Bölüm 4.2: Akış Diyagramları

Bilgisayar ve Programlamaya Giriş

Simgeler | Sıralı akış | Karar | Döngü | Doğrulama

Sercan KÜLCÜ · Giresun Üniversitesi

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü: https://sercankulcu.github.io/files/computer/slides/Bolum_04_02_Akis_Diyagramlari.html

GİRİŞ

Öğrenme Çıktıları

Simgeler

Temel akış diyagramı simgelerini tanımak.

Akış

Oklarla işlem sırasını kurmak.

Kontrol

Karar ve döngü yapılarını göstermek.

Dönüşüm

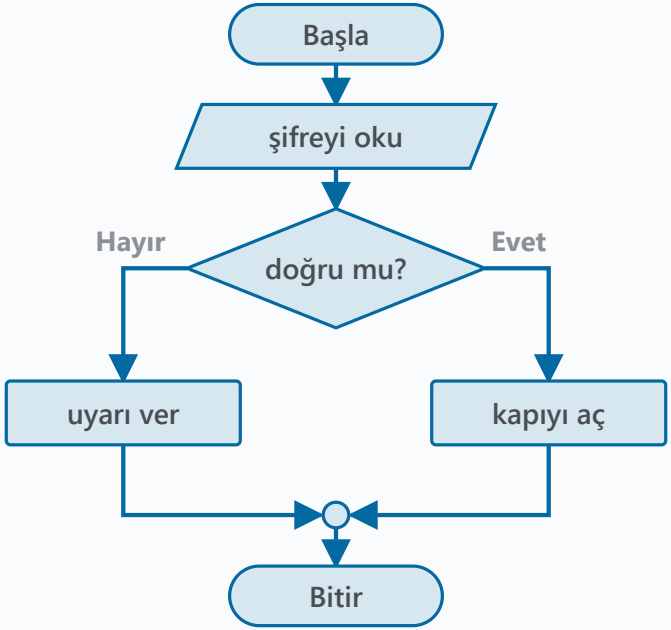
Diyagramı sözde kod ve programa çevirmek.

TEMELLER

Akış Diyagramı Nedir?

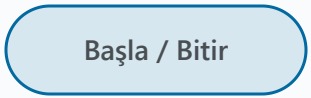
- Bir algoritmanın işlem sırasını grafiksel olarak gösterir.
- Karmaşık mantığı koddan bağımsız biçimde tartışmayı sağlar.
- Simgeler adımın **türünü**; oklar yürütme **yönünü** gösterir.

Yandaki örnek: günlük bir işin (kapı şifresi) algoritması. Karar kutusundan iki yol çıkar ve yollar yeniden birleşir.



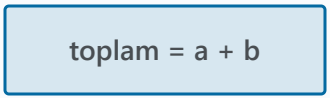
SİMGELER

Temel Simgeler



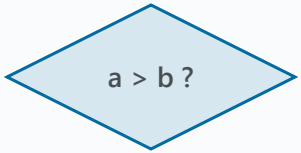
Başlangıç / bitiş

Her diyagramda bir başla, en az bir bitir



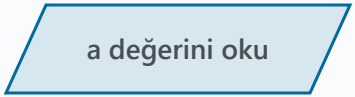
İşlem

Hesaplama, atama



Karar

Evet / Hayır iki çıkış



Giriş / çıkış

Veri alma (oku), gösterme (yaz)



Akış çizgisi

Yürütme yönü; uçtaki ok önemlidir



Bağlayıcı

Yolların birleşmesi / sayfa geçişi

Simgenin **biçimi** adımın türünü söyler; içine yalnızca o adımın kısa ifadesi yazılır.

SİMGELER

Simge ve Görev

Simge	Görev	Örnek
 Oval	Başlangıç veya bitiş	Başla
 Dikdörtgen	İşlem (hesaplama, atama)	toplam = a + b
 Eşkenar dörtgen	Karar (koşul)	a > b ?
 Paralelkenar	Giriş / çıkış	a değerini oku
 Ok	Yürütme yönü	sonraki adıma geç

Karar simgesinin içine **soru** (a > b ?), işlem simgesine **emir** (toplam = a + b) yazılır.

SİMGELER

Ek Simgeler

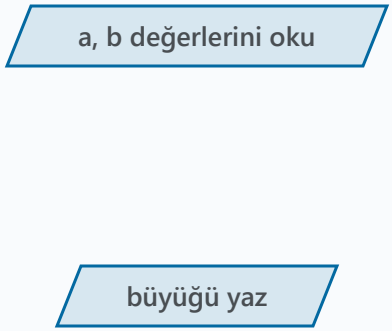
Simge	Görev	Örnek
 Sayfa dışı bağlayıcı	Diyagram başka sayfada sürer; içine sayfa ve nokta yazılır.	2-A
 Önceden tanımlı işlem	Ayrıca çizilmiş bir alt süreci çağırır.	Ortalama hesapla
 Hazırlık	Döngü sayacının başlangıcını ve sınırını tek simgede verir.	i = 1 → n
 Belge	Yazdırılan çıktı ya da rapor.	Not çizelgesi
 Veri deposu	Diskte ya da veri tabanında saklanan veri.	Öğrenci kayıtları
 Açıklama	Kesikli çizgiyle bir simgeye not ekler; akışa katılmaz.	not 0–100 arası

Bu simgeler ISO 5807 ve yaygın çizim araçlarında bulunur. Küçük bir algoritma için beş temel simge yeter; ek simgeler diyagram büyüyüp sayfalara bölündüğünde ya da dosya ve rapor gibi dış ortamlar gösterilmek istendiğinde kullanılır.

TASARIM · ADIM ADIM

Diyagram Oluşturma: İki Sayının Büyüğü

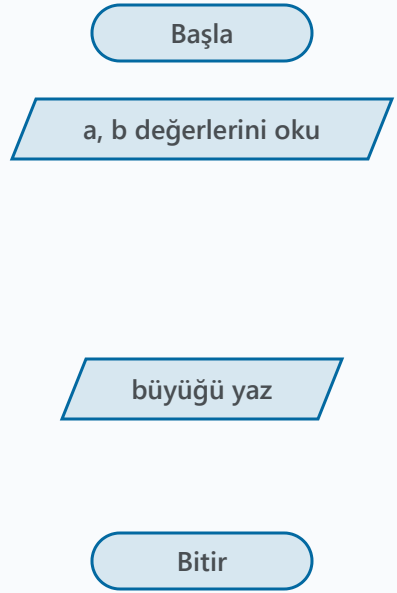
1. Girdi ve çıktıyı belirle
2. Adımları sırala
3. Kararları ayır
4. Okları bağla
5. Örnekle izle



Problem: iki sayının büyüğünü yazmak. **Girdi** a ve b (paralelkenar: oku), **çıktı** büyük olan sayı (paralelkenar: yaz).

Diyagram Oluşturma: İki Sayının Büyüğü

1. Girdi ve çıktıyı belirle
2. Adımları sırala
3. Kararları ayır
4. Okları bağla
5. Örnekle izle

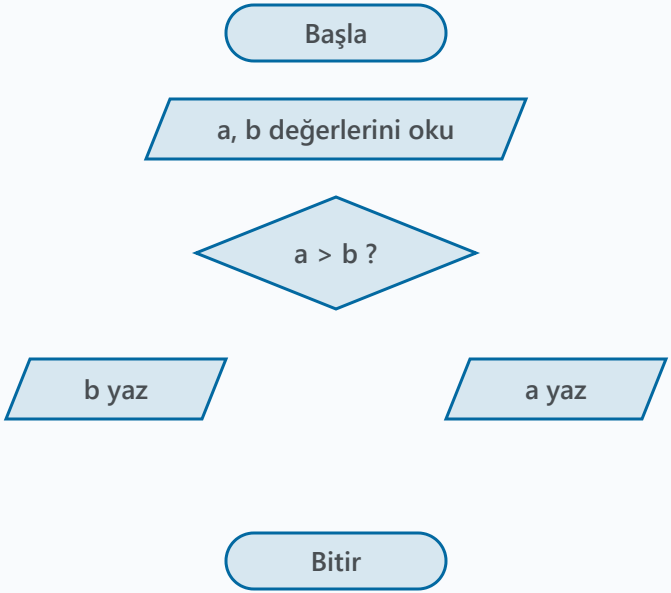


Başla ve Bitir eklenir; adımlar yukarıdan aşağıya sıralanır: önce oku, en son yaz.

TASARIM · ADIM ADIM

Diyagram Oluşturma: İki Sayının Büyüğü

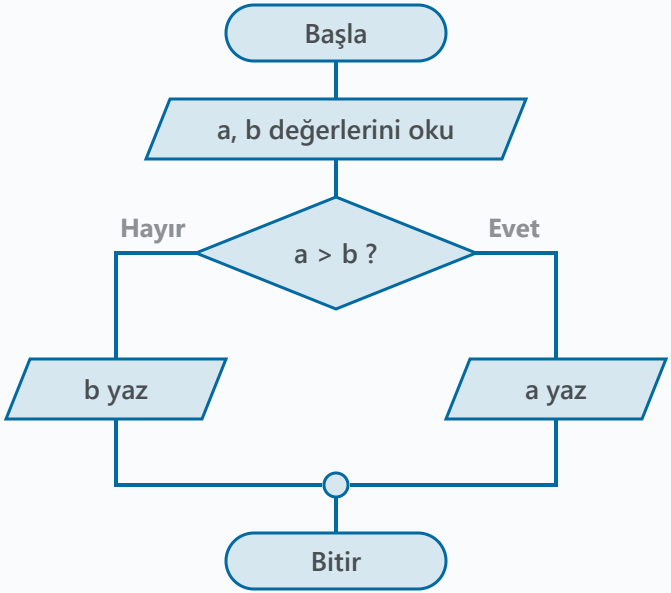
1. Girdi ve çıktıyı belirle
2. Adımları sırala
3. Kararları ayır
4. Okları bağla
5. Örnekle izle



Hangi sayının yazılacağı bir soruya bağlı: **a > b ?** karar kutusu eklenir; çıktı iki kutuya ayrılır.

Diyagram Oluşturma: İki Sayının Büyüğü

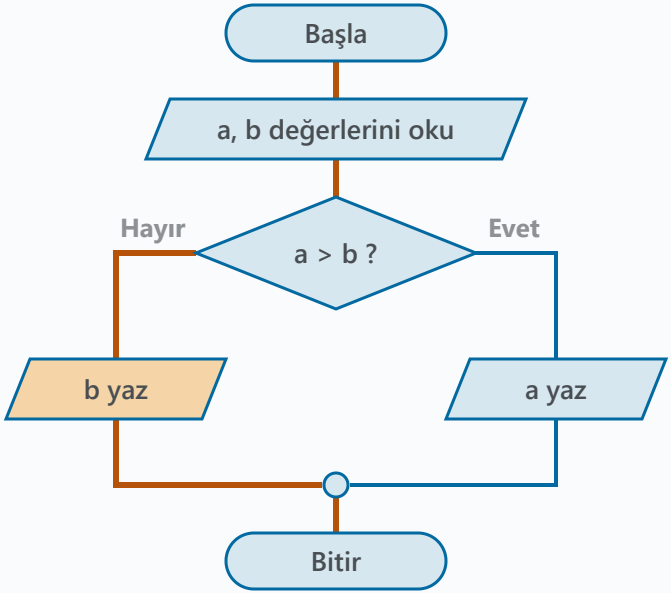
1. Girdi ve çıktıyı belirle
2. Adımları sırala
3. Kararları ayır
4. Okları bağla
5. Örnekle izle



Oklar yürütme yönünü verir. Karar çıkışları **Evet/Hayır** diye etiketlenir, yollar bir bağlayıcıda birleşir.

Diyagram Oluşturma: İki Sayının Büyüğü

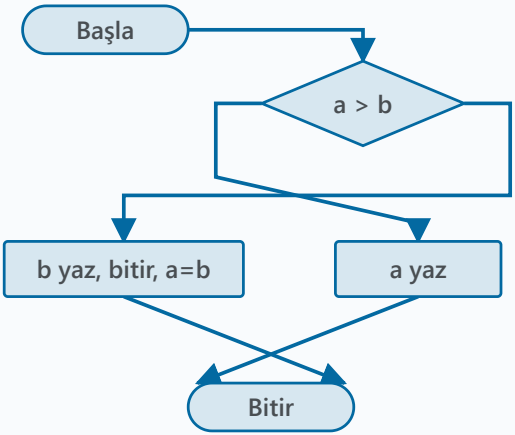
1. Girdi ve çıktıyı belirle
2. Adımları sırala
3. Kararları ayır
4. Okları bağla
5. Örnekle izle



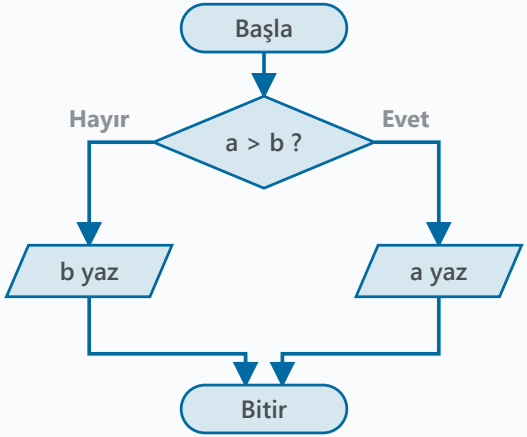
Örnekle izle: **a = 4, b = 9** → 4 > 9 yanlış → Hayır yolu → "b yaz" → çıktı **9**. Turuncu yol izlenen yoldur.

TASARIM

Okunabilirlik Kuralları



✗ Kesişen oklar, etiketsiz karar, bir kutuda üç işlem



✓ Yukarıdan aşağı, Evet/Hayır etiketli, tek işlem

- Akış mümkün olduğunca yukarıdan aşağıya ve soldan sağa ilerlemelidir.
- Karar çıkışları Evet/Hayır gibi anlamlı etiketler taşımalıdır; kesişen oklar azaltılmalıdır.
- Bir simgeye bir temel işlem yazılmalı; büyük diyagramlar alt süreçlere bölünmelidir.

TASARIM

Akış Diyagramı mı, Sözde Kod mu?

	Akış diyagramı	Sözde kod
Biçim	Simgeler ve oklar	Girintili kısa cümleler
Güçlü yanı	Dallanma ve döngü bir bakışta görülür	Hızlı yazılır, koda çok yakındır
Zayıf yanı	Büyüyünce sayfaya sığmaz, değiştirmek zahmetli	Akışın yapısı girintiye bağlıdır
Uygun olduğu yer	Öğrenme, sunum, iş süreçleri	Uzun algoritmalar, kodlamadan hemen önce

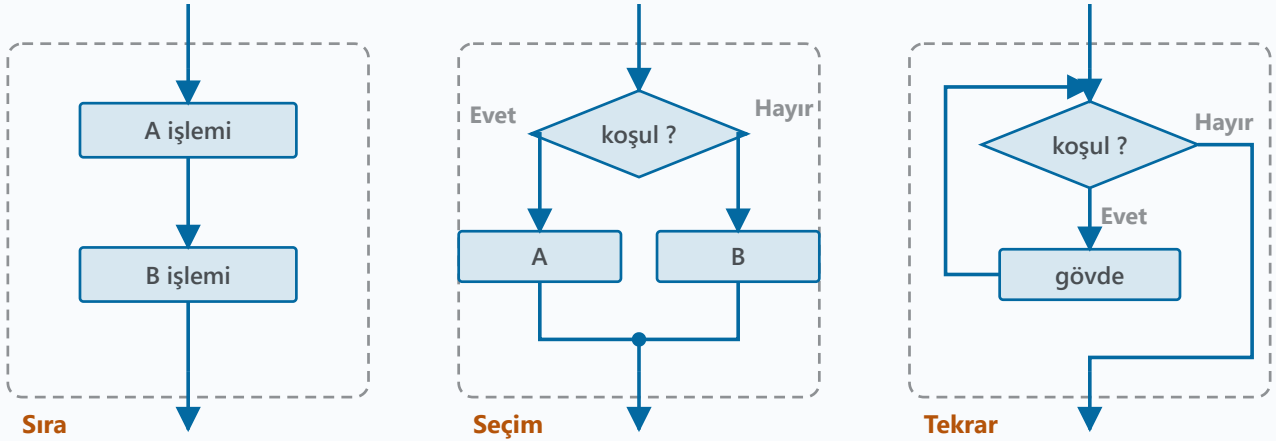
İkisi aynı algoritmayı anlatır; biri diğerine satır satır çevrilebilir. Sözde kodun bir standardı yoktur, yeter ki her adım açık olsun.




```
oku x
eğer x < 0 ise
    x = −x
yaz x
```

TASARIM

Yapısal Programlamanın Üç Yapısı



- Böhm ve Jacopini (1966): her algoritma yalnızca **sıra**, **seçim** ve **tekrar** yapılarıyla yazılabilir.
- Her yapının **tek girişi ve tek çıkışı** vardır; bu yüzden bir yapı, başka bir yapının içindeki kutunun yerine konabilir (iç içe yapılar).
- Okları diyagramın rastgele bir noktasına atlatan çizimler (**spagetti** akış) okunması ve test edilmesi zor diyagramlar üretir. Java'da `goto` ayrılmış bir sözcüktür ama kullanılamaz.

TASARIM

Akış Diyagramı Çizim Araçları

Kâğıt ve kalem

İlk taslak için en hızlısı; silip yeniden çizmek kolaydır. Sınavda da diyagram elle çizilir.

Word ve PowerPoint

Ekle → Şekiller menüsündeki **Akış Çizelgesi** grubu; bağlayıcı oklar kutu taşınınca birlikte kayar.

draw.io (diagrams.net)

Ücretsiz, tarayıcıda çalışır; hazır akış diyagramı şekilleri ve dışa aktarma (PNG, SVG, PDF) sunar.

Microsoft Visio

Kurumsal süreç ve iş akışı diyagramları için ticari araç.

Flowgorithm

Diyagramı **çalıştırır**, değişkenleri adım adım gösterir ve Java dahil birçok dile koda çevirir.

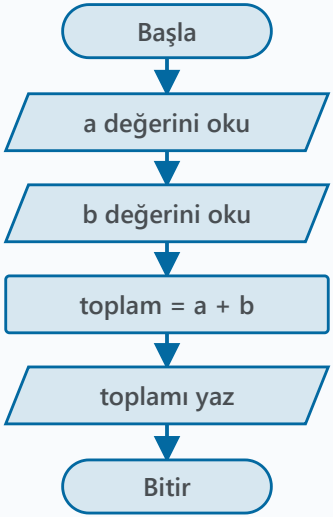
Mermaid

Diyagram metinle tanımlanır, araç çizer; belgeler ve kod depoları içinde sık kullanılır.

Araç ne olursa olsun simgelerin anlamı ve okunabilirlik kuralları aynıdır.

AKIŞ TÜRLERİ

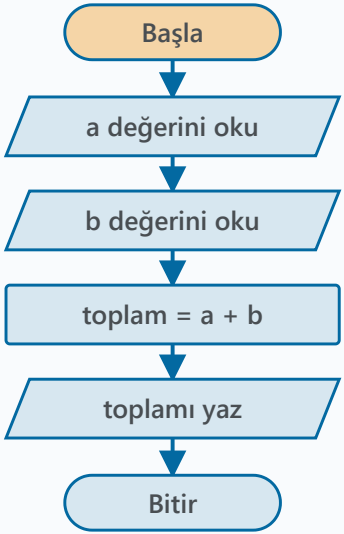
Sıralı Akış




```
Başla
  a değerini oku
  b değerini oku
  toplam = a + b
  toplamı yaz
Bitir
```

- Her adım bir önceki tamamlandıktan sonra çalışır.
- Dallanma veya tekrar bulunmaz: tek giriş, tek çıkış.
- Oklar hep aşağı doğrudur.
- Sıra önemlidir: `toplam`, a ve b okunmadan hesaplanamaz.

Sıralı Akış Adım Adım (a = 4, b = 7)



Yürütme **Başla** kutusundan başlar. Henüz hiçbir değişken yok.

Adım 1 / 6

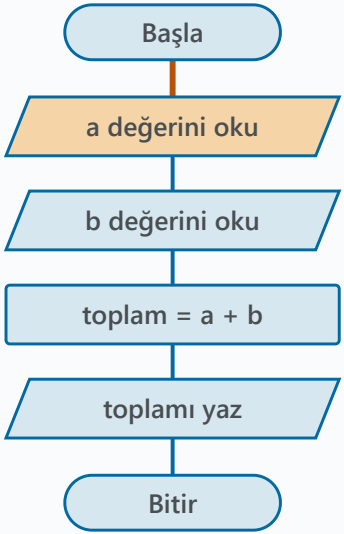
```
▶ Başla
2   a değerini oku
3   b değerini oku
4   toplam = a + b
5   toplamı yaz
6 Bitir
```

DEĞİŞKENLER

—

ÇIKTI

Sıralı Akış Adım Adım (a = 4, b = 7)



Giriş kutusu: kullanıcı **4** girer → **a = 4**.

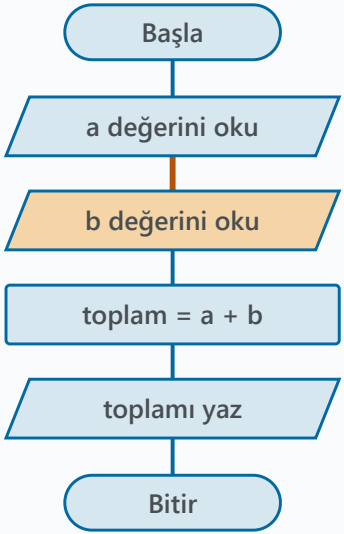
```
1 Başla
▶   a değerini oku
3   b değerini oku
4   toplam = a + b
5   toplamı yaz
6 Bitir
```

DEĞİŞKENLER

`a = 4`

ÇIKTI

Sıralı Akış Adım Adım (a = 4, b = 7)



Okun gösterdiği sonraki kutu: **b = 7** okunur.

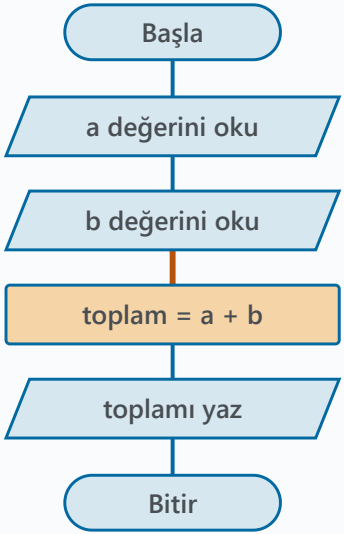
```
1 Başla
2   a değerini oku
▶   b değerini oku
4   toplam = a + b
5   toplamı yaz
6 Bitir
```

DEĞİŞKENLER

a = 4 b = 7

ÇIKTI

Sıralı Akış Adım Adım (a = 4, b = 7)



İşlem kutusu: **toplam = 4 + 7 = 11**.

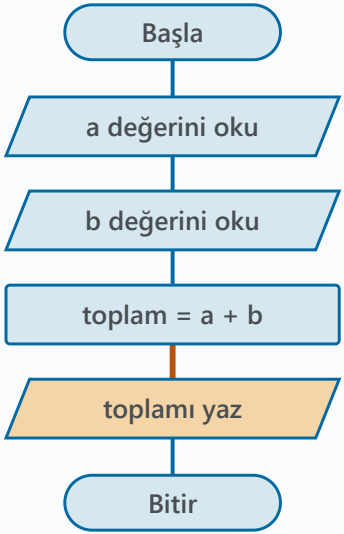
```
1 Başla
2   a değerini oku
3   b değerini oku
▶   toplam = a + b
5   toplamı yaz
6 Bitir
```

DEĞİŞKENLER

`a = 4` `b = 7` `toplam = 11`

ÇIKTI

Sıralı Akış Adım Adım (a = 4, b = 7)



Çıkış kutusu: ekrana **11** yazılır.

```
1 Başla
2   a değerini oku
3   b değerini oku
4   toplam = a + b
▶   toplamı yaz
6 Bitir
```

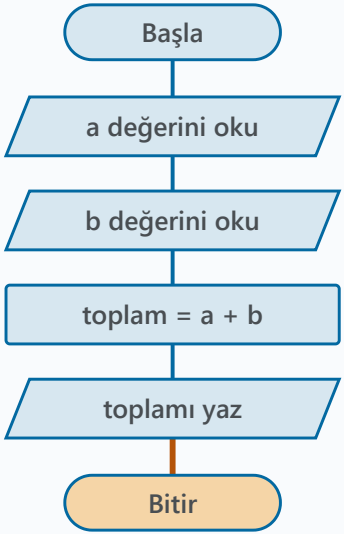
DEĞİŞKENLER

`a = 4` `b = 7` `toplam = 11`

ÇIKTI

```
11
```

Sıralı Akış Adım Adım (a = 4, b = 7)




```
1 Başla
2   a değerini oku
3   b değerini oku
4   toplam = a + b
5   toplamı yaz
▶ Bitir
```

DEĞİŞKENLER

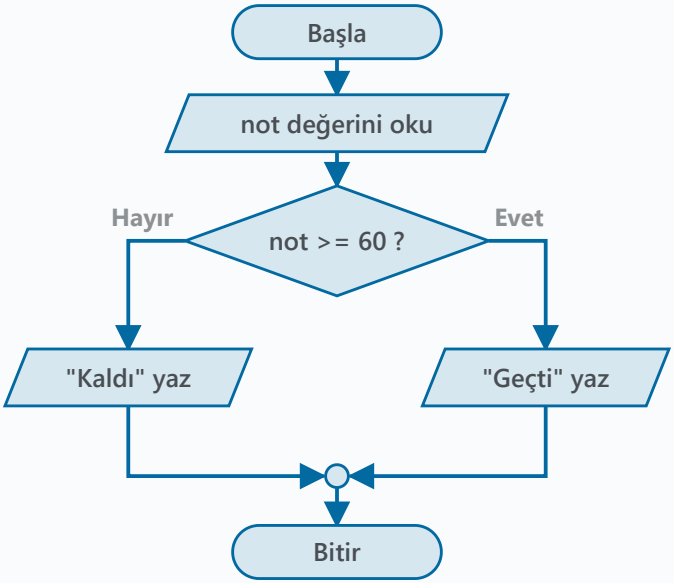
`a = 4` `b = 7` `toplam = 11`

ÇIKTI

```
11
```

Bitir kutusuna ulaşıldı. Her kutu **tam bir kez** çalıştı: sıralı akışın özelliği.

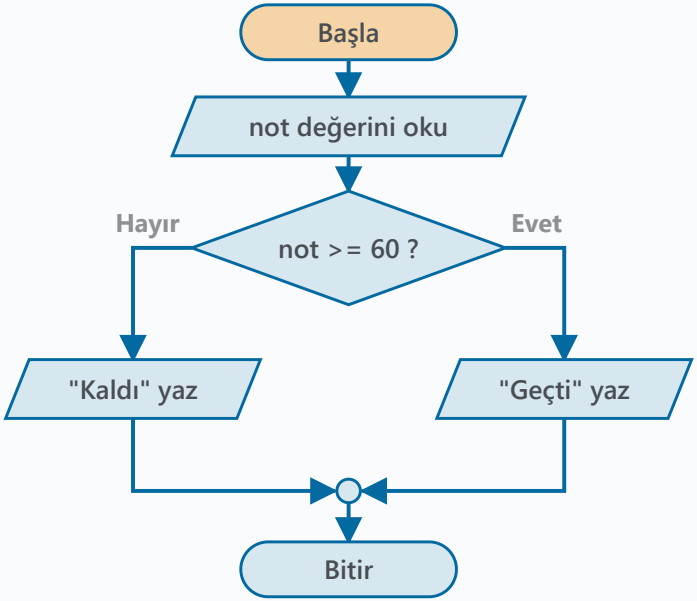
Koşullu Akış




```
Başla
  not değerini oku
  not >= 60 ?
    Evet → "Geçti" yaz
    Hayır → "Kaldı" yaz
Bitir
```

- Karar simgesinden iki yol çıkar; her çalıştırmada **yalnızca biri** izlenir.
- Yollar alttaki bağlayıcıda (●) yeniden birleşir.
- Sınır değeri ayrıca denenir: not = 60 iken `>=` "Geçti", `>` ise "Kaldı" verir.

Koşullu Akış Adım Adım (not = 72, sonra 45)



1. çalıştırma (not = 72): yürütme Başla kutusundan başlar.

Adım 1 / 10

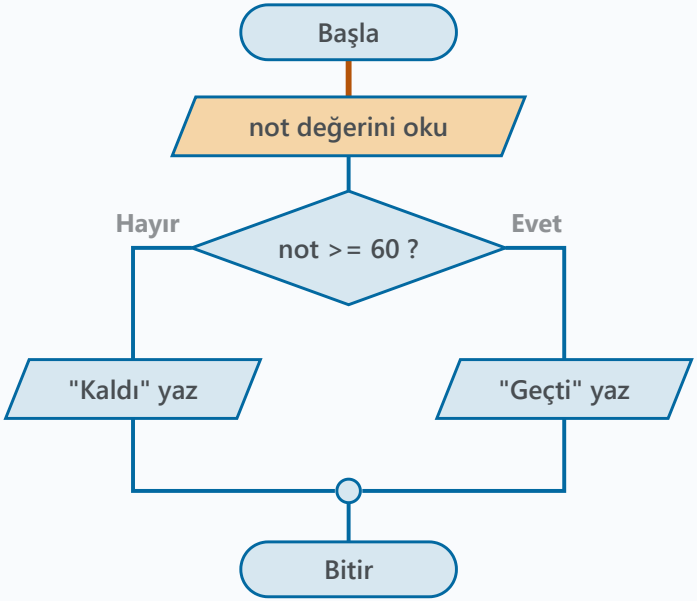
```
▶ Başla
2   not değerini oku
3   not >= 60 ?
4     Evet → "Geçti" yaz
5     Hayır → "Kaldı" yaz
6 Bitir
```

DEĞİŞKENLER

—

ÇIKTI

Koşullu Akış Adım Adım (not = 72, sonra 45)




```
1 Başla
▶   not değerini oku
3   not >= 60 ?
4     Evet → "Geçti" yaz
5     Hayır → "Kaldı" yaz
6 Bitir
```

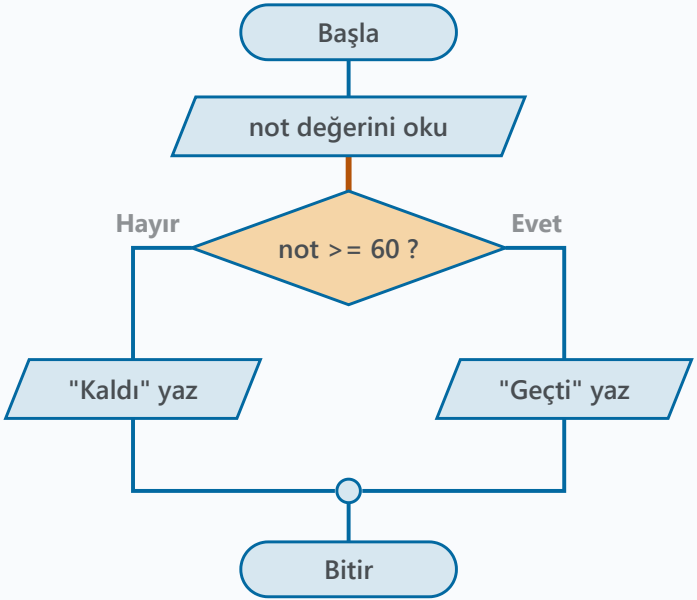
DEĞİŞKENLER

`not = 72`

ÇIKTI

Kullanıcı **72** girer → **not = 72**.

Koşullu Akış Adım Adım (not = 72, sonra 45)




```
1 Başla
2   not değerini oku
▶   not >= 60 ?
4     Evet → "Geçti" yaz
5     Hayır → "Kaldı" yaz
6 Bitir
```

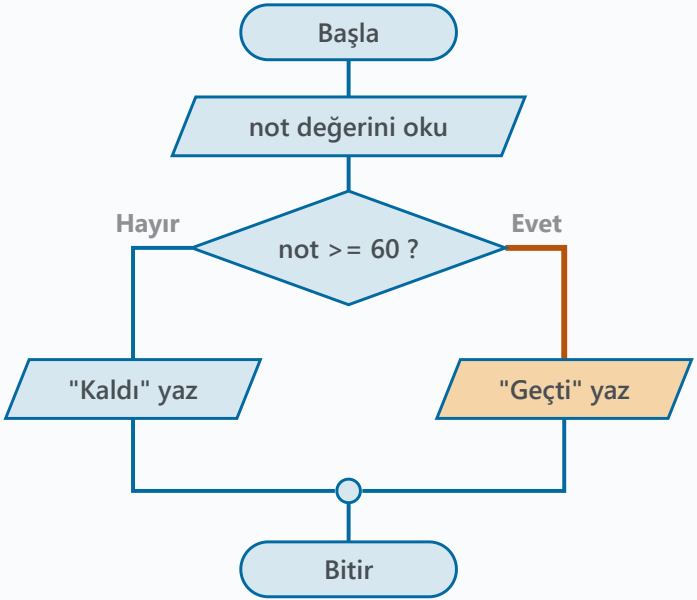
DEĞİŞKENLER

`not = 72` `kosul = doğru`

ÇIKTI

Karar: **72 >= 60** → **doğru**. Akış **Evet** okundan devam eder.

Koşullu Akış Adım Adım (not = 72, sonra 45)




```
1 Başla
2   not değerini oku
3   not >= 60 ?
▶     Evet → "Geçti" yaz
5     Hayır → "Kaldı" yaz
6 Bitir
```

DEĞİŞKENLER

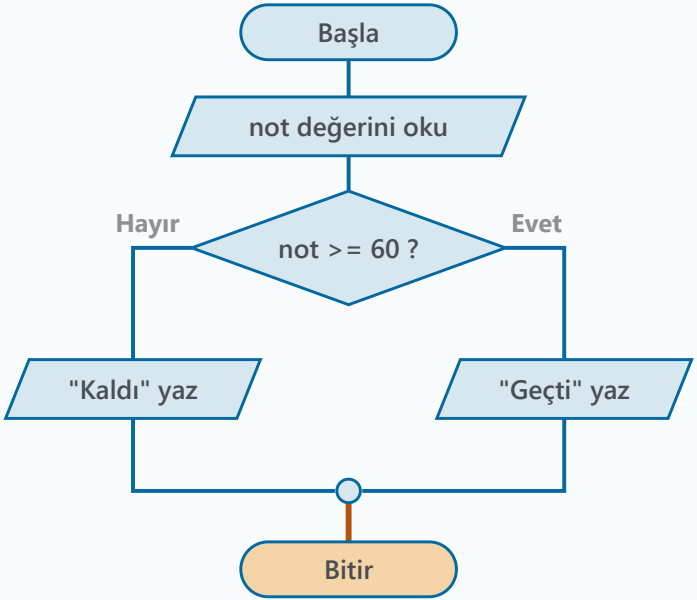
`not = 72` `kosul = doğru`

ÇIKTI

```
Geçti
```

"Geçti" yazılır. Hayır dalındaki kutu bu çalıştırmada **hiç çalışmaz**.

Koşullu Akış Adım Adım (not = 72, sonra 45)



İki yol bağlayıcıda birleşir ve Bitir'e ulaşılır. Şimdi diğer dalı görmek için not = 45 ile yeniden çalıştıralım.

```
1 Başla
2   not değerini oku
3   not >= 60 ?
4     Evet → "Geçti" yaz
5     Hayır → "Kaldı" yaz
▶ Bitir
```

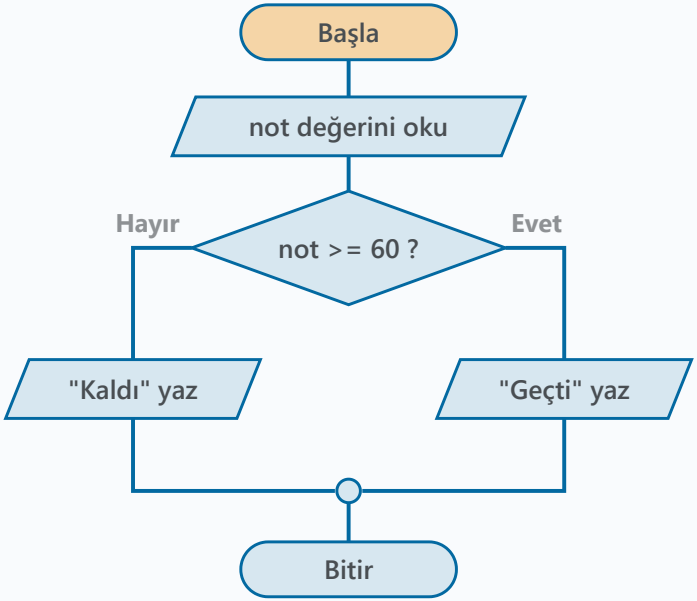
DEĞİŞKENLER

`not = 72` `kosul = doğru`

ÇIKTI

```
Geçti
```

Koşullu Akış Adım Adım (not = 72, sonra 45)




```
▶ Başla
2   not değerini oku
3   not >= 60 ?
4     Evet → "Geçti" yaz
5     Hayır → "Kaldı" yaz
6 Bitir
```

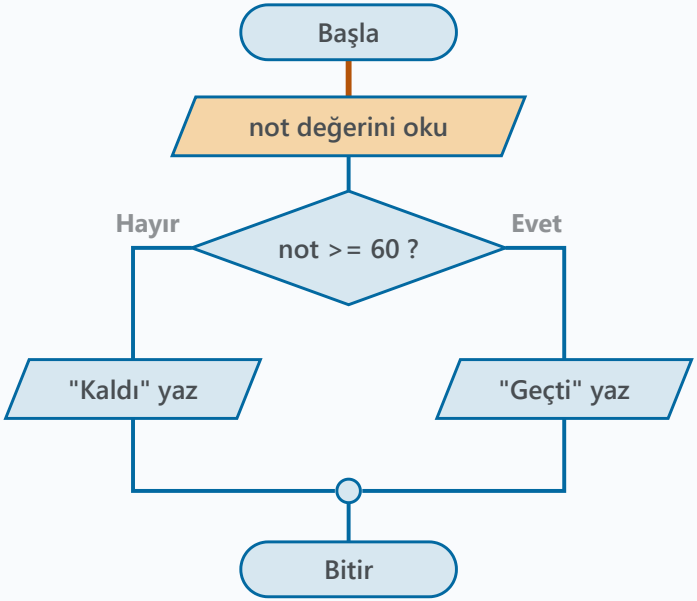
DEĞİŞKENLER

—

ÇIKTI

2. çalıştırma (not = 45): yürütme Başla kutusundan başlar.

Koşullu Akış Adım Adım (not = 72, sonra 45)




```
1 Başla
▶   not değerini oku
3   not >= 60 ?
4     Evet → "Geçti" yaz
5     Hayır → "Kaldı" yaz
6 Bitir
```

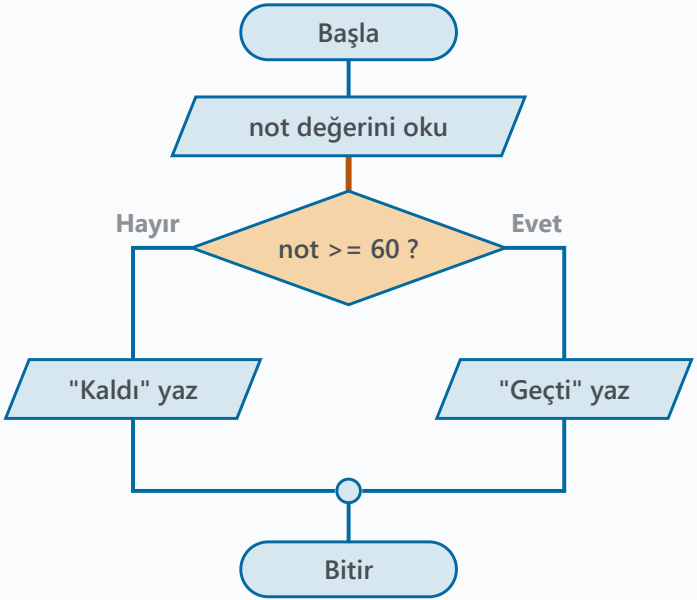
DEĞİŞKENLER

`not = 45`

ÇIKTI

Kullanıcı **45** girer → **not = 45**.

Koşullu Akış Adım Adım (not = 72, sonra 45)




```
1 Başla
2   not değerini oku
▶   not >= 60 ?
4     Evet → "Geçti" yaz
5     Hayır → "Kaldı" yaz
6 Bitir
```

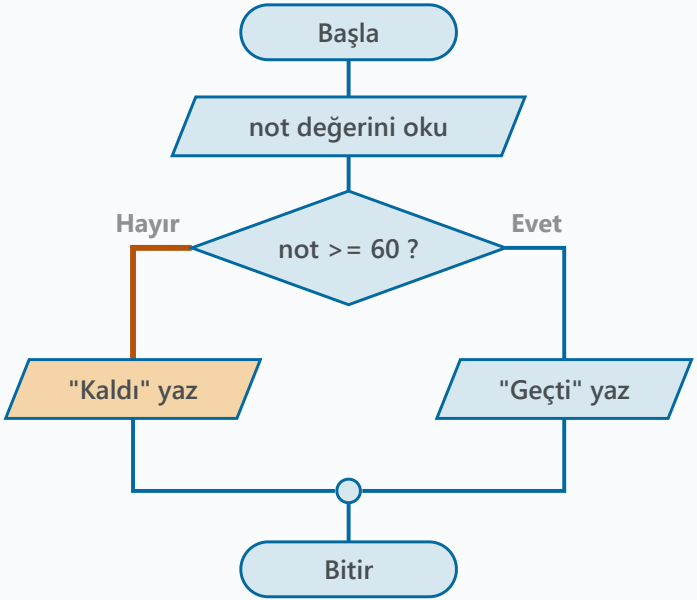
DEĞİŞKENLER

`not = 45` `kosul = yanlış`

ÇIKTI

Karar: **45 >= 60** → **yanlış**. Akış **Hayır** okundan devam eder.

Koşullu Akış Adım Adım (not = 72, sonra 45)




```
1 Başla
2   not değerini oku
3   not >= 60 ?
4     Evet → "Geçti" yaz
▶     Hayır → "Kaldı" yaz
6 Bitir
```

DEĞİŞKENLER

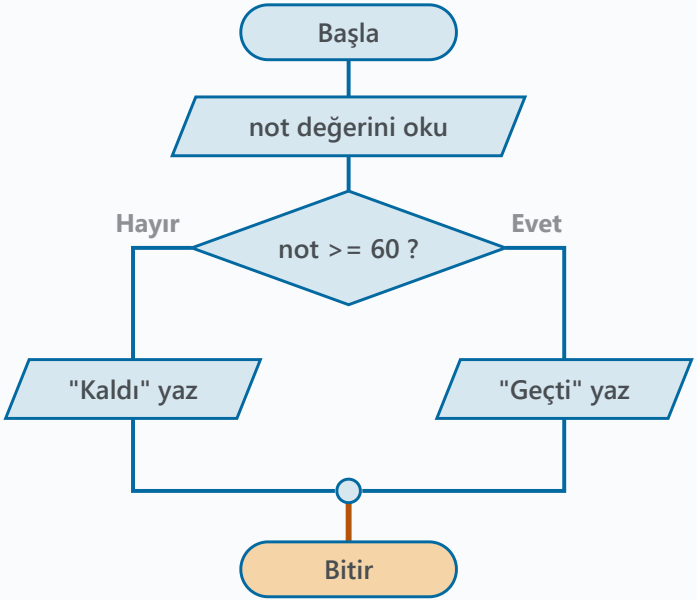
`not = 45` `kosul = yanlış`

ÇIKTI

```
Kaldı
```

"Kaldı" yazılır. Evet dalındaki kutu bu çalıştırmada **hiç çalışmaz**.

