



**2023-2024 Eğitim-Öğretim Yılı  
Başiskele Gübretaş Ortaokulu  
Bilişim Teknolojileri ve Yazılım Dersi  
6.Sınıf**

**Algoritma ve Programlama**

# Ders İeriđi:

- Problem özme(yazılım, problem).
- Algoritma kavramı.
- Bir problem özümünde algoritmayı kullanma.
- Akıř řeması bileřenleri ve iřlevleri.
- Akıř řeması çizme.
- Scratch arayüzü ve kod blokları tanıtımı.
- Programlama ile ilgili temel kavramlar.
  - a) Döngü yapısı
  - b) Karakteri Hareket Ettirme, başlatıcılar
  - c) Karakterin Yönü
  - d) Sahnede konum
  - e) Karakteri yön tuřları ile hareket ettirme
  - f) Karar-Mantık yapısı
  - g) Deđişkenler
  - h) Sayaçlar
- Projeler

**YAZILIM:** Çeşitli görevleri gerçekleştirmek amacıyla hazırlanmış programlara yazılım adı verilir. Her yazılım bir **problemi** çözmek amacıyla geliştirilmiştir.

➤ **PROBLEM:** çözümlmesi gereken sorun ya da aşılması gereken engel anlamına gelir.

➤ Bir Problemin Çözümü İçin:

- Problemi iyi anlamak
- Sonucun doğruluğunu kontrol etmek.
- Kısa ve anlaşılır şekilde çözmek.

➤ Günlük yaşamda karşılaştığımız problemleri bilerek veya farkında olmadan **adım adım** çözmeye çalışırız.

➤ Örneğin yazı yazarken kaleminizin ucu kırıldığında şu adımları takip ederek bu sorunu çözersiniz.

1. **Kalemıraşı çıkar.**
2. **Kalemi al.**
3. **Çöp kovasının yanına git.**
4. **Kalemin ucunu aç.**
5. **Sırana geri dön.**
6. **Yazmaya devam et.**



# Kurt, Kuzu, Ot Problemi



- Tekneye her defasında sadece 1 tanesi alınabiliyor.
- Yani ya kurdu, ya kuzuyu yada otu almanız gerekiyor.
- Eğer kurt ile kuzuyu baş başa bırakırsanız, kurt kuzuyu yer.
- Eğer kuzu ile otu baş başa bırakırsanız, kuzu otu yer.

**Çözümü yapalım:** <https://www.rekoroyun.com/kurt-kuzu-ot.html>

# Kurt, Kuzu, Ot Problemi Çözümü



- ▶ 1) Kuzuyu karşıya geçir.
- ▶ 2) Geri dön.
- ▶ 3) Otu karşıya geçir.
- ▶ 4) Kuzuyu geri getir.
- ▶ 5)Kurdu karşıya geçir.
- ▶ 6) Geri dön.
- ▶ 7)Kuzuyu karşıya geçir.

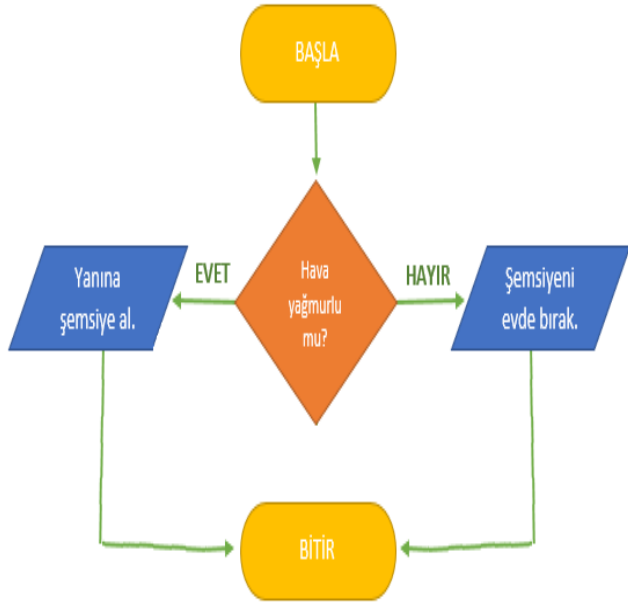
**Çözümü test edelim:** <https://www.rekoroyun.com/kurt-kuzu-ot.html>

Bilgisayarlar da problemleri tıpkı bizler gibi adım adım çözmeye çalışır.

Kullandığımız yazılımların tamamı «**kod**» adı verilen bilgisayarın anlayacağı dilde yazılmış özel komutlardan oluşur.

Bu kodlar bilgisayar yazılımcıları tarafından yazılır.

```
31 self.file = None
32 self.fingerprints = set()
33 self.logdupes = True
34 self.debug = debug
35 self.logger = logging.getLogger(__name__)
36 if path:
37     self.file = open(os.path.join(path, 'requests.txt'),
38                     'a')
39     self.file.seek(0)
40     self.fingerprints.update(set(requests))
41
42 @classmethod
43 def from_settings(cls, settings):
44     debug = settings.getbool('debug', True)
45     return cls(job_dir(settings), debug)
46
47 def request_seen(self, request):
48     fp = self.request_fingerprint(request)
49     if fp in self.fingerprints:
50         return True
51     self.fingerprints.add(fp)
52     if self.file:
53         self.file.write(fp + os.linesep)
54
55 def request_fingerprint(self, request):
56     return request_fingerprint(request)
```



Kodlamaya başlamadan önce oluşturacağımız yazılımın adım adım ne yapacağını tasarlamamız gerekir.

İşte açık ve net ifadelerle problemin adım adım çözümünü gösteren bu taslağa «**algoritma**» adı verilir.

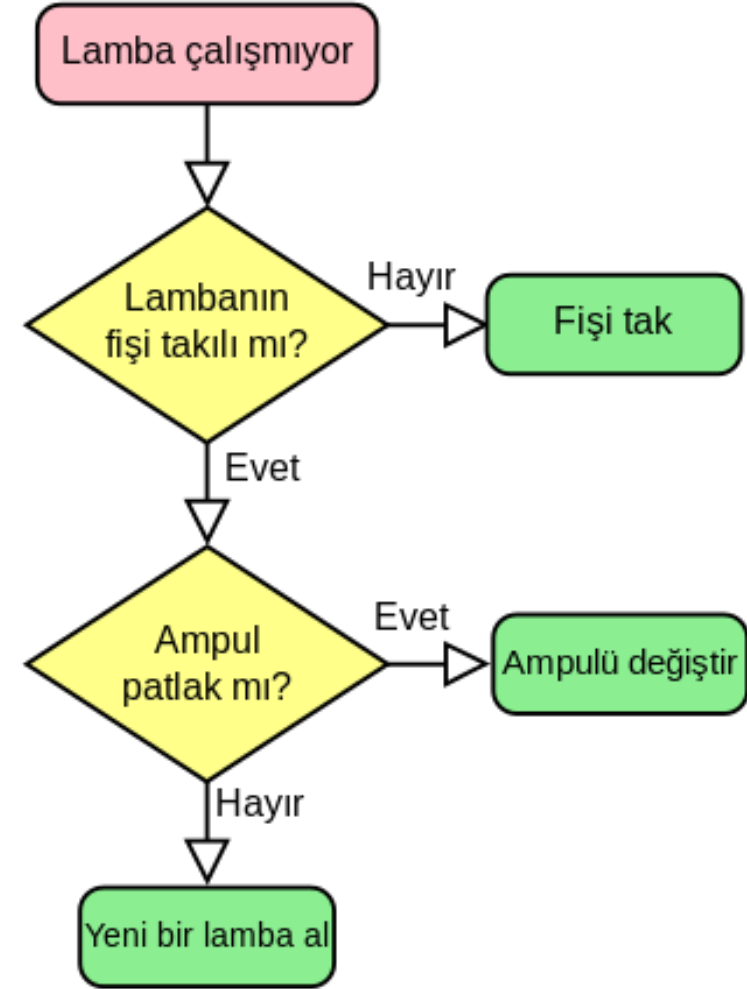
Programlamanın ilk adımı algoritma oluşturmaktır.

► **ALGORİTMA:** Bir problemin çözümünde izlenecek yol anlamına gelir ve problemin çözümünün adımlar halinde yazılmasıyla oluşturulur.

► Algoritma basamaklarının bir başlangıcı ve sonu bulunur.





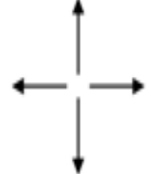


► Her adımda yapılacak işlem açıkça belirtilir.

