den 5'e kadar olan sayılar birer kez kullanılarak dairesel bölümlerden oluşan aşağıdaki şekle yerleştirilecektir.

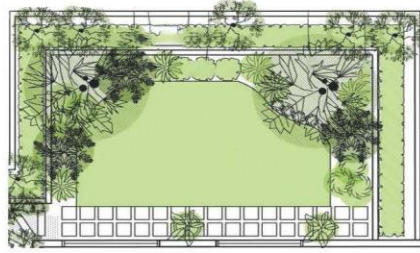


Verilen şekilde birbirine yatay veya dikey olarak temas eden dairelerin içindeki sayılar aralarında asal olacaktır.

1 ve 5 sayıları yerleştirildiğine göre, geri kalan sayılar bu şekle kaç farklı şekilde yerleştirilebilir?

- A) 4 B) 3 C) 2 D) 1

10. Aşağıda dikdörtgen şekilden bir bahçe verilmiştir.

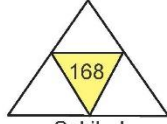


Alanı 612 m^2 olan dikdörtgen şeklindeki bir bahçenin uzun kenar uzunluğu ile kısa kenarının uzunluğu aralarında asaldır.

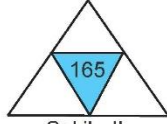
Bahçenin kısa kenarının uzunluğu metre cinsinden aşağıdakilerden hangisi olabilir?

- A) 12 B) 15 C) 17 D) 19

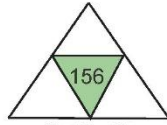
11.



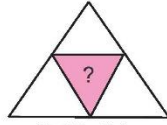
Şekil - I



Şekil - II



Şekil - III



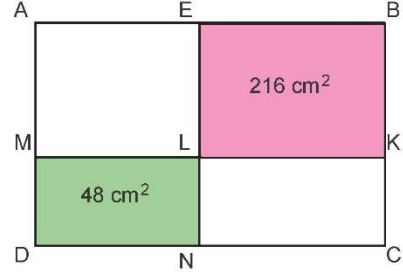
Şekil - IV

Berra, şekilde yazan her sayının asal çarpanlarını bulup etrafındaki üçgenlere yazıyor ve ilk üç şeklin her birinden, birbirinden farklı asal sayılar alarak Şekil IV'e yazıyor ve asal çarpanları bu sayılar olan bir sayı oluşturuyor.

Berra'nın elde ettiği bu sayı aşağıdakilerden hangisi olabilir?

- A) 204 B) 147 C) 385 D) 693

12.



Yukarıda verilen şekilde EBKL, KLNC, MLND dikdörtgen, AELM ise karedir.

ABCD dörtgeninin çevresi en az kaç santimetredir?

- A) 118 B) 112 C) 92 D) 80

13. Aşağıda 2010 yılının ağustos ayına ait bir takvim yaprağı görseli verilmiştir.

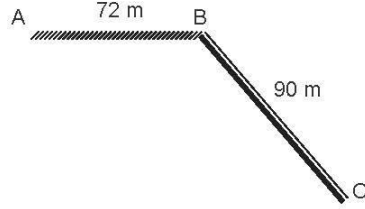
pt	sa	ça	pe	cu	ct	pz
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

Takvim yaprağında işaretlenen gün ile Gülefer'in doğum gününün ortak bölenlerinin en büyüğü 5 ise

Gülefer'in doğum günü aşağıdaki günlerden hangisi olamaz?

- A) Pazar B) Pazartesi C) Cuma D) Cumartesi

14.

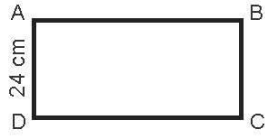


Şekildeki yola eşit aralıklarla aralarındaki mesafe metre cinsinden tam sayı olacak şekilde aydınlatma direkleri dikilecektir.

A, B ve C noktalarına da bu direklerden konması şartıyla gerekli olan toplam direk sayısı aşağıdakilerden hangisi olamaz?

- A) 10 B) 19 C) 29 D) 55

15.



Kısa kenarlarından birisi 24 cm ve çevresi 192 cm olan dikdörtgen şeklindeki bir karton, kenar uzunlukları santimetre cinsinden tam sayı olan eş kare parçalara bölünecektir.