Koşullu Akış Adım Adım (not = 72, sonra 45)



İki yol bağlayıcıda birleşir ve Bitir'e ulaşılır. İki farklı girdiyle iki dalın da çalıştığını gördük.


```
1 Başla
2   not değerini oku
3   not >= 60 ?
4     Evet → "Geçti" yaz
5     Hayır → "Kaldı" yaz
▶ Bitir
```

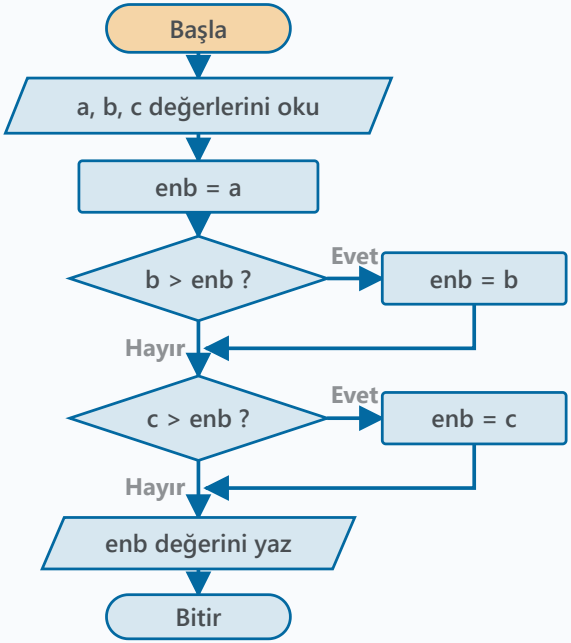
DEĞİŞKENLER

`not = 45` `kosul = yanlış`

ÇIKTI

```
Kaldı
```

Üç Sayının En Büyüğü Adım Adım



1. çalıştırma (4, 9, 7): Başla.

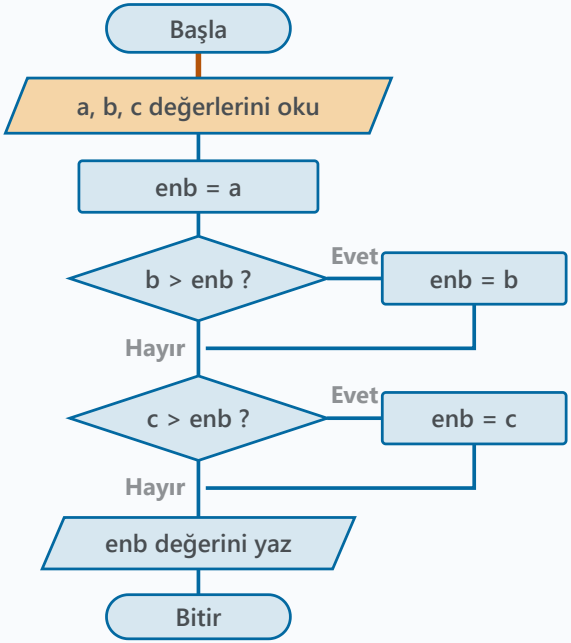
```
▶ Başla
2   a, b, c değerlerini oku
3   enb = a
4   b > enb ?  Evet → enb = b
5   c > enb ?  Evet → enb = c
6   enb değerini yaz
7 Bitir
```

DEĞİŞKENLER

—

ÇIKTI

Üç Sayının En Büyüğü Adım Adım




```
1 Başla
▶   a, b, c değerlerini oku
3   enb = a
4   b > enb ?  Evet → enb = b
5   c > enb ?  Evet → enb = c
6   enb değerini yaz
7 Bitir
```

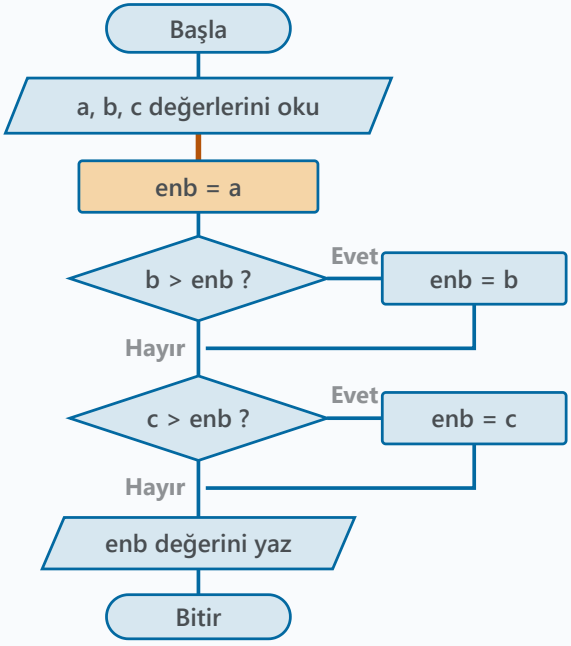
DEĞİŞKENLER

a = 4 b = 9 c = 7

ÇIKTI

Üç sayı okunur: a = 4, b = 9, c = 7.

Üç Sayının En Büyüğü Adım Adım



İlk aday en büyük: **enb = a = 4**.

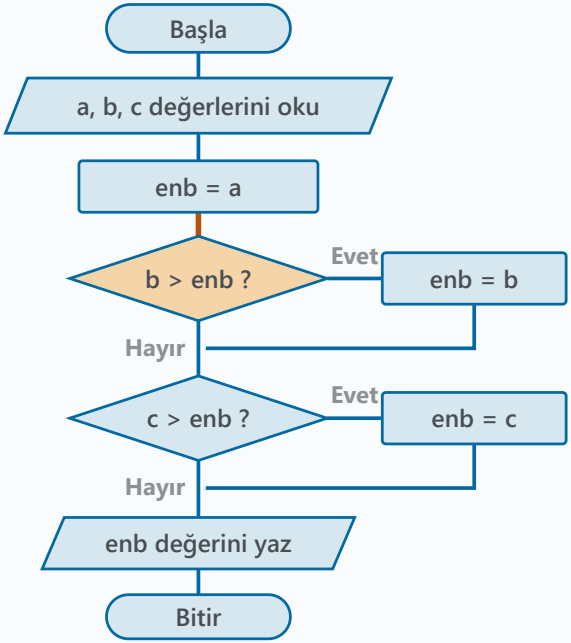
```
1  Başla
2    a, b, c değerlerini oku
▶    enb = a
4    b > enb ?  Evet → enb = b
5    c > enb ?  Evet → enb = c
6    enb değerini yaz
7  Bitir
```

DEĞİŞKENLER

`a = 4` `b = 9` `c = 7` `enb = 4`

ÇIKTI

Üç Sayının En Büyüğü Adım Adım



Karar: **9 > 4** → **doğru** → Evet.

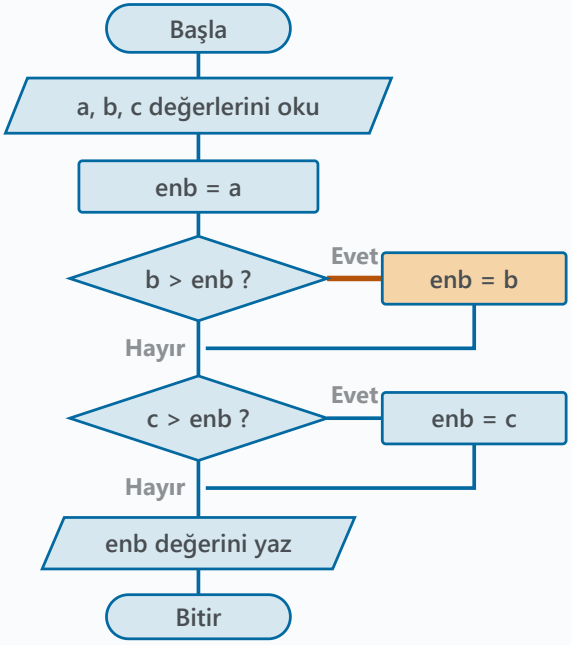
```
1 Başla
2   a, b, c değerlerini oku
3   enb = a
▶   b > enb ?  Evet → enb = b
5   c > enb ?  Evet → enb = c
6   enb değerini yaz
7 Bitir
```

DEĞİŞKENLER

`a = 4` `b = 9` `c = 7` `enb = 4`

ÇIKTI

Üç Sayının En Büyüğü Adım Adım




```
1 Başla
2   a, b, c değerlerini oku
3   enb = a
▶   b > enb ?  Evet → enb = b
5   c > enb ?  Evet → enb = c
6   enb değerini yaz
7 Bitir
```

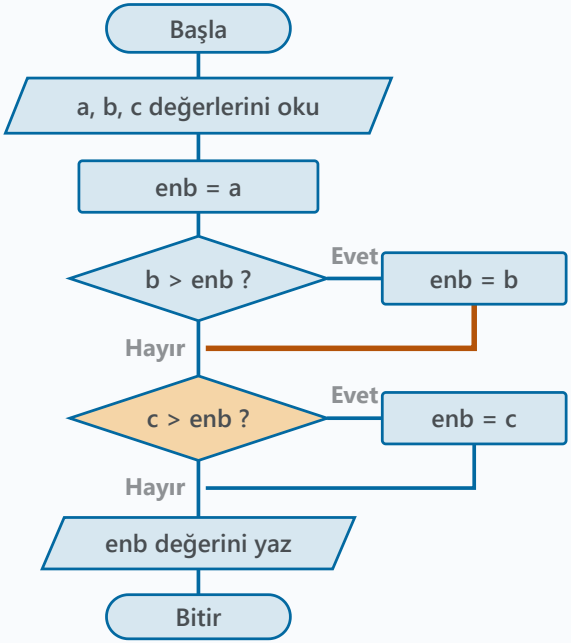
DEĞİŞKENLER

`a = 4` `b = 9` `c = 7` `enb = 9`

ÇIKTI

enb = b = 9. Yol aşağıda birleşir.

Üç Sayının En Büyüğü Adım Adım



Karar: **7 > 9** → **yanlış** → Hayır.

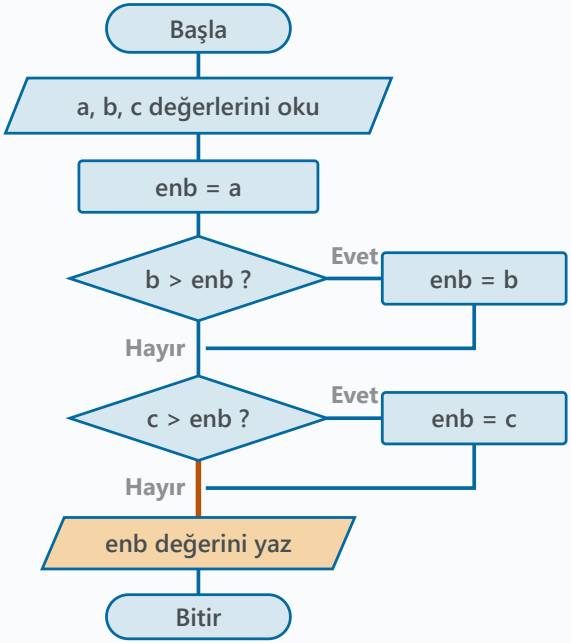
```
1 Başla
2   a, b, c değerlerini oku
3   enb = a
4   b > enb ?  Evet → enb = b
▶   c > enb ?  Evet → enb = c
6   enb değerini yaz
7 Bitir
```

DEĞİŞKENLER

`a = 4` `b = 9` `c = 7` `enb = 9`

ÇIKTI

Üç Sayının En Büyüğü Adım Adım




```
1 Başla
2   a, b, c değerlerini oku
3   enb = a
4   b > enb ?  Evet → enb = b
5   c > enb ?  Evet → enb = c
▶   enb değerini yaz
7 Bitir
```

DEĞİŞKENLER

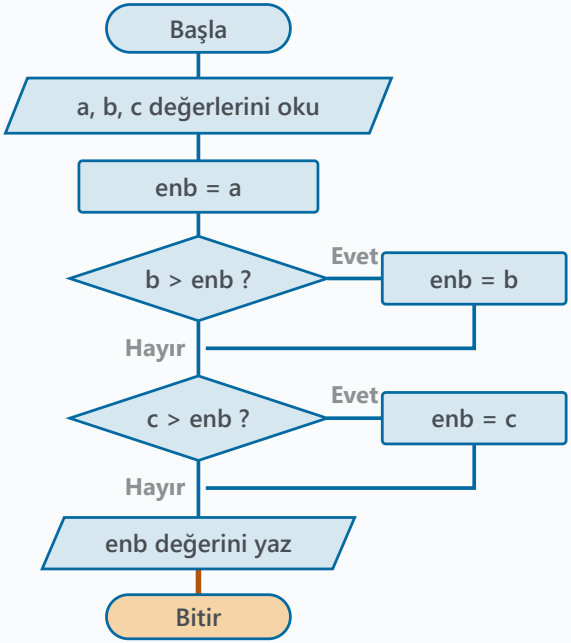
`a = 4` `b = 9` `c = 7` `enb = 9`

ÇIKTI

```
9
```

9 yazılır.

Üç Sayının En Büyüğü Adım Adım




```
1  Başla
2    a, b, c değerlerini oku
3    enb = a
4    b > enb ?  Evet → enb = b
5    c > enb ?  Evet → enb = c
6    enb değerini yaz
▶  Bitir
```

DEĞİŞKENLER

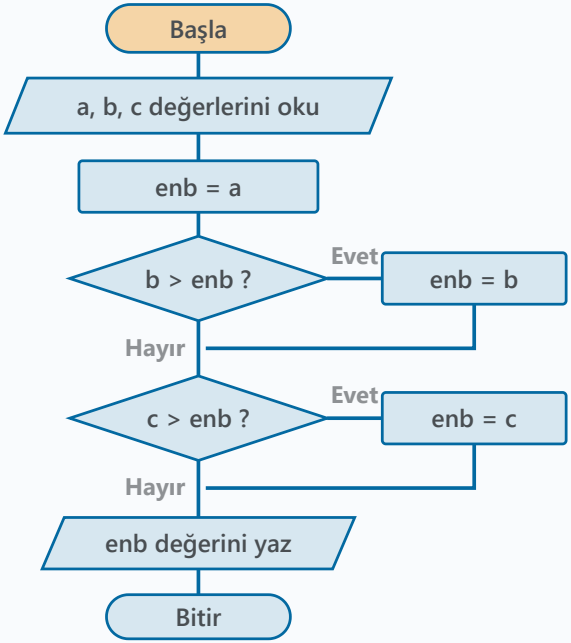
`a = 4` `b = 9` `c = 7` `enb = 9`

ÇIKTI

```
9
```

Bitir. Şimdi en büyüğün c olduğu bir girdiyle deneyelim.

Üç Sayının En Büyüğü Adım Adım




```
▶ Başla
2   a, b, c değerlerini oku
3   enb = a
4   b > enb ?  Evet → enb = b
5   c > enb ?  Evet → enb = c
6   enb değerini yaz
7 Bitir
```

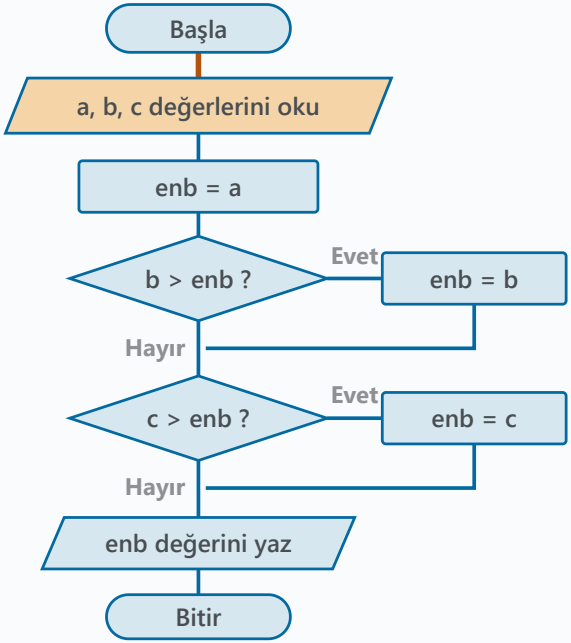
DEĞİŞKENLER

—

ÇIKTI

2. çalıştırma (3, 2, 8): Başla.

Üç Sayının En Büyüğü Adım Adım




```
1 Başla
▶   a, b, c değerlerini oku
3   enb = a
4   b > enb ?  Evet → enb = b
5   c > enb ?  Evet → enb = c
6   enb değerini yaz
7 Bitir
```

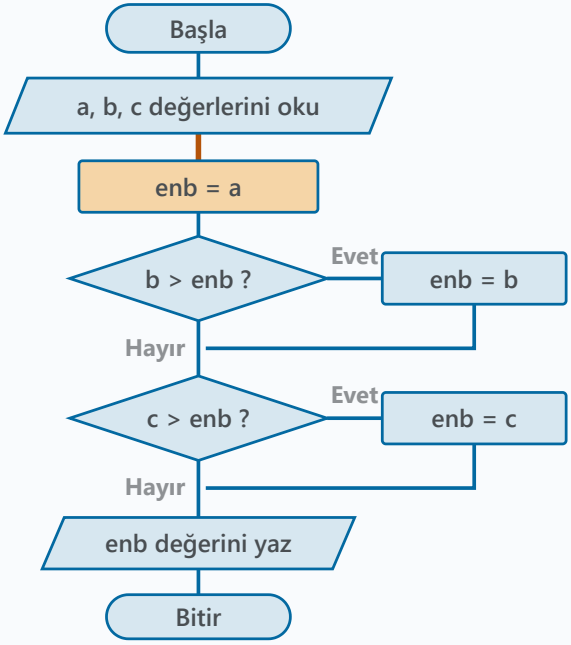
DEĞİŞKENLER

a = 3 b = 2 c = 8

ÇIKTI

Üç sayı okunur: a = 3, b = 2, c = 8.

Üç Sayının En Büyüğü Adım Adım



İlk aday en büyük: **enb = a = 3**.

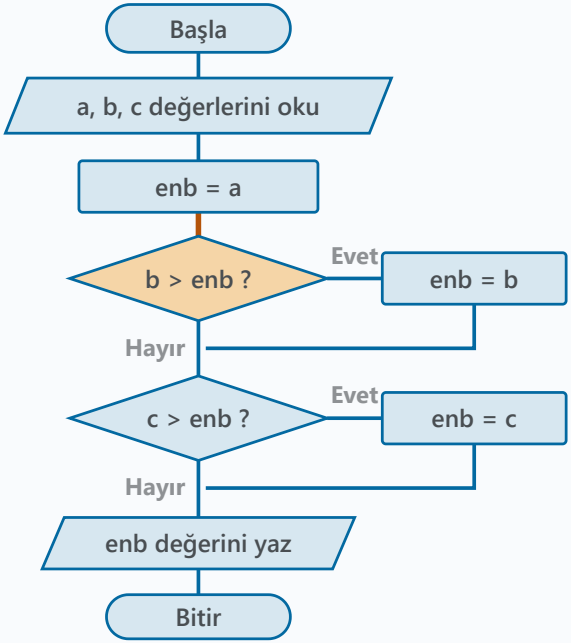
```
1 Başla
2   a, b, c değerlerini oku
    enb = a
4   b > enb ?  Evet → enb = b
5   c > enb ?  Evet → enb = c
6   enb değerini yaz
7 Bitir
```

DEĞİŞKENLER

a = 3 | b = 2 | c = 8 | enb = 3

ÇIKTI

Üç Sayının En Büyüğü Adım Adım




```
1 Başla
2   a, b, c değerlerini oku
3   enb = a
▶   b > enb ?  Evet → enb = b
5   c > enb ?  Evet → enb = c
6   enb değerini yaz
7 Bitir
```

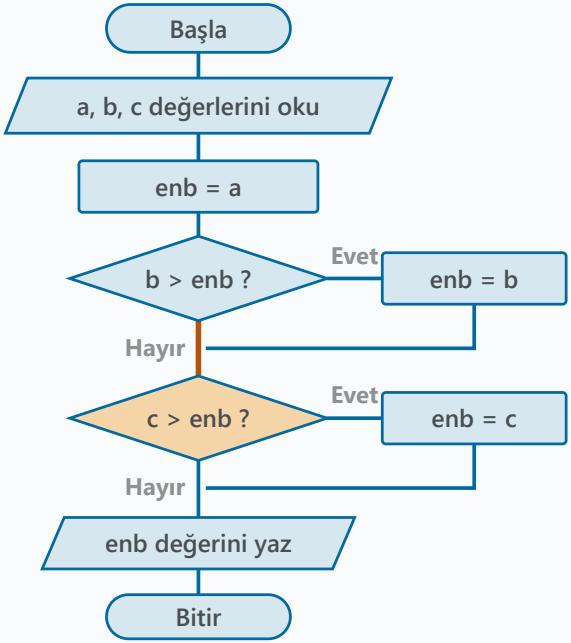
DEĞİŞKENLER

`a = 3` `b = 2` `c = 8` `enb = 3`

ÇIKTI

Karar: **2 > 3** → **yanlış** → Hayır, enb değişmez.

Üç Sayının En Büyüğü Adım Adım



Karar: **8 > 3** → **doğru** → Evet.

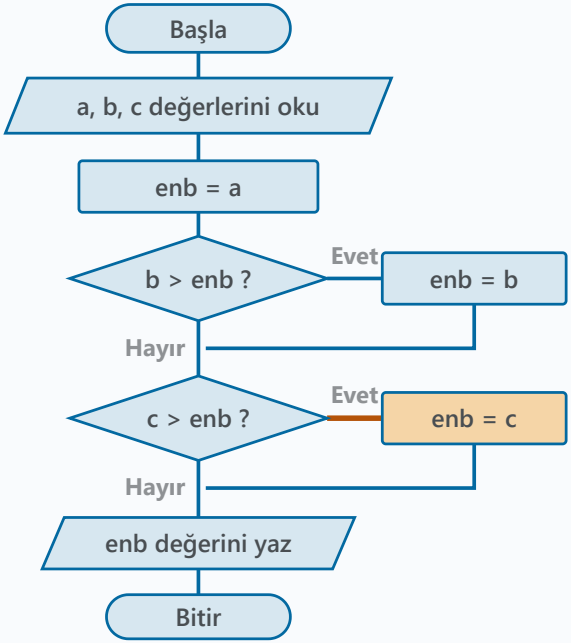
```
1 Başla
2   a, b, c değerlerini oku
3   enb = a
4   b > enb ?  Evet → enb = b
▶   c > enb ?  Evet → enb = c
6   enb değerini yaz
7 Bitir
```

DEĞİŞKENLER

`a = 3` `b = 2` `c = 8` `enb = 3`

ÇIKTI

Üç Sayının En Büyüğü Adım Adım




```
1 Başla
2   a, b, c değerlerini oku
3   enb = a
4   b > enb ?  Evet → enb = b
▶   c > enb ?  Evet → enb = c
6   enb değerini yaz
7 Bitir
```

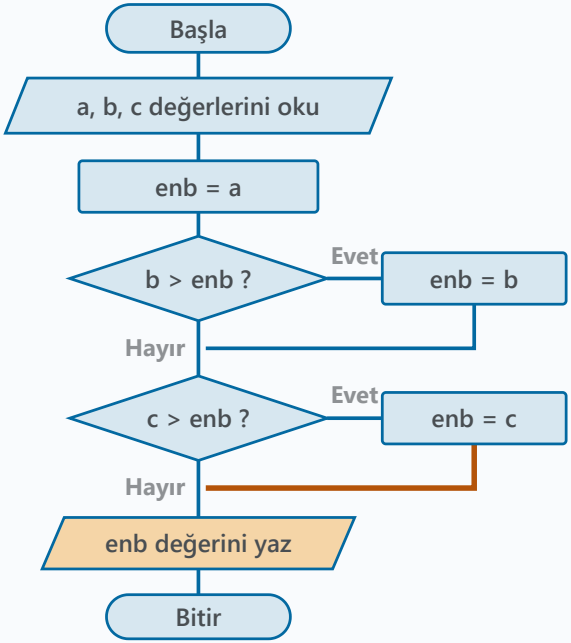
DEĞİŞKENLER

a = 3 | b = 2 | c = 8 | enb = 8

ÇIKTI

enb = c = 8.

Üç Sayının En Büyüğü Adım Adım




```
1 Başla
2   a, b, c değerlerini oku
3   enb = a
4   b > enb ?  Evet → enb = b
5   c > enb ?  Evet → enb = c
▶   enb değerini yaz
7 Bitir
```

DEĞİŞKENLER

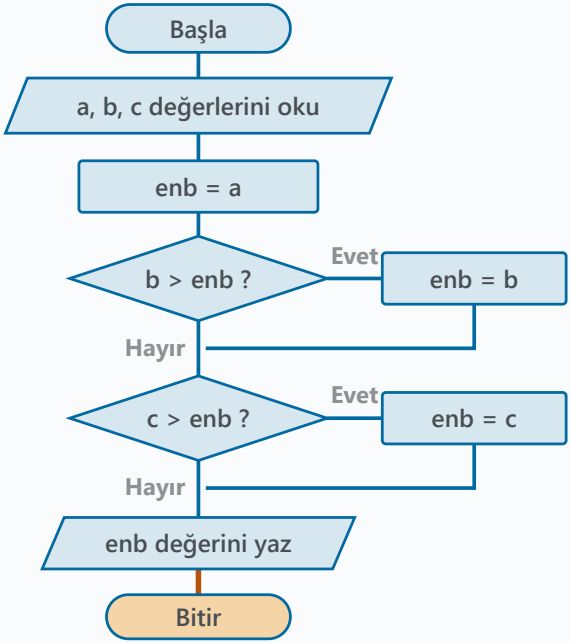
`a = 3` `b = 2` `c = 8` `enb = 8`

ÇIKTI

```
8
```

8 yazılır.

Üç Sayının En Büyüğü Adım Adım




```
1  Başla
2    a, b, c değerlerini oku
3    enb = a
4    b > enb ?  Evet → enb = b
5    c > enb ?  Evet → enb = c
6    enb değerini yaz
▶  Bitir
```

DEĞİŞKENLER

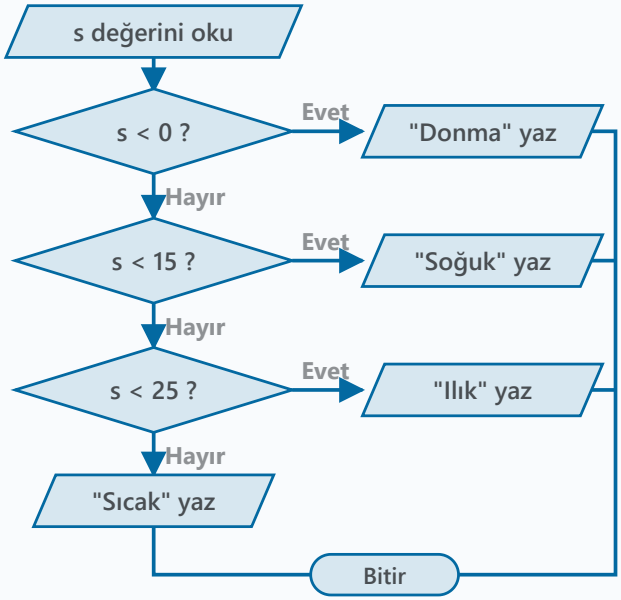
`a = 3` `b = 2` `c = 8` `enb = 8`

ÇIKTI

```
8
```

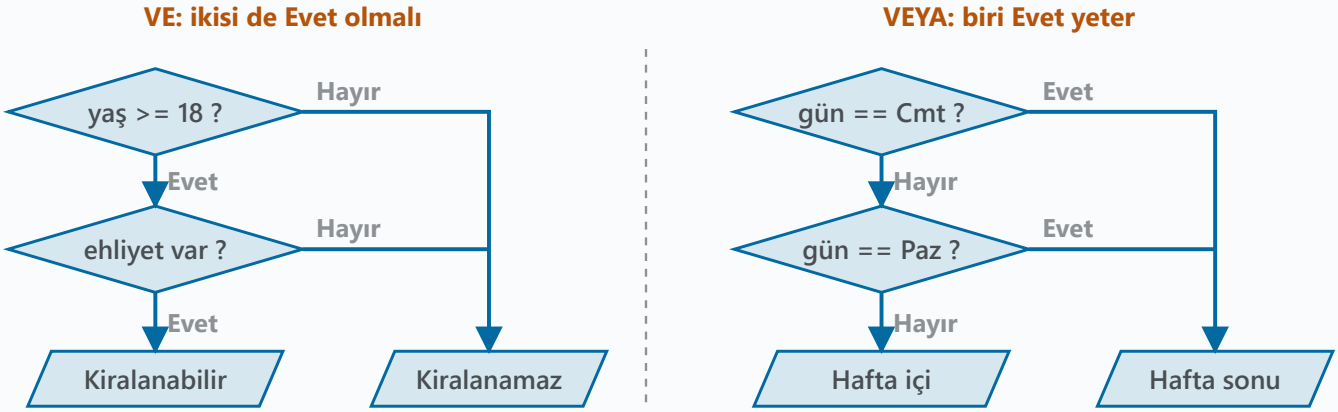
Bitir. Her karar yalnızca bir kez sorulur; sayı sayısı artsa da yöntem aynı kalır.

Çok Yollu Karar: Karar Zinciri



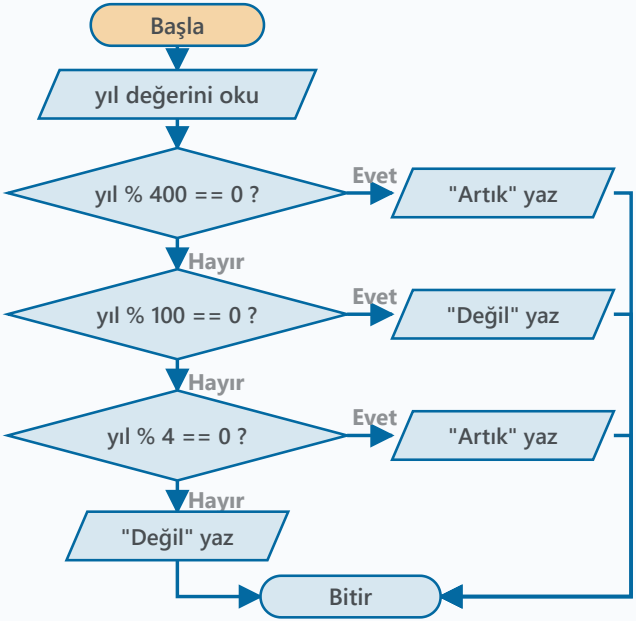
- Seçenek sayısı ikiden fazlaysa kararlar **zincir** biçiminde dizilir: her karar bir öncekinin **Hayır** dalındadır.
- İlk doğru koşulun dalı çalışır, sonraki kararlar hiç sorulmaz. s = 18 için üç karar sorulur: 18 < 0 ve 18 < 15 yanlış, 18 < 25 doğru → **Ilık**.
- Her karar yalnızca üst sınırı denetler; alt sınır zaten bilinir (s < 15 sorulurken s ≥ 0 olduğu kesindir).
- **Sıra önemlidir:** s < 25 en başta sorulsaydı s = −5 için de "Ilık" yazılırdı.
- n seçenek için n − 1 karar gerekir. Java karşılığı `if … else if … else` zinciridir.

Bileşik Koşul: VE / VEYA



- Bileşik koşul tek bir karar kutusuna da yazılabilir: **yaş >= 18 VE ehliyet var ?**. Diyagramda ayrıntı gerekirse iki karara açılır.
- **VE**: ilk karar Hayır ise ikinci karar **hiç sorulmaz**, sonuç zaten Hayır'dır. **VEYA**: ilk karar Evet ise ikinciye gerek kalmaz. Java'daki `&&` ve `||` da böyle çalışır: kısa devre değerlendirme.
- Test için her kararın iki çıkışı da denenir: VE örneğinde (20, var), (20, yok) ve (16, —) üç farklı yolu çalıştırır.

Artık Yıl Adım Adım (2024, 1900, 2000)



Artık yıl kuralı: 4'e bölünen yıllar artıktır; ama 100'e bölünenler değildir, 400'e bölünenler yine artıktır. Üç kararlı bir zincir. **1. çalıştırma: 2024.**

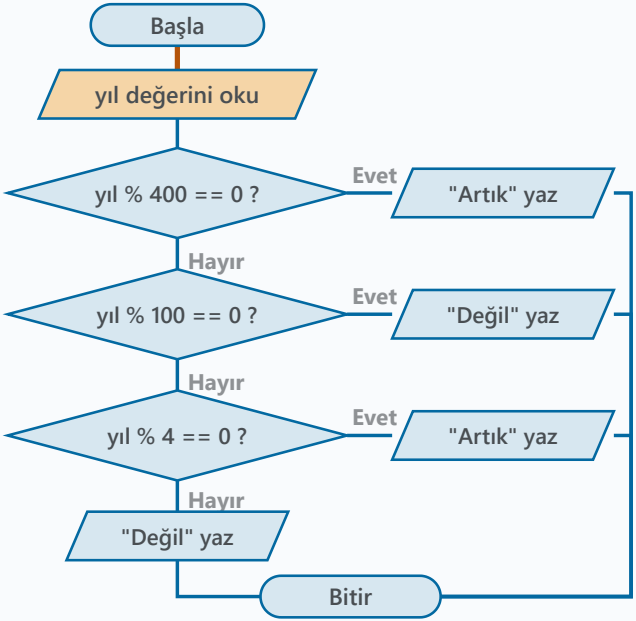
```
▶ Başla
2   yıl değerini oku
3   yıl % 400 == 0 ?  Evet → "Artık" yaz
4   yıl % 100 == 0 ?  Evet → "Değil" yaz
5   yıl % 4 == 0 ?    Evet → "Artık" yaz
6   Hayır → "Değil" yaz
7 Bitir
```

DEĞİŞKENLER

—

ÇIKTI

Artık Yıl Adım Adım (2024, 1900, 2000)




```
1  Başla
▶    yıl değerini oku
3    yıl % 400 == 0 ?  Evet → "Artık" yaz
4    yıl % 100 == 0 ?  Evet → "Değil" yaz
5    yıl % 4 == 0 ?    Evet → "Artık" yaz
6    Hayır → "Değil" yaz
7  Bitir
```

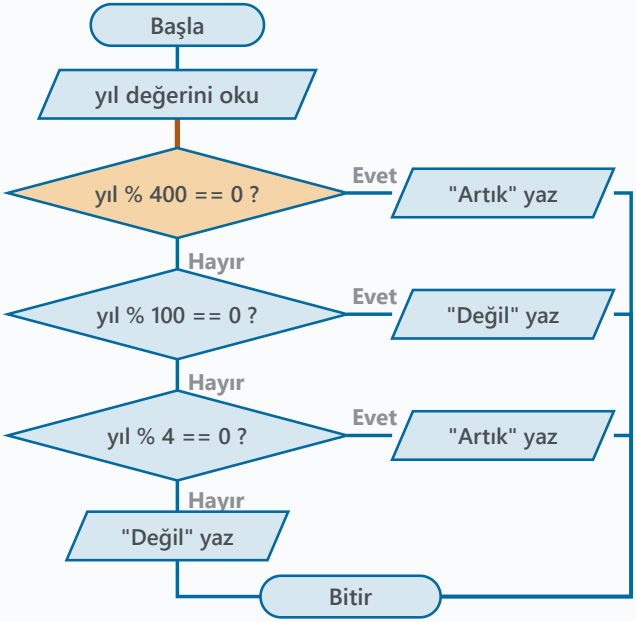
DEĞİŞKENLER

`yıl = 2024`

ÇIKTI

yıl = **2024** okunur.

Artık Yıl Adım Adım (2024, 1900, 2000)




```
1 Başla
2   yıl değerini oku
▶   yıl % 400 == 0 ?   Evet → "Artık" yaz
4   yıl % 100 == 0 ?   Evet → "Değil" yaz
5   yıl % 4 == 0 ?     Evet → "Artık" yaz
6   Hayır → "Değil" yaz
7 Bitir
```

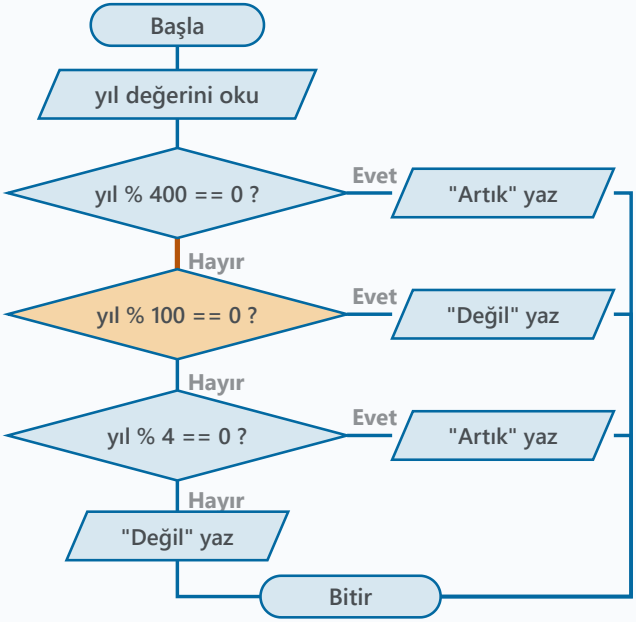
DEĞİŞKENLER

`yıl = 2024`

ÇIKTI

2024 % 400 = 24 → **yanlış** → Hayır.

Artık Yıl Adım Adım (2024, 1900, 2000)



2024 % 100 = 24 → **yanlış** → Hayır.

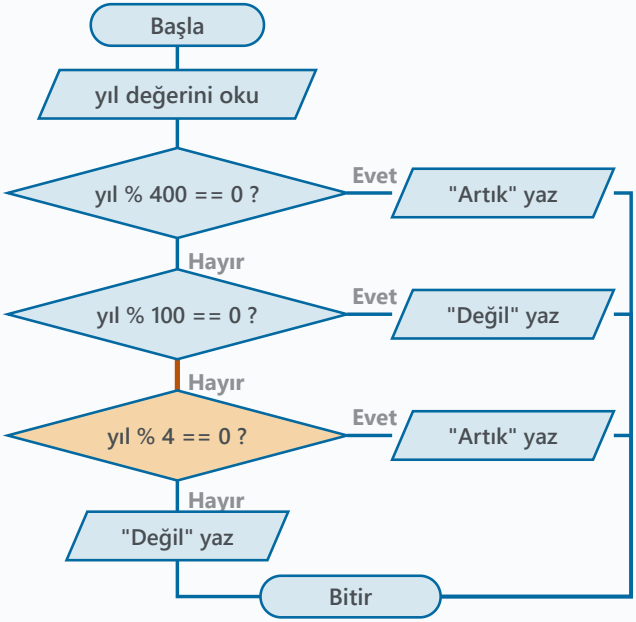
```
1 Başla
2   yıl değerini oku
3   yıl % 400 == 0 ?   Evet → "Artık" yaz
▶   yıl % 100 == 0 ?   Evet → "Değil" yaz
5   yıl % 4 == 0 ?     Evet → "Artık" yaz
6   Hayır → "Değil" yaz
7 Bitir
```

DEĞİŞKENLER

`yıl = 2024`

ÇIKTI

Artık Yıl Adım Adım (2024, 1900, 2000)



2024 % 4 = 0 → **doğru** → Evet.

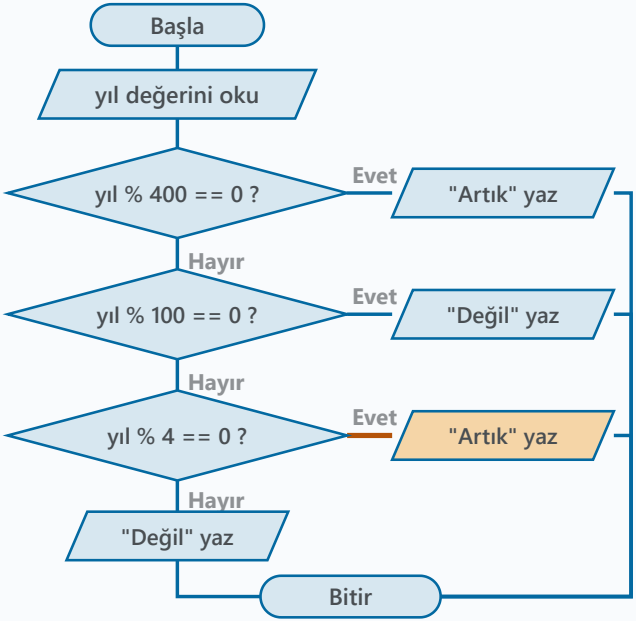
```
1 Başla
2   yıl değerini oku
3   yıl % 400 == 0 ?   Evet → "Artık" yaz
4   yıl % 100 == 0 ?   Evet → "Değil" yaz
▶   yıl % 4 == 0 ?     Evet → "Artık" yaz
6   Hayır → "Değil" yaz
7 Bitir
```

DEĞİŞKENLER

`yıl = 2024`

ÇIKTI

Artık Yıl Adım Adım (2024, 1900, 2000)




```
1 Başla
2   yıl değerini oku
3   yıl % 400 == 0 ?   Evet → "Artık" yaz
4   yıl % 100 == 0 ?   Evet → "Değil" yaz
▶   yıl % 4 == 0 ?     Evet → "Artık" yaz
6   Hayır → "Değil" yaz
7 Bitir
```

DEĞİŞKENLER

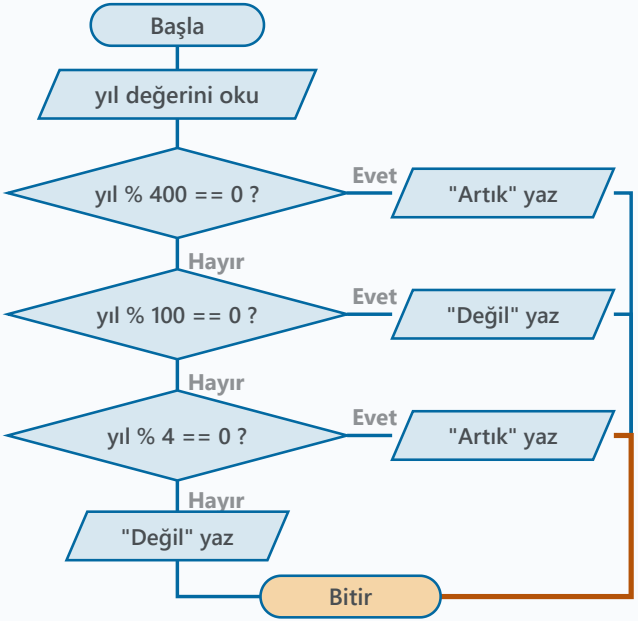
`yıl = 2024`

ÇIKTI

```
Artık
```

"Artık" yazılır. 2024 artık yıldır (Şubat 29 gün).