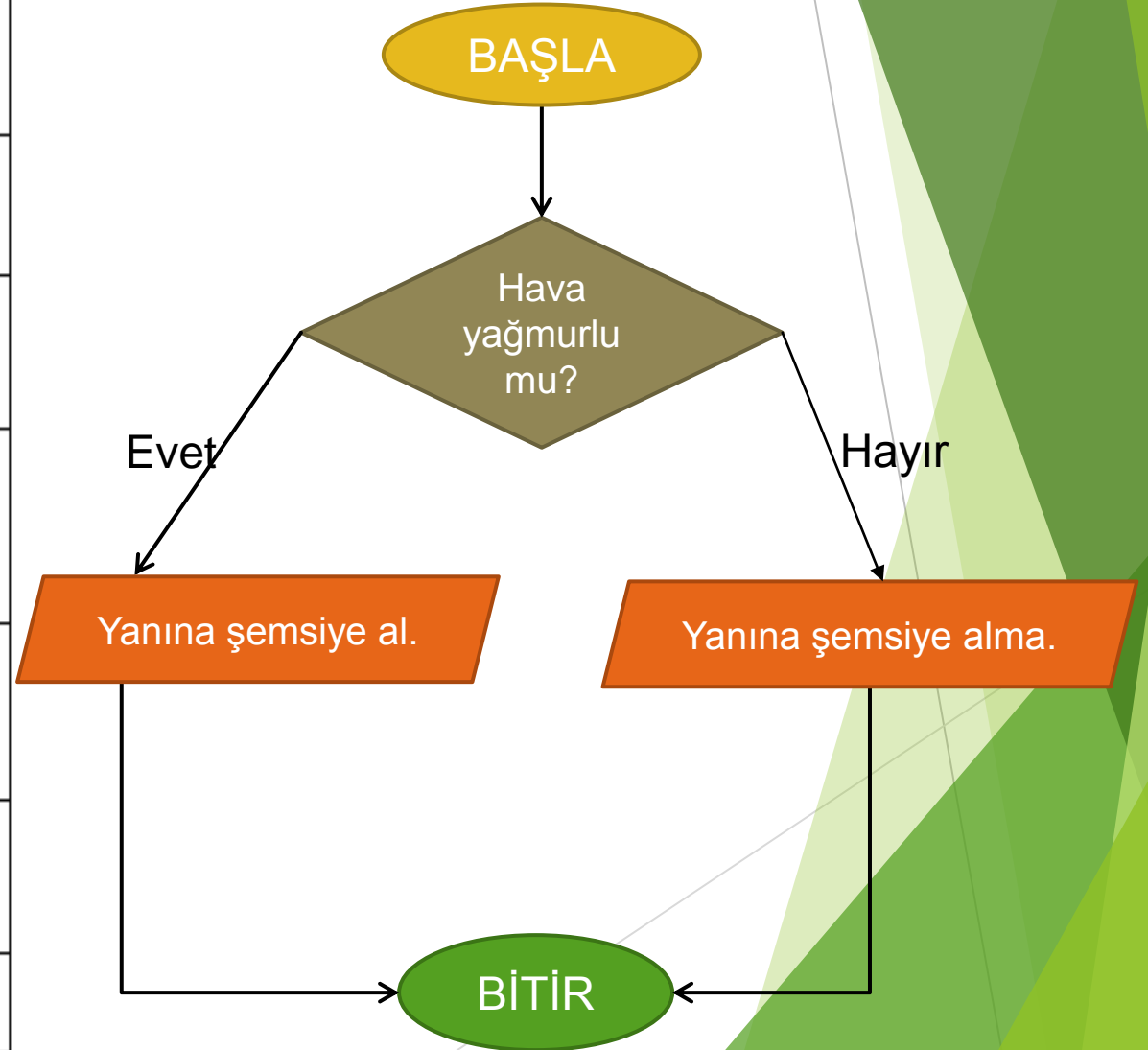
► Oluşturacağımız yazılımın kusursuz olması için öncelikle her adımını gösteren planını, yani algoritmasını hazırlamalıyız.





# AKIŞ DİYAGRAM ŞEKİLLERİ

	Programın başlangıç ve bitiş için kullanılır.
	Bilgi giriş çıkışı için kullanılır.
	Aktarma, aritmetik hesaplama, işlem
	Karar alma için kullanılır.
	Birleştirme çizgileri
	Yazdır
	Bağlantı



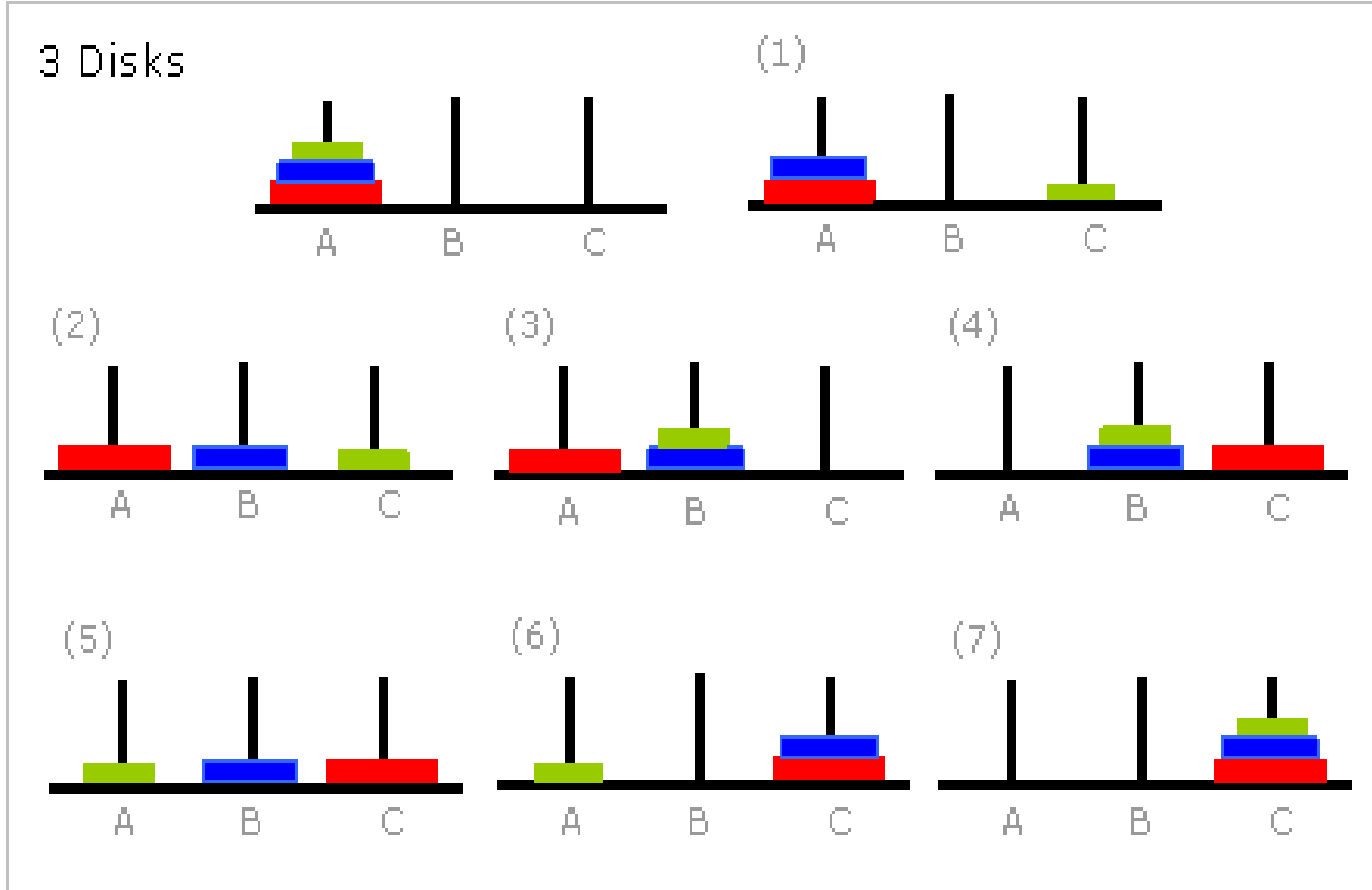
# Hanoi Kuleleri



**Problemi çözelim:** <https://www.rekoroyun.com/hanoi-kuleleri.html>

- Hanoi kuleleri oyununda amaç; En sol taraftaki kulede bulunan diskleri, en sağ tarafta bulunan kuleye büyükten küçüğe doğru olacak şekilde yerleştirmektir.
- Küçük disklerin üstüne kendinden büyük disk koyamazsınız.
- Oyun en az 3 en fazla 6 disk ile oynanmaktadır. Disk sayısı arttıkça oyun zorlaşmaktadır.
- Disk sayısı arttıkça minimum hamle sayınızda buna göre değişmektedir. *Ör: 3 diskte sizden 7 hamlede yapmanız istenirken, 6 diskte ise 63 hamlede tamamlamanız istenmektedir.*
- Oyunda geri alma yoktur. Bu yüzden hamlelerinizi dikkatli yapınız.
- Her hamlede sadece 1 parçayı taşıyabilirsiniz.

# Hanoi Kuleleri Çözüm



- 1) Yeşili C'ye götür.
- 2) Maviyi B'ye götür.
- 3) Yeşili B'ye götür.
- 4) Kırmızıyı C'ye götür.
- 5) Yeşili A'ya götür.
- 6) Maviyi C'ye götür.
- 7) Yeşili C'ye götür.

Algoritmayı test edelim: <https://www.rekoroyun.com/hanoi-kuleleri.html>

**Ödev:** Hanoi kuleleri oyununun 4 diskli olan bölümünü çözerek problemin algoritmasını liste halinde yazmak.



- 1)
- 2)
- 3)
- 4)
- 5)
- 6)
- 7)
- 8)
- 9)
- 10)
- 11)
- 12)
- 13)
- 14)
- 15)