Buna göre eş kare parçaların kenar uzunluklarının alabileceği kaç farklı tam sayı değeri vardır?

- A) 6 B) 8 C) 12 D) 24

16.



Asal çarpanı sadece 2 olan iki basamaklı sayılar torbası En büyük asal çarpanı 3 olan iki basamaklı sayılar torbası En büyük asal çarpanı 5 olan iki basamaklı sayılar torbası

Yukarıda üç torba verilmiştir. Bu torbalarda bulunan sayılarla ilgili bilgiler torbaların altında yazmaktadır.

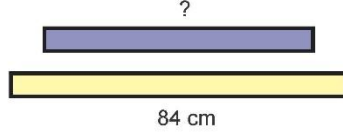
Birinci torbadan A, ikinci torbadan B, üçüncü torbadan C sayısı çekildiğine göre;

- I. $A + B$ en az 41'dir.
II. $A - B$ en çok 52'dir.
III. C'nin alabileceği en büyük ve en küçük değerlerin toplamı 100'dür.

İfadelerinden hangileri doğru olur?

- A) I, II, III B) II ve III C) I ve II D) Yalnız III

17. Aşağıda uzunluğu 84 cm olan ve uzunluğu belli olmayan iki çubuk gösterilmiştir.

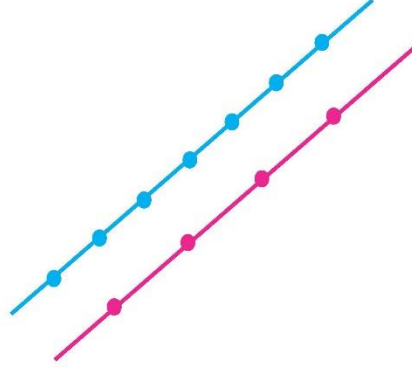


Sündüz, matematik ödevi için görselleri verilen iki çubuğu eşit uzunluklarda keserek uçlarından birleştirmiş ve birbirine eş üç adet kare oluşturmuştur. Bu işlem için toplam 10 kesim yapmıştır.

Buna göre, uzunluğu belli olmayan çubuk kaç santimetredir?

- A) 72 B) 60 C) 48 D) 36

18. Aşağıda eşit uzunluğa sahip mavi ve pembe renkli iki tel verilmiştir.



Mavi tel, parçaların uzunlukları santimetre cinsinden tam sayı olacak şekilde 8 eş parçaya ayrılmak için şekildeki gibi mavi noktalarla işaretleniyor. Pembe tel ise parçaların uzunlukları santimetre cinsinden tam sayı olacak şekilde 5 eş parçaya ayrılmak için şekildeki gibi pembe noktalarla işaretleniyor.

Bu iki tel birer uçları aynı hizada olacak şekilde üst üste konulduğunda mavi ve pembe noktalar arasındaki mesafe en az kaç santimetre olabilir?

- A) 1 B) 5 C) 8 D) 40

19. Hacker Büşra 0 hariç rakamları kullanarak dört basamaklı şifre oluşturacaktır.



Büşra'nın oluşturmak istediği şifrede yan yana gelen rakamlar aralarında asal olduğuna göre şifre aşağıdakilerden hangisi olabilir?

- A) 1397 B) 2465 C) 3456 D) 8936

20.



Yukarıda görünen yüzlerinde iki basamaklı sayıların yazılı olduğu kırmızı ve mavi kartlar verilmiştir. Farklı renkteki kartların görünmeyen yüzleri birbirine yapıştırılarak yeni kartlar elde ediliyor. Oluşan yeni kartların kırmızı ve mavi yüzündeki sayılar aralarında asaldır.

Buna göre oluşturulan yeni kartların herhangi bir yüzeyden görünüşleri aşağıdakilerden hangisi gibi olabilir?

- A)  B)  C)  D) 

EK 3: Araştırma İzni Oluru



T.C.
YOZGAT VALİLİĞİ
İl Millî Eğitim Müdürlüğü

Sayı : E-55005497-605.01-75227809
Konu : Araştırma İzni
(Elif ÇELİK)

28.04.2023

DAĞITIM YERLERİNE

- İlgi : a) Millî Eğitim Bakanlığı'nın 21/01/2020 tarihli ve 1563890 sayılı Araştırma Uygulama İzinleri 2020/2 Nolu Genelgesi.
b) 27/04/2023 tarihli ve E-55005497-20-75187249 sayılı Makam Oluru.
c) Erciyes Üniversitesi Rektörlüğü Öğrenci İşleri Daire Başkanlığı'nın 13.04.2023 tarihli ve E-14065294-044-427264 (74809317) sayılı yazısı.