Artık Yıl Adım Adım (2024, 1900, 2000)




```
1 Başla
2   yıl değerini oku
3   yıl % 400 == 0 ?  Evet → "Artık" yaz
4   yıl % 100 == 0 ?  Evet → "Değil" yaz
5   yıl % 4 == 0 ?    Evet → "Artık" yaz
6   Hayır → "Değil" yaz
▶ Bitir
```

DEĞİŞKENLER

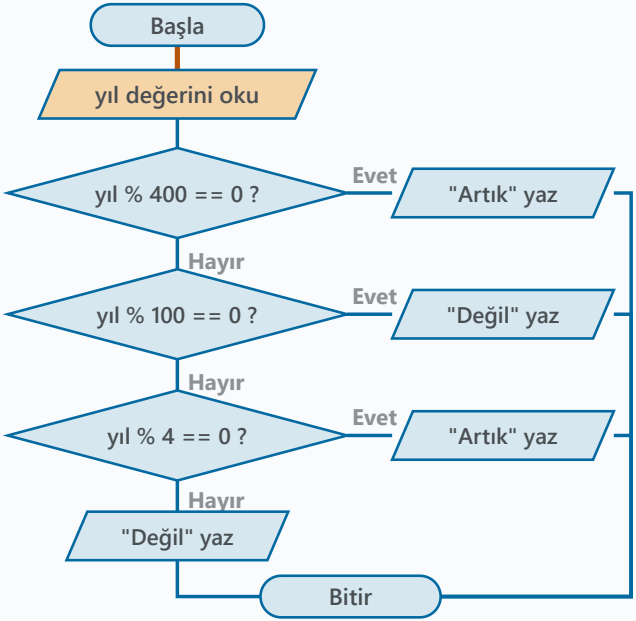
`yıl = 2024`

ÇIKTI

```
Artık
```

Bitir. **2. çalıştırma: 1900.**

Artık Yıl Adım Adım (2024, 1900, 2000)




```
1 Başla
▶   yıl değerini oku
3   yıl % 400 == 0 ?  Evet → "Artık" yaz
4   yıl % 100 == 0 ?  Evet → "Değil" yaz
5   yıl % 4 == 0 ?    Evet → "Artık" yaz
6   Hayır → "Değil" yaz
7 Bitir
```

DEĞİŞKENLER

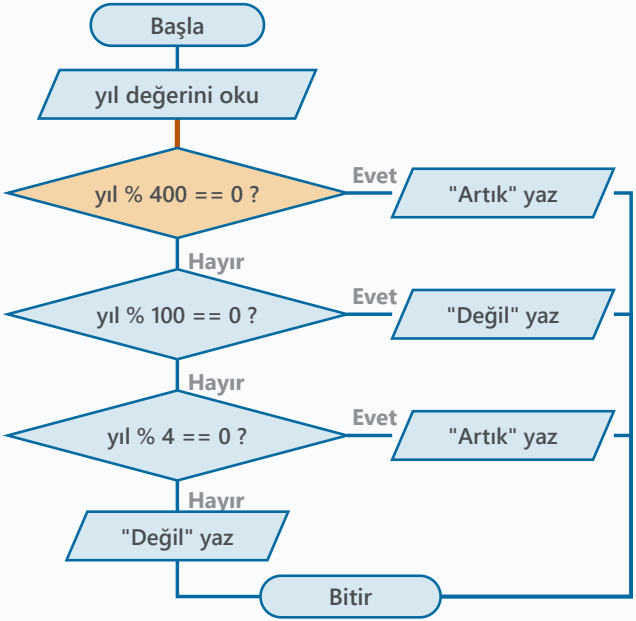
`yıl = 1900`

ÇIKTI

```
Artık
```

yıl = **1900** okunur.

Artık Yıl Adım Adım (2024, 1900, 2000)



1900 % 400 = 300 → **yanlış** → Hayır.

```
1 Başla
2   yıl değerini oku
▶   yıl % 400 == 0 ?  Evet → "Artık" yaz
4   yıl % 100 == 0 ?  Evet → "Değil" yaz
5   yıl % 4 == 0 ?    Evet → "Artık" yaz
6   Hayır → "Değil" yaz
7 Bitir
```

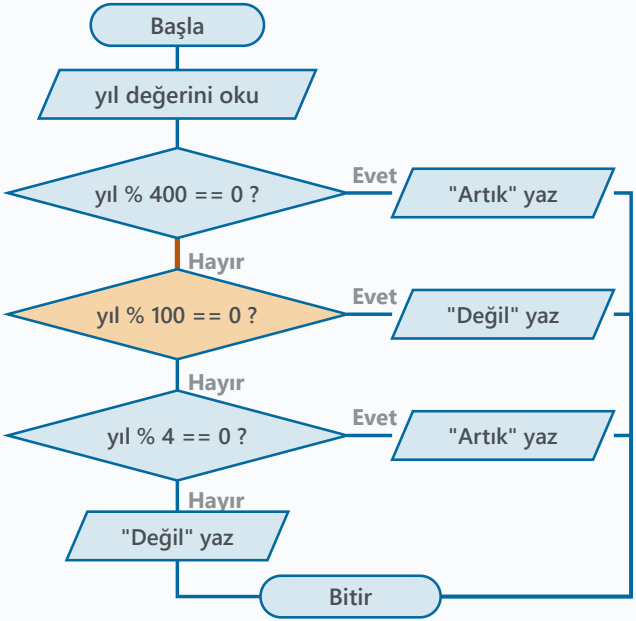
DEĞİŞKENLER

`yıl = 1900`

ÇIKTI

```
Artık
```

Artık Yıl Adım Adım (2024, 1900, 2000)



1900 % 100 = 0 → **doğru** → Evet. Üçüncü karar hiç sorulmaz.

```
1 Başla
2   yıl değerini oku
3   yıl % 400 == 0 ?  Evet → "Artık" yaz
▶   yıl % 100 == 0 ?  Evet → "Değil" yaz
5   yıl % 4 == 0 ?    Evet → "Artık" yaz
6   Hayır → "Değil" yaz
7 Bitir
```

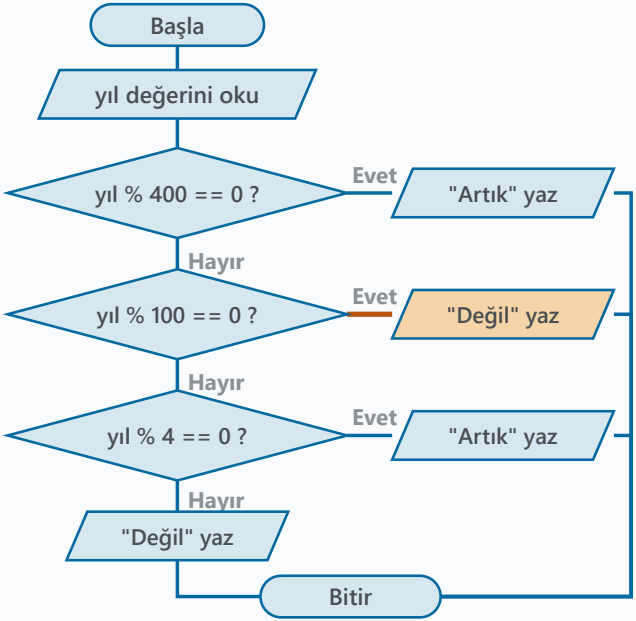
DEĞİŞKENLER

`yıl = 1900`

ÇIKTI

```
Artık
```

Artık Yıl Adım Adım (2024, 1900, 2000)




```
1 Başla
2   yıl değerini oku
3   yıl % 400 == 0 ?   Evet → "Artık" yaz
▶   yıl % 100 == 0 ?   Evet → "Değil" yaz
5   yıl % 4 == 0 ?     Evet → "Artık" yaz
6   Hayır → "Değil" yaz
7 Bitir
```

DEĞİŞKENLER

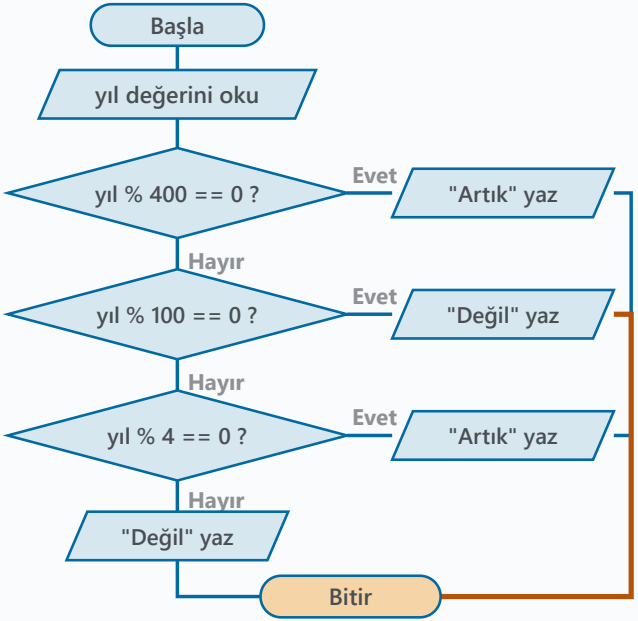
`yıl = 1900`

ÇIKTI

```
Artık
Değil
```

“Değil” yazılır. 1900 4’e bölünse de artık yıl **değildir**; yalnız % 4’e bakan bir diyagram burada yanılırdı.

Artık Yıl Adım Adım (2024, 1900, 2000)




```
1 Başla
2   yıl değerini oku
3   yıl % 400 == 0 ?  Evet → "Artık" yaz
4   yıl % 100 == 0 ?  Evet → "Değil" yaz
5   yıl % 4 == 0 ?    Evet → "Artık" yaz
6   Hayır → "Değil" yaz
▶ Bitir
```

DEĞİŞKENLER

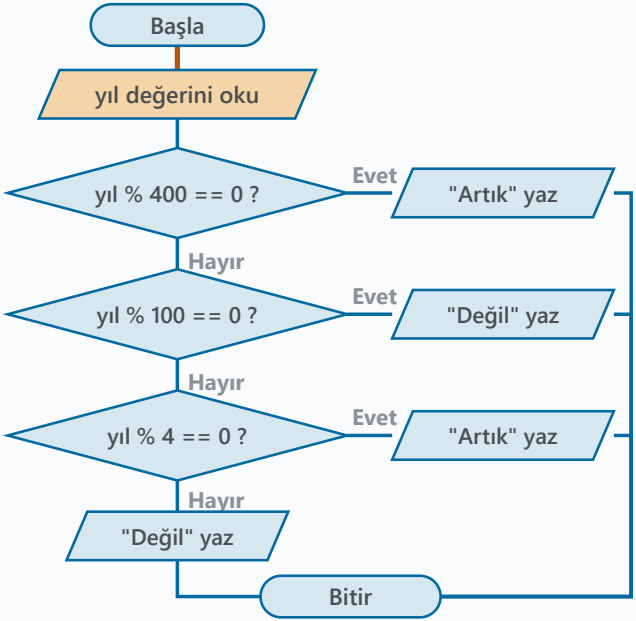
`yıl = 1900`

ÇIKTI

```
Artık
Değil
```

Bitir. **3. çalıştırma: 2000.**

Artık Yıl Adım Adım (2024, 1900, 2000)



yıl = **2000** okunur.

```
1 Başla
▶   yıl değerini oku
3   yıl % 400 == 0 ?  Evet → "Artık" yaz
4   yıl % 100 == 0 ?  Evet → "Değil" yaz
5   yıl % 4 == 0 ?    Evet → "Artık" yaz
6   Hayır → "Değil" yaz
7 Bitir
```

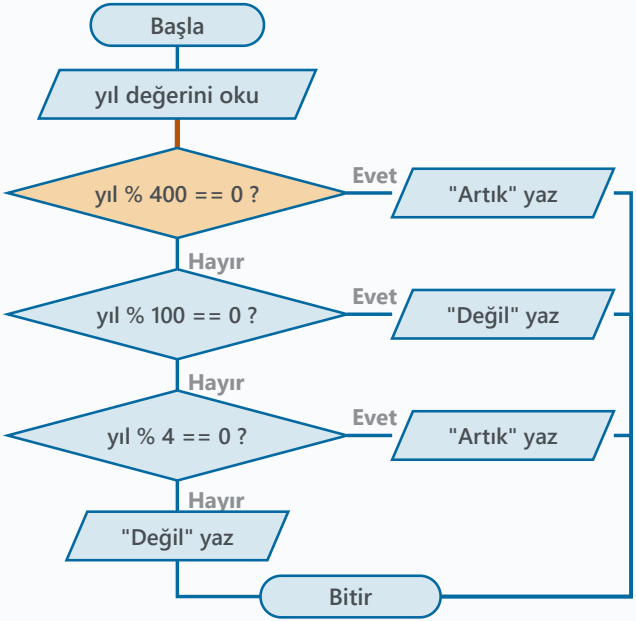
DEĞİŞKENLER

`yıl = 2000`

ÇIKTI

```
Artık
Değil
```

Artık Yıl Adım Adım (2024, 1900, 2000)



2000 % 400 = 0 → **doğru** → Evet. Kalan iki karar sorulmaz.

```
1 Başla
2   yıl değerini oku
▶   yıl % 400 == 0 ?  Evet → "Artık" yaz
4   yıl % 100 == 0 ?  Evet → "Değil" yaz
5   yıl % 4 == 0 ?    Evet → "Artık" yaz
6   Hayır → "Değil" yaz
7 Bitir
```

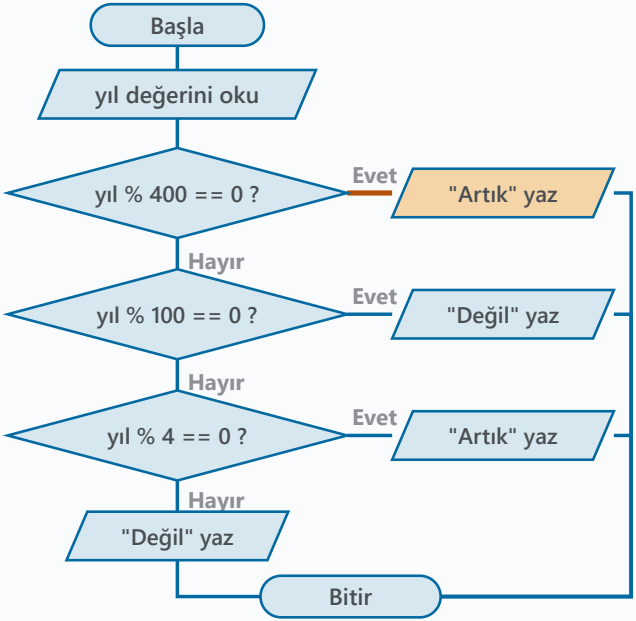
DEĞİŞKENLER

`yıl = 2000`

ÇIKTI

```
Artık
Değil
```

Artık Yıl Adım Adım (2024, 1900, 2000)



“Artık” yazılır. 2000, 100’e bölündüğü hâlde 400’e de bölündüğü için artık yıldır.

```
1 Başla
2   yıl değerini oku
▶   yıl % 400 == 0 ?  Evet → "Artık" yaz
4   yıl % 100 == 0 ?  Evet → "Değil" yaz
5   yıl % 4 == 0 ?    Evet → "Artık" yaz
6   Hayır → "Değil" yaz
7 Bitir
```

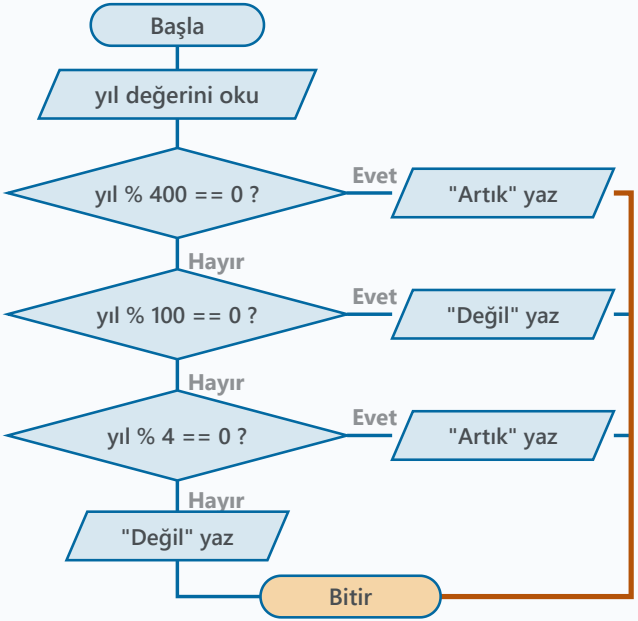
DEĞİŞKENLER

`yıl = 2000`

ÇIKTI

```
Artık
Değil
Artık
```

Artık Yıl Adım Adım (2024, 1900, 2000)




```
1 Başla
2   yıl değerini oku
3   yıl % 400 == 0 ?  Evet → "Artık" yaz
4   yıl % 100 == 0 ?  Evet → "Değil" yaz
5   yıl % 4 == 0 ?    Evet → "Artık" yaz
6   Hayır → "Değil" yaz
▶ Bitir
```

DEĞİŞKENLER

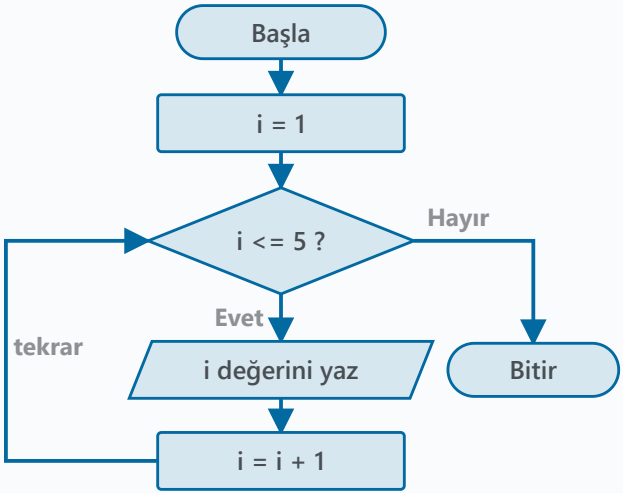
`yıl = 2000`

ÇIKTI

```
Artık
Değil
Artık
```

Bitir. Üç girdi dört çıkıştan üçünü çalıştırdı; 2023 gibi bir yıl son Hayır dalını dener. Kararların **sırası** kuralın istisnalarını doğru yakalar.

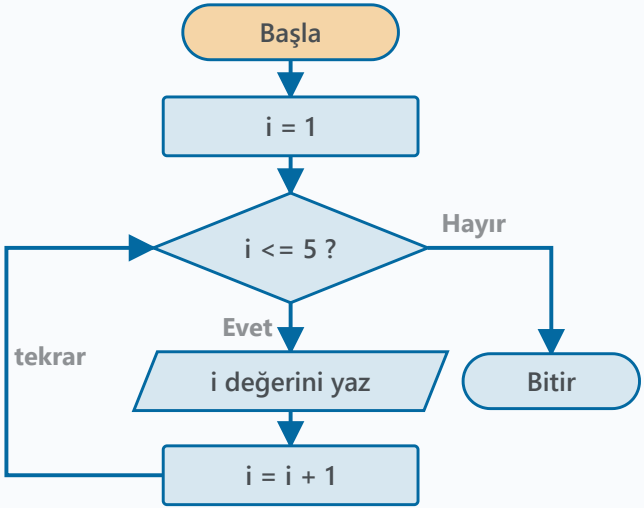
Tekrarlı Akış




```
Başla
  i = 1
  i <= 5 ?
    Evet → i değerini yaz
           i = i + 1
           karara dön
    Hayır → Bitir
```

- Geri dönen ok (tekrar) döngüyü oluşturur; koşul gövdeden önce sorulur (ön koşullu döngü) ve yanlış olunca döngü sona erer.
- Tekrar yolunda koşulu değiştiren bir işlem (**i = i + 1**) bulunmalıdır.

Tekrarlı Akış Adım Adım



Başla.

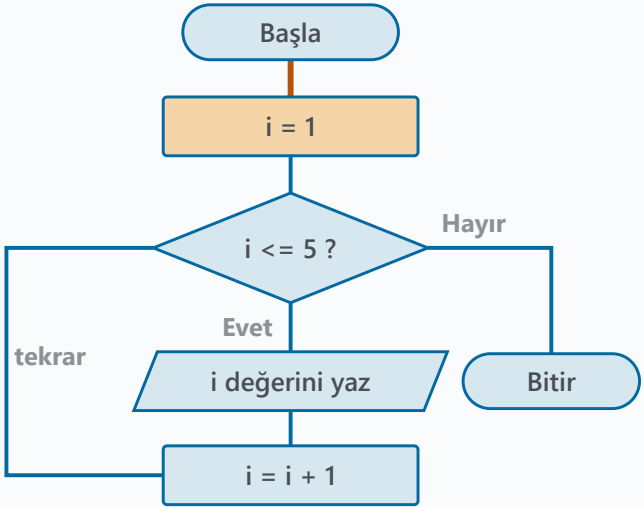
```
▶ Başla
2   i = 1
3   i <= 5 ?
4     Evet → i değerini yaz
5             i = i + 1
6             karara dön
7     Hayır → Bitir
```

DEĞİŞKENLER

—

ÇIKTI

Tekrarlı Akış Adım Adım



Sayaç ilk değerini alır: **i = 1**.

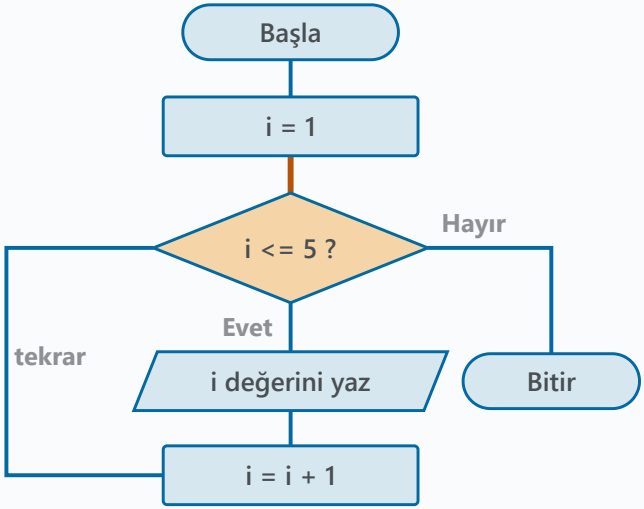
```
1 Başla
▶   i = 1
3   i <= 5 ?
4     Evet → i değerini yaz
5             i = i + 1
6             karara dön
7     Hayır → Bitir
```

DEĞİŞKENLER

`i = 1`

ÇIKTI

Tekrarlı Akış Adım Adım



Karar: **1 <= 5** → **doğru** → Evet, döngü gövdesine gir.

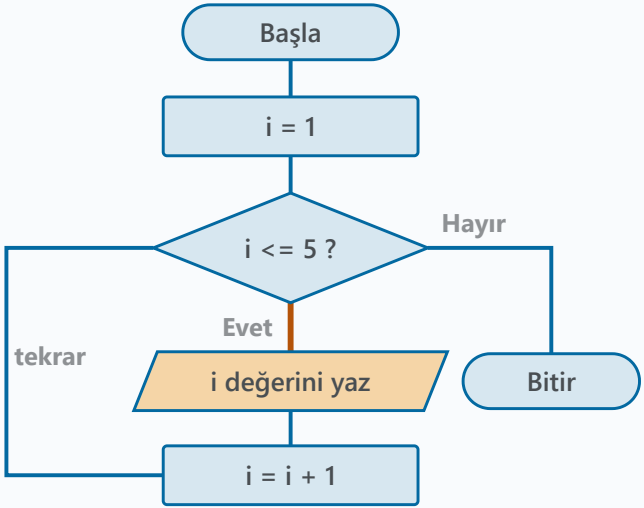
```
1 Başla
2   i = 1
▶   i <= 5 ?
4     Evet → i değerini yaz
5             i = i + 1
6             karara dön
7     Hayır → Bitir
```

DEĞİŞKENLER

`i = 1`

ÇIKTI

Tekrarlı Akış Adım Adım



1 yazılır.

```
1  Başla
2    i = 1
3    i <= 5 ?
▶      Evet → i değerini yaz
5              i = i + 1
6              karara dön
7      Hayır → Bitir
```

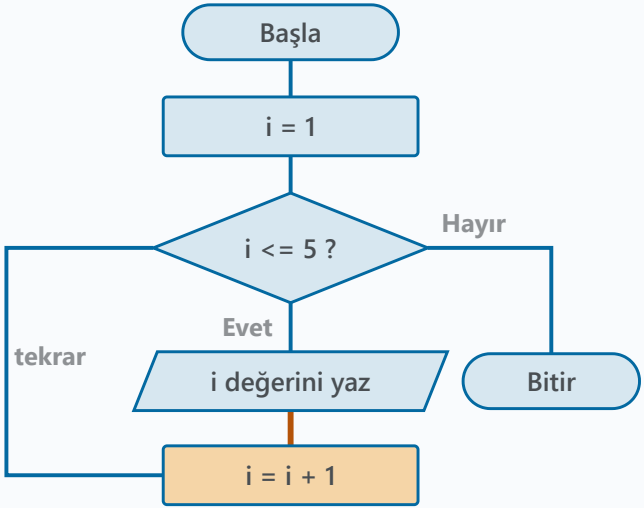
DEĞİŞKENLER

`i = 1`

ÇIKTI

```
1
```

Tekrarlı Akış Adım Adım



i = 1 + 1 = 2. Geri dönen ok yeniden karara götürür.

```
1 Başla
2   i = 1
3   i <= 5 ?
4     Evet → i değerini yaz
▶               i = i + 1
6               karara dön
7     Hayır → Bitir
```

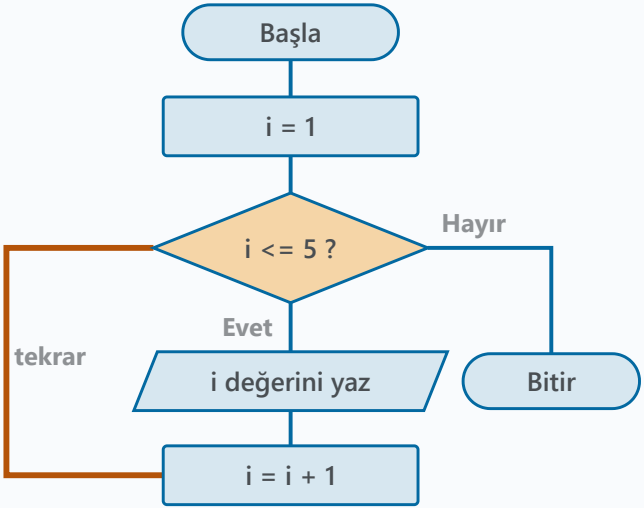
DEĞİŞKENLER

`i = 2`

ÇIKTI

```
1
```

Tekrarlı Akış Adım Adım



Karar: **2 <= 5** → **doğru** → Evet, döngü gövdesine gir.

```
1 Başla
2   i = 1
▶   i <= 5 ?
4     Evet → i değerini yaz
5             i = i + 1
6             karara dön
7     Hayır → Bitir
```

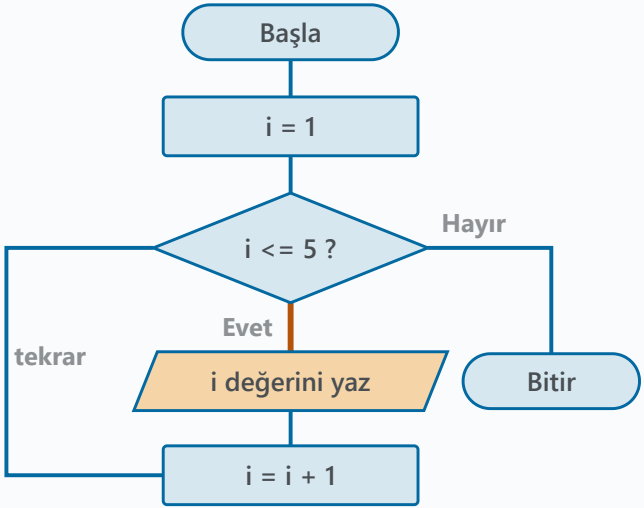
DEĞİŞKENLER

`i = 2`

ÇIKTI

```
1
```

Tekrarlı Akış Adım Adım



2 yazılır.

```
1 Başla
2   i = 1
3   i <= 5 ?
▶     Evet → i değerini yaz
5             i = i + 1
6             karara dön
7     Hayır → Bitir
```

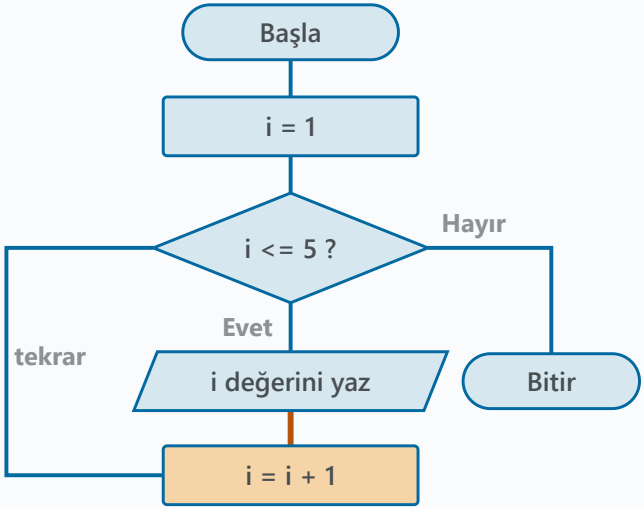
DEĞİŞKENLER

`i = 2`

ÇIKTI

```
1 2
```

Tekrarlı Akış Adım Adım



i = 2 + 1 = 3. Geri dönen ok yeniden karara götürür.

```
1 Başla
2   i = 1
3   i <= 5 ?
4     Evet → i değerini yaz
▶               i = i + 1
6               karara dön
7     Hayır → Bitir
```

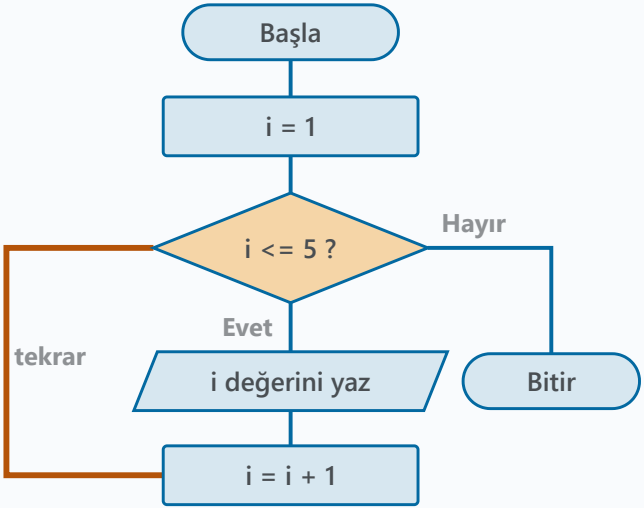
DEĞİŞKENLER

`i = 3`

ÇIKTI

```
1 2
```

Tekrarlı Akış Adım Adım



Karar: **3 <= 5** → **doğru** → Evet, döngü gövdesine gir.

```
1 Başla
2   i = 1
▶   i <= 5 ?
4     Evet → i değerini yaz
5               i = i + 1
6               karara dön
7     Hayır → Bitir
```

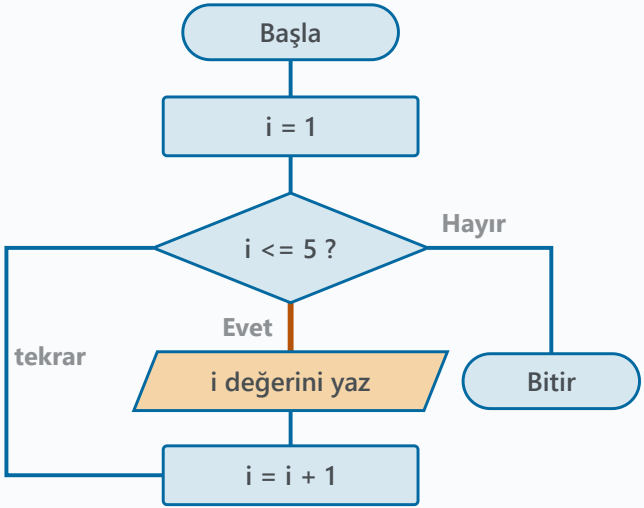
DEĞİŞKENLER

`i = 3`

ÇIKTI

```
1 2
```

Tekrarlı Akış Adım Adım



3 yazılır.

```
1 Başla
2   i = 1
3   i <= 5 ?
▶     Evet → i değerini yaz
5             i = i + 1
6             karara dön
7     Hayır → Bitir
```

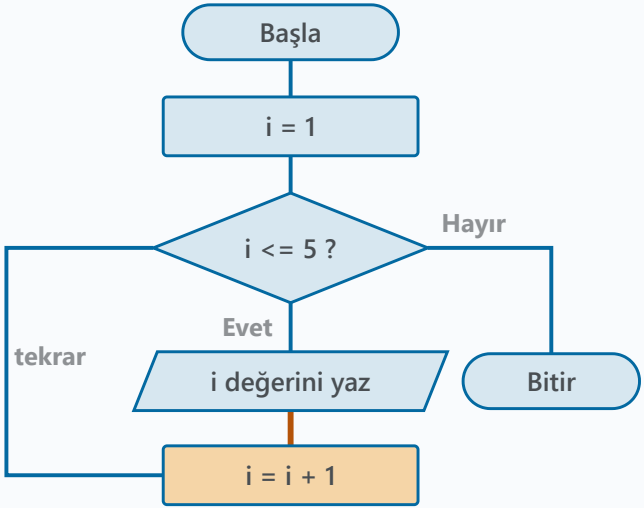
DEĞİŞKENLER

`i = 3`

ÇIKTI

```
1 2 3
```

Tekrarlı Akış Adım Adım



i = 3 + 1 = 4. Geri dönen ok yeniden karara götürür.

```
1 Başla
2   i = 1
3   i <= 5 ?
4     Evet → i değerini yaz
▶               i = i + 1
6               karara dön
7     Hayır → Bitir
```

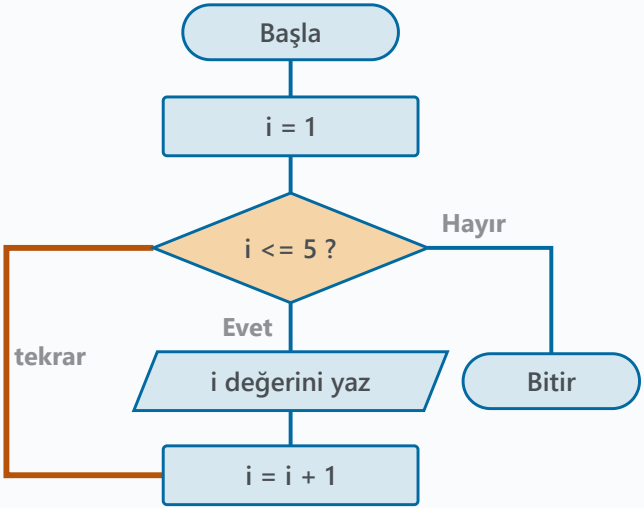
DEĞİŞKENLER

`i = 4`

ÇIKTI

```
1 2 3
```

Tekrarlı Akış Adım Adım



Karar: **4 <= 5** → **doğru** → Evet, döngü gövdesine gir.

```
1 Başla
2   i = 1
▶   i <= 5 ?
4     Evet → i değerini yaz
5             i = i + 1
6             karara dön
7     Hayır → Bitir
```

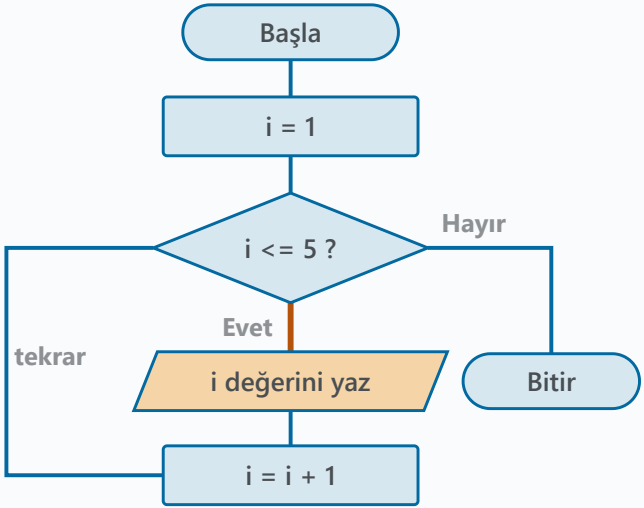
DEĞİŞKENLER

`i = 4`

ÇIKTI

```
1 2 3
```

Tekrarlı Akış Adım Adım



4 yazılır.

```
1 Başla
2   i = 1
3   i <= 5 ?
▶     Evet → i değerini yaz
5             i = i + 1
6             karara dön
7     Hayır → Bitir
```

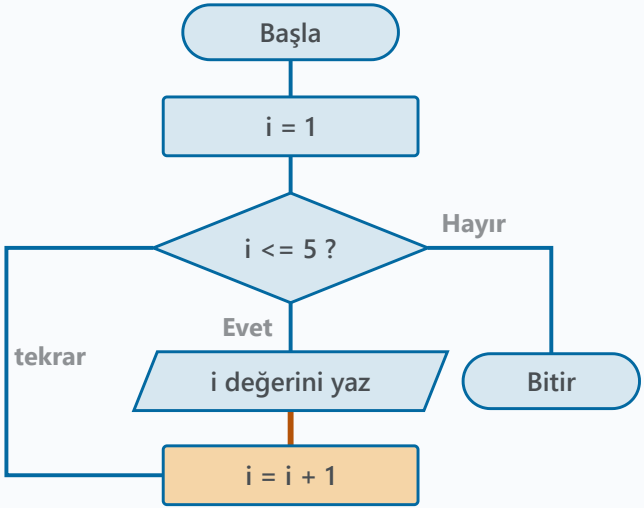
DEĞİŞKENLER

`i = 4`

ÇIKTI

```
1 2 3 4
```

Tekrarlı Akış Adım Adım



i = 4 + 1 = 5. Geri dönen ok yeniden karara götürür.

```
1 Başla
2   i = 1
3   i <= 5 ?
4     Evet → i değerini yaz
▶               i = i + 1
6               karara dön
7     Hayır → Bitir
```

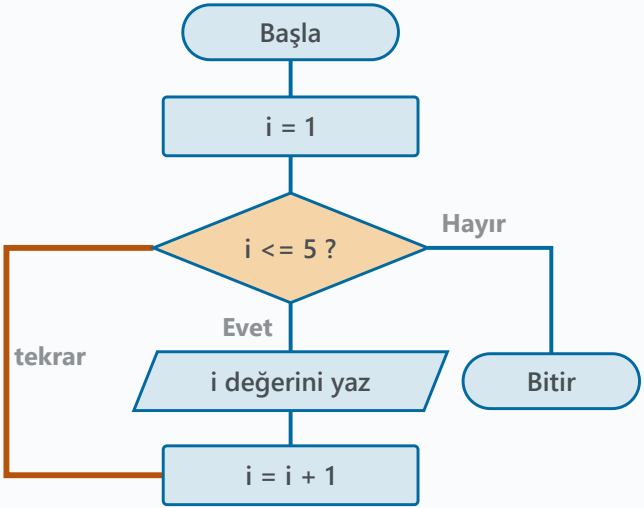
DEĞİŞKENLER

`i = 5`

ÇIKTI

```
1 2 3 4
```

Tekrarlı Akış Adım Adım



Karar: **5 <= 5** → **doğru** → Evet, döngü gövdesine gir.

```
1 Başla
2   i = 1
    i <= 5 ?
4     Evet → i değerini yaz
5             i = i + 1
6             karara dön
7     Hayır → Bitir
```

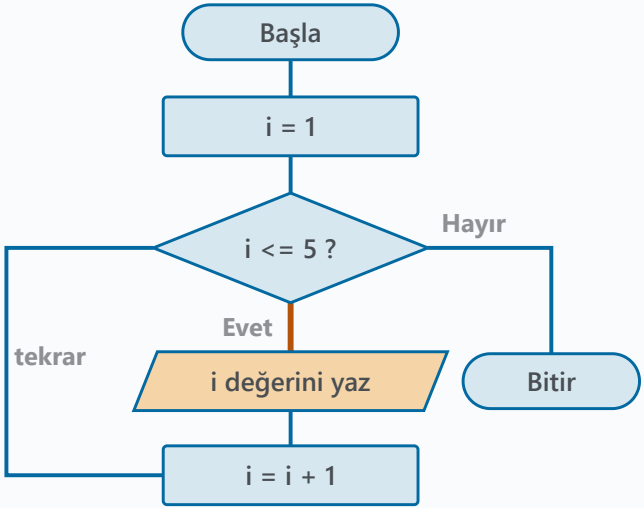
DEĞİŞKENLER

`i = 5`

ÇIKTI

```
1 2 3 4
```

Tekrarlı Akış Adım Adım



5 yazılır.

```
1 Başla
2   i = 1
3   i <= 5 ?
▶     Evet → i değerini yaz
5             i = i + 1
6             karara dön
7     Hayır → Bitir
```

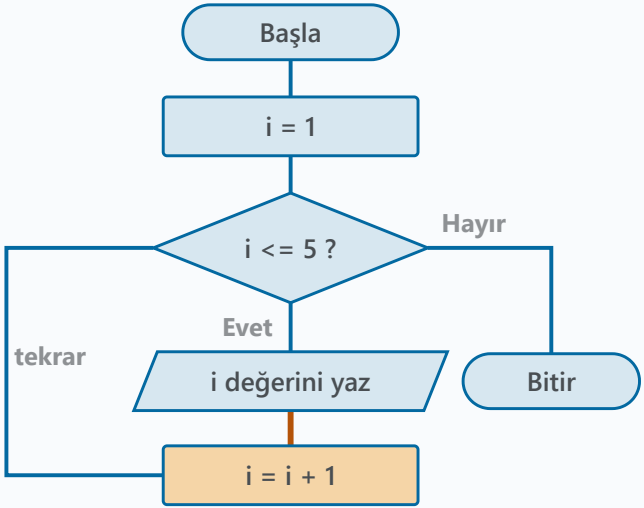
DEĞİŞKENLER

`i = 5`

ÇIKTI

```
1 2 3 4 5
```

Tekrarlı Akış Adım Adım



i = 5 + 1 = 6. Geri dönen ok yeniden karara götürür.

```
1 Başla
2   i = 1
3   i <= 5 ?
4     Evet → i değerini yaz
▶               i = i + 1
6               karara dön
7     Hayır → Bitir
```

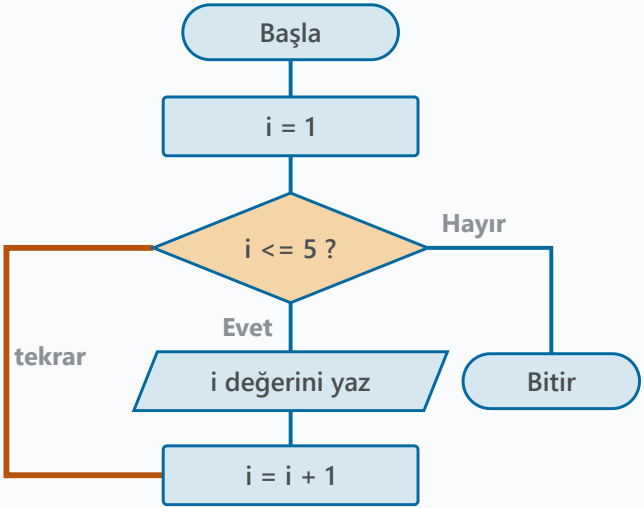
DEĞİŞKENLER

`i = 6`

ÇIKTI

```
1 2 3 4 5
```

Tekrarlı Akış Adım Adım



Karar: **6 <= 5** → **yanlış** → Hayır, döngüden çık.

```
1 Başla
2   i = 1
▶   i <= 5 ?
4     Evet → i değerini yaz
5             i = i + 1
6             karara dön
7     Hayır → Bitir
```

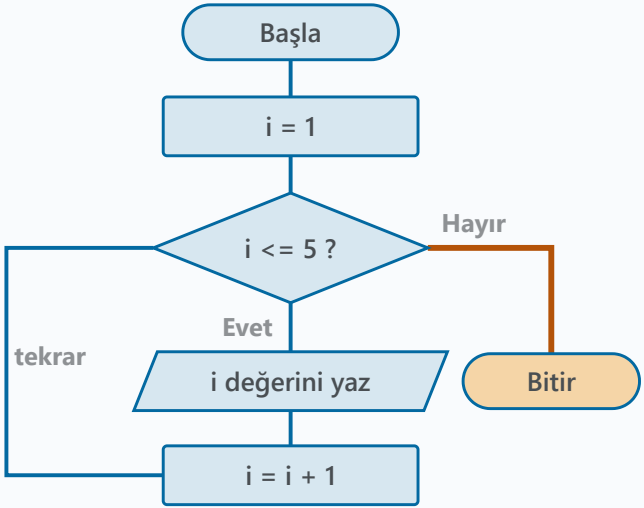
DEĞİŞKENLER

`i = 6`

ÇIKTI

```
1 2 3 4 5
```

Tekrarlı Akış Adım Adım



Bitir. Gövde **5 kez** çalıştı; koşul **6 kez** denendi (son deneme yanlış, i = 6).

```
1 Başla
2   i = 1
3   i <= 5 ?
4     Evet → i değerini yaz
5             i = i + 1
6             karara dön
▶     Hayır → Bitir
```

DEĞİŞKENLER

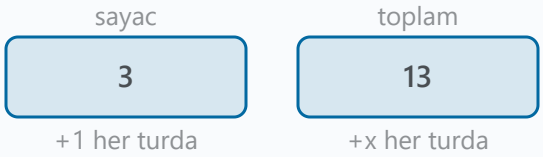
`i = 6`

ÇIKTI

```
1 2 3 4 5
```

Sayaç ve Biriktirici

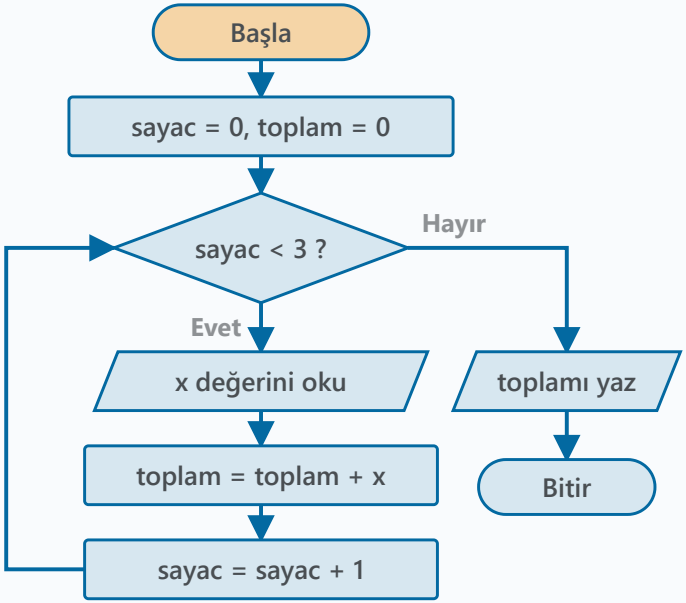
- **Sayaç** kaç tekrar yapıldığını izler: her turda 1 artar.
- **Biriktirici** kısmi sonucu tutar: her turda yeni değer eklenir.
- Başlangıç değeri işleme uygun seçilir: toplam için **0**, çarpım için **1**.