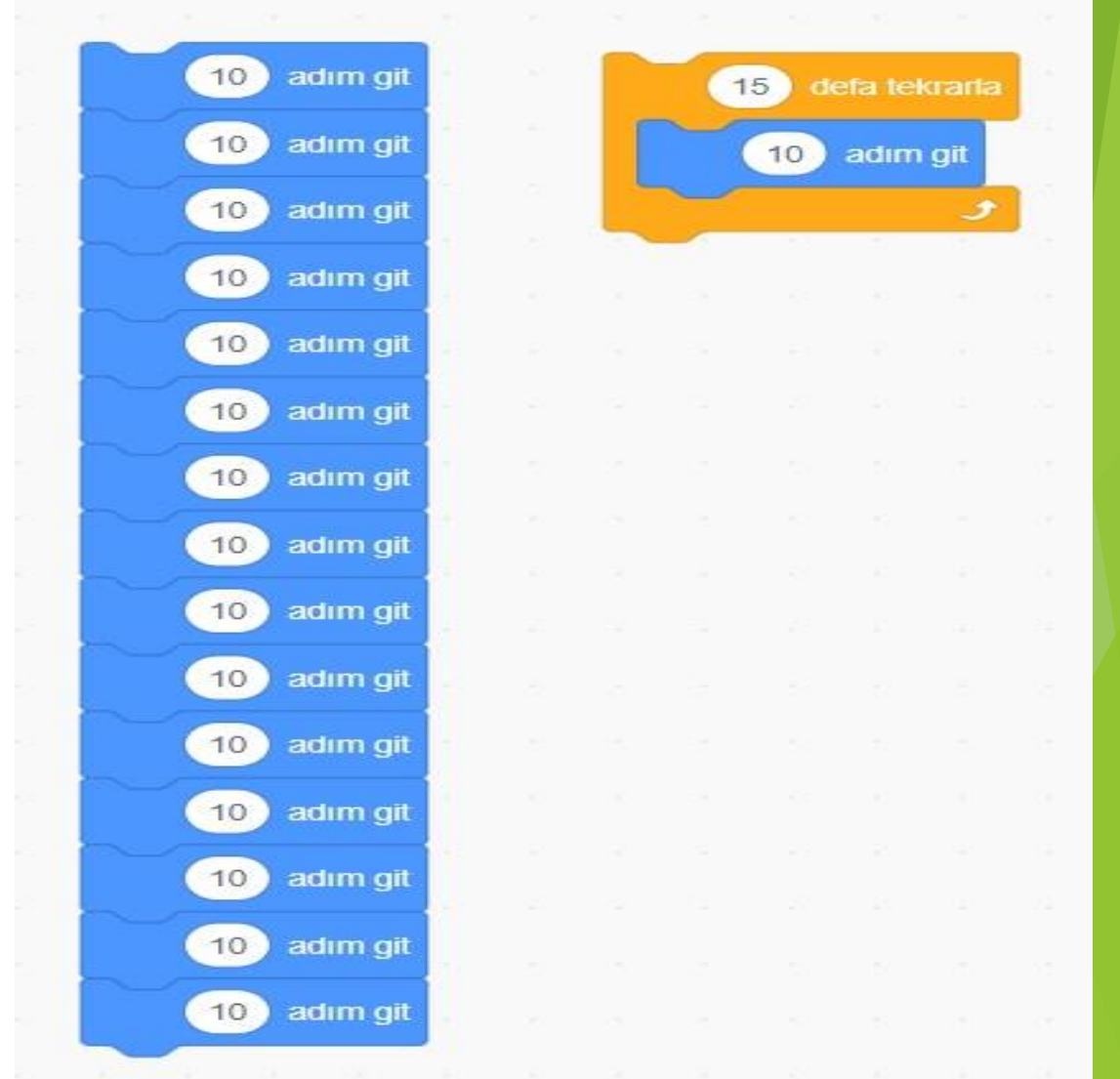
# Scratch Arayüz ve Bloklar

Aşağıdaki bağlantıdan tüm kodlara ulaşabilirsiniz.

<https://blockodlama.com/scratch-3-kod-bloklari-tanitimi/>

# PROGRAMLAMANNIN TEMELLERİ

- **a) Döngü:** Bir veya birden fazla işlemi, bir koşula bağlı olarak, belirli sayıda veya bir koşul sağlandığı sürece tekrarlayarak çalıştıran kalıplara **döngü** adı verilir.



# DÖNGÜ TÜRLERİ



Yazılan sayı kadar içerisinde ki blokları tekrarlatır.

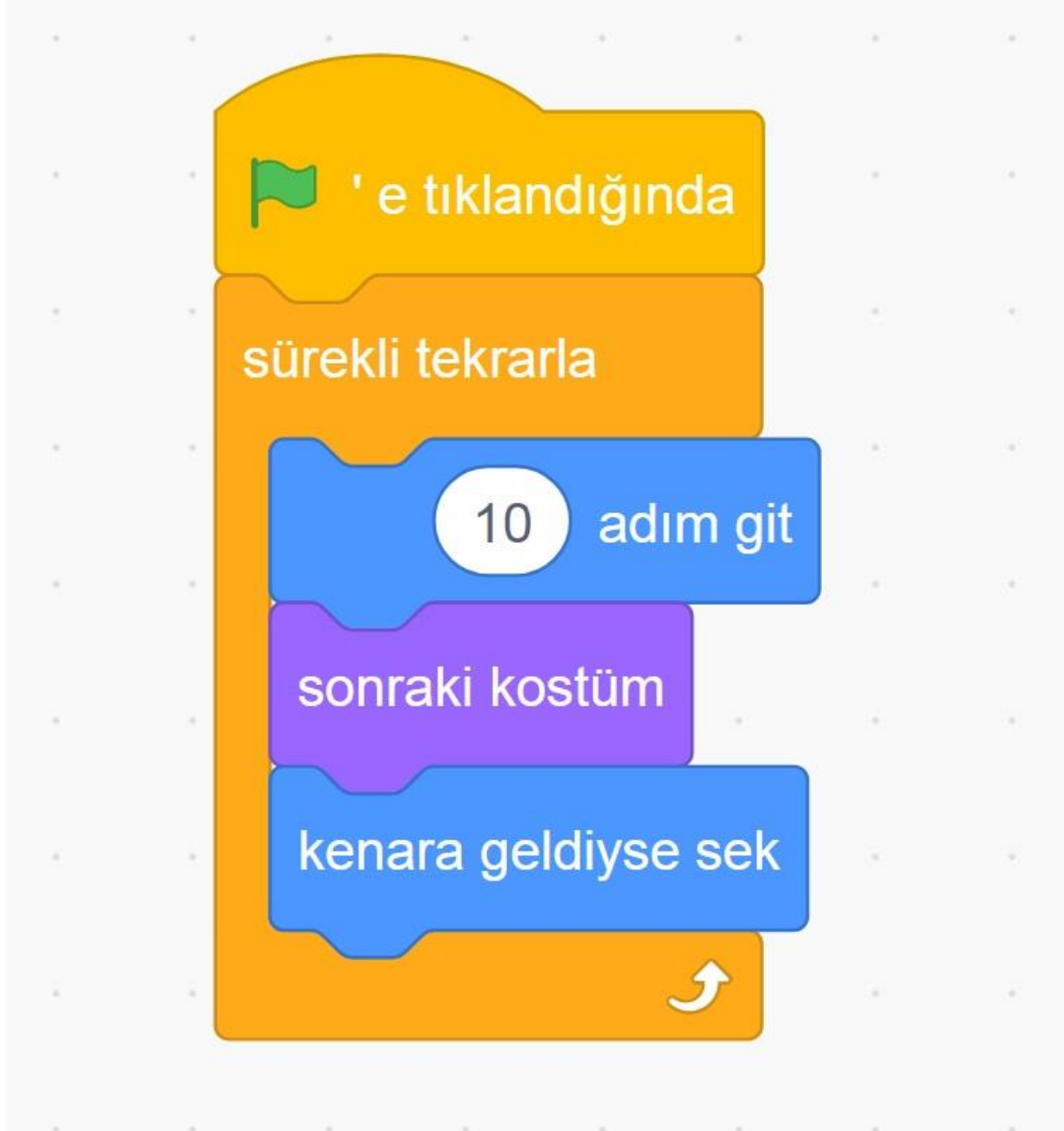


Program durdurulana kadar içerisinde ki blokları tekrarlatır.



Yazılan koşul gerçekleşene kadar içerisinde ki blokları tekrarlatır. Koşul gerçekleştiği anda tekrarlatmayı durdurur.

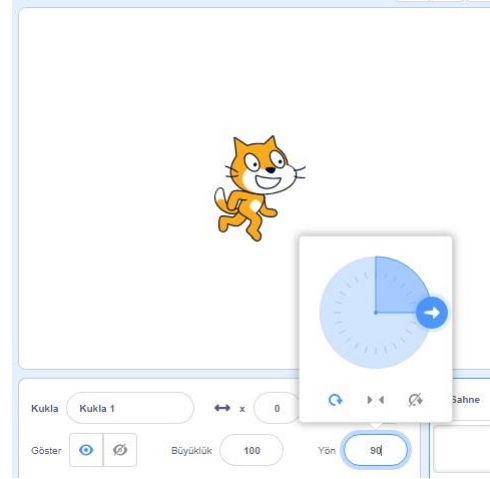
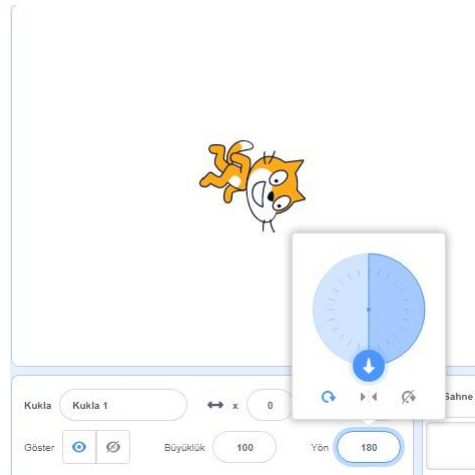
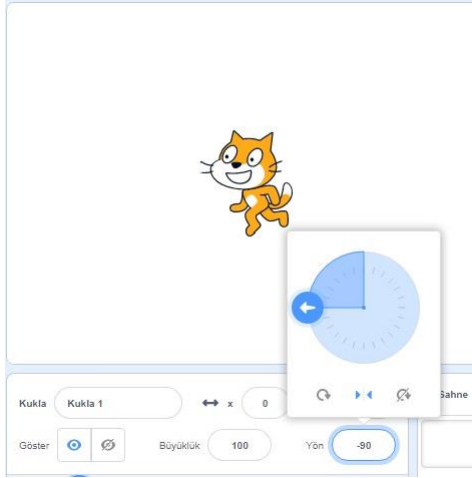
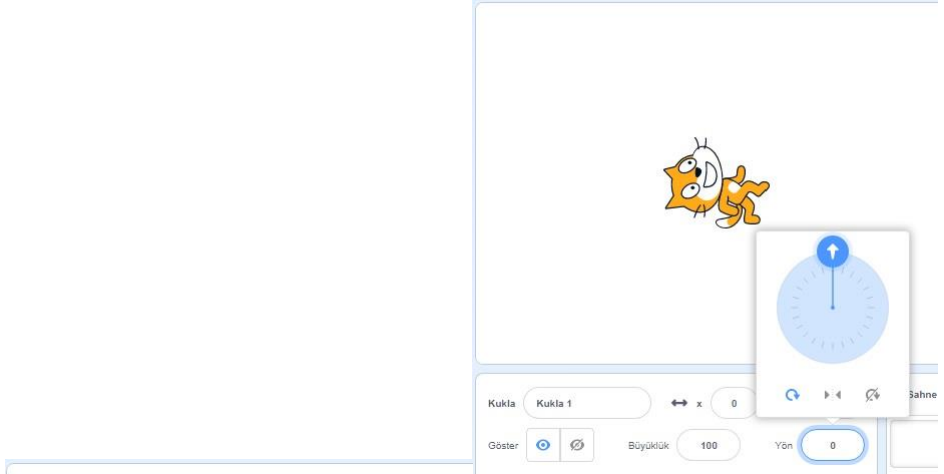
## b) Karakteri Hareket Ettirme, Başlatıcılar