Erciyes Üniversitesi Eğitim Bilimleri Enstitüsü Matematik ve Fen Bilimleri Eğitimi Anabilim Dalı Matematik Eğitimi Bilim Dalı Yüksek Lisans Programı öğrencilerinden Elif ÇELİK'in "Stratht Kullanımının Matematiksel Problem Çözme Becerilerine Etkisi" konulu tez çalışması hakkındaki ilgi (b) Makam Oluru ekte gönderilmiştir.

Öğrencilerin katılımıyla yapılması planlanan söz konusu uygulamanın 2022-2023 eğitim öğretim yılı içerisinde; denetimi okul idaresinde olmak ve kurum faaliyetlerini aksatmamak kaydıyla; gönüllülük esasına dayalı olarak, onaylı bir örneği Müdürlüğümüzde muhafaza edilen ve uygulama sırasında da mühürlü ve imzalı örnekten çoğaltılan veri toplama araçlarının ilgi (a) Genelge doğrultusunda ilimiz Merkez Gazi Mustafa Kemal İmam-Hatip Ortaokulunda uygulanması hususunda;

Bilgilerinizi ve gereğini rica ederim.

Latif ÇETİNGÖZ
Müdür a.
Şube Müdürü

Ekler:

- 1 - Makam Oluru (1 sayfa)
2 - Mühürlü Form (9 sayfa)

Dağıtım:

Gereği:
Merkez Gazi Mustafa Kemal İHO Müd.ne

Bilgi:
Din Öğretimi Şube Müd.ne

Bu belge güvenli elektronik imza ile imzalanmıştır.

Adres : Yeni Valilik Binası İl Millî Eğitim Müdürlüğü
Strateji Geliştirme Şube Müdürlüğü (AR-GE)
Telefon No : 0 (354) 280 66 78
E-Posta: arge66@meb.gov.tr
Kep Adresi : meb@hs01.kep.tr

Belge Doğrulama Adresi : <https://www.turkiye.gov.tr/meb-ebys>
Bilgi için: E.YILDIZ
Unvan : Teknisyen
Faks: 0 (354) 280 66 69
İnternet Adresi: yozgat.meb.gov.tr

Bu evrak güvenli elektronik imza ile imzalanmıştır. <https://evraksorgu.meb.gov.tr> adresinden 2546-10e5-32f2-8dac-05e6 kodu ile teyit edilebilir.

EK 4: Wilcoxon Testi Bulguları

Wilcoxon İşaretli Sıralar Testi İstatistiksel Sonuçları

Öntest-sontest	N	Sıra Ortalaması	Sıra Toplamı	z	p
Negatif Sıralar	2	5	10	-1,15	,251
Pozitif Sıralar	6	4,33	26		
Eşit	0				
Toplam	8				

Wilcoxon test sonuçlarına göre sontest değerleri ile öntest değerleri arasında anlamlı bir fark olmadığı görülmüştür. $z=-1,15$, $p > .05$

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı:	Elif Çelik
Uyruğu:	Türkiye Cumhuriyeti

EĞİTİM

Derece	Kurum	Mezuniyet Tarihi
Yüksek Lisans	Erciyes Üniversitesi Eğitim Bilimleri Enstitüsü- Matematik Eğitimi Bölümü	2024
Lisans	Erciyes Üniversitesi- İlköğretim Matematik Öğretmenliği	2012
Lise	Yozgat Lisesi	2007

İŞ DENEYİMLERİ

Yıl	Kurum	Görev
2020-Halen	Yozgat Gazi Mustafa Kemal İmam Hatip Ortaokulu	Matematik Öğretmeni
2015-2020	Yozgat Saray Yibitaş Lafarge Ortaokulu	Matematik Öğretmeni
2012-2015	Menderes Ortaokulu	Matematik öğretmeni

YABANCI DİL

İngilizce (90)

YAYINLAR

Sevgi, S., ve Çelik, E. (2024). Scratch Kullanımının Matematiksel Problem Çözme Becerilerine Etkisi. *XI International Eurasian Educational Research Congress*.