Girdiler 4, 7, 2 için tur tur değerler:

An	x	sayac	toplam
başlangıç	—	0	0
1. tur	4	1	4
2. tur	7	2	11
3. tur	2	3	13

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



Başla. Üç sayı okunup toplanacak; girdiler sırasıyla 4, 7, 2.

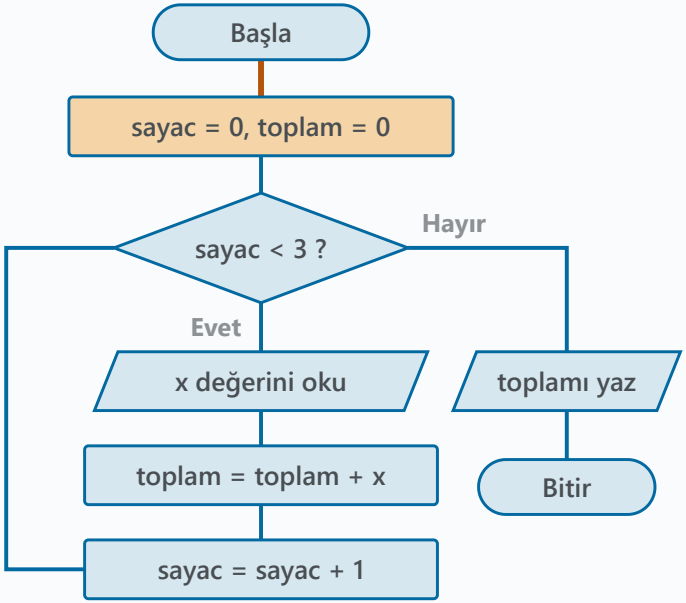
```
▶ Başla
2   sayac = 0, toplam = 0
3   sayac < 3 ?
4     Evet → x değerini oku
5               toplam = toplam + x
6               sayac = sayac + 1
7               karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

—

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



Sayaç ve biriktirici **0** ile başlar.

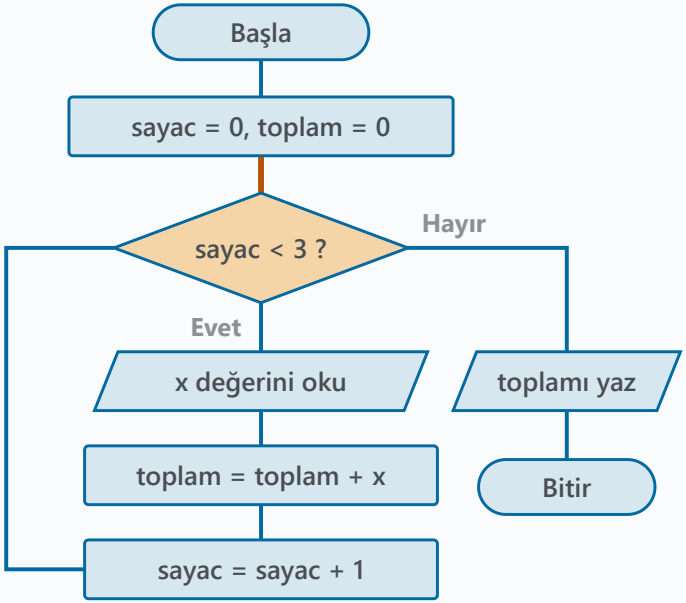
```
1 Başla
▶   sayac = 0, toplam = 0
3   sayac < 3 ?
4     Evet → x değerini oku
5               toplam = toplam + x
6               sayac = sayac + 1
7               karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`sayac = 0` `toplam = 0`

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



Karar: **0 < 3** → **doğru** → Evet.

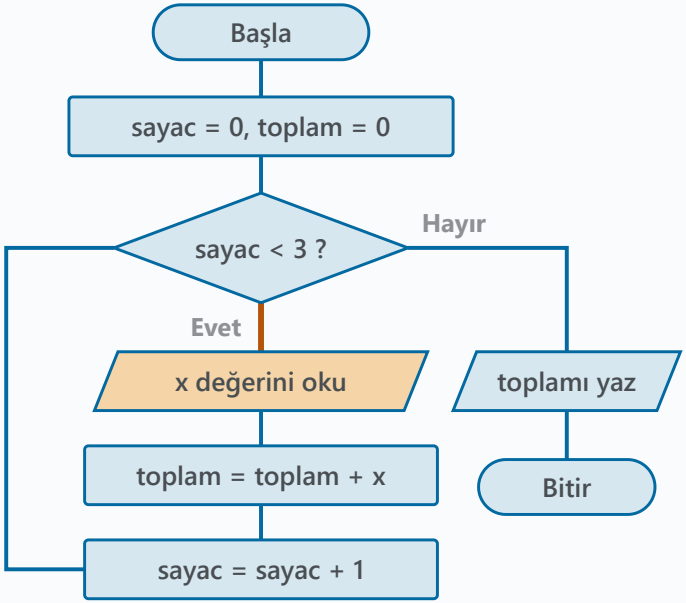
```
1 Başla
2   sayac = 0, toplam = 0
▶   sayac < 3 ?
4     Evet → x değerini oku
5               toplam = toplam + x
6               sayac = sayac + 1
7               karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`sayac = 0` `toplam = 0`

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



x = 4 okunur.

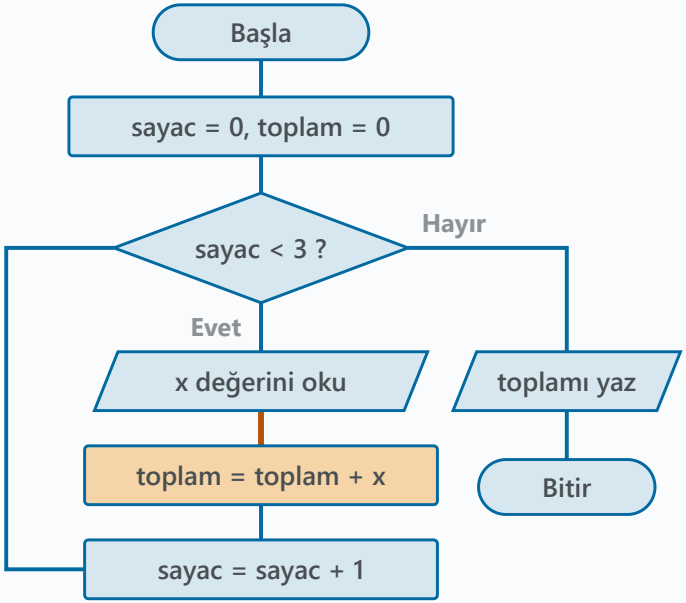
```
1 Başla
2   sayac = 0, toplam = 0
3   sayac < 3 ?
▶     Evet → x değerini oku
5             toplam = toplam + x
6             sayac = sayac + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`sayac = 0` `toplam = 0` `x = 4`

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



Biriktirici: **toplam = 0 + 4 = 4**.

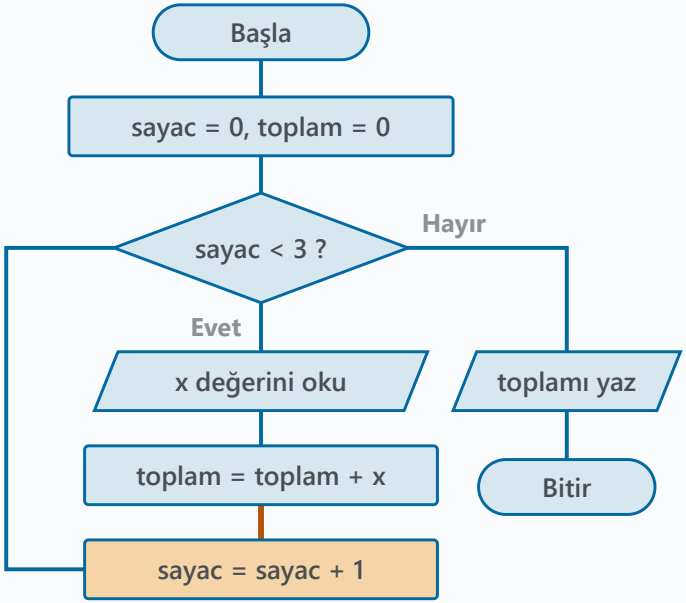
```
1  Başla
2    sayac = 0, toplam = 0
3    sayac < 3 ?
4      Evet → x değerini oku
5              toplam = toplam + x
6              sayac = sayac + 1
7              karara dön
8      Hayır → toplamı yaz
9  Bitir
```

DEĞİŞKENLER

`sayac = 0` `toplam = 4` `x = 4`

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



Sayaç: **sayac = 0 + 1 = 1**. Geri dön.

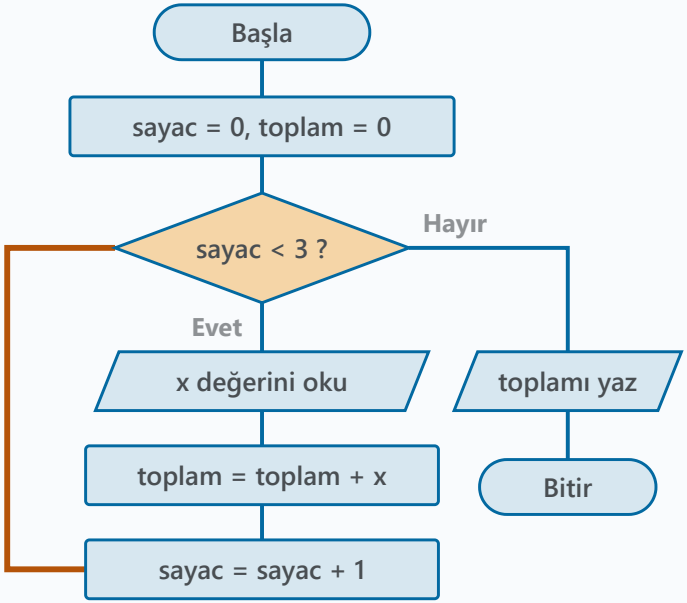
```
1 Başla
2   sayac = 0, toplam = 0
3   sayac < 3 ?
4     Evet → x değerini oku
5               toplam = toplam + x
6               sayac = sayac + 1
7               karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`sayac = 1` `toplam = 4` `x = 4`

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



Karar: **1 < 3** → **doğru** → Evet.

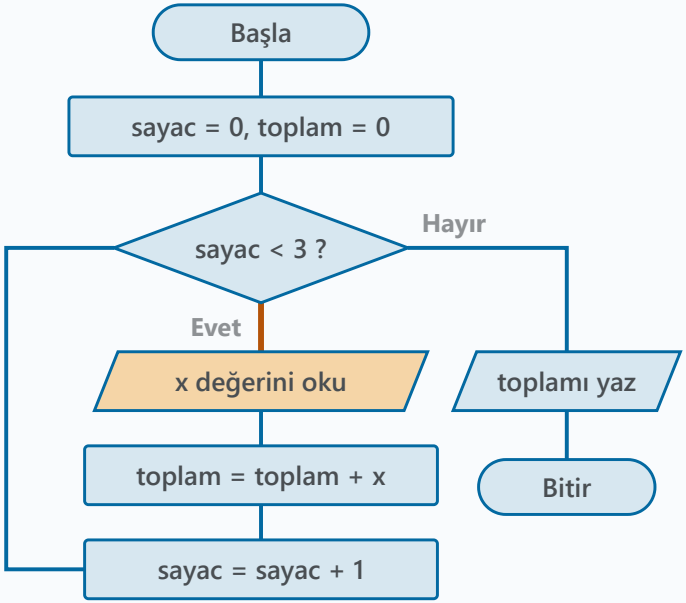
```
1 Başla
2   sayac = 0, toplam = 0
▶   sayac < 3 ?
4     Evet → x değerini oku
5               toplam = toplam + x
6               sayac = sayac + 1
7               karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`sayac = 1` `toplam = 4` `x = 4`

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



x = 7 okunur.

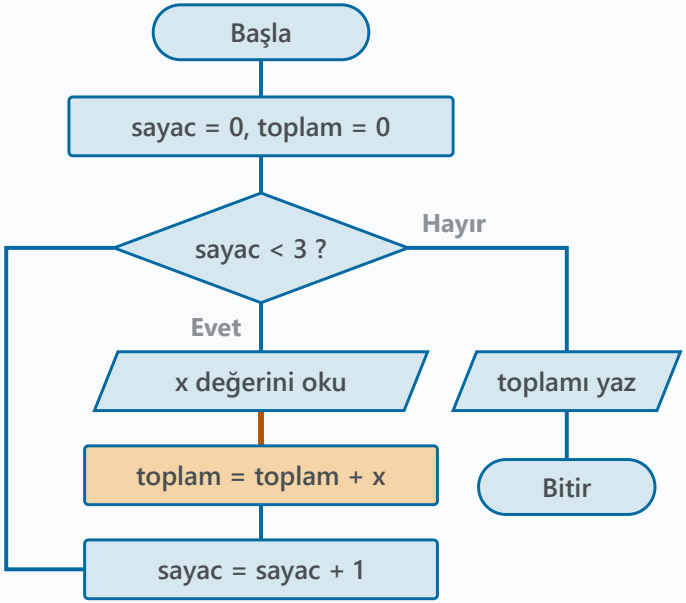
```
1 Başla
2   sayac = 0, toplam = 0
3   sayac < 3 ?
▶     Evet → x değerini oku
5               toplam = toplam + x
6               sayac = sayac + 1
7               karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`sayac = 1` `toplam = 4` `x = 7`

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



Biriktirici: **toplam = 4 + 7 = 11**.

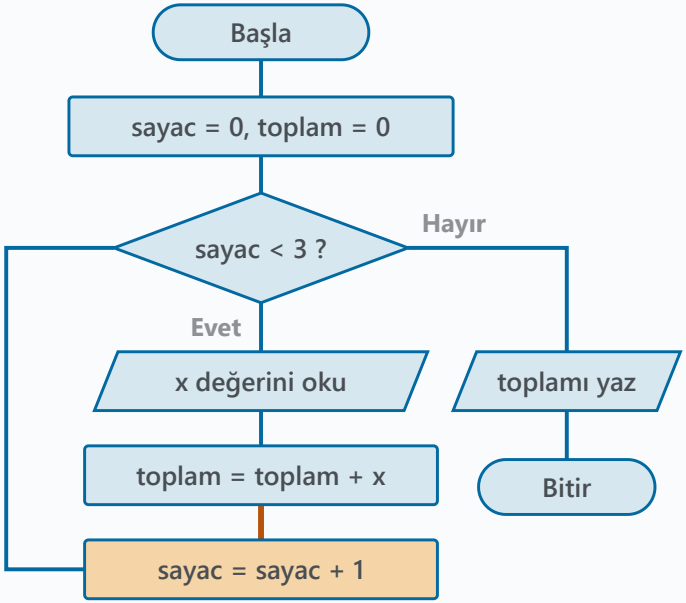
```
1 Başla
2   sayac = 0, toplam = 0
3   sayac < 3 ?
4     Evet → x değerini oku
▶               toplam = toplam + x
6               sayac = sayac + 1
7               karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`sayac = 1` `toplam = 11` `x = 7`

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



Sayaç: **sayac = 1 + 1 = 2**. Geri dön.

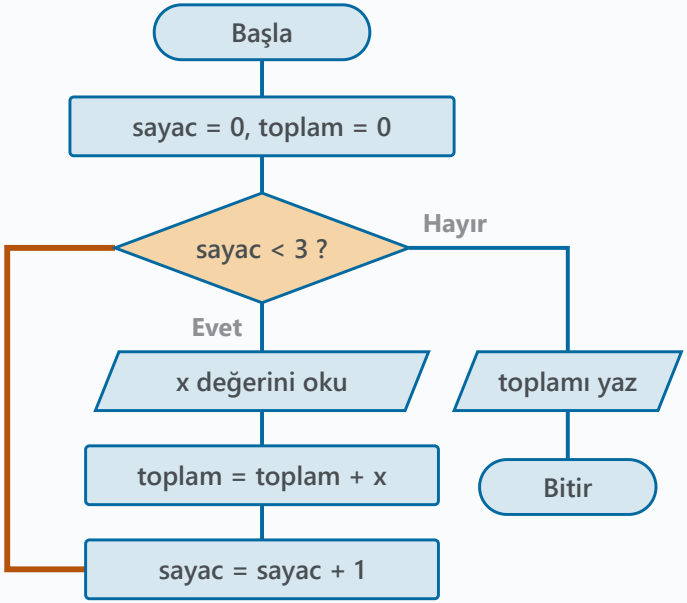
```
1  Başla
2    sayac = 0, toplam = 0
3    sayac < 3 ?
4      Evet → x değerini oku
5               toplam = toplam + x
6               sayac = sayac + 1
7               karara dön
8      Hayır → toplamı yaz
9  Bitir
```

DEĞİŞKENLER

`sayac = 2` `toplam = 11` `x = 7`

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



Karar: **2 < 3** → **doğru** → Evet.

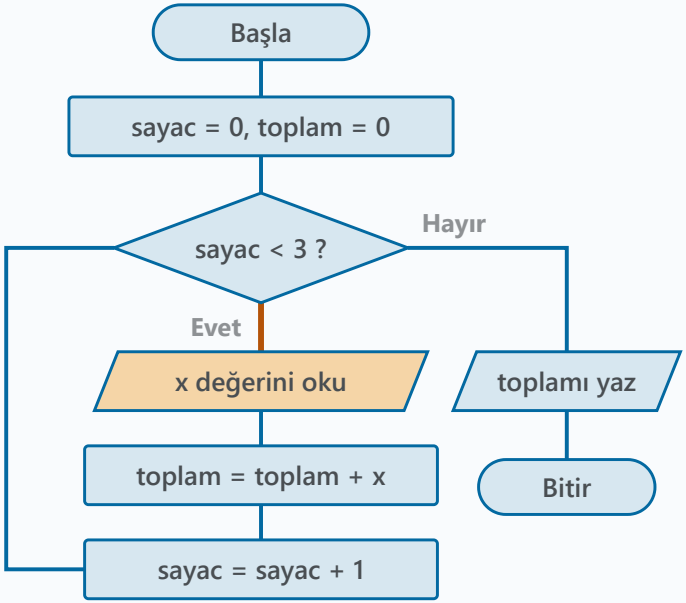
```
1 Başla
2   sayac = 0, toplam = 0
▶   sayac < 3 ?
4     Evet → x değerini oku
5               toplam = toplam + x
6               sayac = sayac + 1
7               karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`sayac = 2` `toplam = 11` `x = 7`

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



x = 2 okunur.

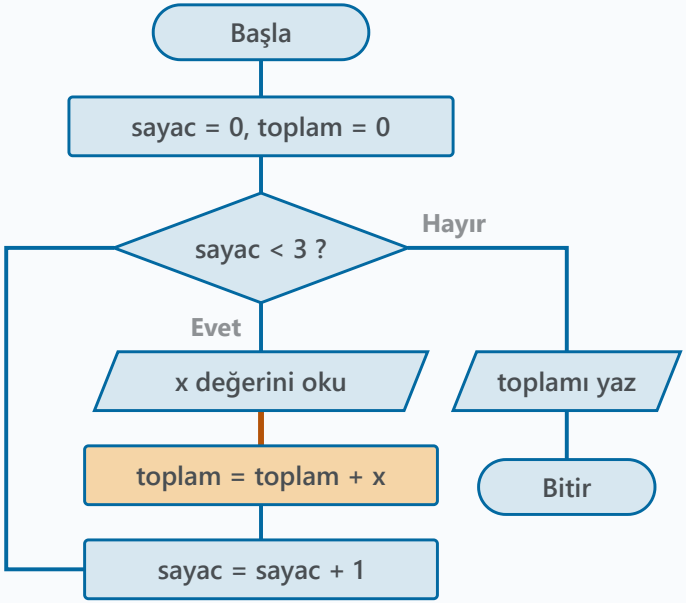
```
1 Başla
2   sayac = 0, toplam = 0
3   sayac < 3 ?
▶     Evet → x değerini oku
5             toplam = toplam + x
6             sayac = sayac + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`sayac = 2` `toplam = 11` `x = 2`

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



Biriktirici: **toplam = 11 + 2 = 13**.

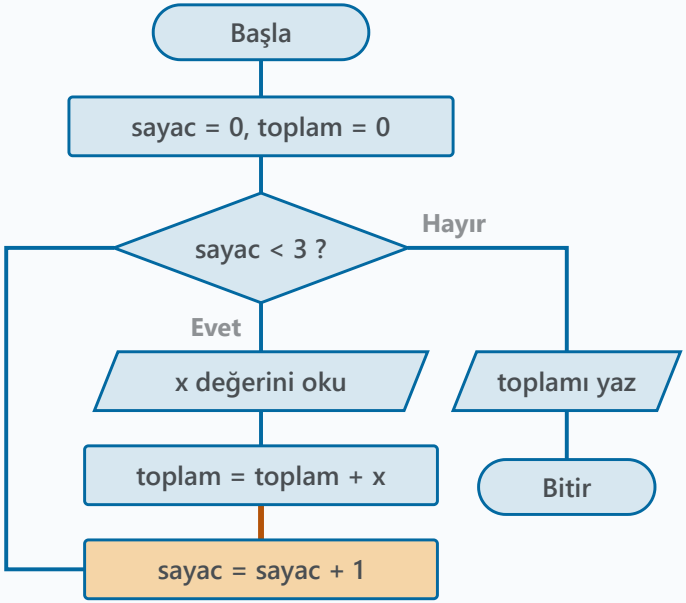
```
1 Başla
2   sayac = 0, toplam = 0
3   sayac < 3 ?
4     Evet → x değerini oku
▶               toplam = toplam + x
6               sayac = sayac + 1
7               karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`sayac = 2` `toplam = 13` `x = 2`

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



Sayaç: **sayac = 2 + 1 = 3**. Geri dön.

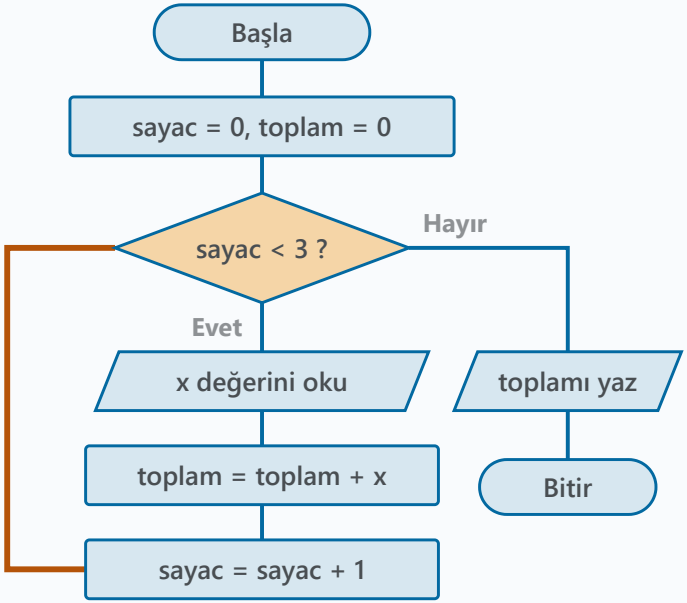
```
1  Başla
2    sayac = 0, toplam = 0
3    sayac < 3 ?
4      Evet → x değerini oku
5               toplam = toplam + x
6               sayac = sayac + 1
7               karara dön
8      Hayır → toplamı yaz
9  Bitir
```

DEĞİŞKENLER

`sayac = 3` `toplam = 13` `x = 2`

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



Karar: **3 < 3** → **yanlış** → Hayır, döngü biter.

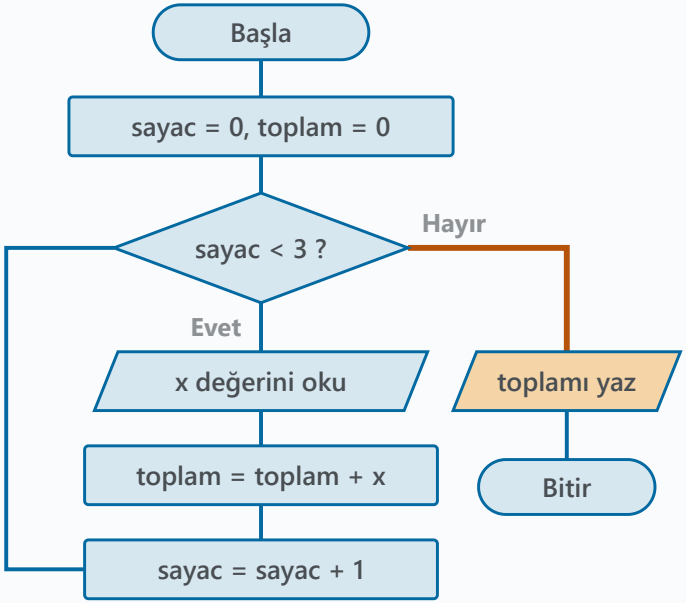
```
1 Başla
2   sayac = 0, toplam = 0
▶   sayac < 3 ?
4     Evet → x değerini oku
5               toplam = toplam + x
6               sayac = sayac + 1
7               karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`sayac = 3` `toplam = 13` `x = 2`

ÇIKTI

Sayaç ve Biriktirici Adım Adım (4, 7, 2)



Toplam yazılır: **13**.

```
1 Başla
2   sayac = 0, toplam = 0
3   sayac < 3 ?
4     Evet → x değerini oku
5               toplam = toplam + x
6               sayac = sayac + 1
7               karara dön
▶     Hayır → toplamı yaz
9 Bitir
```

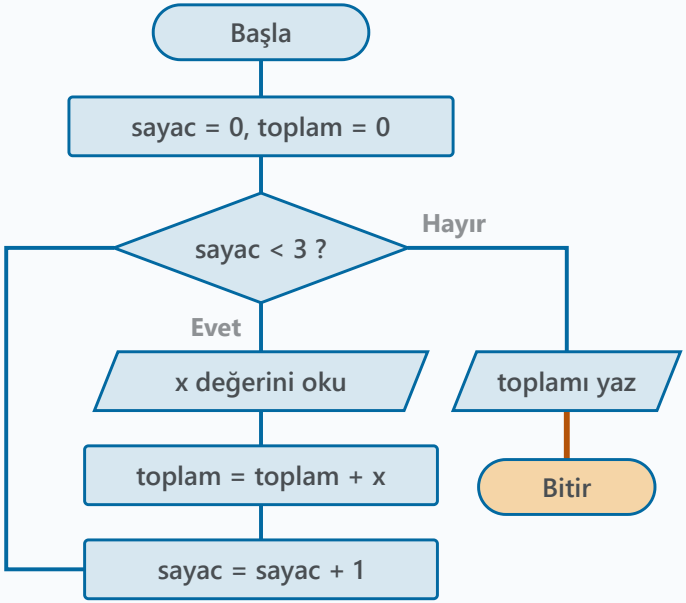
DEĞİŞKENLER

`sayac = 3` `toplam = 13` `x = 2`

ÇIKTI

```
13
```

Sayaç ve Biriktirici Adım Adım (4, 7, 2)




```
1  Başla
2    sayac = 0, toplam = 0
3    sayac < 3 ?
4      Evet → x değerini oku
5                toplam = toplam + x
6                sayac = sayac + 1
7                karara dön
8      Hayır → toplamı yaz
▶  Bitir
```

DEĞİŞKENLER

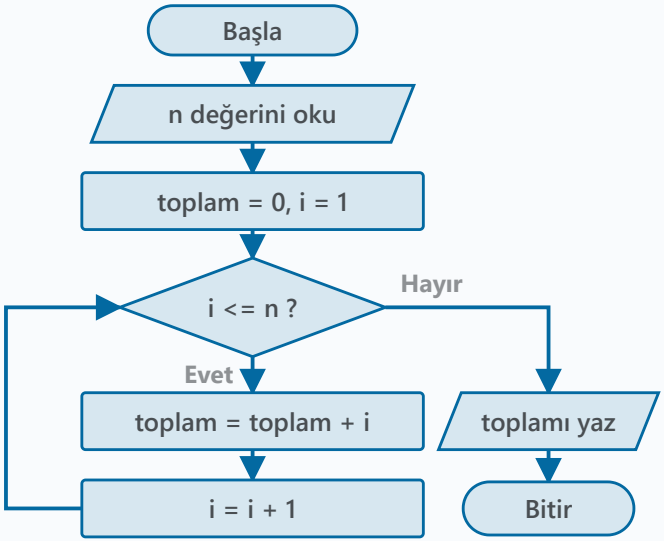
`sayac = 3` `toplam = 13` `x = 2`

ÇIKTI

```
13
```

Bitir. Sayaç döngünün **kaç kez** döndüğünü, biriktirici **sonucu** tuttu.

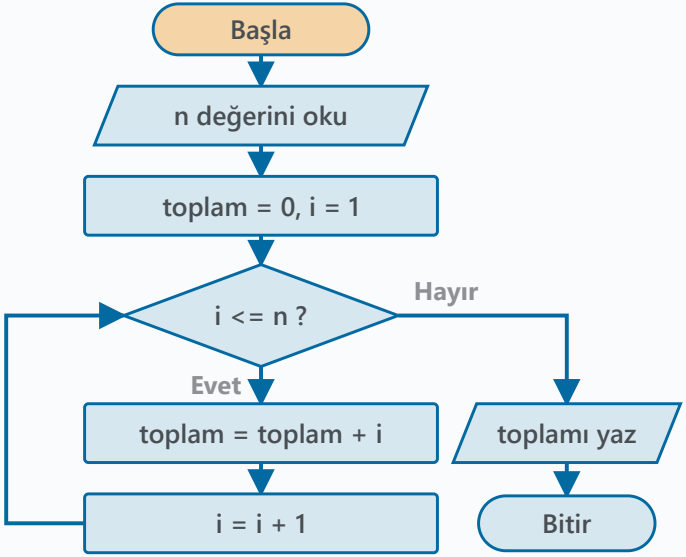
Örnek: 1'den n'ye Toplam




```
Başla
  n değerini oku
  toplam = 0, i = 1
  i <= n ?
    Evet → toplam = toplam + i
           i = i + 1
           karara dön
    Hayır → toplamı yaz
Bitir
```

- **i** sayaç, **toplam** biriktiricidir.
- n = 0 girilirse koşul ilk denemede yanlış olur, 0 yazılır.

1’den n’ye Toplam Adım Adım (n = 4)



Başla. Girdi: n = 4. Beklenen sonuç 1 + 2 + 3 + 4 = 10.

```
▶ Başla
2   n değerini oku
3   toplam = 0, i = 1
4   i <= n ?
5     Evet → toplam = toplam + i
6             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

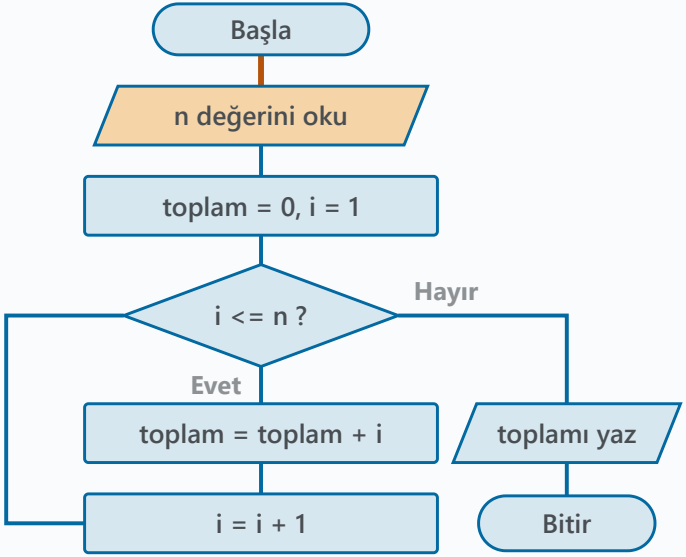
DEĞİŞKENLER

—

ÇIKTI

Adım 1 / 18

1’den n’ye Toplam Adım Adım (n = 4)



n = 4 okunur.

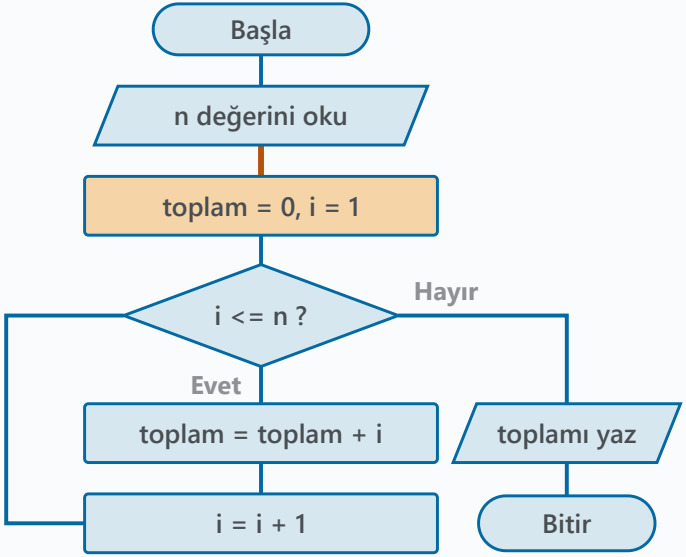
```
1 Başla
▶   n değerini oku
3   toplam = 0, i = 1
4   i <= n ?
5     Evet → toplam = toplam + i
6             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



Biriktirici **toplam = 0**, sayaç **i = 1**.

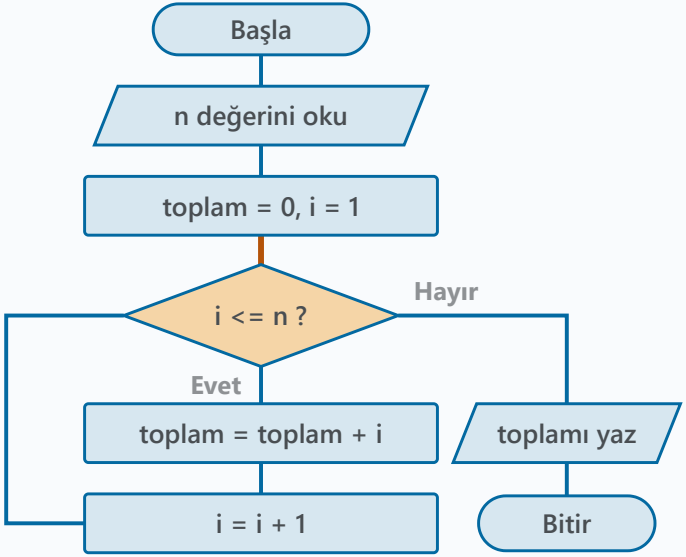
```
1 Başla
2   n değerini oku
▶   toplam = 0, i = 1
4   i <= n ?
5     Evet → toplam = toplam + i
6             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 0` `i = 1`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



Karar: **1 <= 4** → **doğru** → Evet.

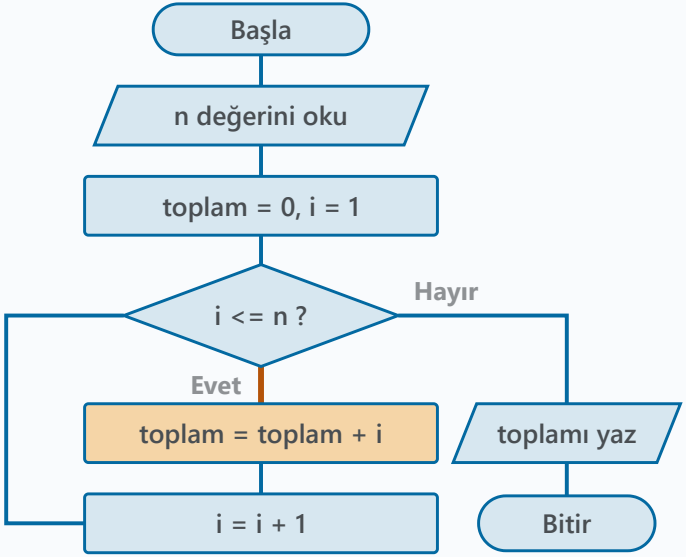
```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
▶   i <= n ?
5     Evet → toplam = toplam + i
6             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 0` `i = 1`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



toplam = 0 + 1 = 1.

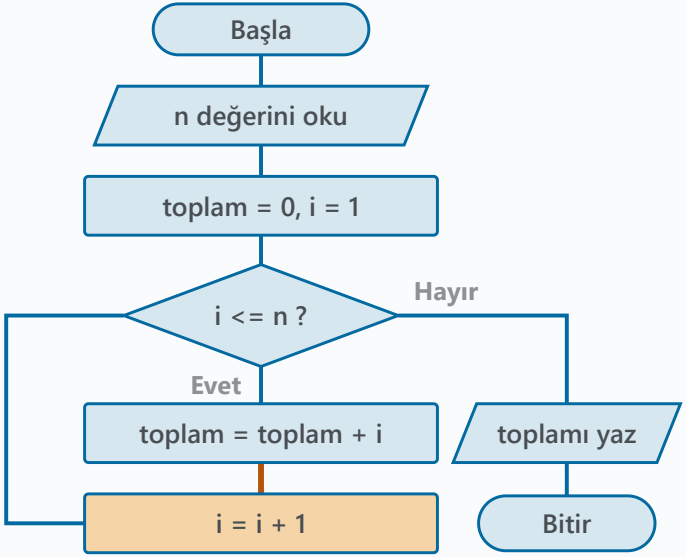
```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
4   i <= n ?
      Evet → toplam = toplam + i
6             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 1` `i = 1`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



i = 2; karara dön.

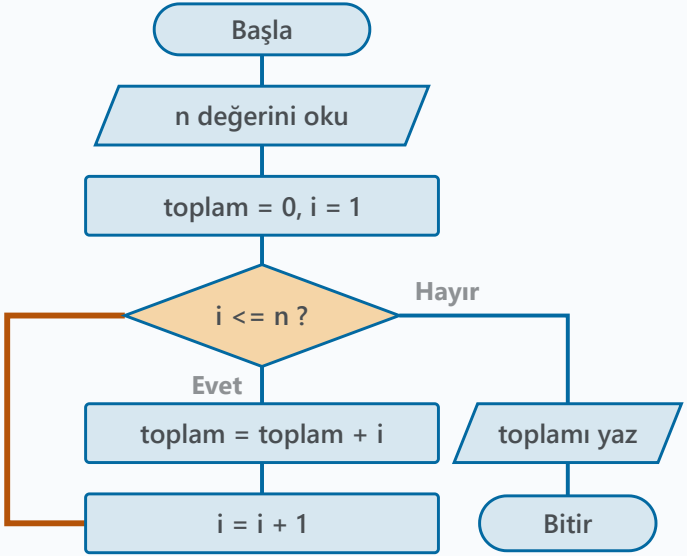
```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
4   i <= n ?
5     Evet → toplam = toplam + i
▶             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 1` `i = 2`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



Karar: **2 <= 4** → **doğru** → Evet.

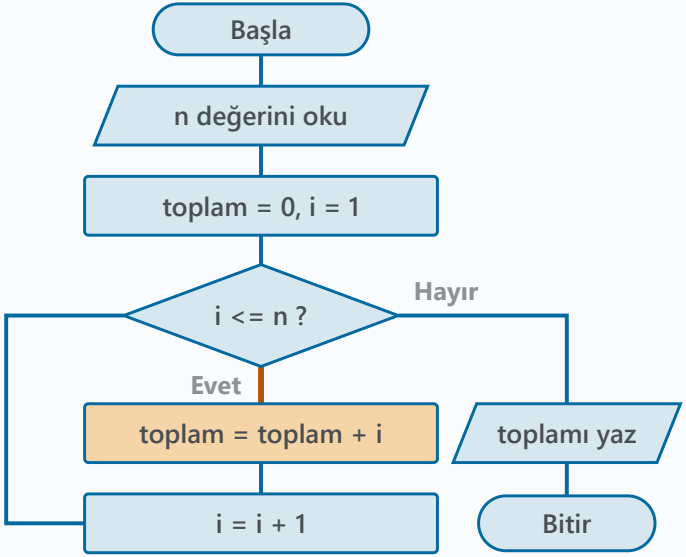
```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
▶   i <= n ?
5     Evet → toplam = toplam + i
6             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 1` `i = 2`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



toplam = 1 + 2 = 3.

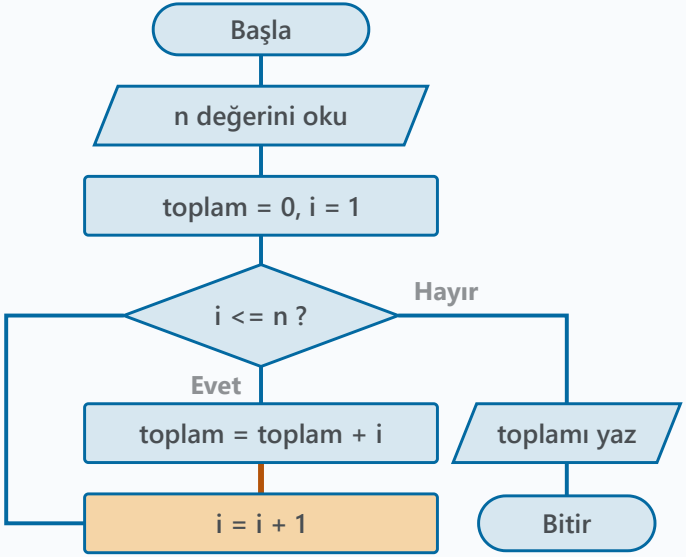
```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
4   i <= n ?
      Evet → toplam = toplam + i
6             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 3` `i = 2`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



i = 3; karara dön.