Yeşil bayrak tıklandığında oyun başlar.  
Program durdurulana kadar sürekli tekrarlar bloğu çalışır.  
Her çalıştığında sonraki kostüme gelir.  
Karakter sahnenin köşesine geldiğinde geri döner.



## c) Karakterin Yönü



Karakterimizin sahnede hangi yöne bakacağı 0 - 180 derece ve 0 - -179 derece arasında belirlenir.

0 derece ekrana göre yukarı bakar.

90 derece ekrana göre sağa bakar.

180 derece ekrana göre aşağı bakar.

-90 derece ekrana göre sola bakar.

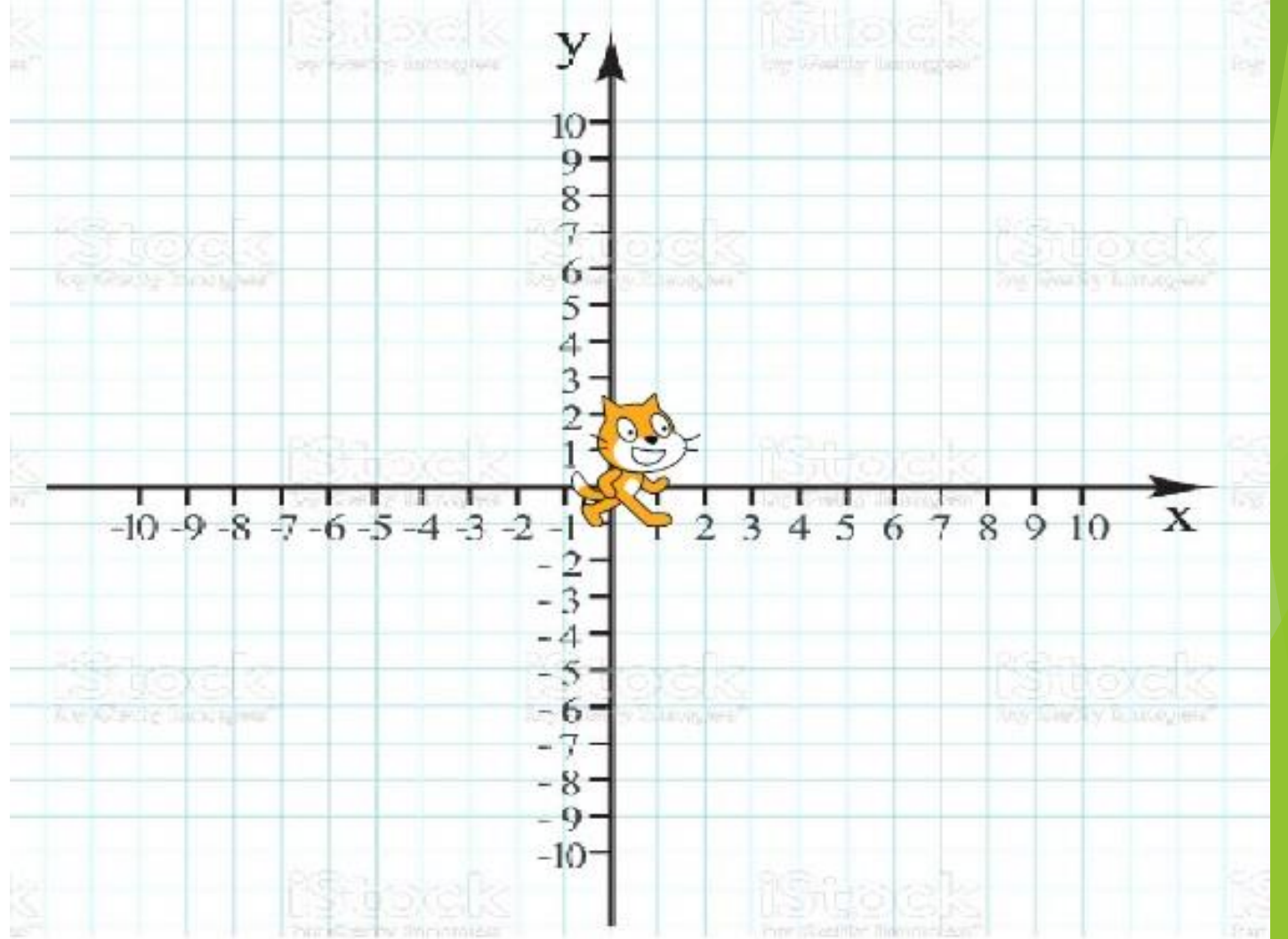
## d) Sahnede Konum

Sahnede yer alan karakterimizin konumu önce x eksenine, sonra y eksenine göre belirlenir.

Tam orta noktasında olmasını istiyorsak (0,0) noktasına getirmeliyiz.

X değeri artarsa karakterimiz sağa, azalırsa sola hareket eder. Y değeri artarsa karakterimiz yukarı, azalırsa aşağı yönde hareket eder.

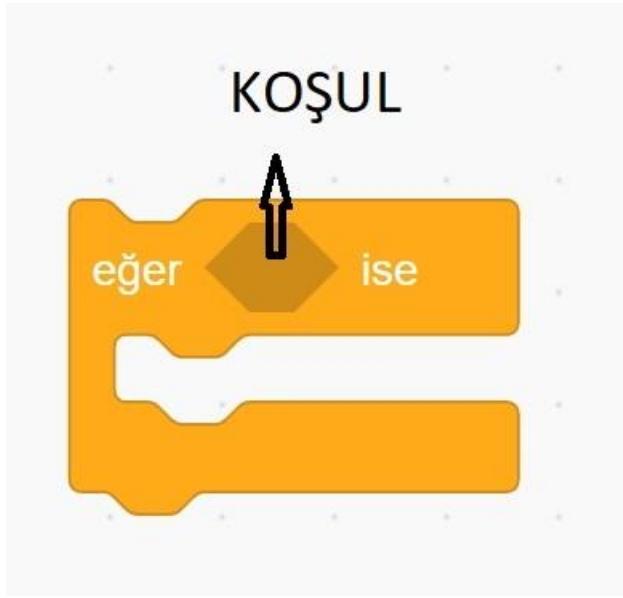
Çapraz yönlerde hareket ettirmek istersek x ve y değerlerini aynı anda artırmalı veya azaltmalıyız.



## e) Karar - Mantık Yapısı

- Bilgisayara iki yada daha fazla seçenek arasından seçim yaptırmak için kurulan mantık yapılarıdır.