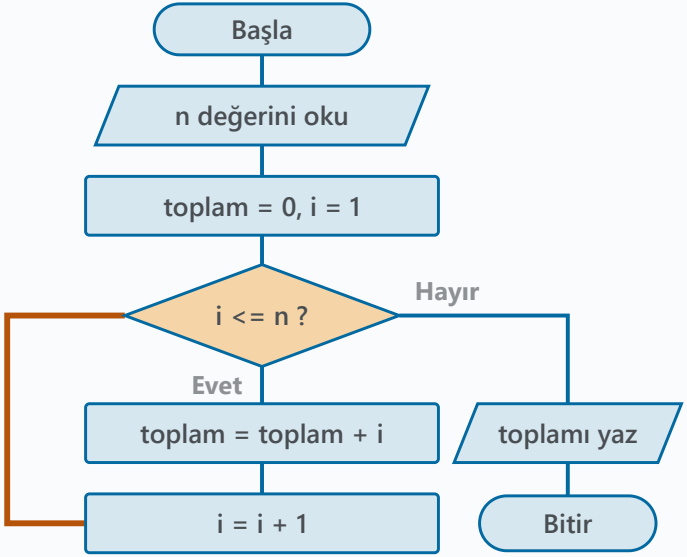
```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
4   i <= n ?
5     Evet → toplam = toplam + i
▶             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 3` `i = 3`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



Karar: **3 <= 4** → **doğru** → Evet.

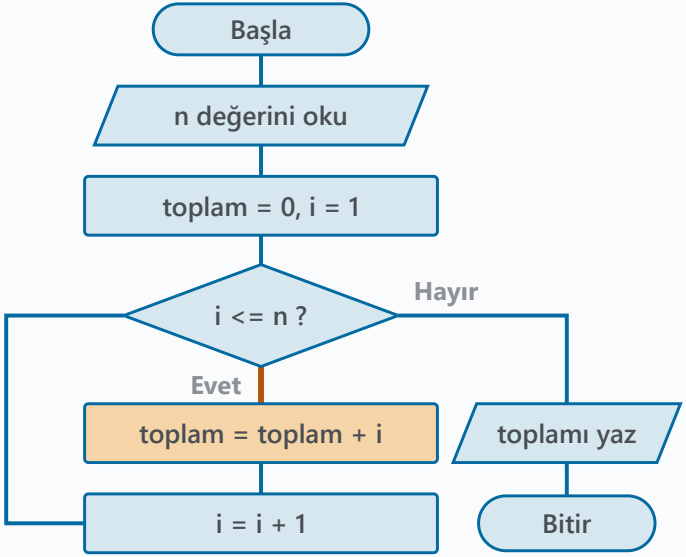
```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
▶   i <= n ?
5     Evet → toplam = toplam + i
6             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 3` `i = 3`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



toplam = 3 + 3 = 6.

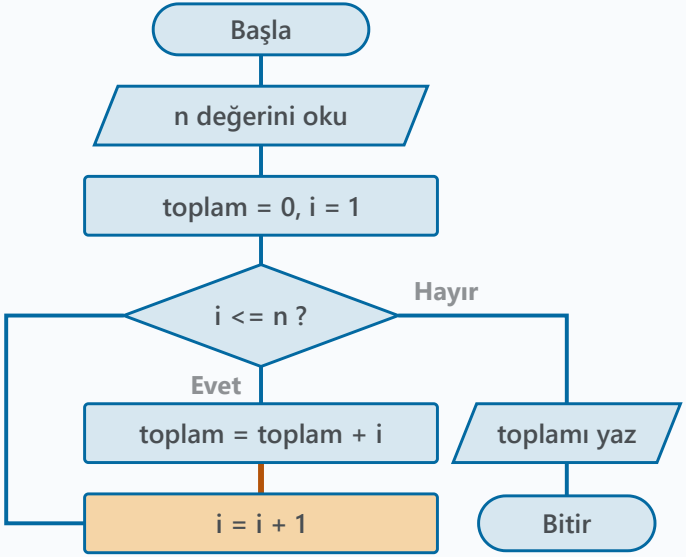
```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
4   i <= n ?
      Evet → toplam = toplam + i
6             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 6` `i = 3`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



i = 4; karara dön.

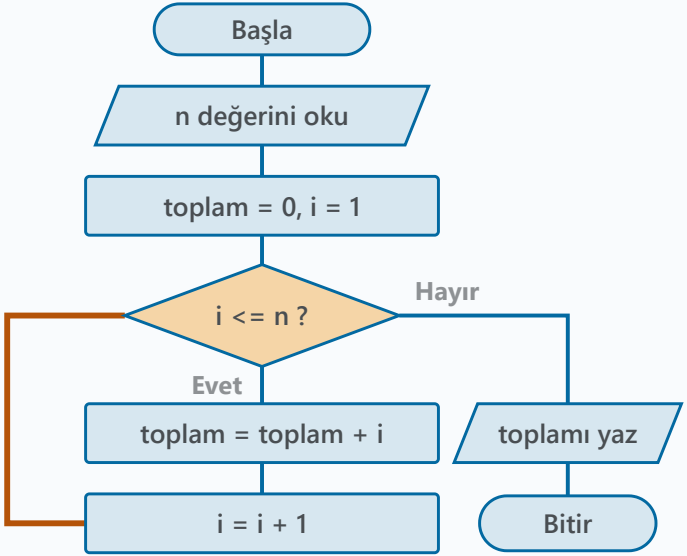
```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
4   i <= n ?
5     Evet → toplam = toplam + i
▶             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 6` `i = 4`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



Karar: **4 <= 4** → **doğru** → Evet.

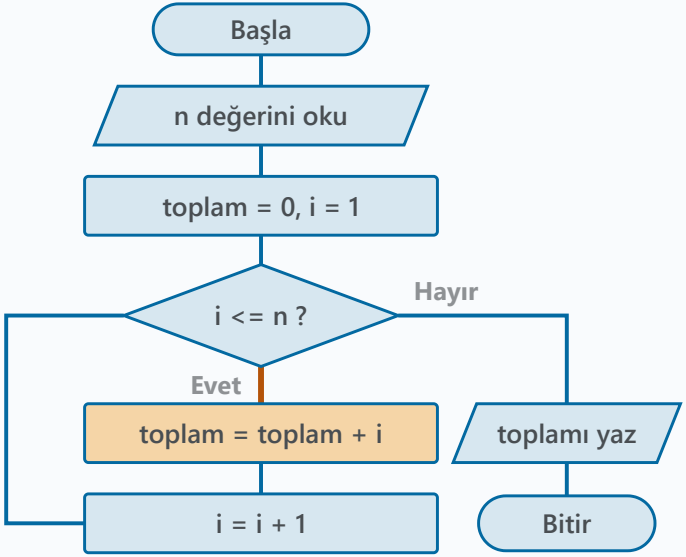
```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
▶   i <= n ?
5     Evet → toplam = toplam + i
6             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 6` `i = 4`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



toplam = 6 + 4 = 10.

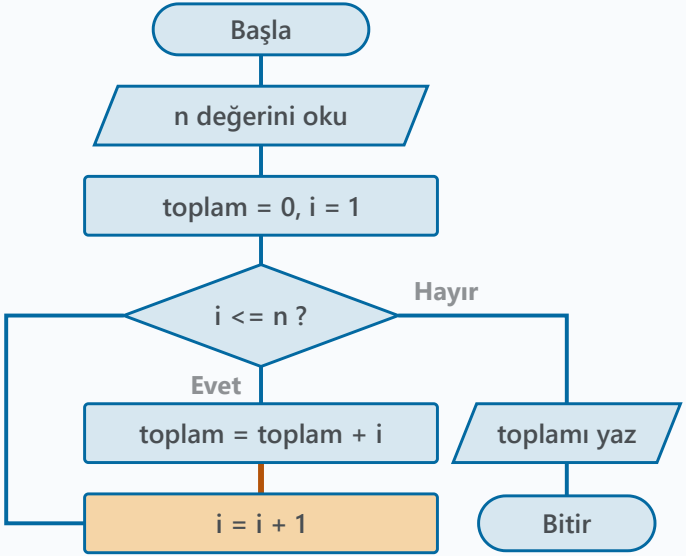
```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
4   i <= n ?
      Evet → toplam = toplam + i
6             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 10` `i = 4`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



i = 5; karara dön.

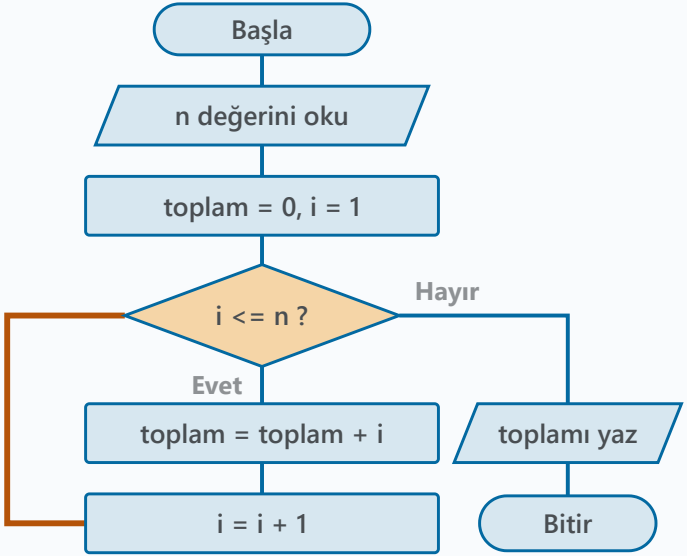
```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
4   i <= n ?
5     Evet → toplam = toplam + i
▶             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 10` `i = 5`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



Karar: **5 <= 4** → **yanlış** → Hayır, döngüden çık.

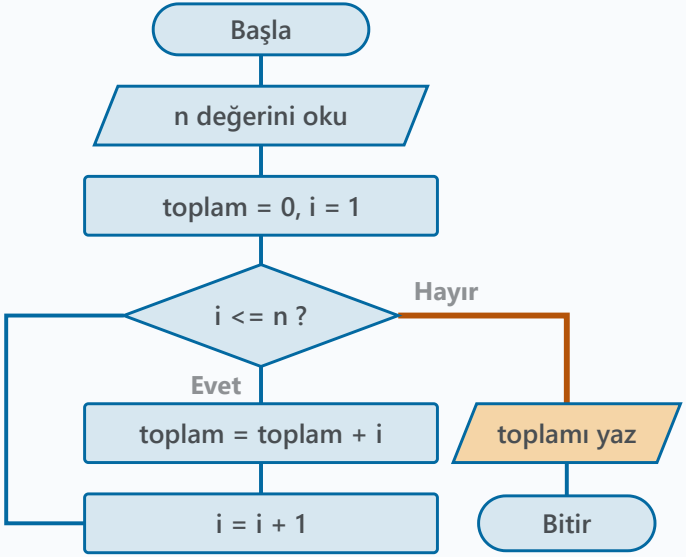
```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
▶   i <= n ?
5     Evet → toplam = toplam + i
6             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
9 Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 10` `i = 5`

ÇIKTI

1’den n’ye Toplam Adım Adım (n = 4)



10 yazılır.

```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
4   i <= n ?
5     Evet → toplam = toplam + i
6             i = i + 1
7             karara dön
▶     Hayır → toplamı yaz
9 Bitir
```

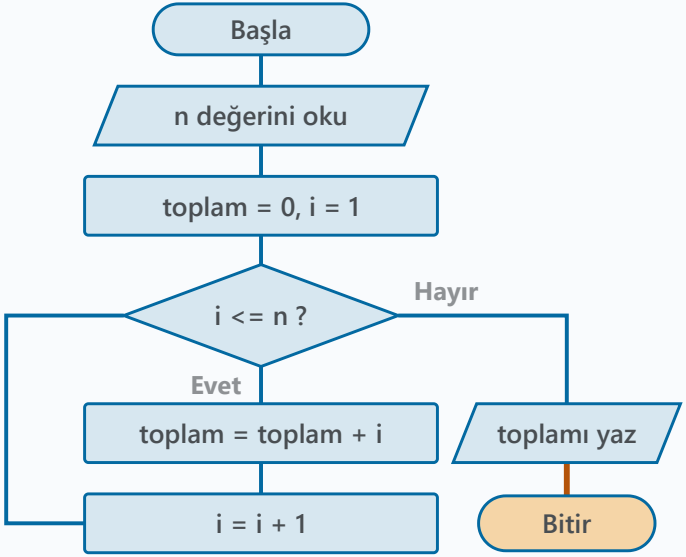
DEĞİŞKENLER

`n = 4` `toplam = 10` `i = 5`

ÇIKTI

```
10
```

1’den n’ye Toplam Adım Adım (n = 4)



Bitir. Gövde n = 4 kez çalıştı; sonuç formülle de doğrulanır: n(n+1)/2 = 10.

Adım 18 / 18

```
1 Başla
2   n değerini oku
3   toplam = 0, i = 1
4   i <= n ?
5     Evet → toplam = toplam + i
6             i = i + 1
7             karara dön
8     Hayır → toplamı yaz
▶ Bitir
```

DEĞİŞKENLER

`n = 4` `toplam = 10` `i = 5`

ÇIKTI

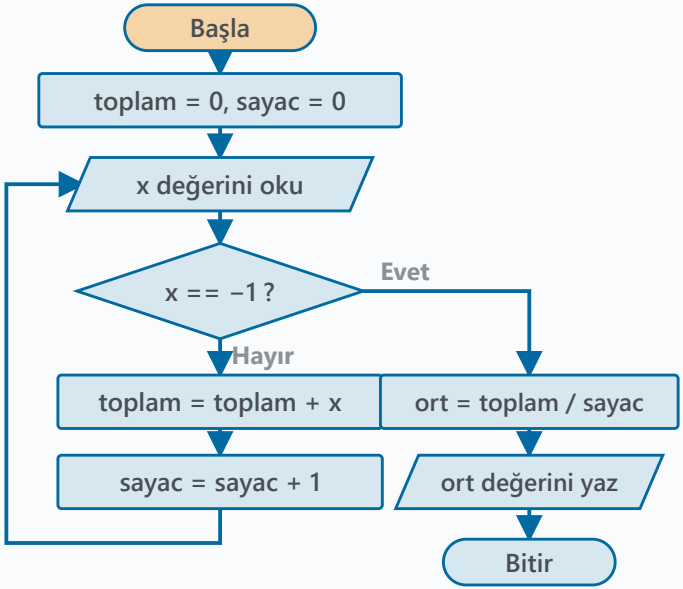
```
10
```

Sayaçlı ve Bekçi Değerli Döngü

	Sayaçla denetlenen	Bekçi değerle denetlenen
Tekrar sayısı	Önceden bilinir (n kez)	Bilinmez; özel bir değer gelince durur
Karar	i <= n ?	x == −1 ? (bekçi geldi mi?)
Örnek	Sınıftaki 30 öğrencinin notunu oku	Kasada okutulan ürünler; "bitti" tuşuna kadar
Java'da çoğunlukla	`for`	`while`

- **Bekçi değer**, gerçek veride görülemeyecek bir değer seçilir: notlar 0–100 arasındaysa −1 uygundur, 0 uygun değildir.
- Bekçi değer veriye katılmamalıdır: karar, toplama ve sayma işlemlerinden **önce** sorulur; geri dönen ok yeniden **oku** kutusuna gider.
- İlk girdi bekçi değerse döngü gövdesi hiç çalışmaz ve sayaç 0 kalır; ortalama alınacaksa bölmeden önce **sayac > 0 ?** diye sorulmalıdır.

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)



Başla. Kullanıcı notları girecek, bitince **−1** yazacak. Girdiler: 80, 64, 90, −1.

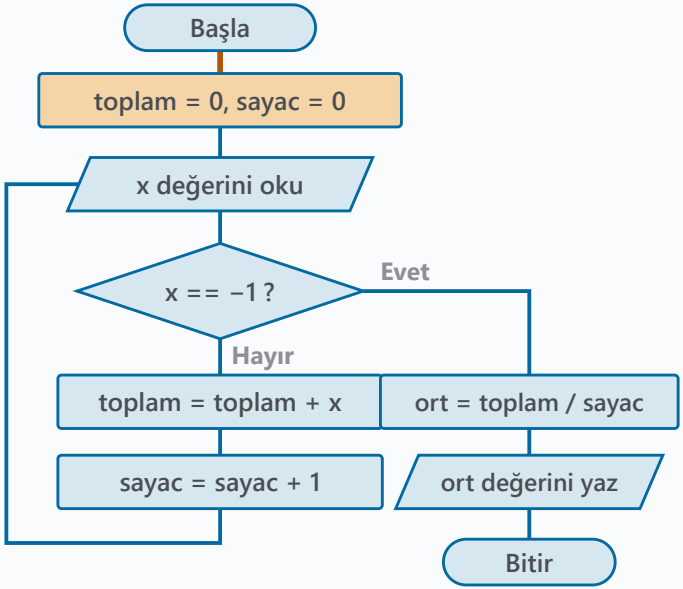
```
▶ Başla
2   toplam = 0, sayac = 0
3   x değerini oku
4   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

DEĞİŞKENLER

—

ÇIKTI

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)



Biriktirici ve sayaç 0 ile başlar.

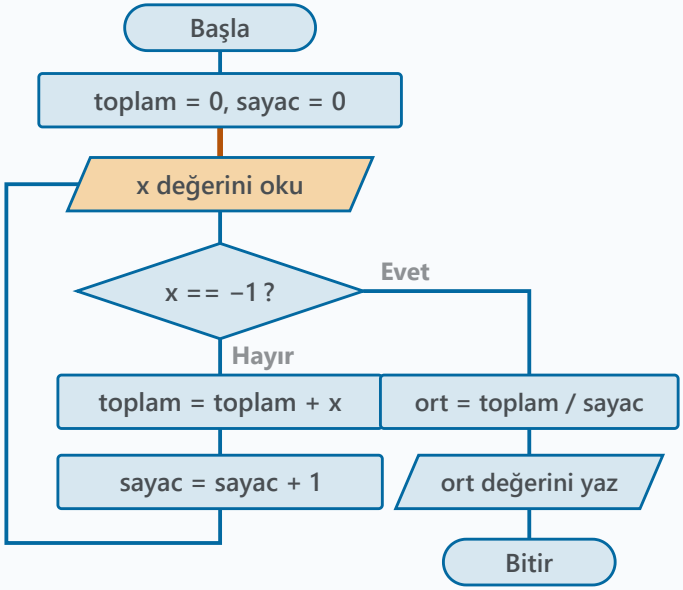
```
1 Başla
▶   toplam = 0, sayac = 0
3   x değerini oku
4   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

DEĞİŞKENLER

`toplam = 0` `sayac = 0`

ÇIKTI

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)



x = **80** okunur.

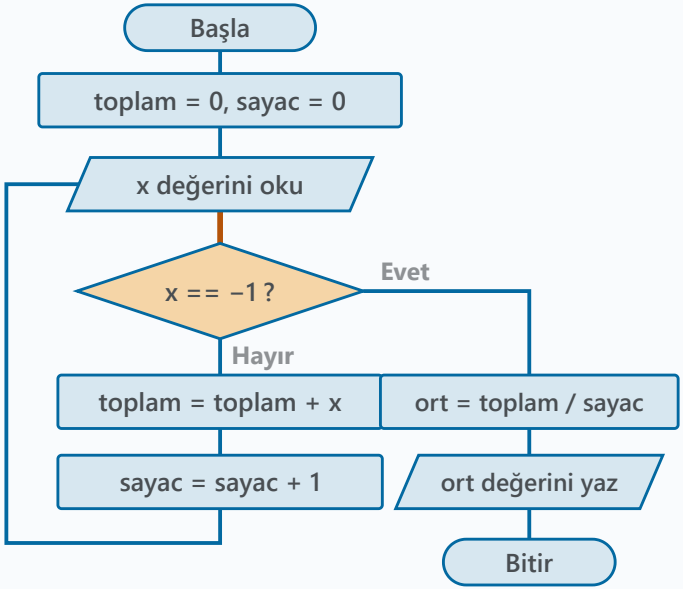
```
1 Başla
2   toplam = 0, sayac = 0
▶   x değerini oku
4   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

DEĞİŞKENLER

`toplam = 0` `sayac = 0` `x = 80`

ÇIKTI

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)



80 == −1 → **yanlış** → Hayır: bu gerçek bir not.

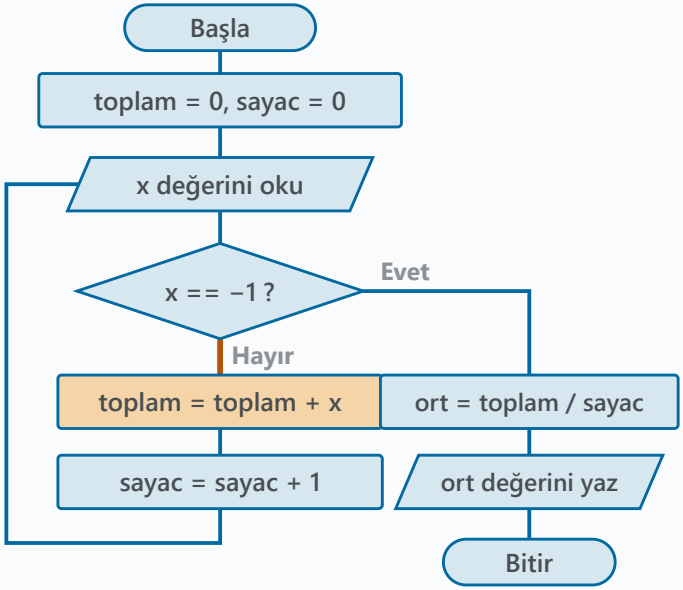
```
1 Başla
2   toplam = 0, sayac = 0
3   x değerini oku
▶   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

DEĞİŞKENLER

`toplam = 0` `sayac = 0` `x = 80`

ÇIKTI

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)




```
1 Başla
2   toplam = 0, sayac = 0
3   x değerini oku
4   x == -1 ?  Evet → 7. satıra git
▶   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

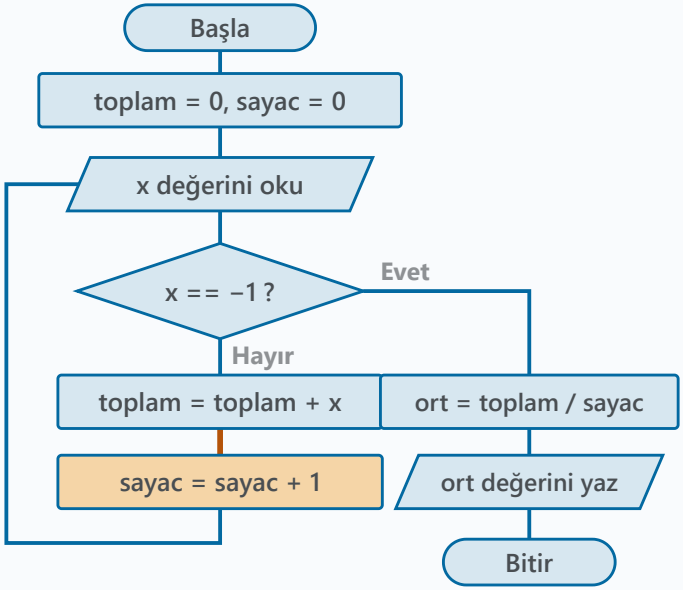
DEĞİŞKENLER

`toplam = 80` `sayac = 0` `x = 80`

ÇIKTI

toplam = 0 + 80 = **80**.

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)



sayac = **1**.

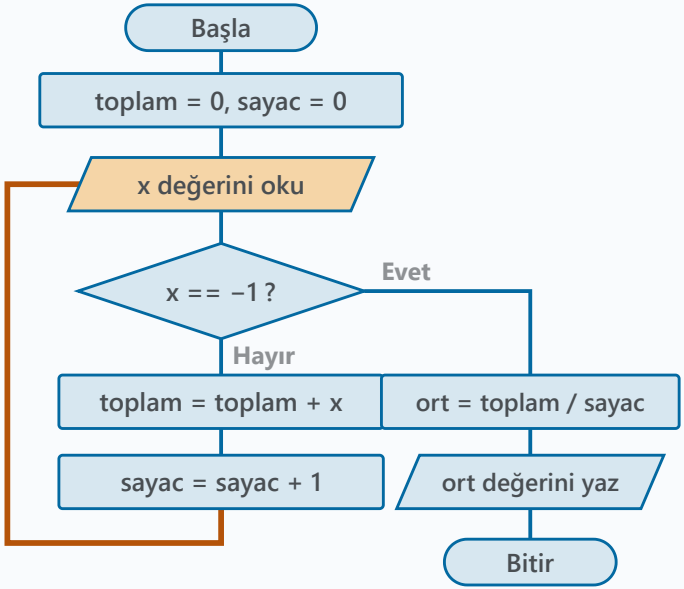
```
1 Başla
2   toplam = 0, sayac = 0
3   x değerini oku
4   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
▶   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

DEĞİŞKENLER

`toplam = 80` `sayac = 1` `x = 80`

ÇIKTI

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)



x = **64** okunur. Geri dönen ok yeniden **oku** kutusuna geldi.

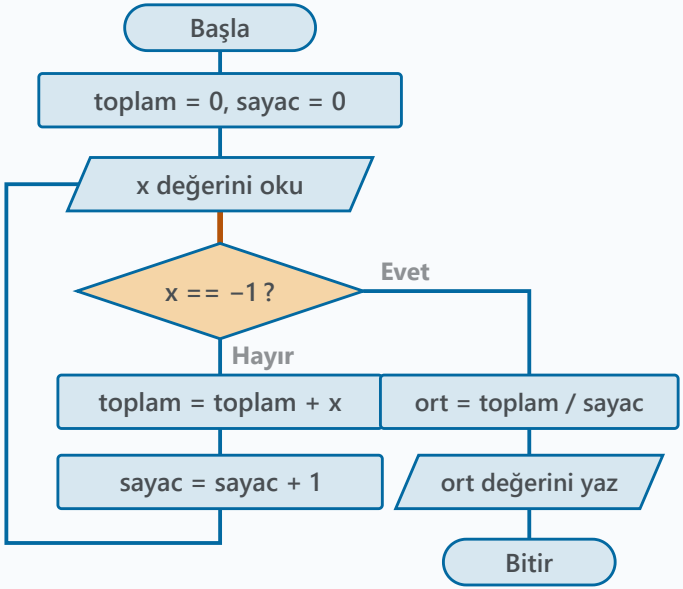
```
1 Başla
2   toplam = 0, sayac = 0
    x değerini oku
4   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

DEĞİŞKENLER

`toplam = 80` `sayac = 1` `x = 64`

ÇIKTI

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)



64 == −1 → **yanlış** → Hayır: bu gerçek bir not.

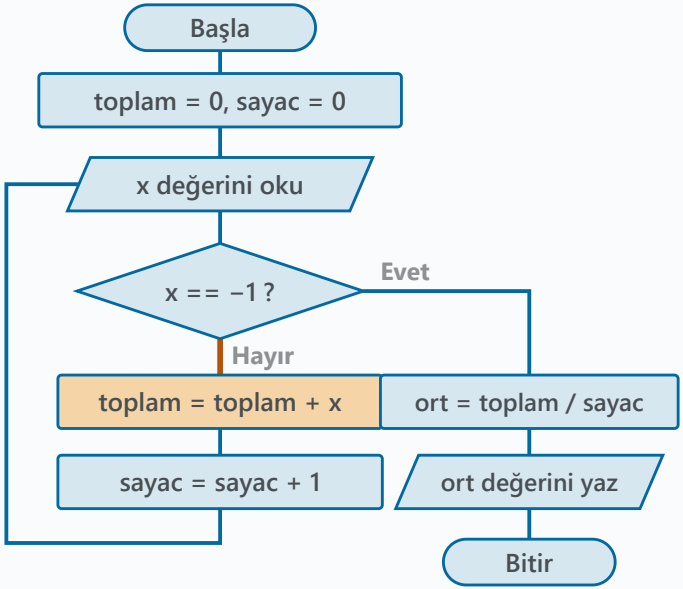
```
1 Başla
2   toplam = 0, sayac = 0
3   x değerini oku
▶   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

DEĞİŞKENLER

`toplam = 80` `sayac = 1` `x = 64`

ÇIKTI

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)




```
1 Başla
2   toplam = 0, sayac = 0
3   x değerini oku
4   x == -1 ?  Evet → 7. satıra git
▶   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

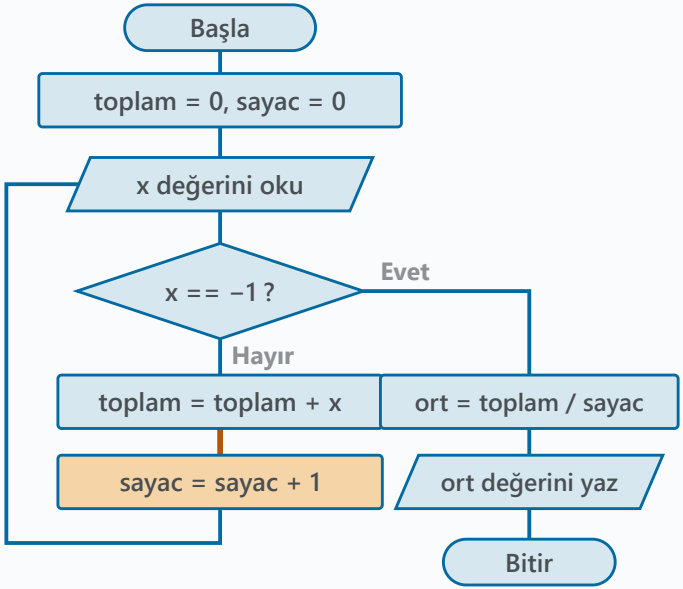
DEĞİŞKENLER

`toplam = 144` `sayac = 1` `x = 64`

ÇIKTI

toplam = 80 + 64 = **144**.

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)




```
1 Başla
2   toplam = 0, sayac = 0
3   x değerini oku
4   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
▶   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

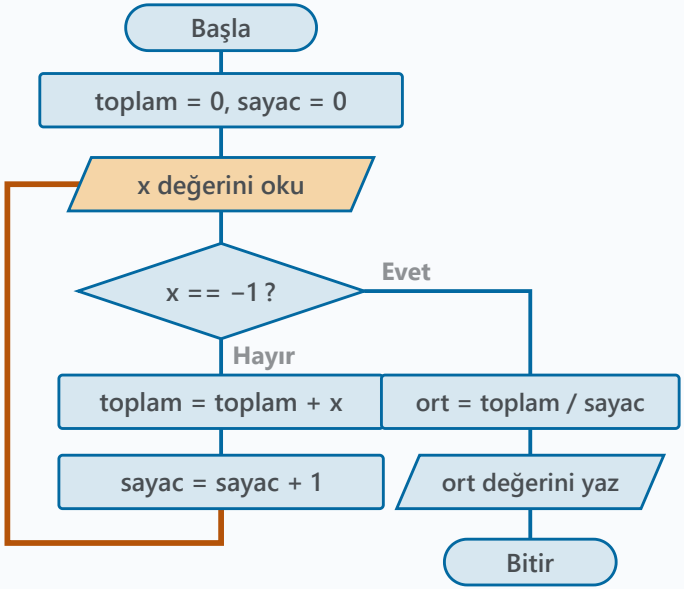
sayac = **2**.

DEĞİŞKENLER

`toplam = 144` `sayac = 2` `x = 64`

ÇIKTI

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)




```
1 Başla
2   toplam = 0, sayac = 0
    x değerini oku
4   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

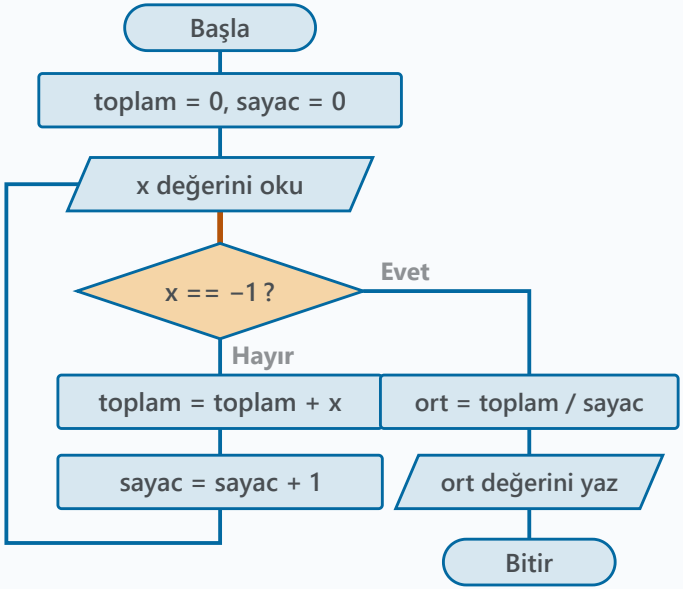
DEĞİŞKENLER

`toplam = 144` `sayac = 2` `x = 90`

ÇIKTI

x = **90** okunur. Geri dönen ok yeniden **oku** kutusuna geldi.

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)




```
1 Başla
2   toplam = 0, sayac = 0
3   x değerini oku
▶   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

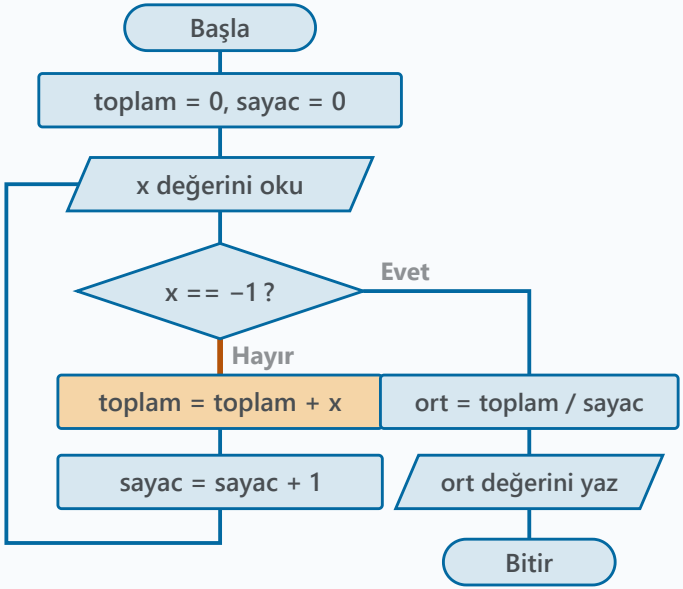
DEĞİŞKENLER

`toplam = 144` `sayac = 2` `x = 90`

ÇIKTI

90 == −1 → **yanlış** → Hayır: bu gerçek bir not.

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)



toplam = 144 + 90 = **234**.

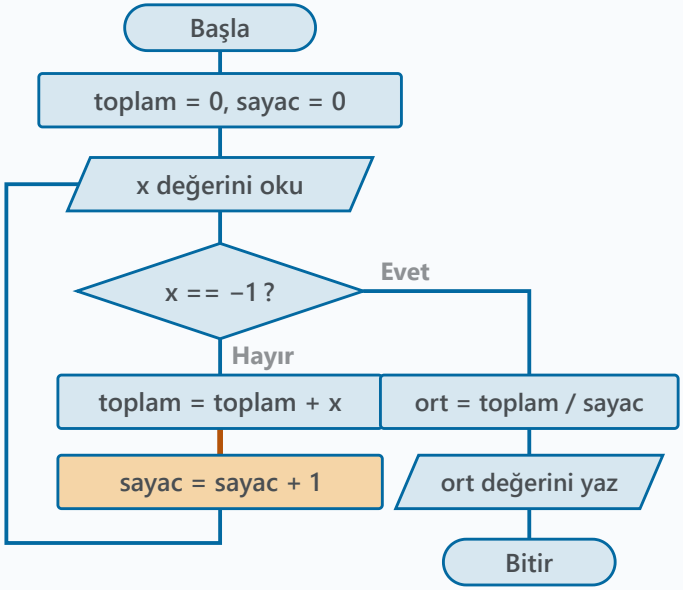
```
1 Başla
2   toplam = 0, sayac = 0
3   x değerini oku
4   x == -1 ?  Evet → 7. satıra git
▶   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

DEĞİŞKENLER

`toplam = 234` `sayac = 2` `x = 90`

ÇIKTI

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)



sayac = **3**.

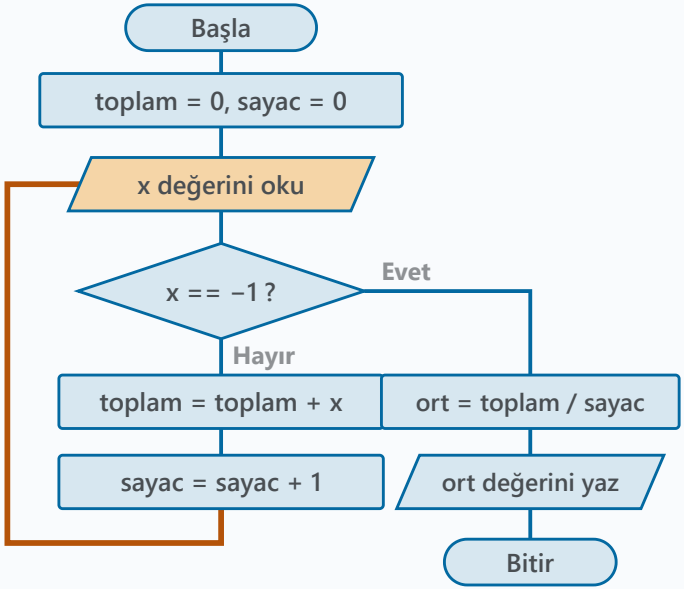
```
1 Başla
2   toplam = 0, sayac = 0
3   x değerini oku
4   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
▶   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

DEĞİŞKENLER

`toplam = 234` `sayac = 3` `x = 90`

ÇIKTI

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)



x = **−1** okunur.

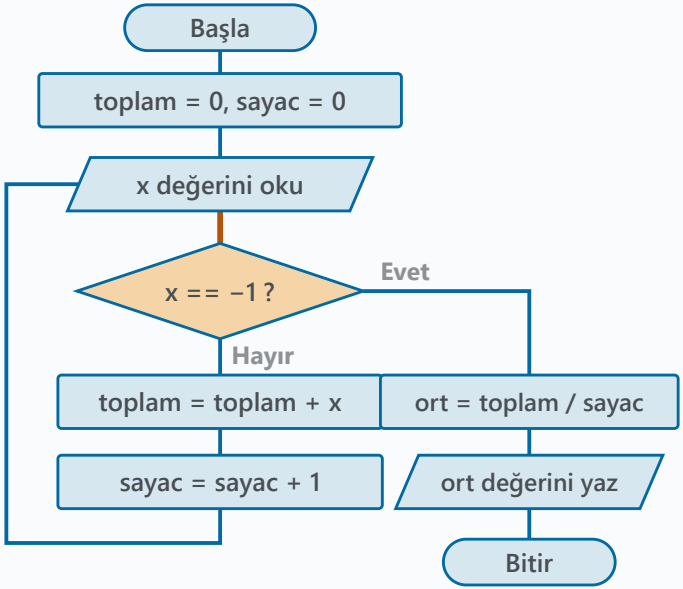
```
1 Başla
2   toplam = 0, sayac = 0
    x değerini oku
4   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

DEĞİŞKENLER

`toplam = 234` `sayac = 3` `x = −1`

ÇIKTI

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)




```
1 Başla
2   toplam = 0, sayac = 0
3   x değerini oku
▶   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

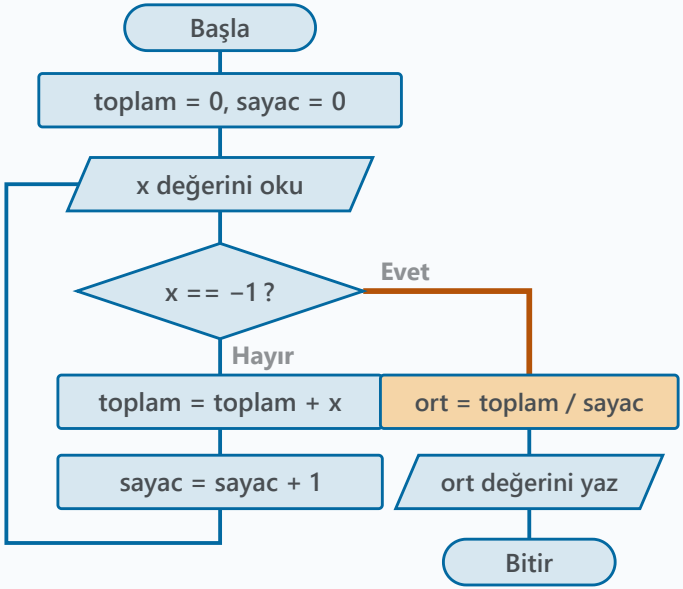
DEĞİŞKENLER

`toplam = 234` `sayac = 3` `x = −1`

ÇIKTI

−1 == −1 → **doğru** → Evet: bekçi geldi, döngü biter.
−1 toplama da sayaca da **katılmadı**.

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)



ort = 234 / 3 = **78**.

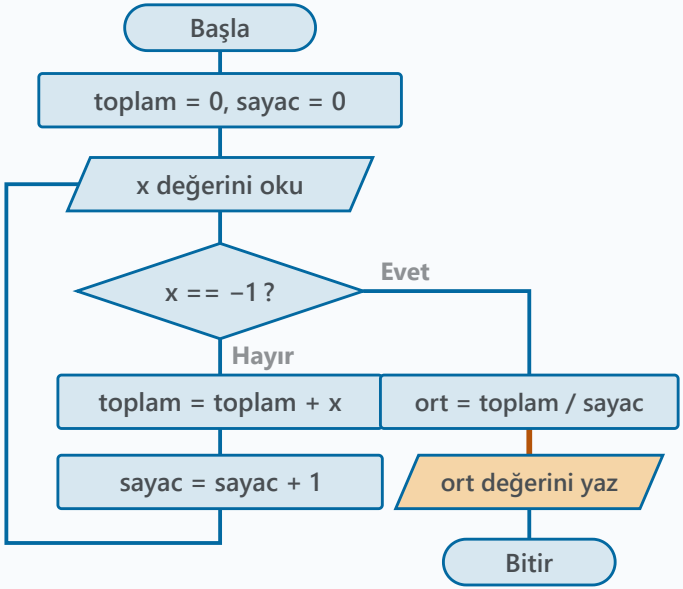
```
1 Başla
2   toplam = 0, sayac = 0
3   x değerini oku
4   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
▶   ort = toplam / sayac
8   ort değerini yaz
9 Bitir
```

DEĞİŞKENLER

`toplam = 234` `sayac = 3` `x = −1` `ort = 78`

ÇIKTI

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)



Ortalama yazılır.

```
1 Başla
2   toplam = 0, sayac = 0
3   x değerini oku
4   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
▶   ort değerini yaz
9 Bitir
```

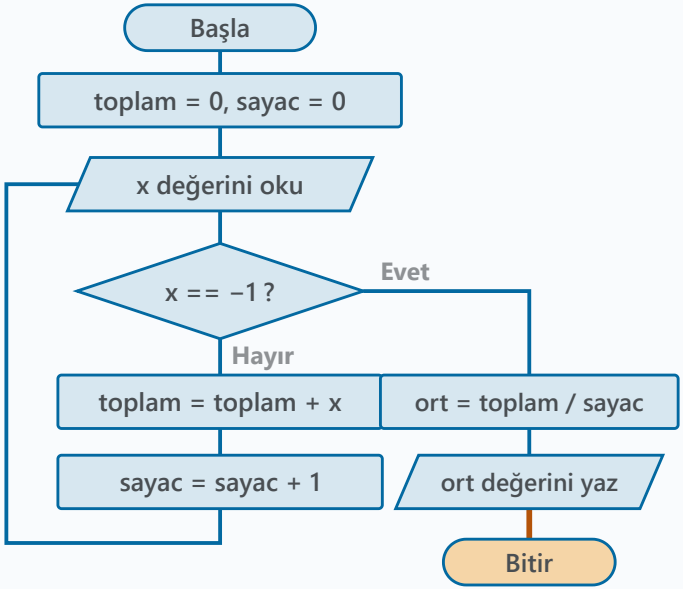
DEĞİŞKENLER

`toplam = 234` `sayac = 3` `x = −1` `ort = 78`

ÇIKTI

```
78
```

Bekçi Değerli Döngü Adım Adım (80, 64, 90, −1)




```
1 Başla
2   toplam = 0, sayac = 0
3   x değerini oku
4   x == -1 ?  Evet → 7. satıra git
5   toplam = toplam + x
6   sayac = sayac + 1;  3. satıra dön
7   ort = toplam / sayac
8   ort değerini yaz
▶ Bitir
```

DEĞİŞKENLER

`toplam = 234` `sayac = 3` `x = −1` `ort = 78`

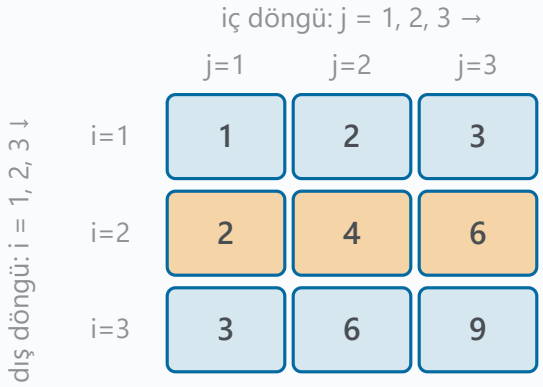
ÇIKTI

```
78
```

Bitir. Gövde 3 kez, karar 4 kez çalıştı. Kaç not girileceği önceden bilinmiyordu; döngüyü **veri** durdurdu.