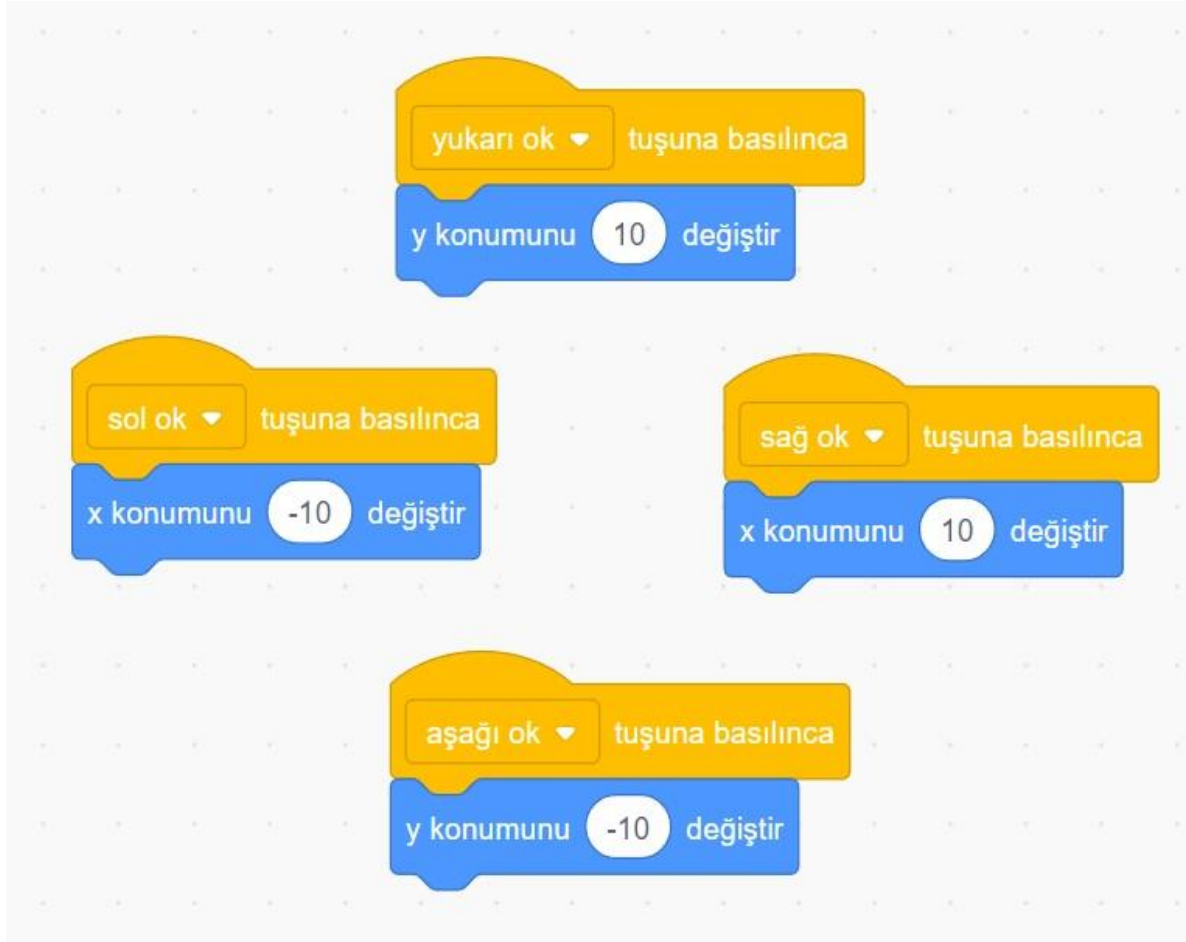


Yazılan koşul **doğru** ise içerisindeki blokları çalıştırır, **yanlış** ise blokları çalıştırmadan alttaki komutlardan devam eder.

Koşul gerçekleşir ise ilk boşlukta yer alan bloklar çalışır, gerçekleşmez ise ikinci boşlukta yer alan bloklar çalışır.

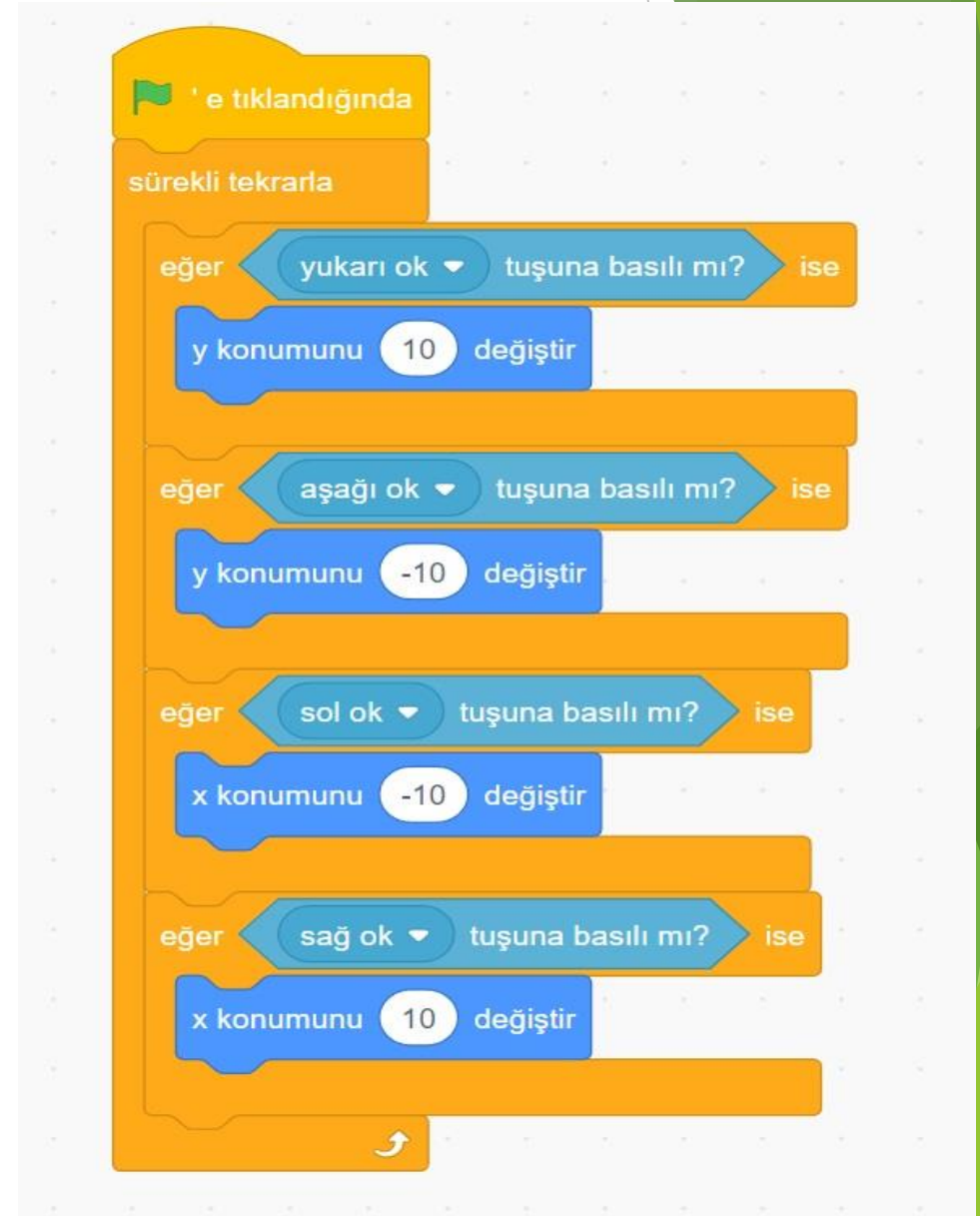


# Yön Tuşları İle Hareket Ettirme



Scratch code blocks for movement:

- Yukarı ok tuşuna basılınca: y konumunu 10 değiştir
- Sol ok tuşuna basılınca: x konumunu -10 değiştir
- Sağ ok tuşuna basılınca: x konumunu 10 değiştir
- Aşağı ok tuşuna basılınca: y konumunu -10 değiştir

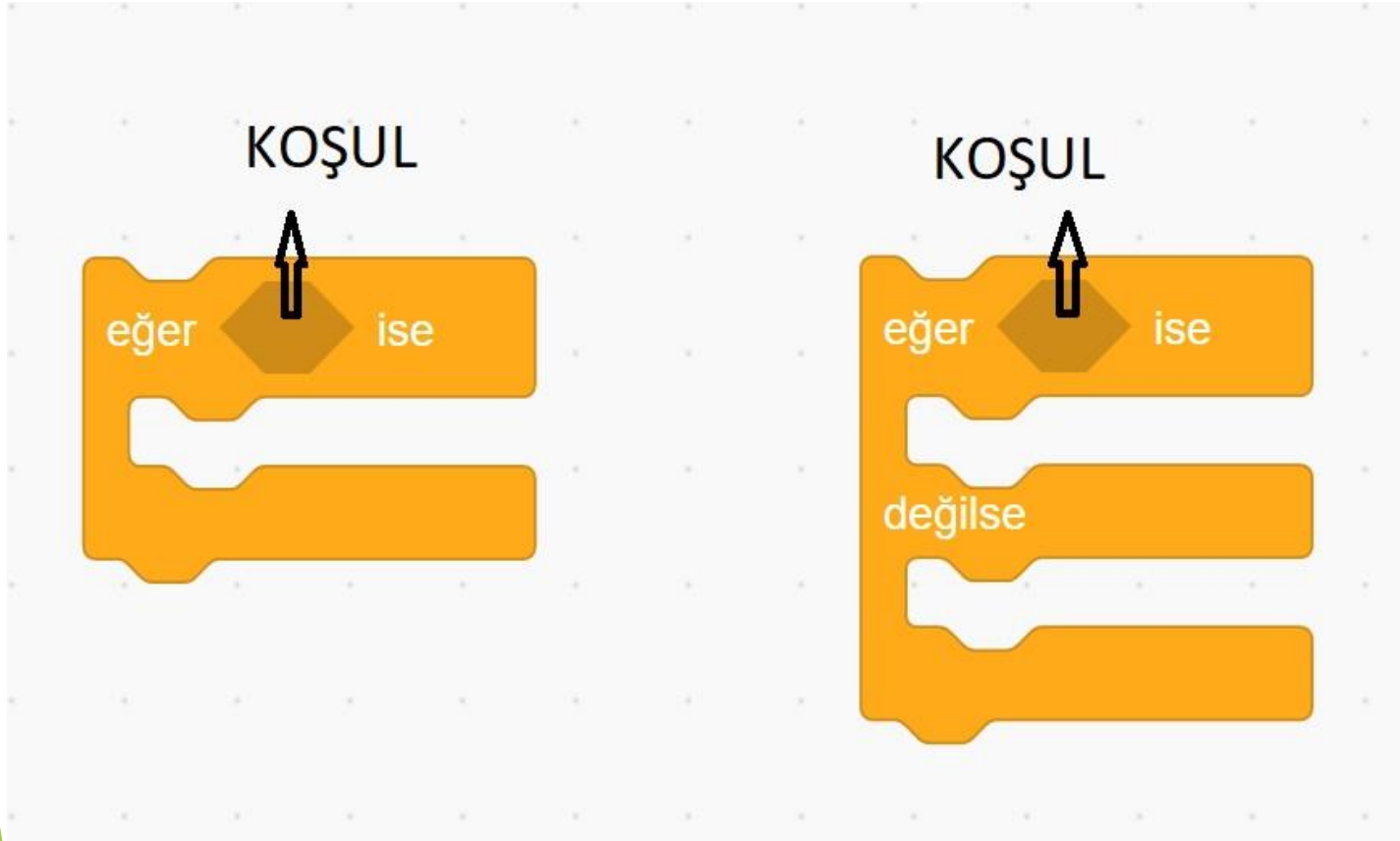


Scratch code blocks for continuous movement:

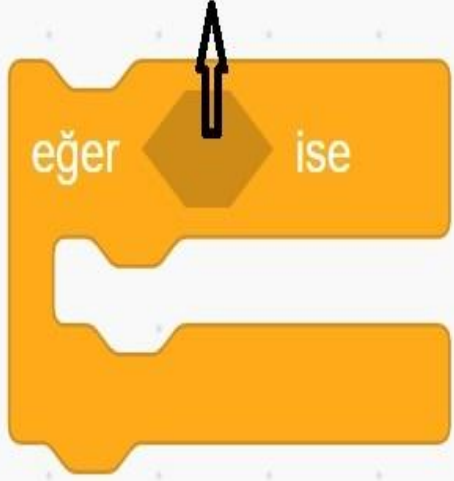
- Bayrak 'e tıkladığında
- Sürekli tekrarla
- Eğer yukarı ok tuşuna basılı mı? ise: y konumunu 10 değiştir
- Eğer aşağı ok tuşuna basılı mı? ise: y konumunu -10 değiştir
- Eğer sol ok tuşuna basılı mı? ise: x konumunu -10 değiştir
- Eğer sağ ok tuşuna basılı mı? ise: x konumunu 10 değiştir

## f) Karar Mantık Yapısı

- Bilgisayara iki yada daha fazla seçenek arasından seçim yaptırmak için kurulan mantık yapılarıdır.



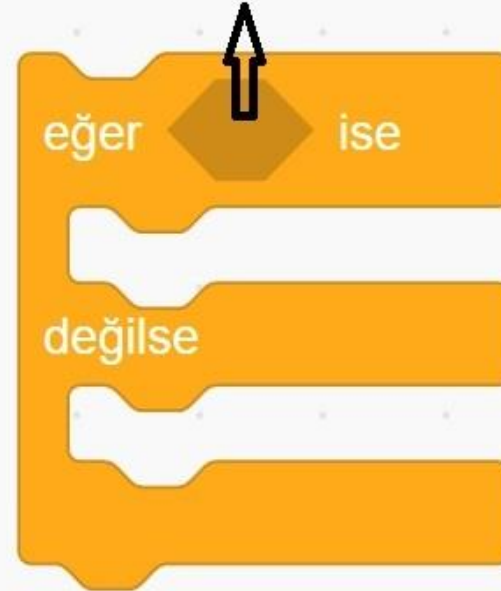
KOŞUL



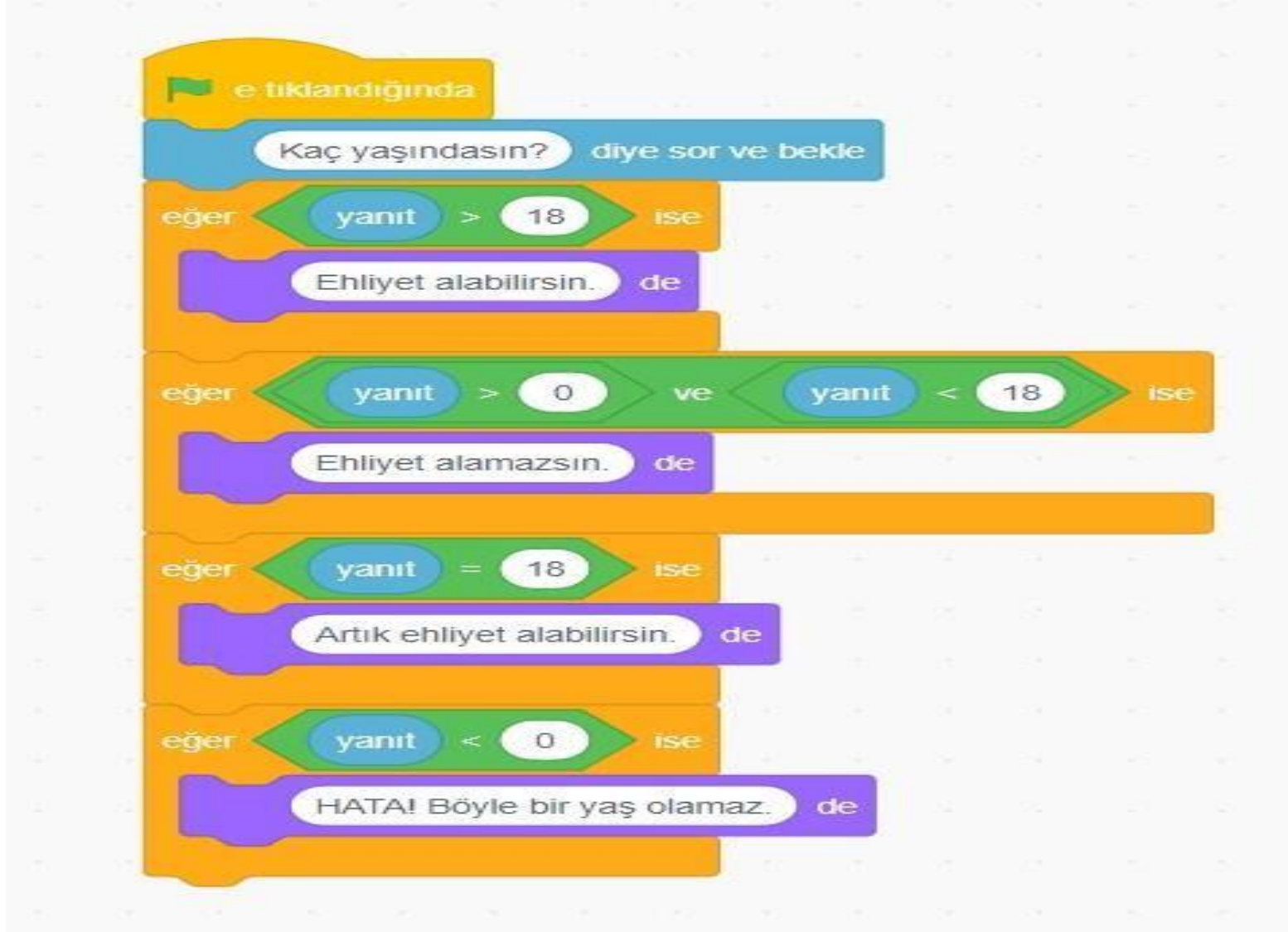
Yazılan koşul **doğru** ise içerisindeki blokları çalıştırır, **yanlış** ise blokları çalıştırmadan alttaki komutlardan devam eder.

Koşul gerçekleşir ise ilk boşlukta yer alan bloklar çalışır, gerçekleşmez ise ikinci boşlukta yer alan bloklar çalışır.

KOŞUL



# Ehliyet Alabilir Misin?





## g) Değişkenler

- ▶ Program, oyun sürecinde girilen sayı, yazı, doğru-yanlış, tarih vb. değerleri tutan programlama birimleridir.
- ▶ Değişkenleri biz tanımlarız ve isim veririz.
- ▶ Aynı isimde birden fazla değişken olamaz.

The image shows a screenshot of the 'Değişkenler' (Variables) panel in a programming environment. The panel has a title 'Değişkenler' and a button 'Bir Değişken Oluştur'. Below this, there is a checkbox labeled 'değişkenim'. Underneath the checkbox, there are four orange blocks, each with a dropdown menu showing 'değişkenim' and a button. The first block has a value of '0' and a button labeled 'yap'. The second block has a value of '1' and a button labeled 'kadar değiştir'. The third block has a button labeled 'değişkenini göster'. The fourth block has a button labeled 'değişkenini gizle'.

## h) Sayaç



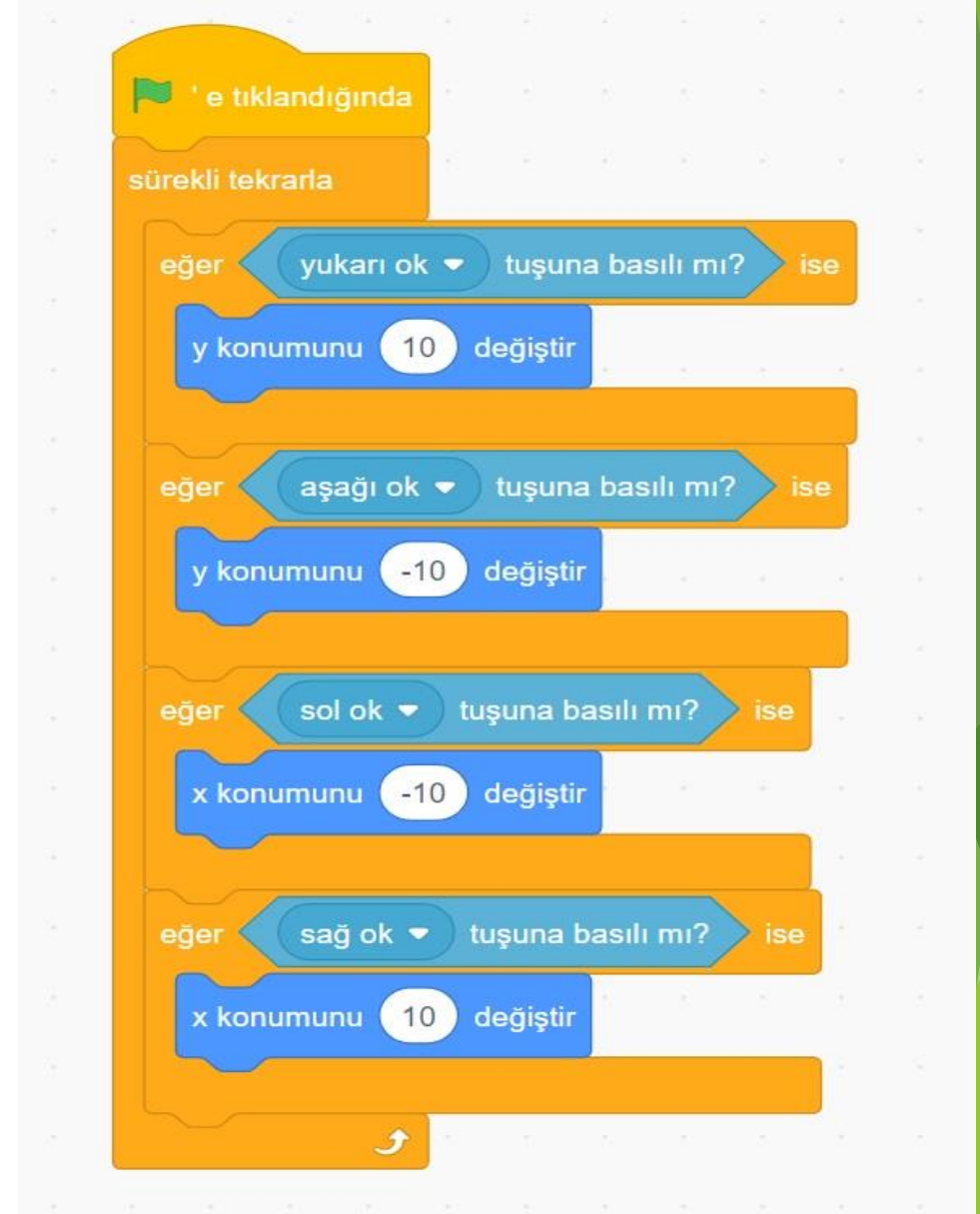
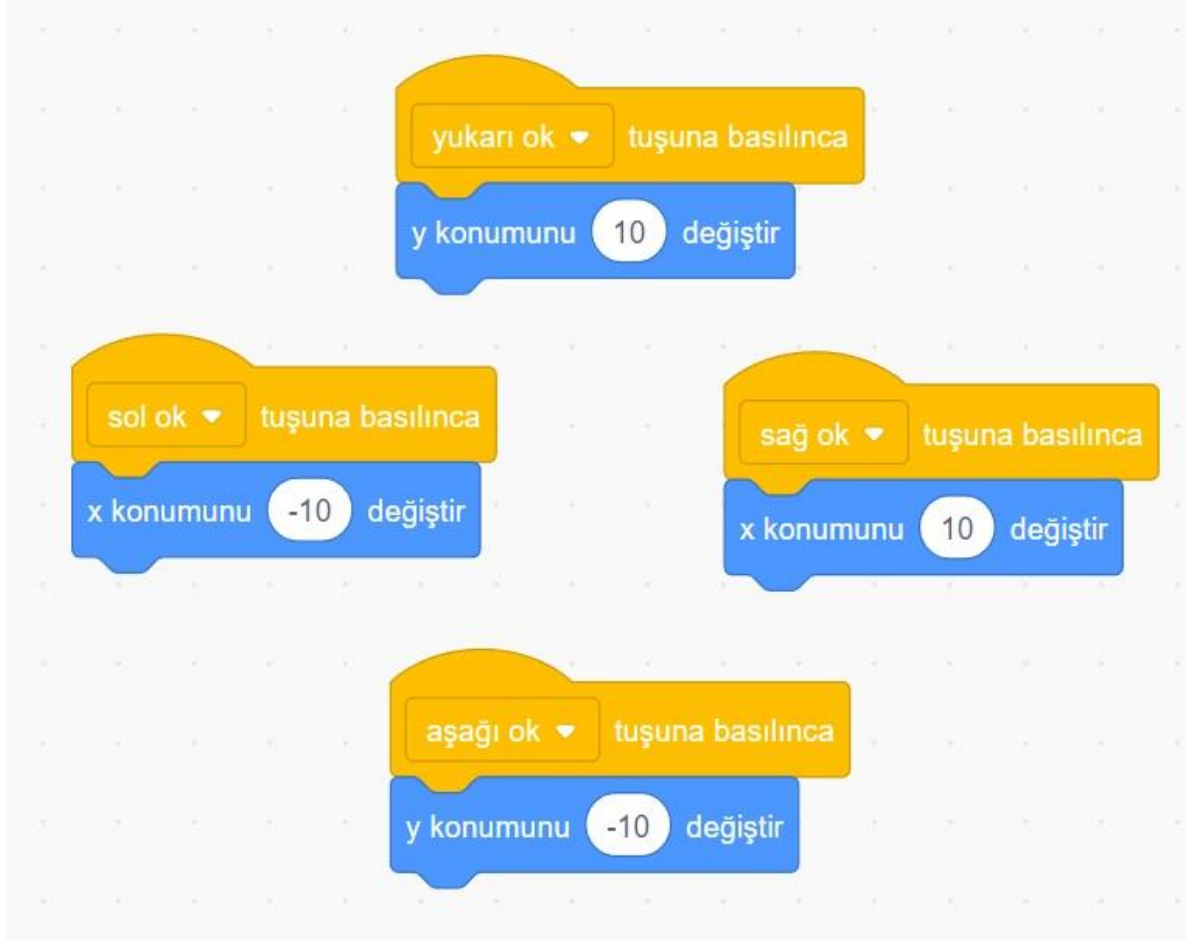
Yeşil bayrak tıklandığında program başlayarak her bir saniyede süre değişkenini 1 artırır. Bu sayede sayaç/kronometre programlanmış olur.

# Projeler

# 1) Karakteri Hareket Ettirme



## 2) Karakteri Tuşlar İle Hareket Ettirme



### 3) Geçti / Kaldı



## 4) Bil Bilebilirsen (Sayı Tahmin)

The image shows a Scratch script for a number guessing game. The script starts with a 'when clicked' event block. It then sets a random number between 0 and 100 to a variable named 'rastgelesayi'. Next, it sets a variable named 'tahmin' to 0. A 'repeat' loop is used to run the following steps:

- Ask the user for a number between 0 and 100 and wait for a response.
- Increment the 'tahmin' variable by 1.
- Check if the user's answer ('yanıt') is greater than the random number ('rastgelesayi'). If true, play a 'Bonk' sound and display 'Daha küçük bir sayı yaz' for 2 seconds.
- Check if the user's answer ('yanıt') is less than the random number ('rastgelesayi'). If true, play a 'Bonk' sound and display 'Daha büyük bir sayı yaz' for 2 seconds.
- Check if the user's answer ('yanıt') is equal to the random number ('rastgelesayi'). If true, play a 'Tada' sound, display 'Tebrikler' for 3 seconds, and stop the script.

```
when clicked
  rastgelesayi = random number from 0 to 100
  tahmin = 0
  repeat
    ask "0 ile 100 arasında bir sayı seç. diye sor ve bekle"
    tahmin = tahmin + 1
    if yanıt > rastgelesayi
      play sound Bonk
      say "Daha küçük bir sayı yaz" for 2 seconds
    if yanıt < rastgelesayi
      play sound Bonk
      say "Daha büyük bir sayı yaz" for 2 seconds
    if yanıt = rastgelesayi
      say "Tebrikler" for 3 seconds
      play sound Tada
      stop all
```

# 5) Futbol Oyunu

```
when green flag clicked
  x: 0 y: 0 konumuna git
  Kırmızı değişkenini 0 yap
  Mavi değişkenini 0 yap
  Referee Whistle sesini bitene kadar çal
  1 ile 360 arasında rastgele bir sayı seç yönüne dön
  sürekli tekrarla
    10 adım git
    kenara geldiyse sek
    eğer Line e değişiyor mu? ise
      45 ile 135 arasında rastgele bir sayı seç derece dön
    eğer Line2 e değişiyor mu? ise
      45 ile 135 arasında rastgele bir sayı seç derece dön
    eğer MaviKale e değişiyor mu? ise
      Kırmızı i 1 kadar değiştir
      Goal Cheer sesini bitene kadar çal
      x: 0 y: 0 konumuna git
      1 ile 360 arasında rastgele bir sayı seç yönüne dön
    eğer KırmızıKale e değişiyor mu? ise
      Mavi i 1 kadar değiştir
      Goal Cheer sesini bitene kadar çal
      x: 0 y: 0 konumuna git
      1 ile 360 arasında rastgele bir sayı seç yönüne dön
```

```
when green flag clicked
  Süre değişkenini 45 yap
  sürekli tekrarla
    1 saniye bekle
    Süre i -1 kadar değiştir
    eğer Süre = 0 ise
      durdur tümünü
```

```
when green flag clicked
  x: 0 y: -150 konumuna git
  sürekli tekrarla
    eğer a tuşuna basıldı mı? ise
      x konumunu -10 değiştir
    eğer d tuşuna basıldı mı? ise
      x konumunu 10 değiştir
```

```
when green flag clicked
  x: 0 y: 145 konumuna git
  sürekli tekrarla
    eğer 4 tuşuna basıldı mı? ise
      x konumunu -10 değiştir
    eğer 6 tuşuna basıldı mı? ise
      x konumunu 10 değiştir
```



**Aşağıdaki kelimeleri tanımlardaki uygun yerlere yerleştiriniz.**

**Problem**

**Akış Şeması**

**Yazılım**

**Algoritma**

- .....: Çeşitli görevleri gerçekleştirmek amacıyla hazırlanmış programlara denir.
- .....: Çözülmesi gereken sorun yada aşılması gereken engeldir.
- .....: Bir problemin çözümünde izlenecek yol anlamına gelir ve problemin adımlar halinde yazılması ile oluşur.
- .....: Bilgisayar programlarının işlem basamaklarınınin geometrik şekillerle gösteren şemadır.