İç İçe Yapılar

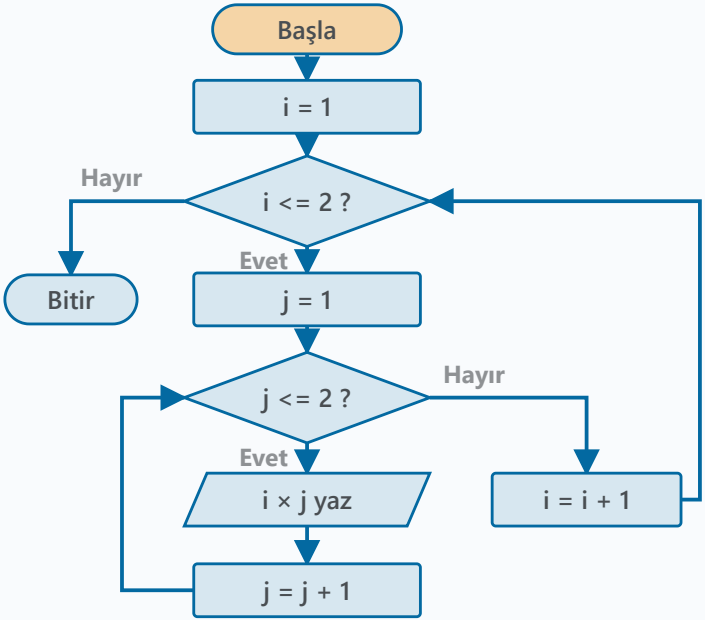
- Bir kararın dalında başka karar bulunabilir.
- Bir döngünün içinde ikinci döngü yer alabilir: dış döngünün **her turunda** iç döngü baştan sona çalışır.
- Her yapının giriş ve çıkış noktaları görsel olarak ayırt edilmelidir.



3 × 3 = 9 hücre: iç döngü gövdesi 9 kez çalışır. Turuncu satır: i = 2 iken j'nin üç turu.

Çok sayıda iç içe yapı, algoritmayı alt süreçlere ayırma gereksinimini gösterebilir.

İç İçe Döngü Adım Adım (i = 1)



Başla. Dış döngü i = 1, 2; iç döngü j = 1, 2. Her satıra i × j yazılacak.

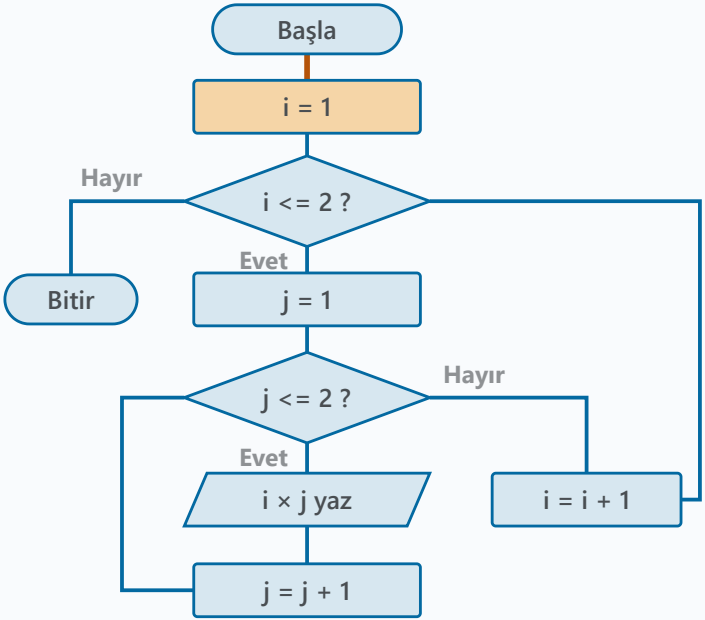
```
▶ Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

DEĞİŞKENLER

—

ÇIKTI

İç İçe Döngü Adım Adım (i = 1)



i = 1: dış sayaç başlar.

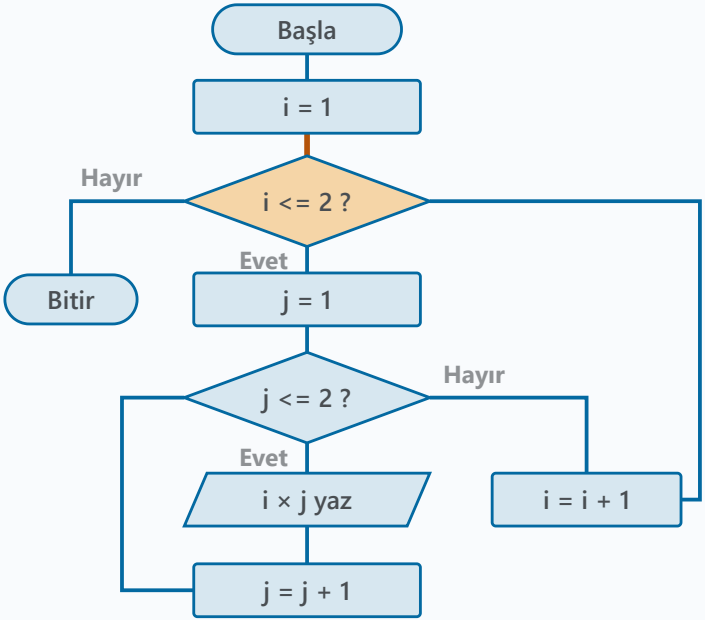
```
1 Başla
▶   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

DEĞİŞKENLER

`i = 1`

ÇIKTI

İç İçe Döngü Adım Adım (i = 1)



Dış karar: **1 <= 2** → **doğru** → iç döngüye gir.

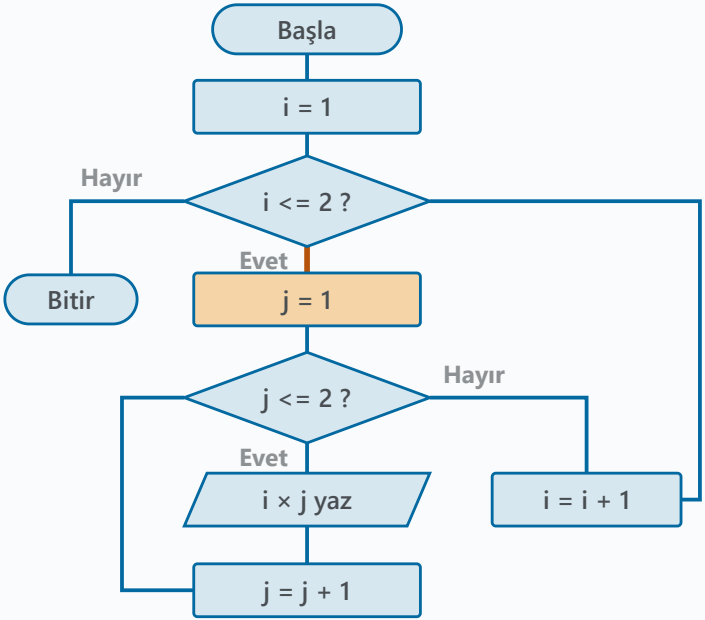
```
1 Başla
2   i = 1
▶   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

DEĞİŞKENLER

`i = 1`

ÇIKTI

İç İçe Döngü Adım Adım (i = 1)



j = 1: iç sayaç her dış turda **yeniden** 1 olur.

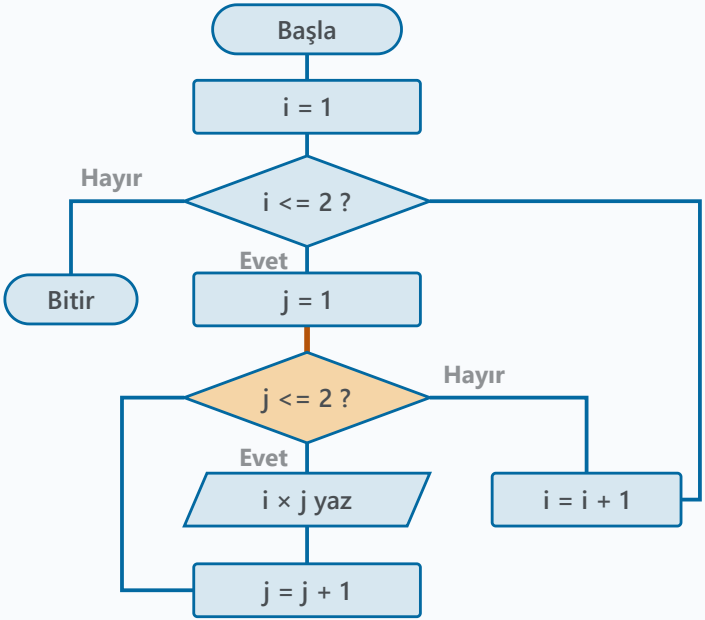
```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
▶     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

DEĞİŞKENLER

`i = 1` `j = 1`

ÇIKTI

İç İçe Döngü Adım Adım (i = 1)



İç karar: **1 <= 2 → doğru**

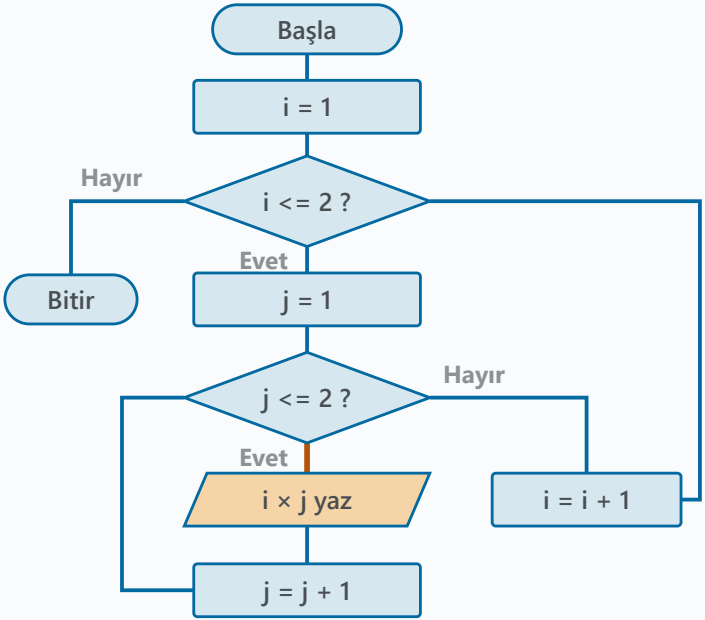
```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
▶     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

DEĞİŞKENLER

`i = 1` `j = 1`

ÇIKTI

İç İçe Döngü Adım Adım (i = 1)



1 × 1 = 1 yazılır.

```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
▶       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

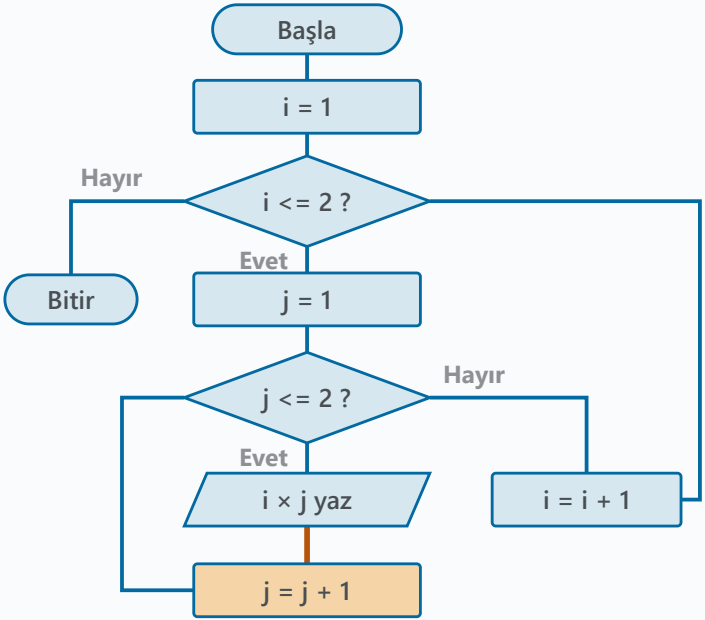
DEĞİŞKENLER

`i = 1` `j = 1`

ÇIKTI

```
1
```

İç İçe Döngü Adım Adım (i = 1)



j = 2; iç karara dön.

```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
▶       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

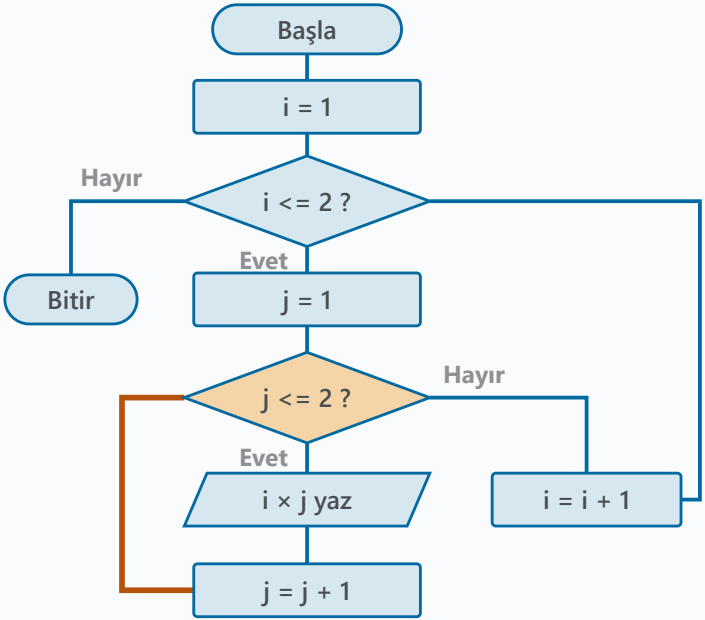
DEĞİŞKENLER

`i = 1` `j = 2`

ÇIKTI

```
1
```

İç İçe Döngü Adım Adım (i = 1)



İç karar: **2 <= 2 → doğru**

```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
▶     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

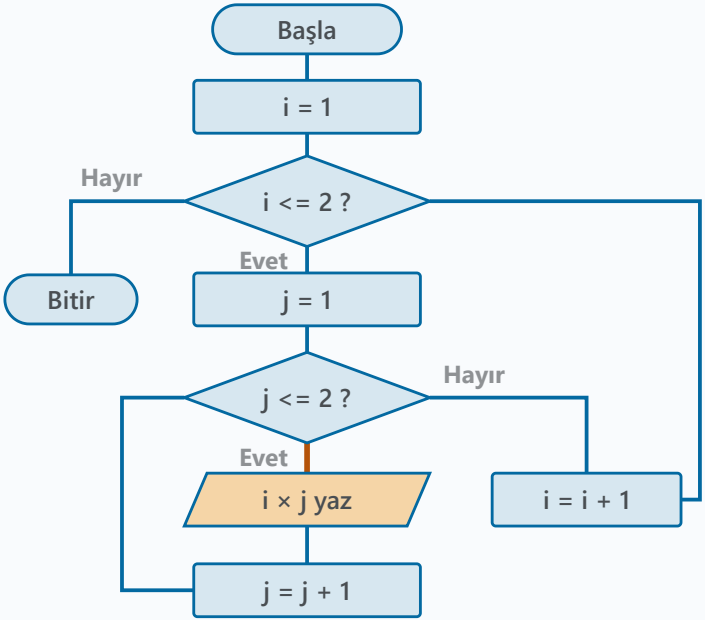
DEĞİŞKENLER

`i = 1` `j = 2`

ÇIKTI

```
1
```

İç İçe Döngü Adım Adım (i = 1)



1 × 2 = 2 yazılır.

```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
▶       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

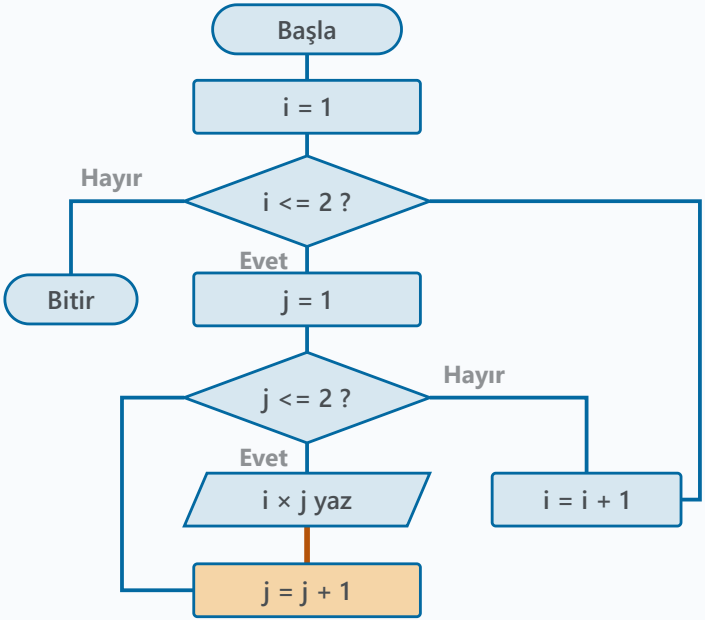
DEĞİŞKENLER

`i = 1` `j = 2`

ÇIKTI

```
1 2
```

İç İçe Döngü Adım Adım (i = 1)



j = 3; iç karara dön.

```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
▶       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

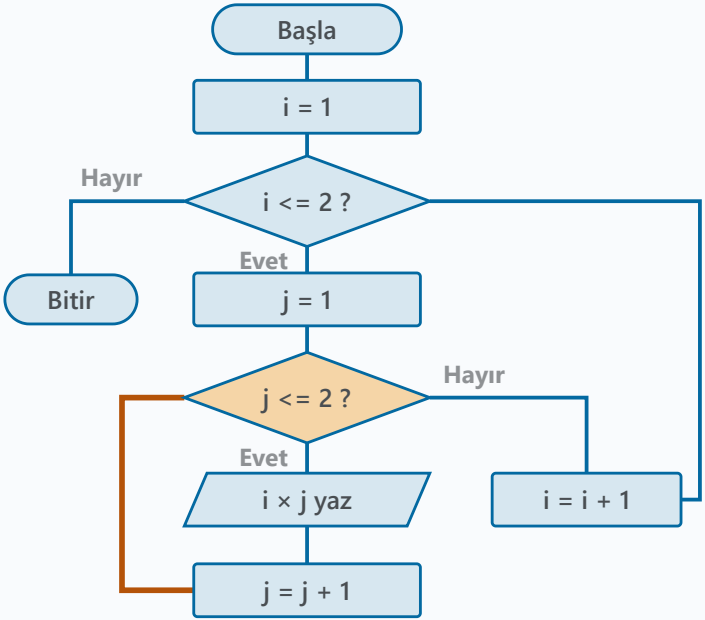
DEĞİŞKENLER

`i = 1` `j = 3`

ÇIKTI

```
1 2
```

İç İçe Döngü Adım Adım (i = 1)



İç karar: **3 <= 2** → **yanlış** → Hayır, iç döngü biter.

```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
▶     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

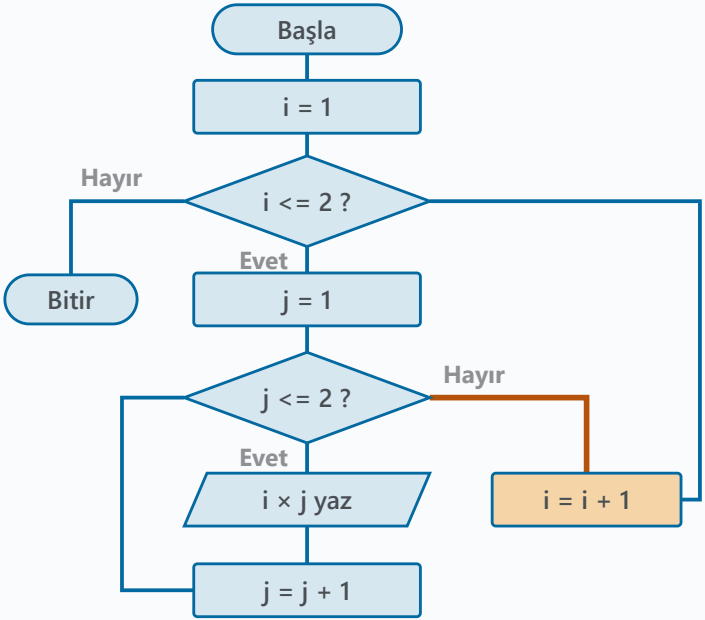
DEĞİŞKENLER

`i = 1` `j = 3`

ÇIKTI

```
1 2
```

İç İçe Döngü Adım Adım (i = 1)




```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
▶     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

Satır tamamlandı: **println()** satır sonunu yazar (imleç alt satıra geçer). Sonra **i = 2**; dış karara dön.

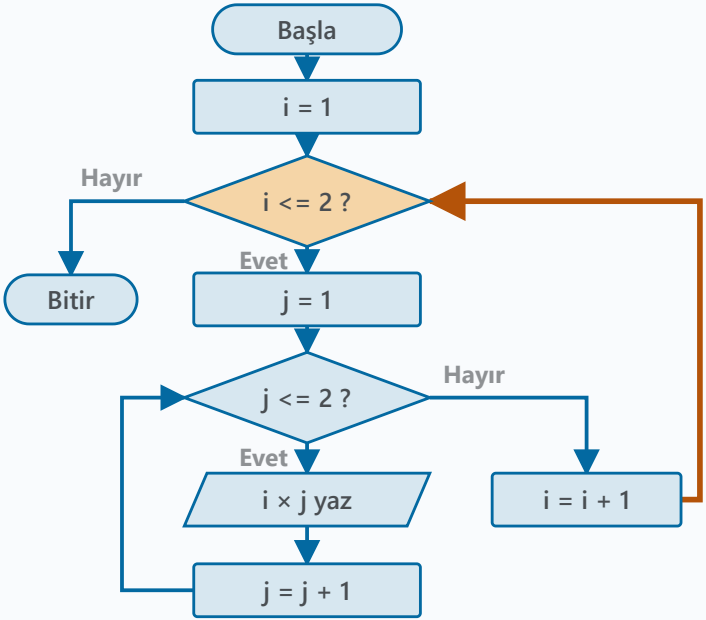
DEĞİŞKENLER

`i = 2` `j = 3`

ÇIKTI

```
1 2
```

İç İçe Döngü Adım Adım (devam: i = 2)



Dış karar: **2 <= 2** → **doğru** → iç döngüye gir.

```
1 Başla
2   i = 1
▶   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

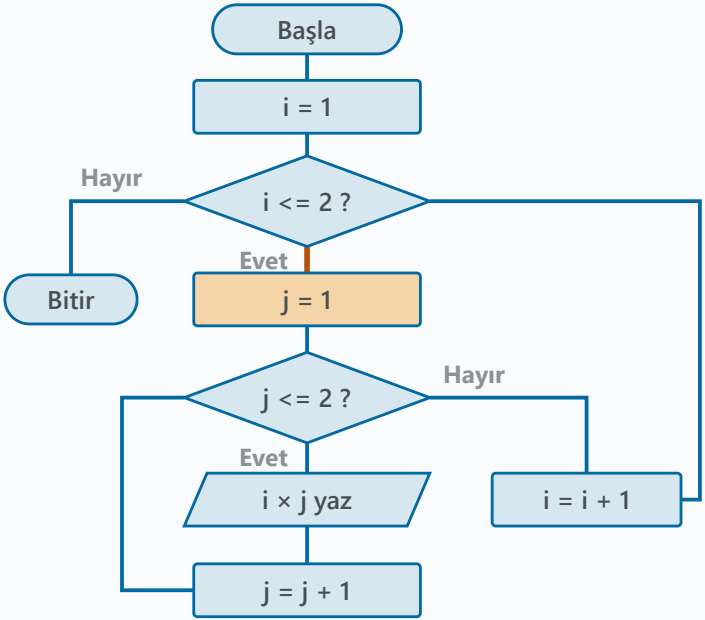
DEĞİŞKENLER

`i = 2`

ÇIKTI

```
1 2
```

İç İçe Döngü Adım Adım (devam: i = 2)



j = 1: iç sayaç her dış turda **yeniden** 1 olur.

```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
▶     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

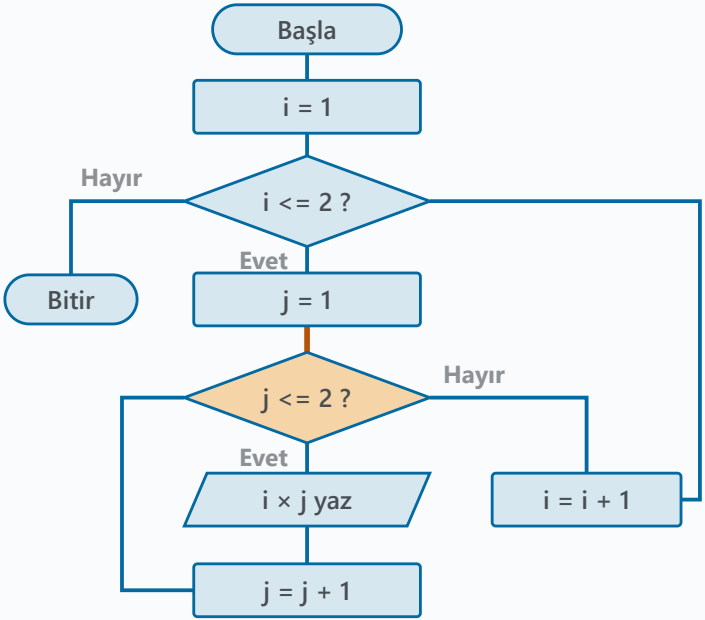
DEĞİŞKENLER

`i = 2` `j = 1`

ÇIKTI

```
1 2
```

İç İçe Döngü Adım Adım (devam: i = 2)



İç karar: **1 <= 2 → doğru**

```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
▶     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

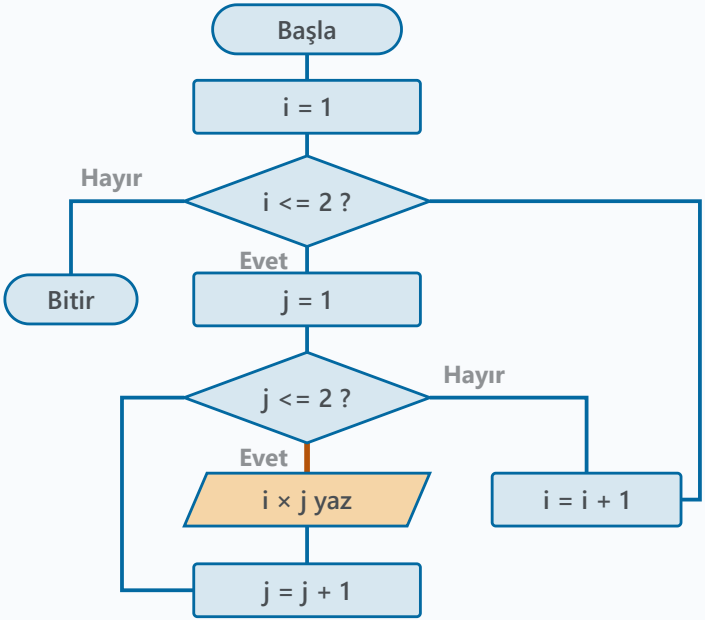
DEĞİŞKENLER

`i = 2` `j = 1`

ÇIKTI

```
1 2
```

İç İçe Döngü Adım Adım (devam: i = 2)



2 × 1 = 2 yazılır.

```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
▶       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

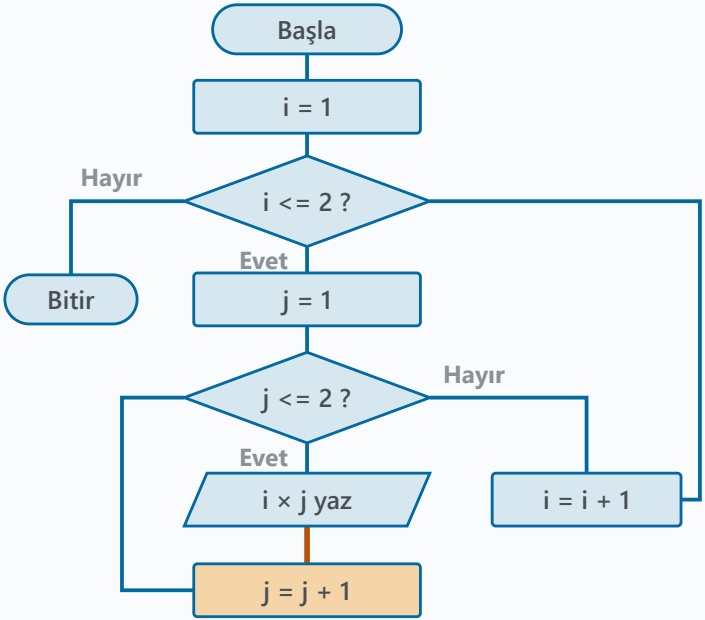
DEĞİŞKENLER

`i = 2` `j = 1`

ÇIKTI

```
1 2
2
```

İç İçe Döngü Adım Adım (devam: i = 2)



j = 2; iç karara dön.

```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
▶       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

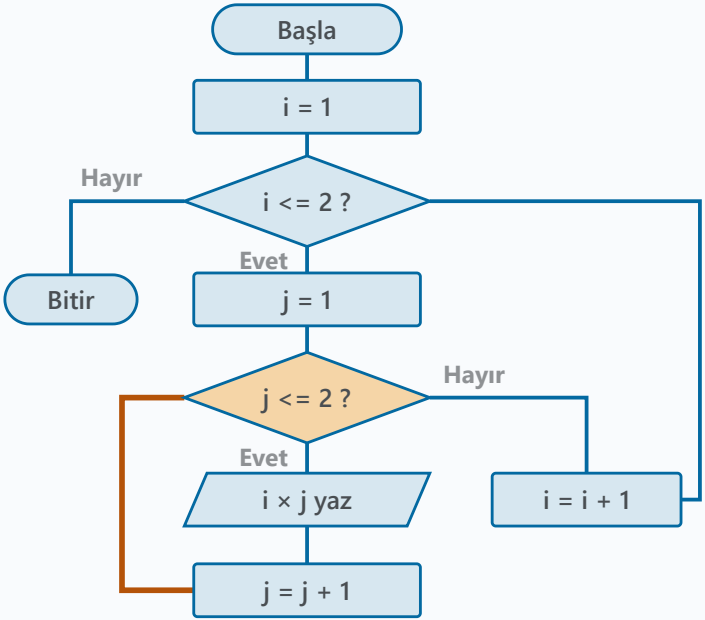
DEĞİŞKENLER

`i = 2` `j = 2`

ÇIKTI

```
1 2
2
```

İç İçe Döngü Adım Adım (devam: i = 2)



İç karar: **2 <= 2 → doğru**

```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
▶     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

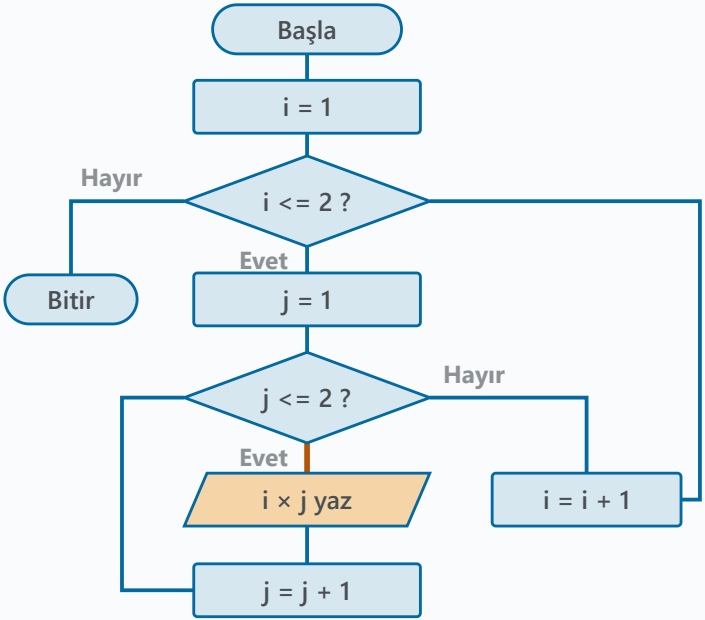
DEĞİŞKENLER

`i = 2` `j = 2`

ÇIKTI

```
1 2
2
```

İç İçe Döngü Adım Adım (devam: i = 2)



2 × 2 = 4 yazılır.

```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
▶       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

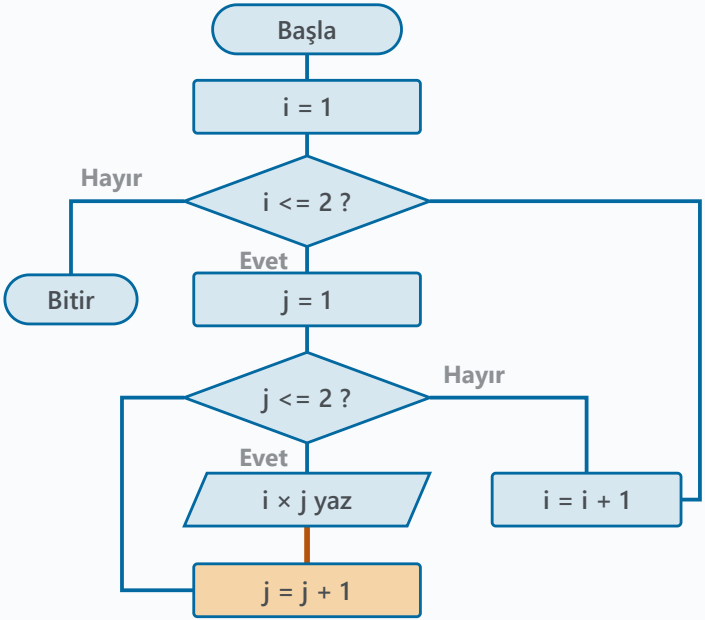
DEĞİŞKENLER

`i = 2` `j = 2`

ÇIKTI

```
1 2
2 4
```

İç İçe Döngü Adım Adım (devam: i = 2)



j = 3; iç karara dön.

```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
▶       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

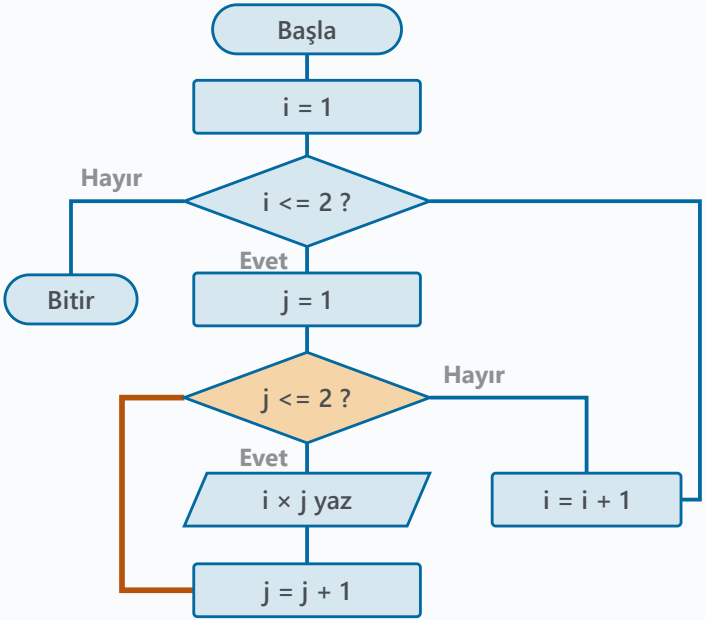
DEĞİŞKENLER

`i = 2` `j = 3`

ÇIKTI

```
1 2
2 4
```

İç İçe Döngü Adım Adım (devam: i = 2)



İç karar: **3 <= 2** → **yanlış** → Hayır, iç döngü biter.

```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
▶     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

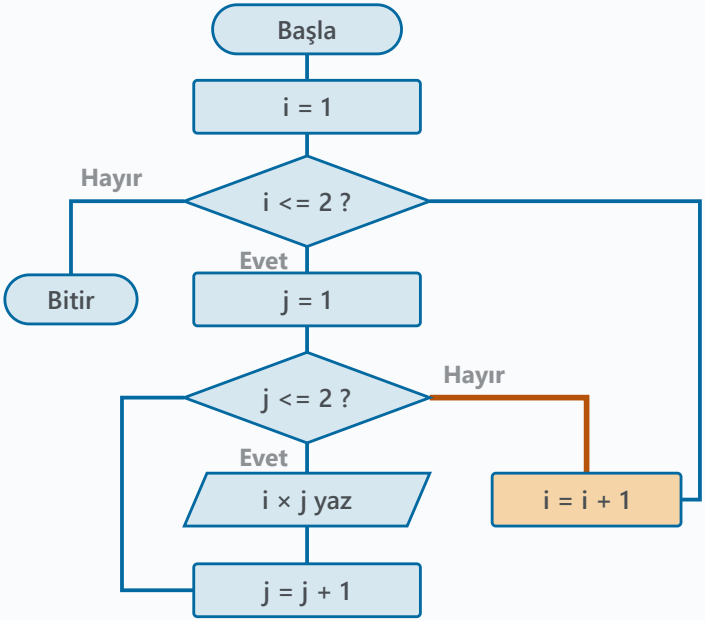
DEĞİŞKENLER

`i = 2` `j = 3`

ÇIKTI

```
1 2
2 4
```

İç İçe Döngü Adım Adım (devam: i = 2)



Satır tamamlandı: **println()** satır sonunu yazar. Sonra **i = 3**; dış karara dön.


```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
▶     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

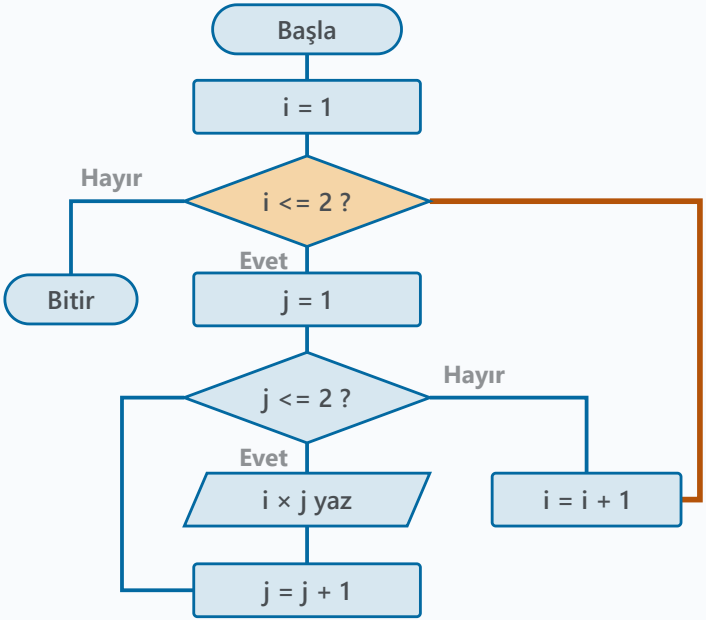
DEĞİŞKENLER

`i = 3` `j = 3`

ÇIKTI

```
1 2
2 4
```

İç İçe Döngü Adım Adım (devam: i = 2)



Dış karar: **3 <= 2** → **yanlış** → Hayır, Bitir'e git.

```
1 Başla
2   i = 1
▶   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
9 Bitir
```

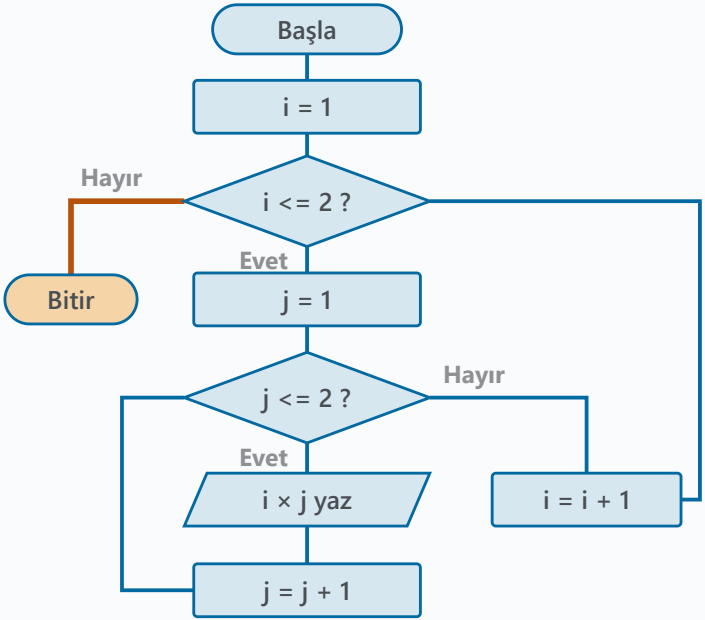
DEĞİŞKENLER

`i = 3`

ÇIKTI

```
1 2
2 4
```

İç İçe Döngü Adım Adım (devam: i = 2)



Bitir. İç gövde 2 × 2 = **4** kez çalıştı; çıktı küçük bir çarpım tablosu.


```
1 Başla
2   i = 1
3   i <= 2 ?  (Hayır → Bitir)
4     j = 1
5     j <= 2 ?  (Hayır → i = i + 1)
6       i × j yaz
7       j = j + 1, iç karara dön
8     satır sonu (println), i = i + 1, dış karara dön
▶ Bitir
```

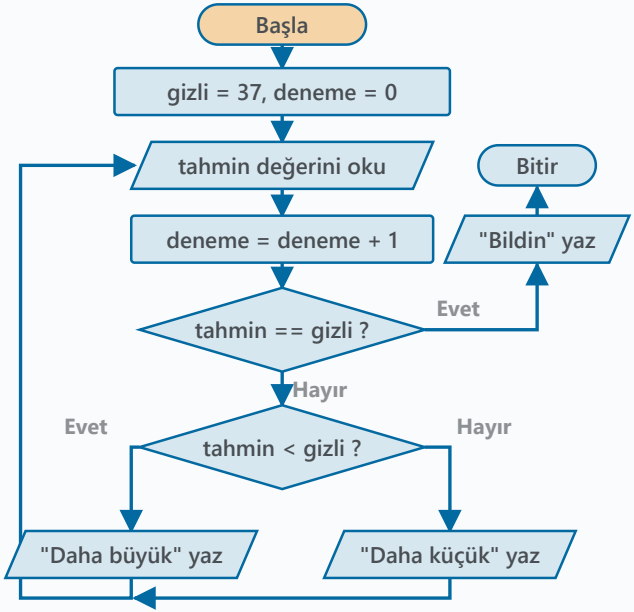
DEĞİŞKENLER

`i = 3` `j = 3`

ÇIKTI

```
1 2
2 4
```

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



Başla. Gizli sayı (normalde rastgele seçilir) 37; oyuncunun tahminleri 50, 25, 37.

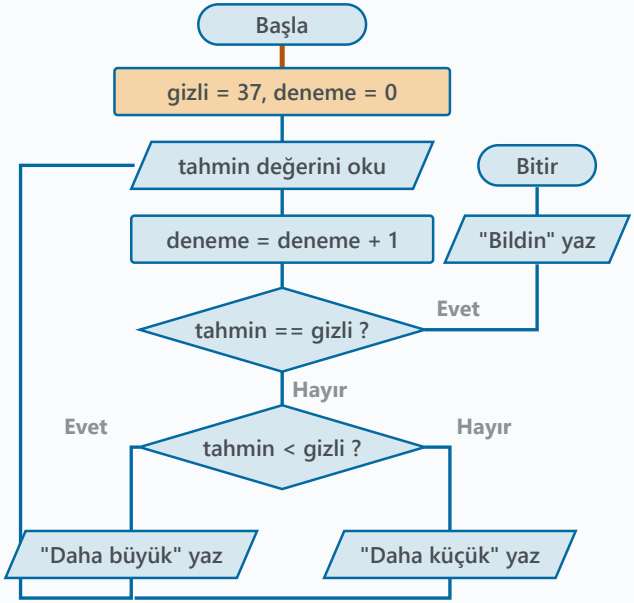

```
▶ Başla
2   gizli = 37, deneme = 0
3   tahmin değerini oku
4   deneme = deneme + 1
5   tahmin == gizli ?
6     Evet → "Bildin (" deneme ". deneme)" yaz,
    Bitir
7   tahmin < gizli ?
8     Evet → "Daha büyük" yaz
9     Hayır → "Daha küçük" yaz
10  3. satıra (oku) dön
11 Bitir
```

DEĞİŞKENLER

—

ÇIKTI

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



Gizli sayı belirlenir, deneme sayacı 0.

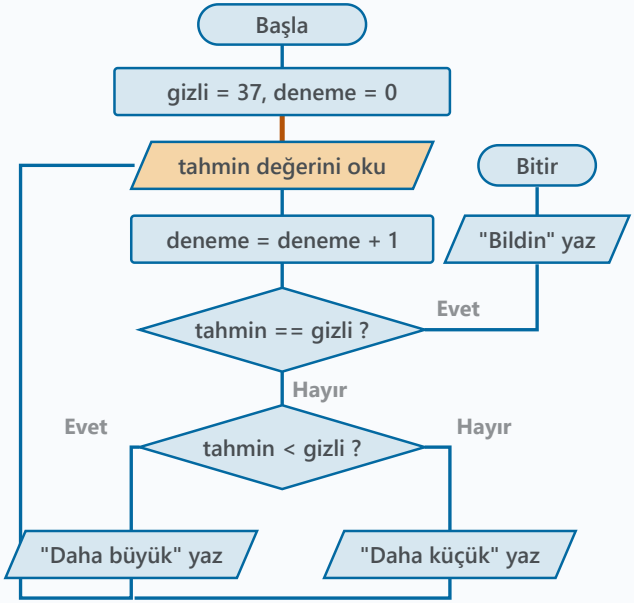
```
1  Başla
▶    gizli = 37, deneme = 0
3    tahmin değerini oku
4    deneme = deneme + 1
5    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
   Bitir
7    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

DEĞİŞKENLER

`gizli = 37` `deneme = 0`

ÇIKTI

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



Oyuncu **50** girer.

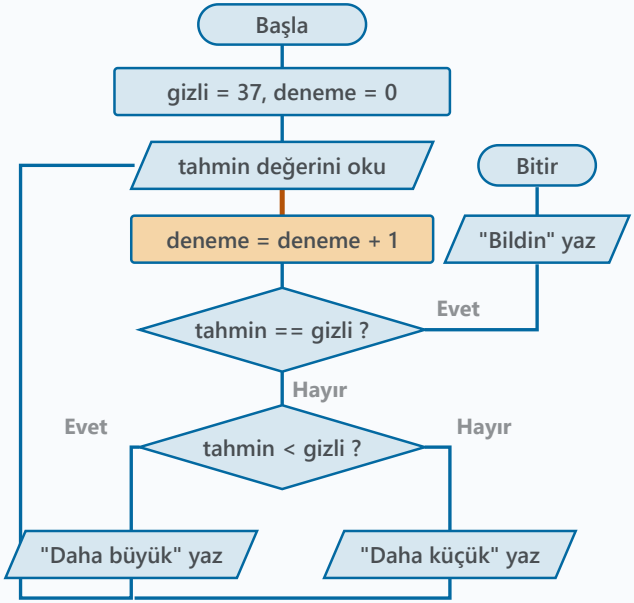
```
1  Başla
2    gizli = 37, deneme = 0
▶    tahmin değerini oku
4    deneme = deneme + 1
5    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
     Bitir
7    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

DEĞİŞKENLER

`gizli = 37` `deneme = 0` `tahmin = 50`

ÇIKTI

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



deneme = 1.

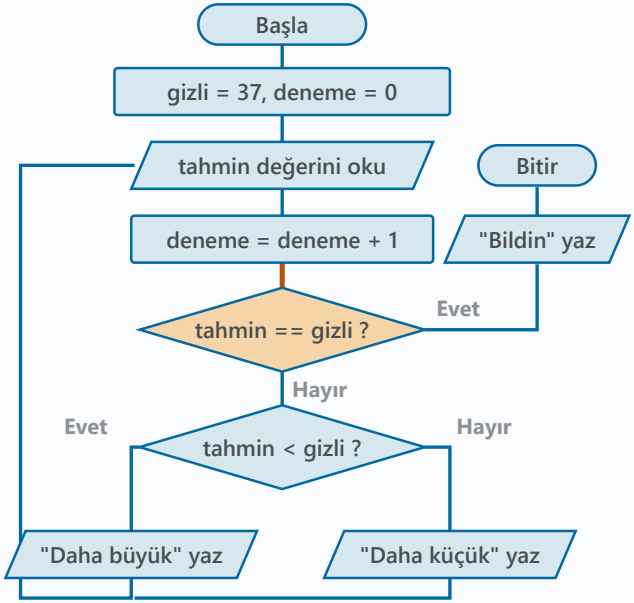
```
1  Başla
2    gizli = 37, deneme = 0
3    tahmin değerini oku
▶    deneme = deneme + 1
5    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
   Bitir
7    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

DEĞİŞKENLER

`gizli = 37` `deneme = 1` `tahmin = 50`

ÇIKTI

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



Karar: **50 == 37** → **yanlış** → Hayır, ipucu verilecek.

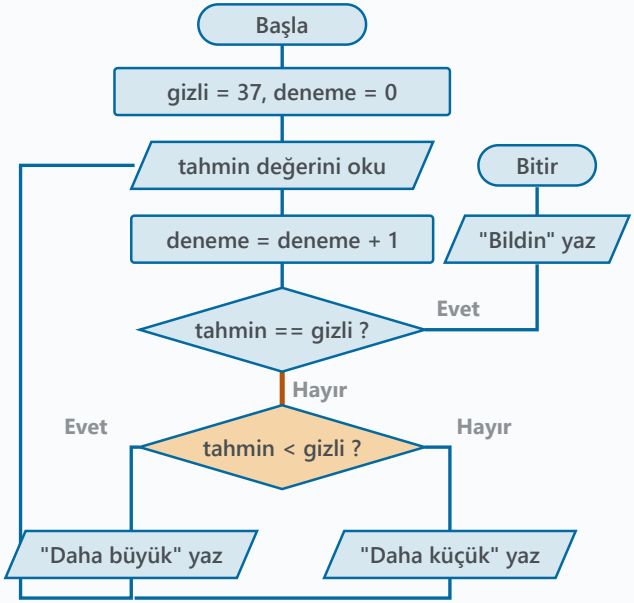
```
1  Başla
2    gizli = 37, deneme = 0
3    tahmin değerini oku
4    deneme = deneme + 1
▶    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
   Bitir
7    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

DEĞİŞKENLER

`gizli = 37` `deneme = 1` `tahmin = 50`

ÇIKTI

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



Karar: **50 < 37 → yanlış**.

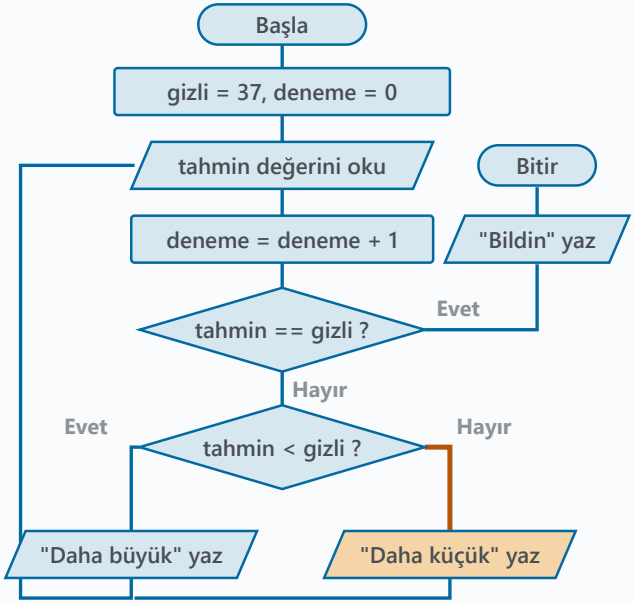
```
1  Başla
2    gizli = 37, deneme = 0
3    tahmin değerini oku
4    deneme = deneme + 1
5    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
   Bitir
▶    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

DEĞİŞKENLER

`gizli = 37` `deneme = 1` `tahmin = 50`

ÇIKTI

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



"Daha küçük" yazılır; ok yeniden **oku** kutusuna döner.


```
1  Başla
2    gizli = 37, deneme = 0
3    tahmin değerini oku
4    deneme = deneme + 1
5    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
   Bitir
7    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
▶      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

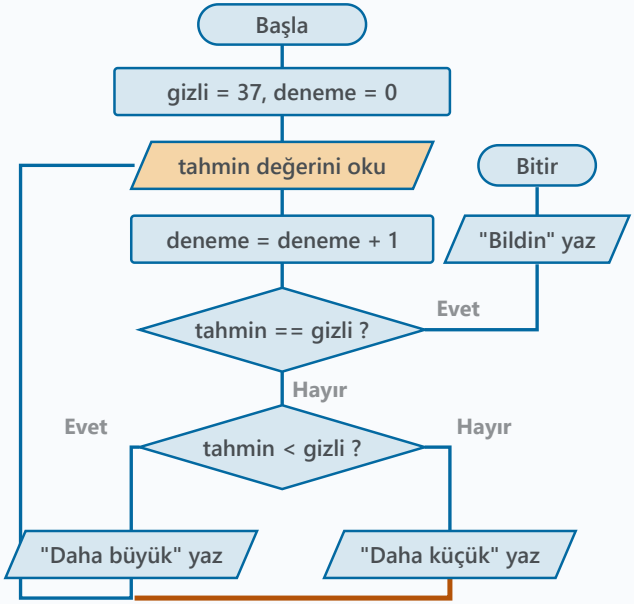
DEĞİŞKENLER

`gizli = 37` `deneme = 1` `tahmin = 50`

ÇIKTI

```
Daha küçük
```

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



Oyuncu **25** girer.

```
1  Başla
2    gizli = 37, deneme = 0
▶    tahmin değerini oku
4    deneme = deneme + 1
5    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
     Bitir
7    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

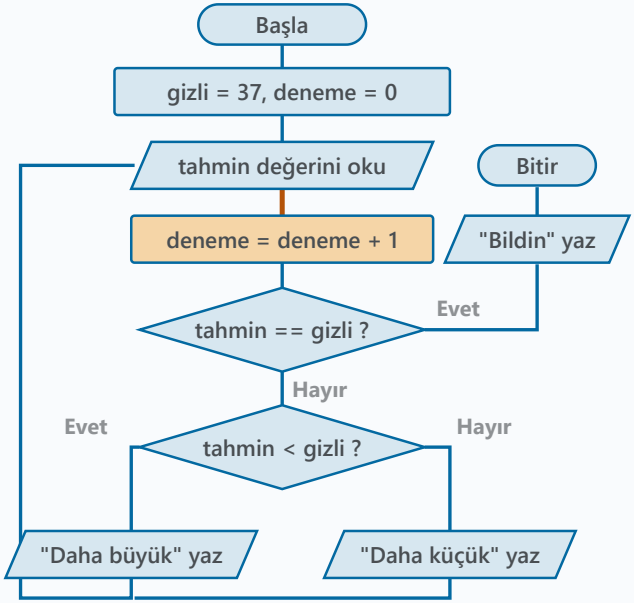
DEĞİŞKENLER

`gizli = 37` `deneme = 1` `tahmin = 25`

ÇIKTI

```
Daha küçük
```

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



deneme = 2.

```
1  Başla
2    gizli = 37, deneme = 0
3    tahmin değerini oku
▶    deneme = deneme + 1
5    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
   Bitir
7    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

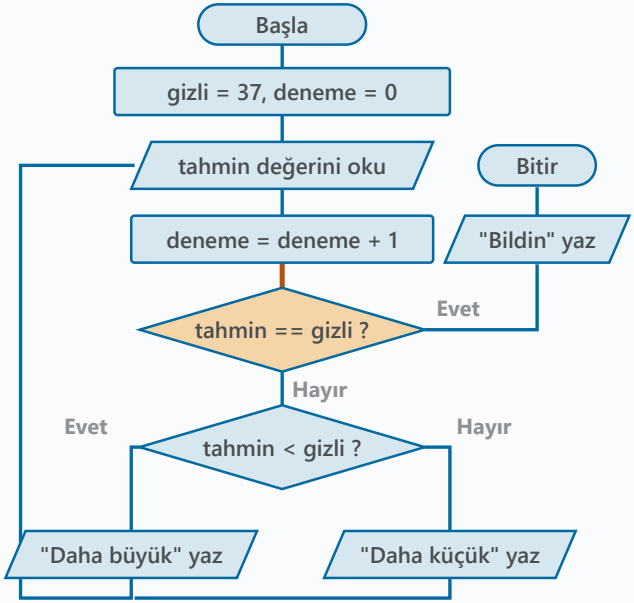
DEĞİŞKENLER

`gizli = 37` `deneme = 2` `tahmin = 25`

ÇIKTI

```
Daha küçük
```

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



Karar: **25 == 37** → **yanlış** → Hayır, ipucu verilecek.

```
1  Başla
2    gizli = 37, deneme = 0
3    tahmin değerini oku
4    deneme = deneme + 1
▶    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
   Bitir
7    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

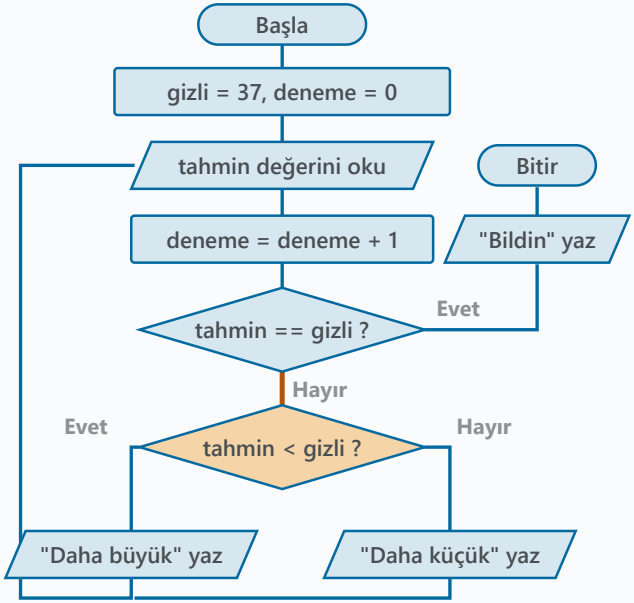
DEĞİŞKENLER

`gizli = 37` `deneme = 2` `tahmin = 25`

ÇIKTI

```
Daha küçük
```

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



Karar: **25 < 37 → doğru**.

```
1  Başla
2    gizli = 37, deneme = 0
3    tahmin değerini oku
4    deneme = deneme + 1
5    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
   Bitir
▶    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

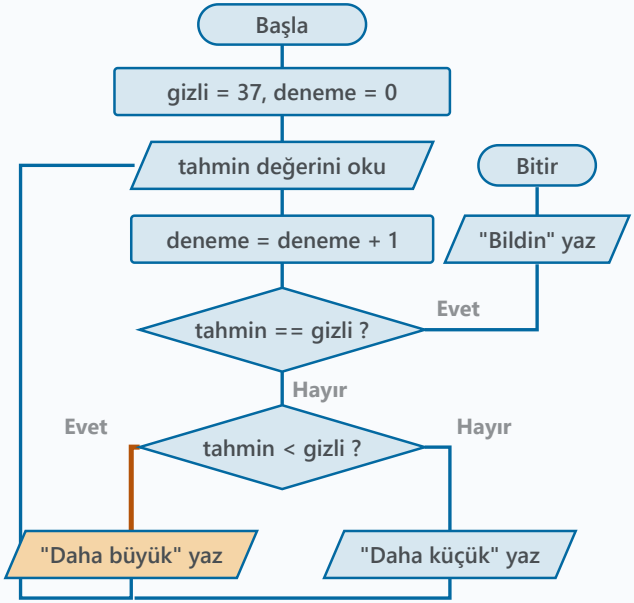
DEĞİŞKENLER

`gizli = 37` `deneme = 2` `tahmin = 25`

ÇIKTI

```
Daha küçük
```

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



"Daha büyük" yazılır; ok yeniden **oku** kutusuna döner.


```
1  Başla
2    gizli = 37, deneme = 0
3    tahmin değerini oku
4    deneme = deneme + 1
5    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
   Bitir
7    tahmin < gizli ?
▶      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

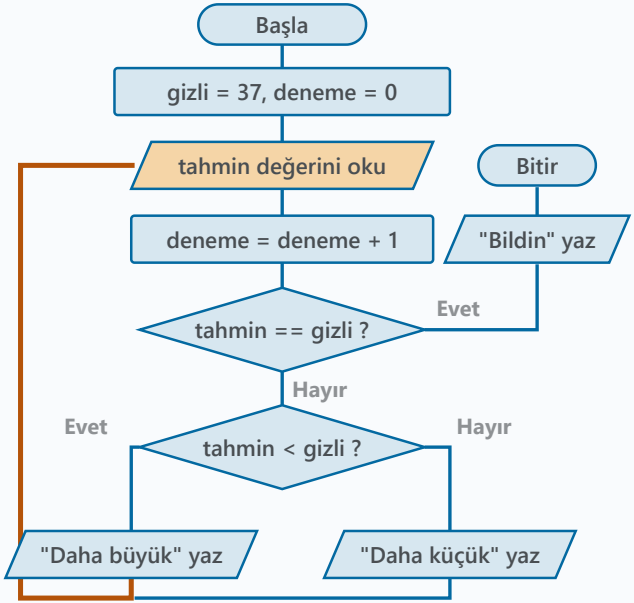
DEĞİŞKENLER

`gizli = 37` `deneme = 2` `tahmin = 25`

ÇIKTI

```
Daha küçük
Daha büyük
```

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



Oyuncu **37** girer.

```
1  Başla
2    gizli = 37, deneme = 0
▶   tahmin değerini oku
4    deneme = deneme + 1
5    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
   Bitir
7    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

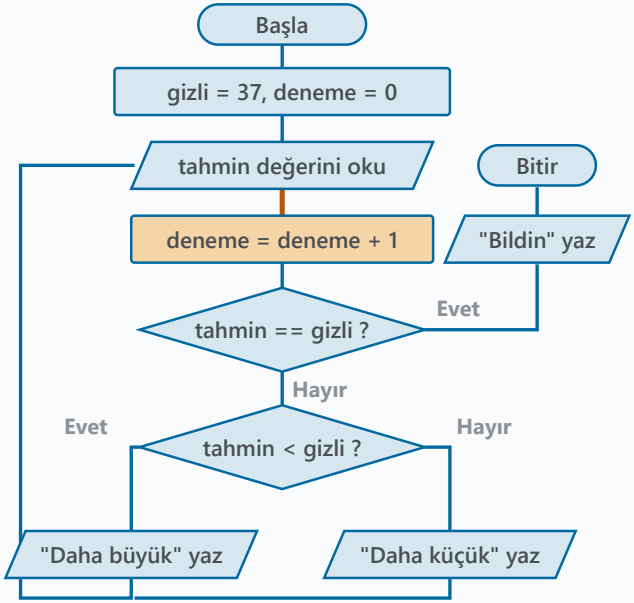
DEĞİŞKENLER

`gizli = 37` `deneme = 2` `tahmin = 37`

ÇIKTI

```
Daha küçük
Daha büyük
```

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



deneme = 3.


```
1  Başla
2    gizli = 37, deneme = 0
3    tahmin değerini oku
▶    deneme = deneme + 1
5    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
   Bitir
7    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

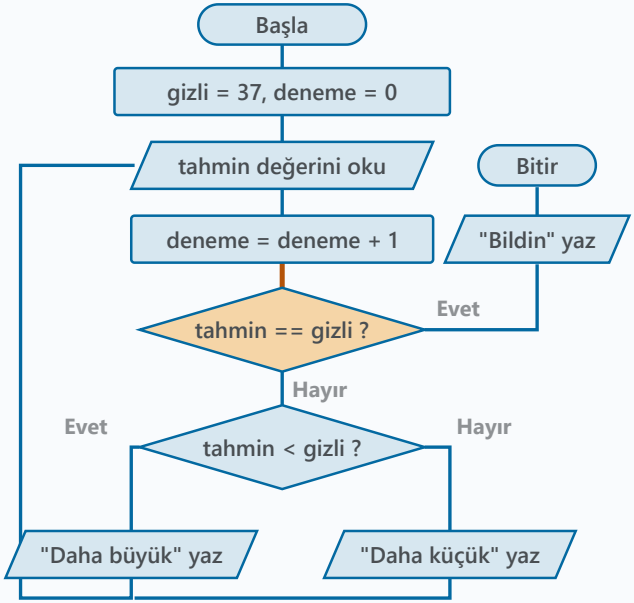
DEĞİŞKENLER

`gizli = 37` `deneme = 3` `tahmin = 37`

ÇIKTI

```
Daha küçük
Daha büyük
```

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



Karar: **37 == 37** → **doğru** → Evet.

```
1  Başla
2    gizli = 37, deneme = 0
3    tahmin değerini oku
4    deneme = deneme + 1
▶    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
   Bitir
7    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

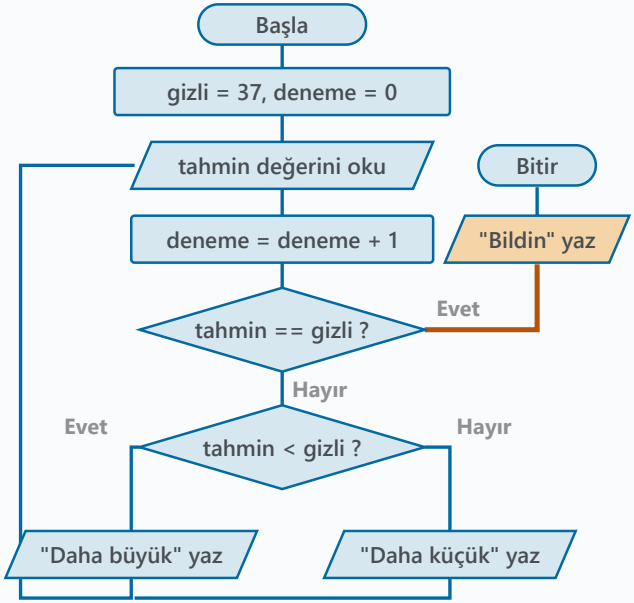
DEĞİŞKENLER

`gizli = 37` `deneme = 3` `tahmin = 37`

ÇIKTI

```
Daha küçük
Daha büyük
```

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



Doğru tahmin! Deneme sayısı **3** ile birlikte yazılır.

```
1  Başla
2    gizli = 37, deneme = 0
3    tahmin değerini oku
4    deneme = deneme + 1
5    tahmin == gizli ?
▶      Evet → "Bildin (" deneme ". deneme)" yaz,
   Bitir
7    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
11 Bitir
```

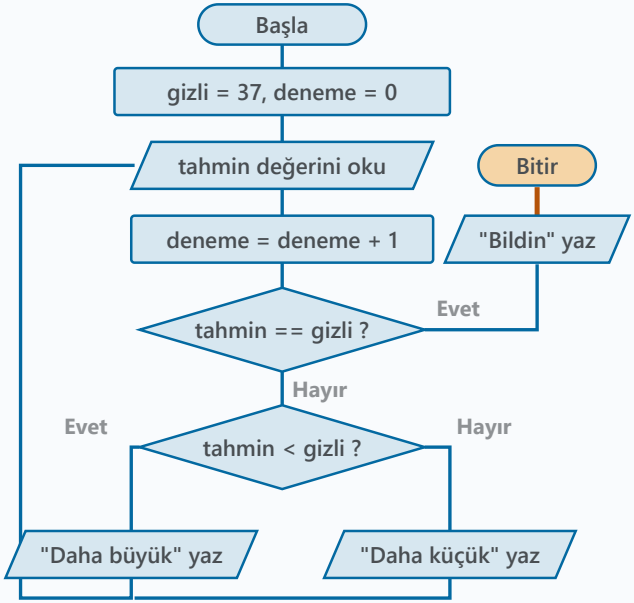
DEĞİŞKENLER

`gizli = 37` `deneme = 3` `tahmin = 37`

ÇIKTI

```
Daha küçük
Daha büyük
Bildin (3. deneme)
```

Sayı Tahmin Oyunu Adım Adım (gizli = 37)



Bitir. Döngü, koşul (tahmin == gizli) doğru olunca sona erdi; kaç tur döneceği önceden bilinmez.


```
1  Başla
2    gizli = 37, deneme = 0
3    tahmin değerini oku
4    deneme = deneme + 1
5    tahmin == gizli ?
6      Evet → "Bildin (" deneme ". deneme)" yaz,
   Bitir
7    tahmin < gizli ?
8      Evet → "Daha büyük" yaz
9      Hayır → "Daha küçük" yaz
10   3. satıra (oku) dön
▶  Bitir
```

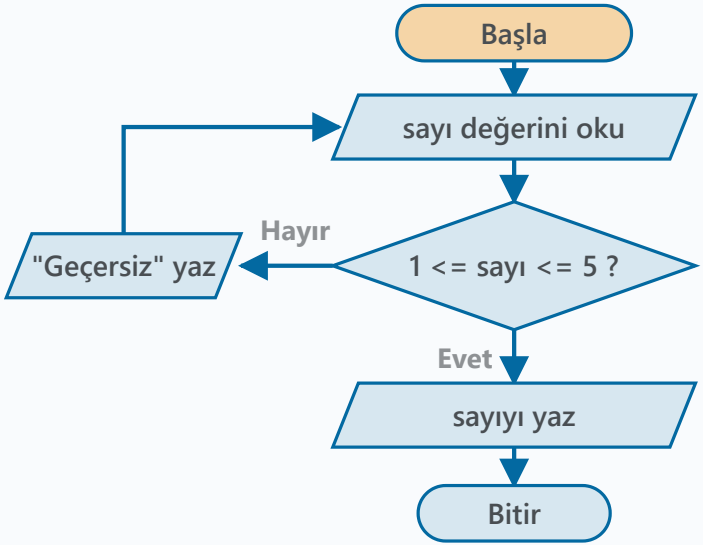
DEĞİŞKENLER

`gizli = 37` `deneme = 3` `tahmin = 37`

ÇIKTI

```
Daha küçük
Daha büyük
Bildin (3. deneme)
```

do-while Akış Diyagramı Adım Adım (girdiler: 7, 0, 3)




```
▶ Başla
2   sayı değerini oku
3   1 <= sayı <= 5 ?
4     Hayır → "Geçersiz" yaz, oku'ya dön
5     Evet → sayıyı yaz
6 Bitir
```

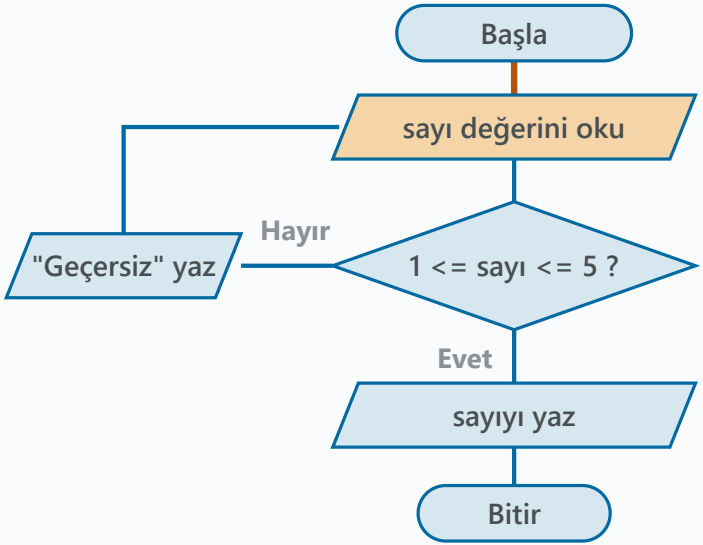
DEĞİŞKENLER

—

ÇIKTI

Başla. Girdi listesi sırayla verilecek: 7, 0, 3. Önce **gövde** çalışır (oku), koşul **sonda** sorulur: son koşullu döngü.

do-while Akış Diyagramı Adım Adım (girdiler: 7, 0, 3)




```
1 Başla
▶   sayı değerini oku
3   1 <= sayı <= 5 ?
4     Hayır → "Geçersiz" yaz, oku'ya dön
5     Evet → sayıyı yaz
6 Bitir
```

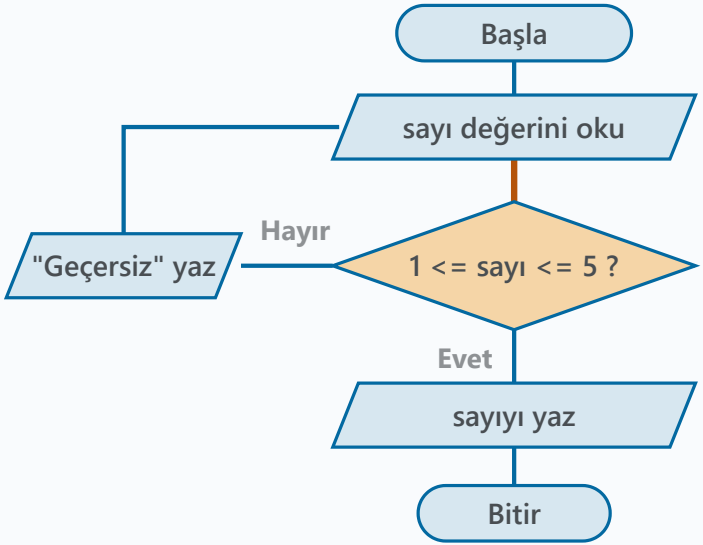
DEĞİŞKENLER

`sayı = 7`

ÇIKTI

Önce gövde: **sayı = 7** okunur. Koşul henüz hiç sorulmadı.

do-while Akış Diyagramı Adım Adım (girdiler: 7, 0, 3)




```
1 Başla
2   sayı değerini oku
▶   1 <= sayı <= 5 ?
4     Hayır → "Geçersiz" yaz, oku'ya dön
5     Evet → sayıyı yaz
6 Bitir
```

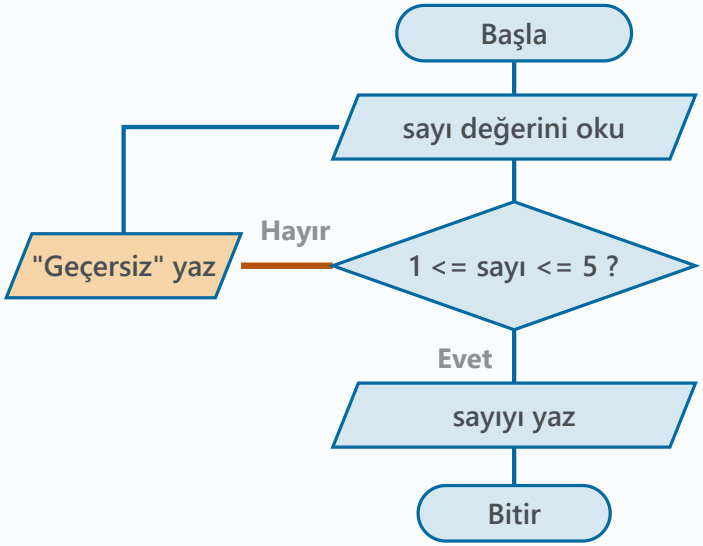
DEĞİŞKENLER

`sayı = 7`

ÇIKTI

Karar: 1 ≤ 7 ≤ 5 ? 7 > 5 olduğundan **yanlış** → Hayır.

do-while Akış Diyagramı Adım Adım (girdiler: 7, 0, 3)




```
1 Başla
2   sayı değerini oku
3   1 <= sayı <= 5 ?
      Hayır → "Geçersiz" yaz, oku'ya dön
5     Evet → sayıyı yaz
6 Bitir
```

DEĞİŞKENLER

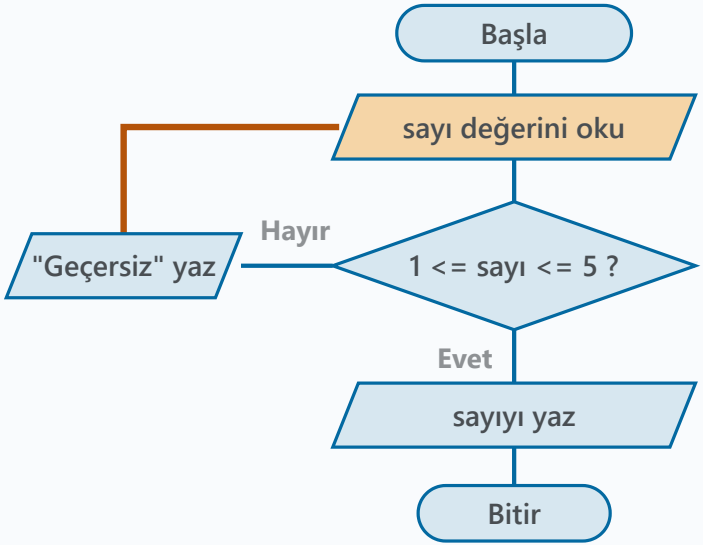
`sayı = 7`

ÇIKTI

```
Geçersiz
```

"Geçersiz" yazılır.

do-while Akış Diyagramı Adım Adım (girdiler: 7, 0, 3)




```
1 Başla
▶   sayı değerini oku
3   1 <= sayı <= 5 ?
4     Hayır → "Geçersiz" yaz, oku'ya dön
5     Evet → sayıyı yaz
6 Bitir
```

DEĞİŞKENLER

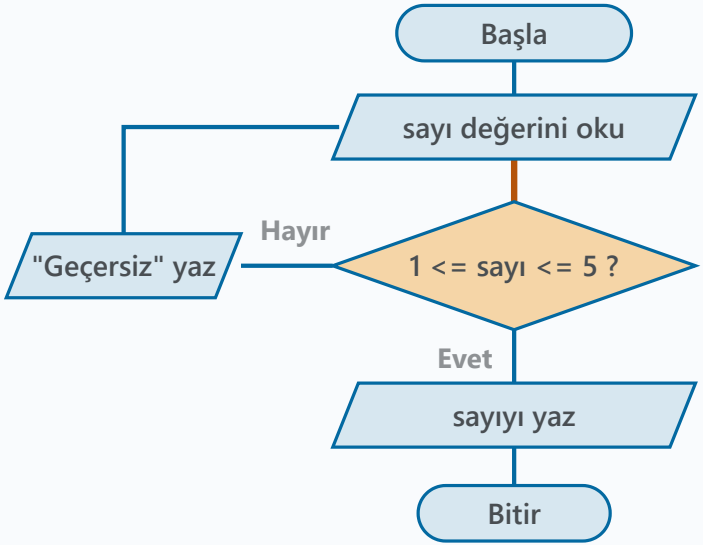
`sayı = 0`

ÇIKTI

```
Geçersiz
```

Ok **oku** kutusuna döner: **sayı = 0** okunur.

do-while Akış Diyagramı Adım Adım (girdiler: 7, 0, 3)




```
1 Başla
2   sayı değerini oku
▶   1 <= sayı <= 5 ?
4     Hayır → "Geçersiz" yaz, oku'ya dön
5     Evet → sayıyı yaz
6 Bitir
```

DEĞİŞKENLER

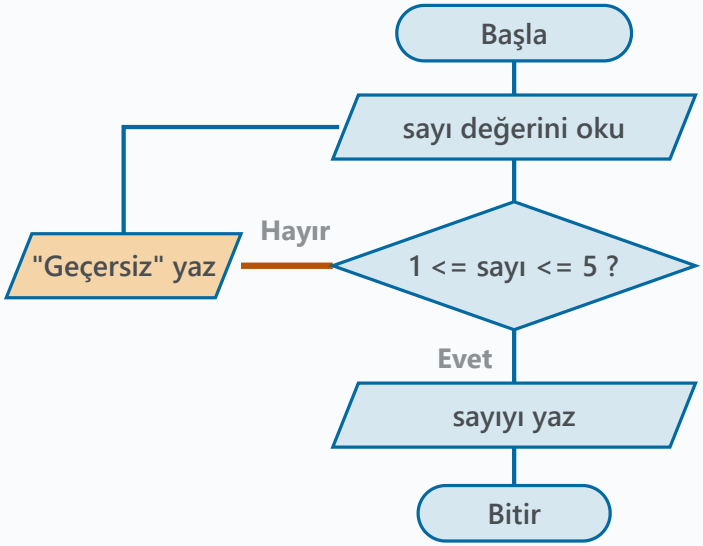
`sayı = 0`

ÇIKTI

```
Geçersiz
```

Karar: 1 ≤ 0 ≤ 5 ? 0 < 1 olduğundan **yanlış** → Hayır.

do-while Akış Diyagramı Adım Adım (girdiler: 7, 0, 3)



"Geçersiz" yeniden yazılır.

```
1 Başla
2   sayı değerini oku
3   1 <= sayı <= 5 ?
      Hayır → "Geçersiz" yaz, oku'ya dön
5     Evet → sayıyı yaz
6 Bitir
```

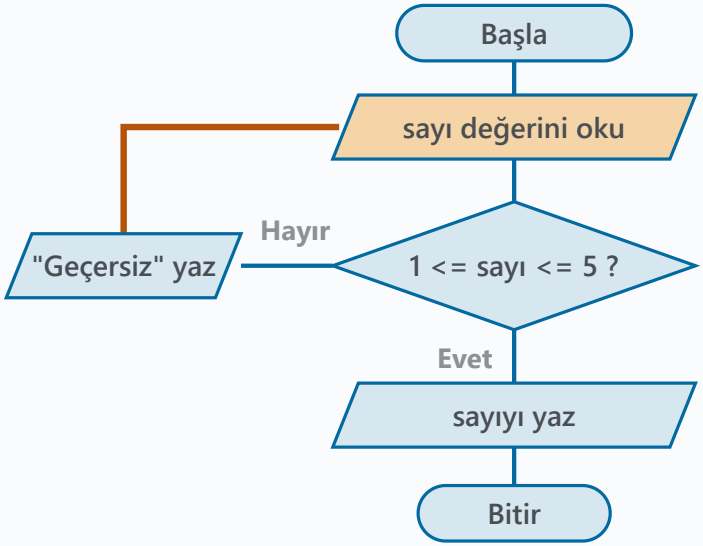
DEĞİŞKENLER

`sayı = 0`

ÇIKTI

```
Geçersiz
Geçersiz
```

do-while Akış Diyagramı Adım Adım (girdiler: 7, 0, 3)




```
1 Başla
▶   sayı değerini oku
3   1 <= sayı <= 5 ?
4     Hayır → "Geçersiz" yaz, oku'ya dön
5     Evet → sayıyı yaz
6 Bitir
```

DEĞİŞKENLER

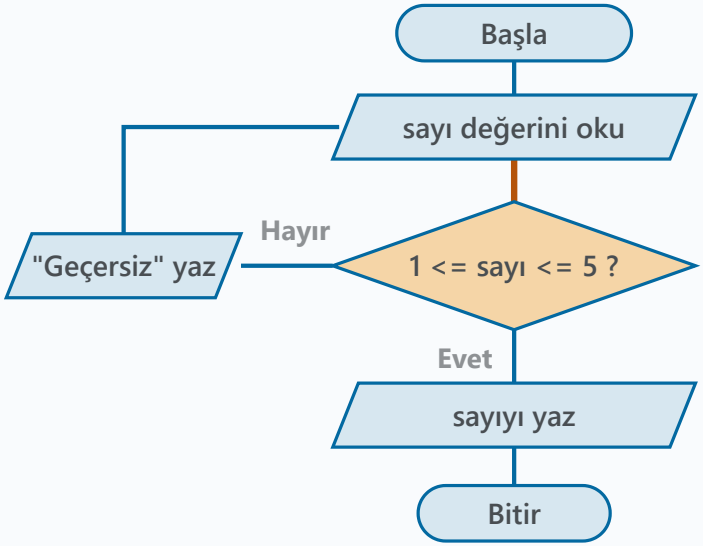
`sayı = 3`

ÇIKTI

```
Geçersiz
Geçersiz
```

Yeniden **oku**: **sayı = 3**.

do-while Akış Diyagramı Adım Adım (girdiler: 7, 0, 3)



Karar: **1 ≤ 3 ≤ 5** → **doğru** → Evet.

```
1 Başla
2   sayı değerini oku
▶   1 <= sayı <= 5 ?
4     Hayır → "Geçersiz" yaz, oku'ya dön
5     Evet → sayıyı yaz
6 Bitir
```

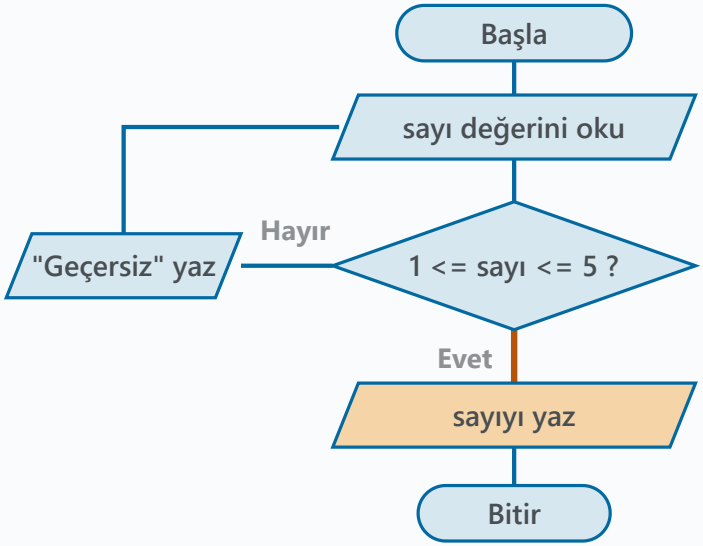
DEĞİŞKENLER

`sayı = 3`

ÇIKTI

```
Geçersiz
Geçersiz
```

do-while Akış Diyagramı Adım Adım (girdiler: 7, 0, 3)



Geçerli sayı yazılır: **3**.

```
1 Başla
2   sayı değerini oku
3   1 <= sayı <= 5 ?
4     Hayır → "Geçersiz" yaz, oku'ya dön
▶     Evet → sayıyı yaz
6 Bitir
```

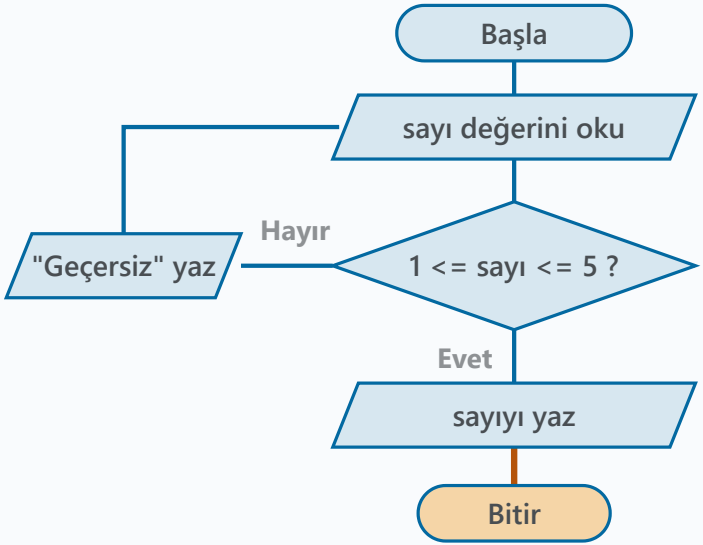
DEĞİŞKENLER

`sayı = 3`

ÇIKTI

```
Geçersiz
Geçersiz
Geçerli: 3
```

do-while Akış Diyagramı Adım Adım (girdiler: 7, 0, 3)




```
1  Başla
2    sayı değerini oku
3    1 <= sayı <= 5 ?
4      Hayır → "Geçersiz" yaz, oku'ya dön
5      Evet → sayıyı yaz
▶  Bitir
```

DEĞİŞKENLER

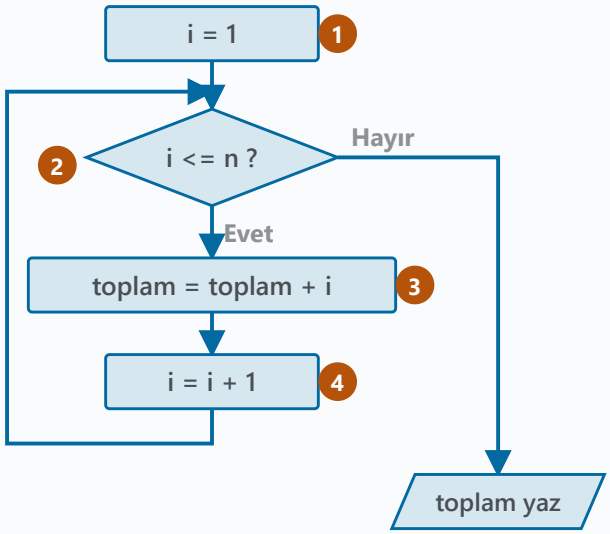
`sayı = 3`

ÇIKTI

```
Geçersiz
Geçersiz
Geçerli: 3
```

Bitir. Gövde 3 kez çalıştı. Koşul gövdeden **sonra** sorulduğu için gövde en az 1 kez çalışır. Java karşılığı: `do { … } while (sayı < 1 || sayı > 5);`

`for` Döngüsünün Akış Diyagramı

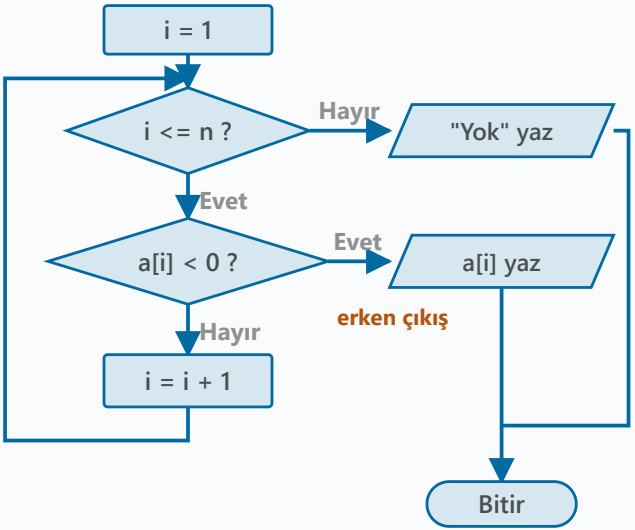



```
for (i = 1; i <= n; i++) {
    toplam = toplam + i;
}
```

1. **Başlangıç** (i = 1): döngüye girerken **bir kez** çalışır.
2. **Koşul** (i <= n): her turdan **önce** sorulur; for ön koşullu bir döngüdür.
3. **Gövde**: koşul doğruysa çalışır.
4. **Güncelleme** (i++): gövdeden **sonra** çalışır, ok koşula döner.