Aşağıdaki problemin algoritmasını liste halinde yanına yazınız.

## KURT, KUZU VE OT

Ahmet Amca'nın çiftliği köyün biraz dışında Kızıldere'nin hemen öbür yanındaymış. Ahmet Amca bir gün kuzusunu, ormandan bahçesine inen kurdu ve kuzusu için ayırdığı bir miktar otu da alıp karşı kıyıya geçmek istemiş. Ancak karşıya geçebileceği tek araç ufacık bir kayıkmış ve hepsinin beraber karşıya geçmesi imkansızmış. Kayığa her defasında birini alabiliyormuş; ya kuzuyu ya kurdu ya da otu yanına alabilecekmış. Ancak bir sorunu daha varmış, kurtla kuzuyu yalnız bırakırsa kurt kuzuyu yemiş, kuzuyla otu yalnız bıraksa bu sefer kuzu da otları yemiş. Peki sizce nasıl olacak dva Ahmet Amca üçünü birden karşıya geçirecek?

1)

2)

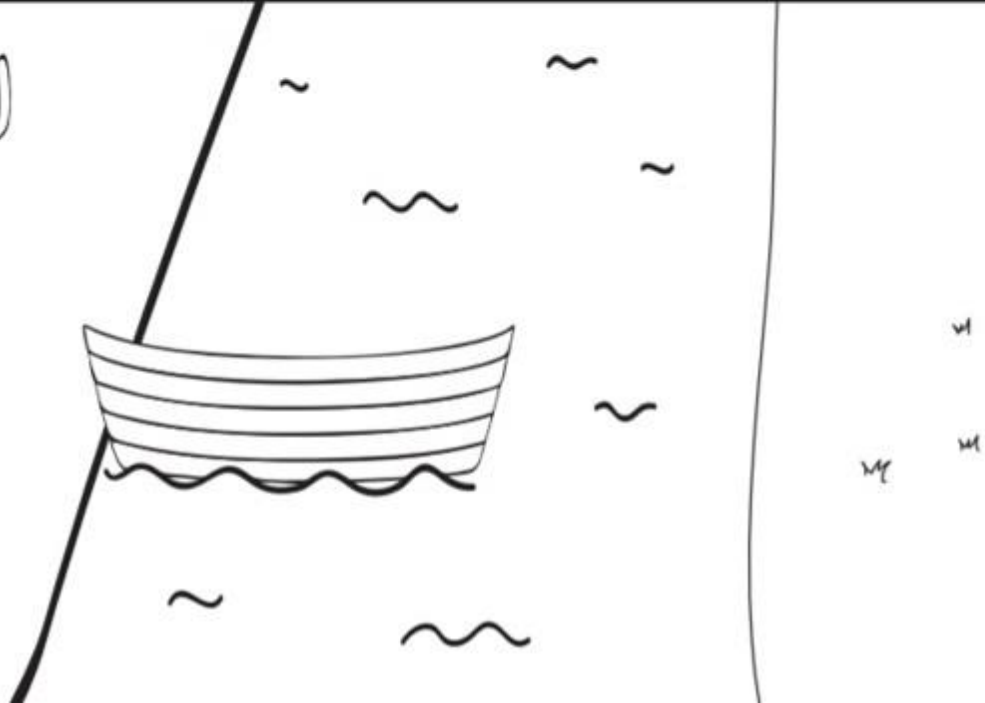
3)

4)

5)

6)

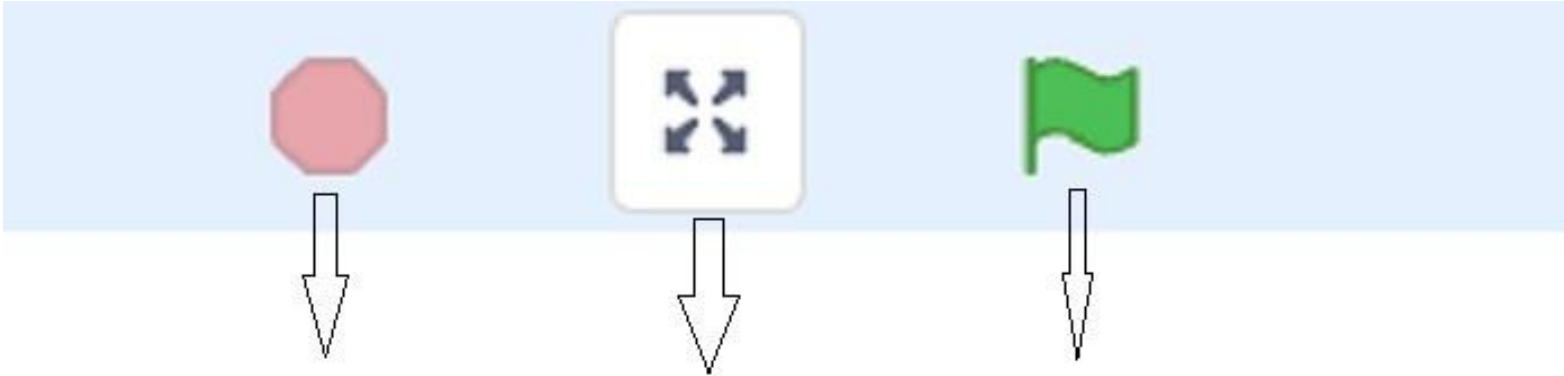
7)



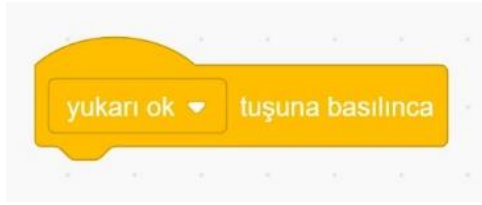
Aşağıda yer alan 2 döngü bloğu arasındaki farkı yazınız.



Aşağıda yer alan butonların görevlerini altlarına yazınız.



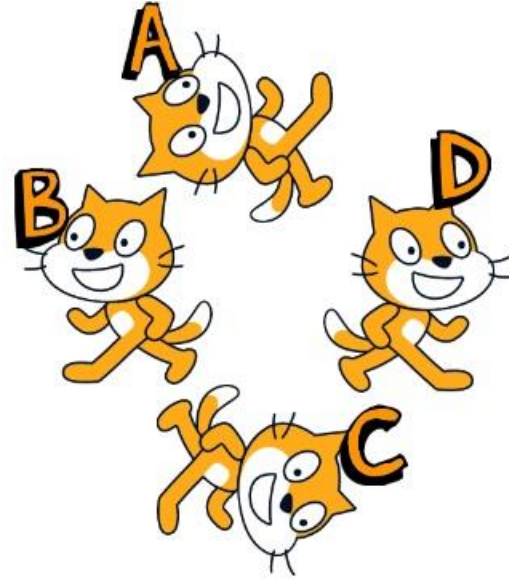
Aşağıda yer alan kod bloklarının görevlerini yanlarına yazınız.



Kedi karakterimizi koşuyormuş gibi göstermek için aşağıda yer alan kod bloklarında soru işaretli olan bölgeye hangi blok gelmelidir?



Aşağıda yer alan karakterlerin sahnede hangi yöne baktıklarını göstermek için kullandığımız açılar yazınız.



x konumunu 10 deęiřtir

y konumunu 10 deęiřtir

y konumunu -10 deęiřtir

x konumunu -10 deęiřtir

Yukarıda yer alan blokları kullanarak, klavyede bulunan yön tuřları ile karakterimizi hareket ettirmek için gereken blokları uygun yerlerine yerleřtiriniz.

yukarı ok ▼ tuřuna basılınca

sol ok ▼ tuřuna basılınca

saę ok ▼ tuřuna basılınca

ařaęı ok ▼ tuřuna basılınca

Aşağıda yer alan bloklar ayrı ayrı çalıştırıldığında karakterimiz hangi yöne hareket eder.

x konumunu 10 değiştir

y konumunu -10 değiştir

y konumunu 10 değiştir

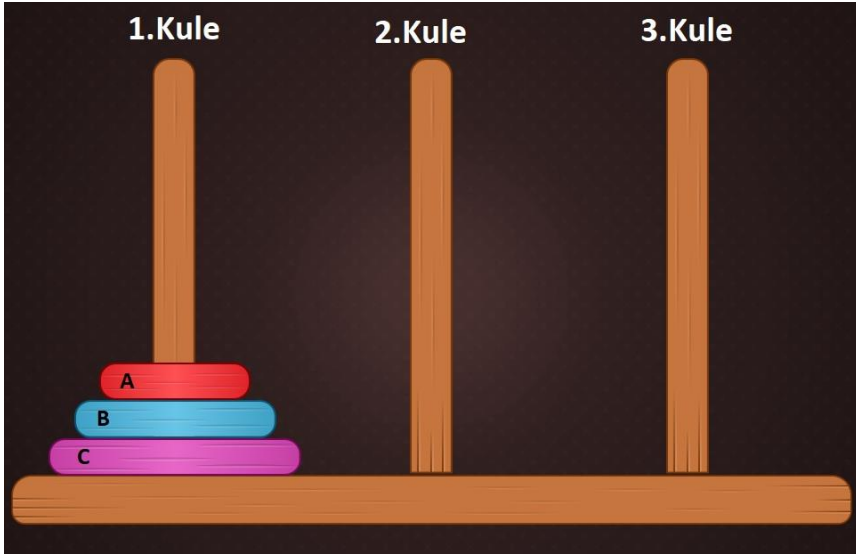
x konumunu -10 değiştir

Aşağıda yer alan döngü bloklarının görevlerinin yazınız.





Aşağıda yer alan hanoi kuleleri probleminin çözüm algoritmasını liste halinde yazınız.



- 1)
- 2)
- 3)
- 4)
- 5)
- 6)
- 7)