Diyagram aynı while diyagramıdır; for yalnızca 1, 2 ve 4'ü tek satırda toplar. Bazı kaynaklar 1, 2 ve 4'ü tek bir **hazırlık** simgesinde (altıgen, "i = 1 → n") gösterir.

Döngüden Erken Çıkış ve Bayrak

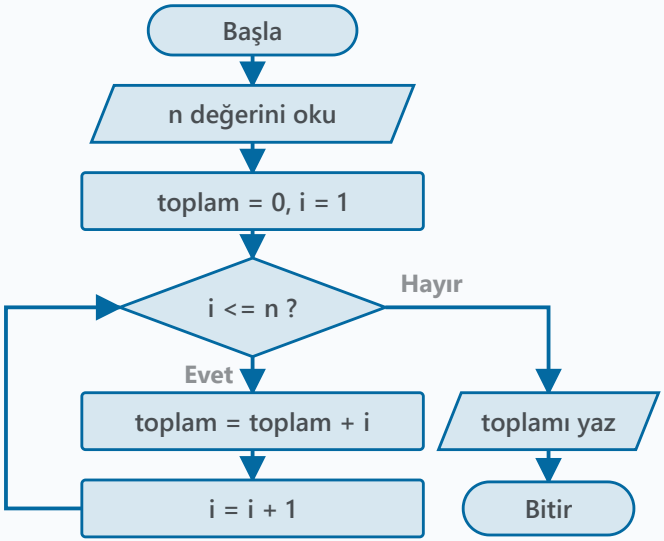


İlk negatif sayıyı arama: a = [4, 7, −2, 5, −8]

- Aranan bulununca kalan turlar gereksizdir. Gövdenin içinden döngü dışına giden ikinci ok **erken çıkış**tır; Java'da `break` (çık).
- Örnekte i = 3'te −2 bulunur ve yazılır; 5 ve −8'e hiç bakılmaz. Hiç negatif yoksa döngü koşuldan çıkar ve "Yok" yazılır.
- Döngünün iki çıkışı olur. Tek çıkış isteniyorsa **bayrak** kullanılır: bulundu = yanlış ile başlanır, bulununca doğru yapılır ve koşul **i <= n VE bulundu değil ?** olur. Döngüden sonra bayrağa bakılarak sonuç yazılır.
- **Atla** (`continue`) ise döngüden çıkmaz: o turun kalanını atlayıp güncellemeye ve koşula döner.

KODA ÇEVİRİ

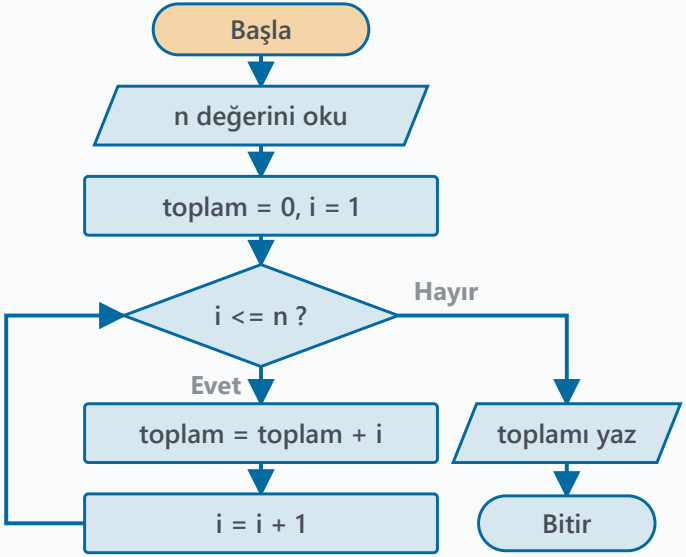
Diyagramdan Java Koduna




```java
public static void main(String[] args) {
    int n = 3;
    int toplam = 0, i = 1;
    while (i <= n) {
        toplam = toplam + i;
        i = i + 1;
    }
    System.out.println(toplam);
}
```

- **Başla / Bitir** → metodun başı ve sonu.
- **Girdi** → değişken (burada sabit değer).
- **İşlem** → atama deyimi.
- **Karar + geri dönen ok** → `while` koşulu ve gövdesi.
- **Çıktı** → `println`.

Diyagramdan Java Koduna Adım Adım (n = 3)



Başla simgesi metodun başlangıcıdır: `main` açılır.

```
▶ public static void main(String[] args) {
2     int n = 3;
3     int toplam = 0, i = 1;
4     while (i <= n) {
5         toplam = toplam + i;
6         i = i + 1;
7     }
8     System.out.println(toplam);
9 }
```

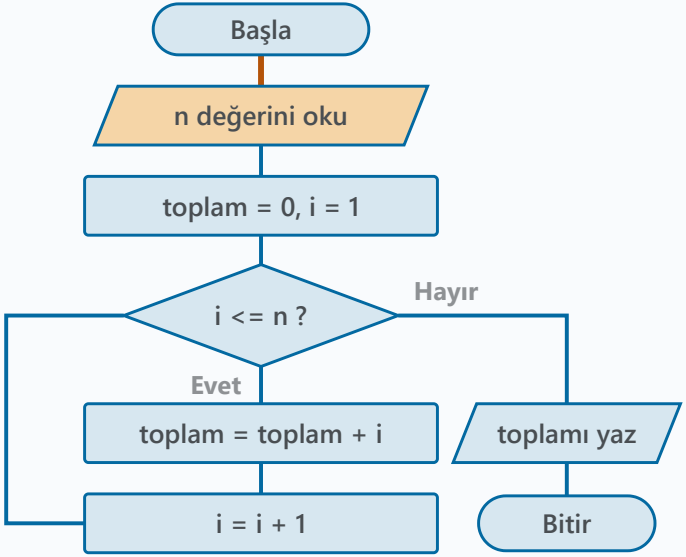
DEĞİŞKENLER

—

ÇIKTI

Adım 1 / 15

Diyagramdan Java Koduna Adım Adım (n = 3)



Girdi simgesi → `int n = 3;` (girdi sabit alındı).

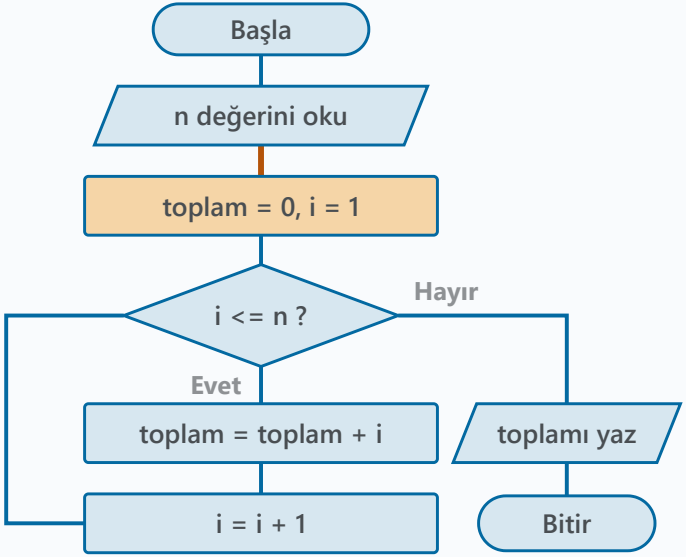
```
1 public static void main(String[] args) {
▶       int n = 3;
3       int toplam = 0, i = 1;
4       while (i <= n) {
5           toplam = toplam + i;
6           i = i + 1;
7       }
8       System.out.println(toplam);
9   }
```

DEĞİŞKENLER

`n = 3`

ÇIKTI

Diyagramdan Java Koduna Adım Adım (n = 3)



İşlem simgesi → iki değişkenin başlangıç değeri.

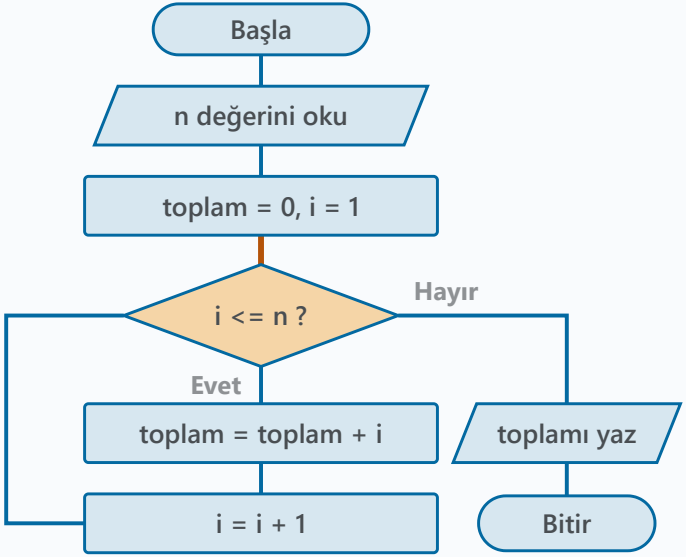
```
1 public static void main(String[] args) {
2     int n = 3;
▶     int toplam = 0, i = 1;
4     while (i <= n) {
5         toplam = toplam + i;
6         i = i + 1;
7     }
8     System.out.println(toplam);
9 }
```

DEĞİŞKENLER

`n = 3` `toplam = 0` `i = 1`

ÇIKTI

Diyagramdan Java Koduna Adım Adım (n = 3)



Karar simgesi → `while (i <= n)` : 1 <= 3 doğru.

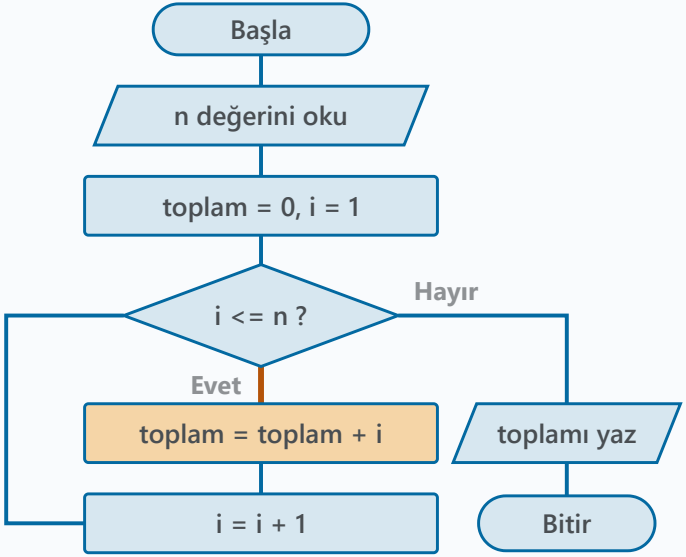
```
1 public static void main(String[] args) {
2     int n = 3;
3     int toplam = 0, i = 1;
▶     while (i <= n) {
5         toplam = toplam + i;
6         i = i + 1;
7     }
8     System.out.println(toplam);
9 }
```

DEĞİŞKENLER

`n = 3` `toplam = 0` `i = 1`

ÇIKTI

Diyagramdan Java Koduna Adım Adım (n = 3)



Evet yolu = döngü gövdesi: `toplam = 0 + 1` = 1.

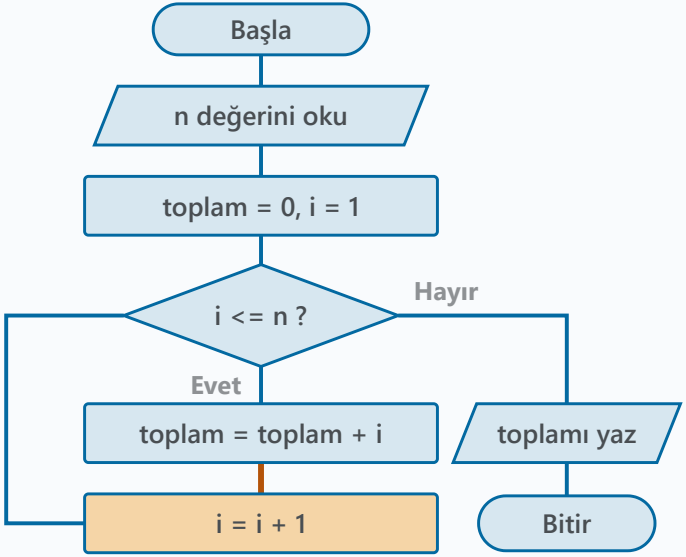
```
1 public static void main(String[] args) {
2     int n = 3;
3     int toplam = 0, i = 1;
4     while (i <= n) {
▶         toplam = toplam + i;
6         i = i + 1;
7     }
8     System.out.println(toplam);
9 }
```

DEĞİŞKENLER

`n = 3` `toplam = 1` `i = 1`

ÇIKTI

Diyagramdan Java Koduna Adım Adım (n = 3)




```
1 public static void main(String[] args) {
2     int n = 3;
3     int toplam = 0, i = 1;
4     while (i <= n) {
5         toplam = toplam + i;
          i = i + 1;
7     }
8     System.out.println(toplam);
9 }
```

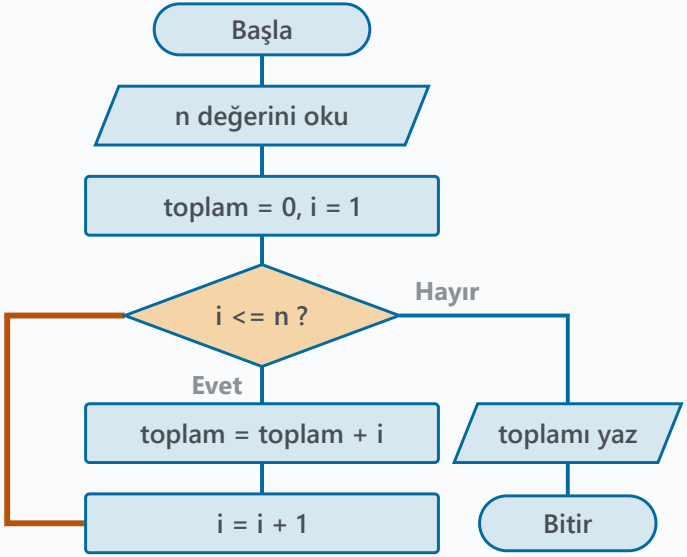
`i = i + 1` → i = 2. Sonraki ok karara **geri** döner.

DEĞİŞKENLER

`n = 3` `toplam = 1` `i = 2`

ÇIKTI

Diyagramdan Java Koduna Adım Adım (n = 3)



Geri dönen ok = döngü bloğunun sonundan koşula dönüş: 2 <= 3 doğru.

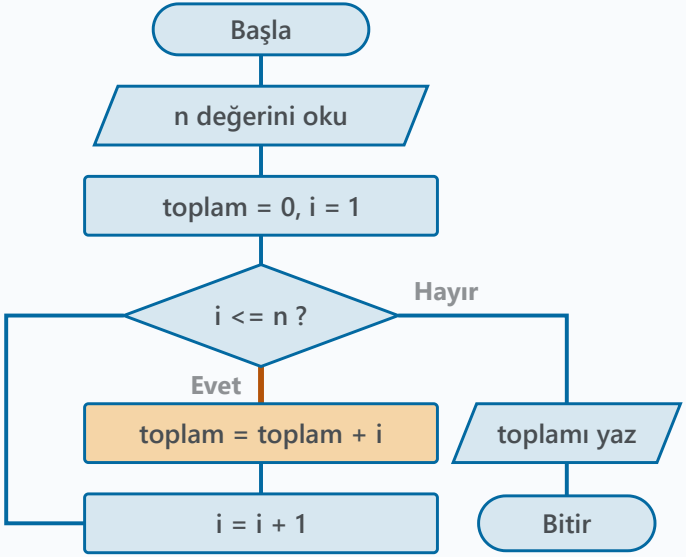
```
1 public static void main(String[] args) {
2     int n = 3;
3     int toplam = 0, i = 1;
▶     while (i <= n) {
5         toplam = toplam + i;
6         i = i + 1;
7     }
8     System.out.println(toplam);
9 }
```

DEĞİŞKENLER

`n = 3` `toplam = 1` `i = 2`

ÇIKTI

Diyagramdan Java Koduna Adım Adım (n = 3)



`toplam = 1 + 2` = 3.

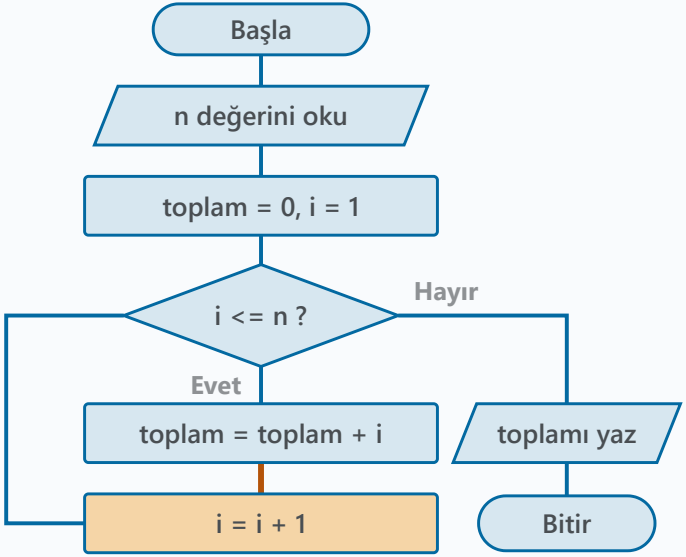
```
1 public static void main(String[] args) {
2     int n = 3;
3     int toplam = 0, i = 1;
4     while (i <= n) {
▶         toplam = toplam + i;
6         i = i + 1;
7     }
8     System.out.println(toplam);
9 }
```

DEĞİŞKENLER

`n = 3` `toplam = 3` `i = 2`

ÇIKTI

Diyagramdan Java Koduna Adım Adım (n = 3)



`i = 3` .

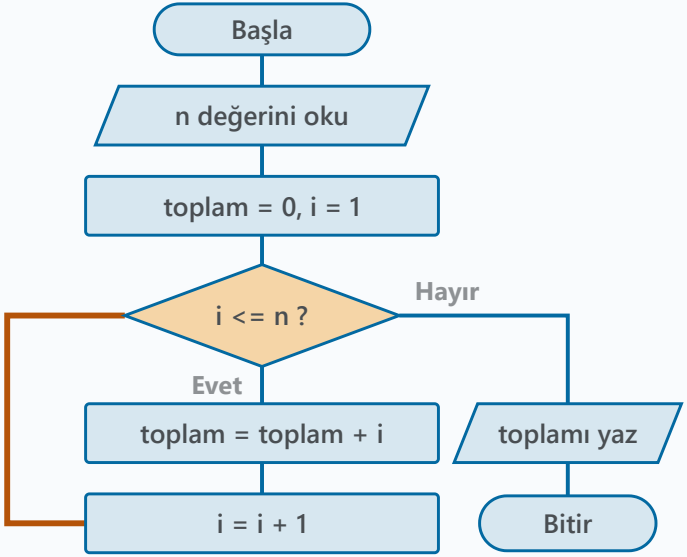
```
1 public static void main(String[] args) {
2     int n = 3;
3     int toplam = 0, i = 1;
4     while (i <= n) {
5         toplam = toplam + i;
          i = i + 1;
7     }
8     System.out.println(toplam);
9 }
```

DEĞİŞKENLER

`n = 3` `toplam = 3` `i = 3`

ÇIKTI

Diyagramdan Java Koduna Adım Adım (n = 3)



3 <= 3 doğru.

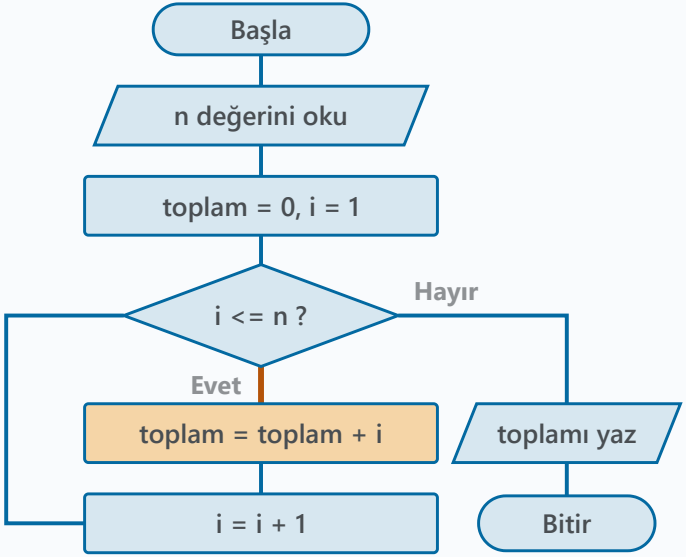
```java
public static void main(String[] args) {
    int n = 3;
    int toplam = 0, i = 1;
    while (i <= n) {
        toplam = toplam + i;
        i = i + 1;
    }
    System.out.println(toplam);
}
```

DEĞİŞKENLER

`n = 3` `toplam = 3` `i = 3`

ÇIKTI

Diyagramdan Java Koduna Adım Adım (n = 3)



`toplam = 3 + 3` = 6.

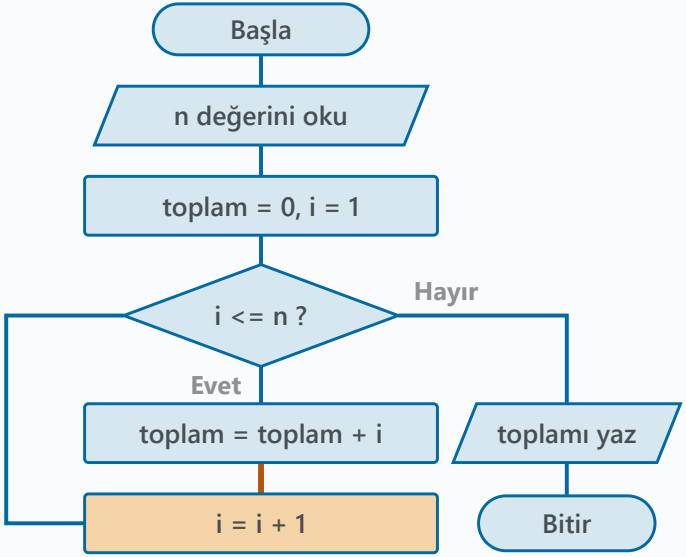
```
1 public static void main(String[] args) {
2     int n = 3;
3     int toplam = 0, i = 1;
4     while (i <= n) {
▶         toplam = toplam + i;
6         i = i + 1;
7     }
8     System.out.println(toplam);
9 }
```

DEĞİŞKENLER

`n = 3` `toplam = 6` `i = 3`

ÇIKTI

Diyagramdan Java Koduna Adım Adım (n = 3)



`i = 4` .

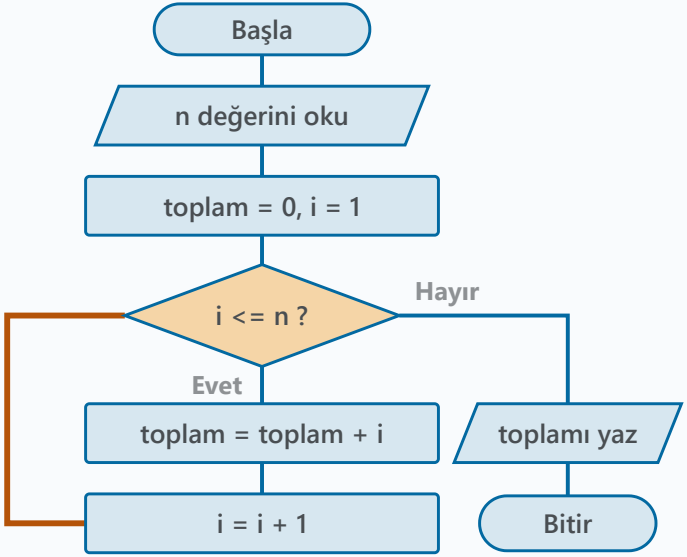
```
1 public static void main(String[] args) {
2     int n = 3;
3     int toplam = 0, i = 1;
4     while (i <= n) {
5         toplam = toplam + i;
          i = i + 1;
7     }
8     System.out.println(toplam);
9 }
```

DEĞİŞKENLER

`n = 3` `toplam = 6` `i = 4`

ÇIKTI

Diyagramdan Java Koduna Adım Adım (n = 3)



4 <= 3 **yanlış** → Hayır yolu, döngü bloğunun dışına çıkılır.

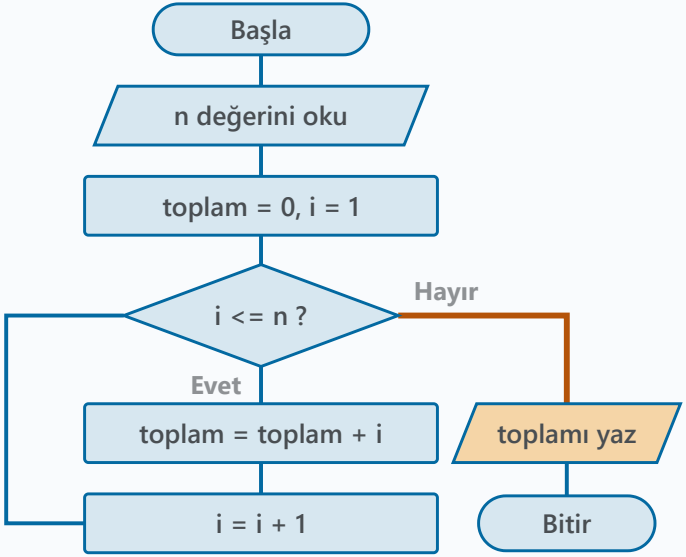
```
1 public static void main(String[] args) {
2     int n = 3;
3     int toplam = 0, i = 1;
▶     while (i <= n) {
5         toplam = toplam + i;
6         i = i + 1;
7     }
8     System.out.println(toplam);
9 }
```

DEĞİŞKENLER

`n = 3` `toplam = 6` `i = 4`

ÇIKTI

Diyagramdan Java Koduna Adım Adım (n = 3)



Çıktı simgesi → `println(toplam)` : **6** yazılır.

```
1 public static void main(String[] args) {
2     int n = 3;
3     int toplam = 0, i = 1;
4     while (i <= n) {
5         toplam = toplam + i;
6         i = i + 1;
7     }
▶     System.out.println(toplam);
9 }
```

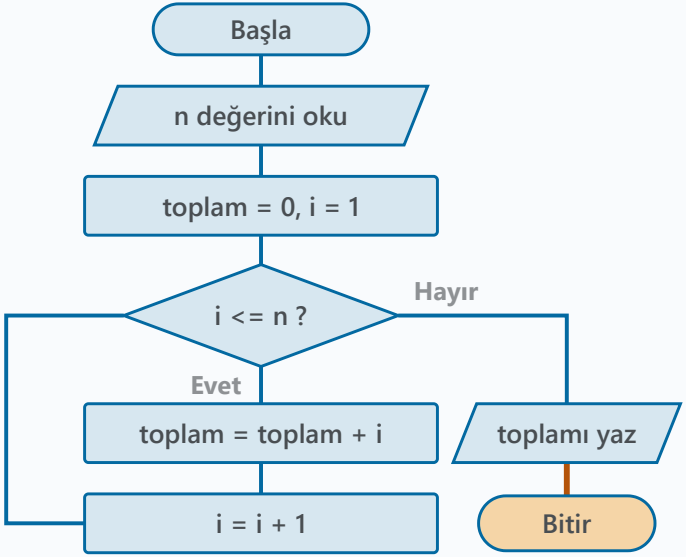
DEĞİŞKENLER

`n = 3` `toplam = 6` `i = 4`

ÇIKTI

```
6
```

Diyagramdan Java Koduna Adım Adım (n = 3)



Bitir → metodun kapanışı. Her simgenin bir kod karşılığı var.

```
1 public static void main(String[] args) {
2     int n = 3;
3     int toplam = 0, i = 1;
4     while (i <= n) {
5         toplam = toplam + i;
6         i = i + 1;
7     }
8     System.out.println(toplam);
▶ }
```

DEĞİŞKENLER

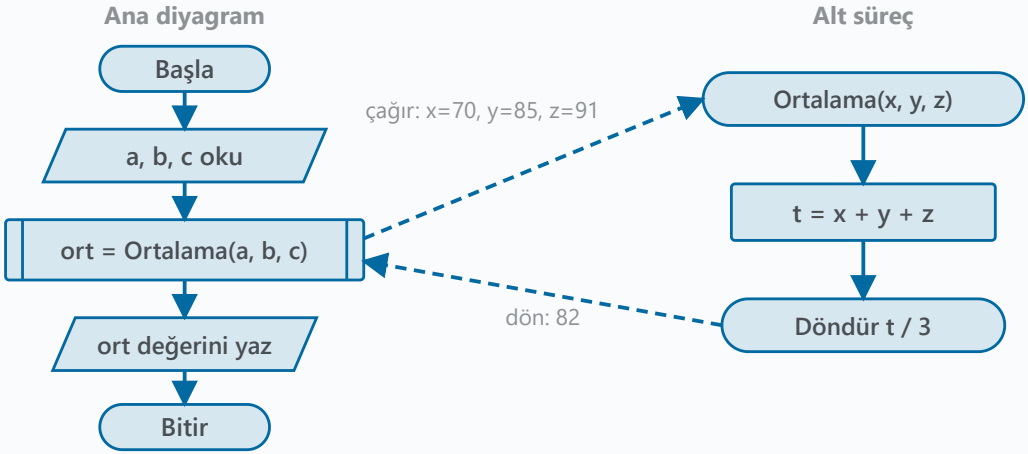
`n = 3` `toplam = 6` `i = 4`

ÇIKTI

```
6
```

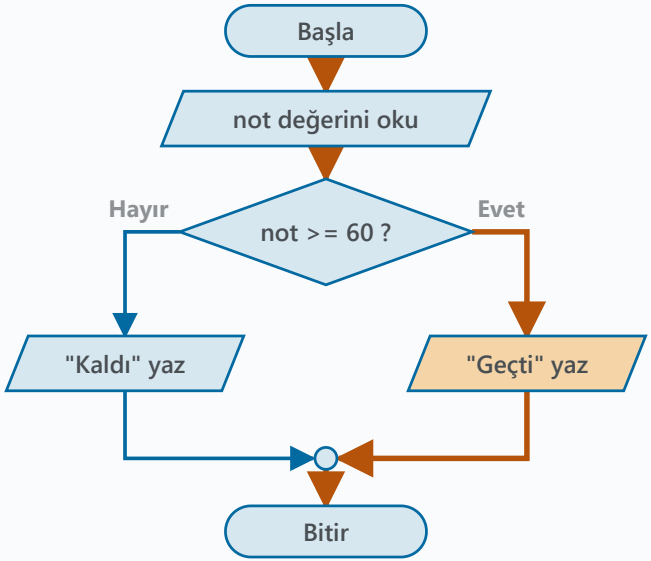
KODA ÇEVİRİ

Alt Süreçle Modüler Diyagram



- **Önceden tanımlı işlem** simgesi ayrı çizilmiş bir diyagramı çağırır; alt süreç bitince akış ana diyagramda **aynı noktadan** sürer.
- Alt diyagram Başla yerine **adı ve parametreleriyle**, Bitir yerine **Döndür** ile biter. a = 70, b = 85, c = 91 için t = 246, dönen değer 82.
- Aynı alt süreç birçok yerden çağrılabilir, bir kez çizilir. Java'da her alt süreç bir **metot** olur: `static int ortalama(int x, int y, int z) { return (x + y + z) / 3; }`

Diyagramı Test Etme

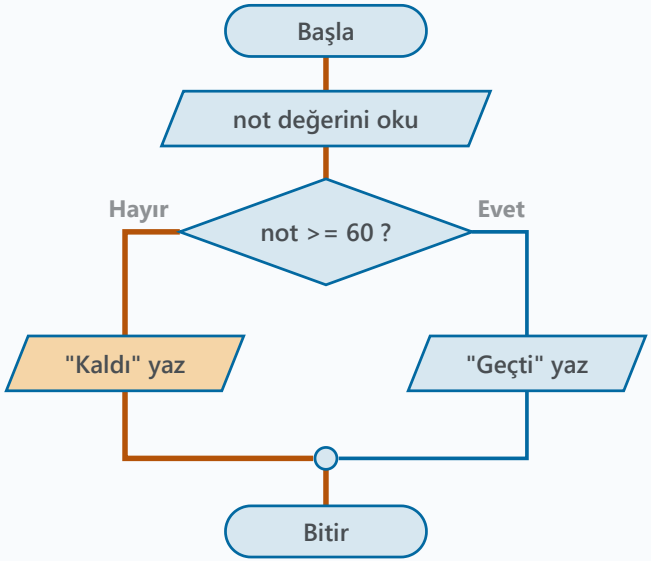


Normal girdi not = 72: Evet yolu çalışır.

Adım 1 / 3

- Normal bir girdide oklar adım adım izlenir.
- Kararın **her çıkışını** çalıştıracak test değeri seçilir.
- Sınır değerleri (60 gibi) ayrıca denenir.
- Döngünün sıfır, bir ve çok tekrar yaptığı durumlar denenir.
- Başlangıca dönmeyen veya bitişe ulaşmayan yollar aranır.

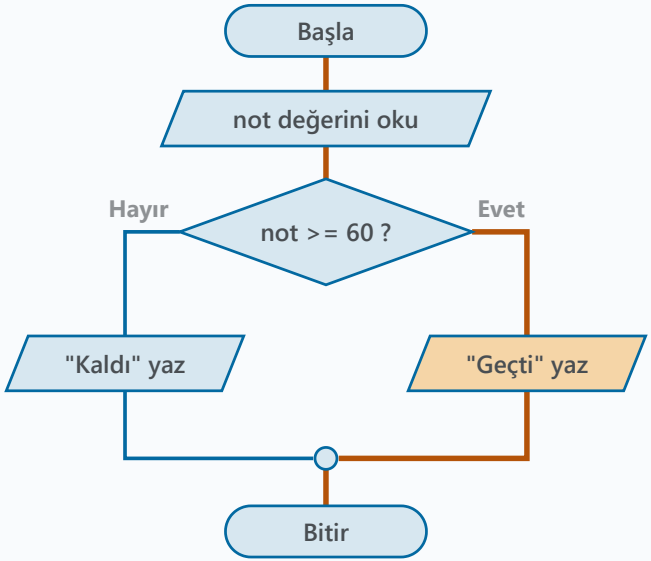
Diyagramı Test Etme



Diğer dal için not = 45: Hayır yolu çalışır. Artık her kutu en az bir kez denendi.

- Normal bir girdide oklar adım adım izlenir.
- Kararın **her çıkışını** çalıştıracak test değeri seçilir.
- Sınır değerleri (60 gibi) ayrıca denenir.
- Döngünün sıfır, bir ve çok tekrar yaptığı durumlar denenir.
- Başlangıca dönmeyen veya bitişe ulaşmayan yollar aranır.

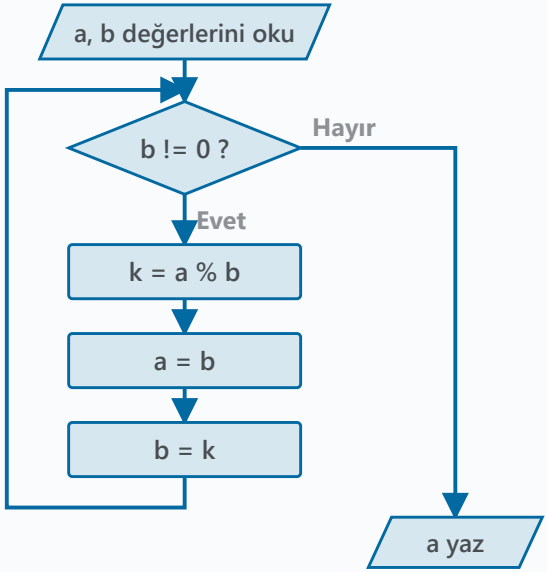
Diyagramı Test Etme



- Normal bir girdide oklar adım adım izlenir.
- Kararın **her çıkışını** çalıştıracak test değeri seçilir.
- Sınır değerleri (60 gibi) ayrıca denenir.
- Döngünün sıfır, bir ve çok tekrar yaptığı durumlar denenir.
- Başlangıca dönmeyen veya bitişe ulaşmayan yollar aranır.

Sınır değeri not = 60: 60 >= 60 doğru → "Geçti". `>` yazılsaydı hata burada yakalanırdı.

İzleme Tablosuyla Elle İzleme: Öklid (48, 18)



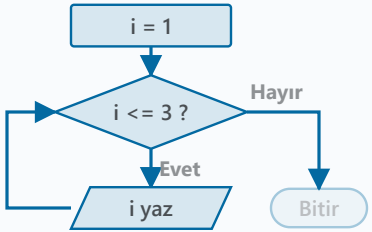
An	a	b	k	b != 0 ?
başlangıç	48	18	—	18 != 0 doğru
1. tur	18	12	12	12 != 0 doğru
2. tur	12	6	6	6 != 0 doğru
3. tur	6	0	0	0 != 0 yanlış

- Her değişken bir sütun, her tur bir satırdır; karar sonucu da yazılır. Sonuç: **6** yazılır, EBOB(48, 18) = 6.
- Tablo, diyagramdaki sıranın önemini gösterir: **a = b** ve **b = k** yer değiştirseydi a, b'nin yeni değerini alırdı ve sonuç bozulurdu.
- Döngü 3 tur döndü, karar 4 kez soruldu: her zamanki gibi karar, gövdeden bir fazla çalışır.

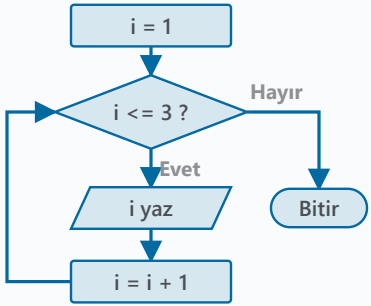
DOĞRULAMA

Sık Hatalar

Hata	Sonuç	Düzeltme
Etiketsiz karar yolu	Hangi koşulun seçildiği belirsiz	Evet/Hayır yaz
Güncellenmeyen sayaç	Sonsuz döngü	Sayaç adımı ekle
Bağlantısız simge	Akış kopar	Okla bağla
Bir simgede çok işlem	Okunabilirlik azalır	Adımlara böl



✗ i hep 1: 1 1 1 1 … Bitir'e hiç ulaşılmaz



✓ i = i + 1 eklendi: 1 2 3, sonra Bitir

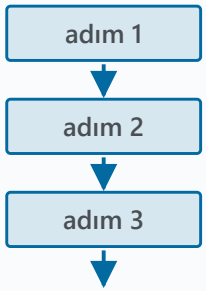
Sık Hatalar (devam)

Hata	Belirti	Düzeltme
Biriktirici döngünün **içinde** sıfırlanıyor	4, 7, 2 için 13 yerine 2 yazılır: yalnız son değer kalır	toplam = 0 kutusunu döngüden önceye al
Geri dönen ok **oku** kutusunu atlıyor	Aynı x tekrar tekrar işlenir; bekçi değer hiç gelmez	Ok, oku kutusunun önüne döner
Karar kutusunda atama: **x = 0 ?**	Soru mu, atama mı belirsiz	Karşılaştırma yaz: x == 0 ?
Bekçi değer işlemden **sonra** sorgulanıyor	−1 toplama katılır, sayaç bir fazla olur	Kararı toplama ve saymadan önce sor
Karar zincirinde geniş koşul önce	Alttaki dallar hiç çalışmaz	Dar koşuldan genişe sırala
Karardan tek ok çıkıyor	Diğer durumda akış kopar	İki çıkışı da çiz, Evet/Hayır yaz
Bölmeden önce sıfır denetimi yok	Hiç veri girilmezse 0'a bölme	sayac > 0 ? kararı ekle

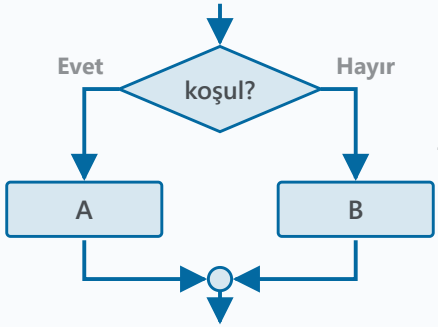
ÖZET

Bölüm Özeti

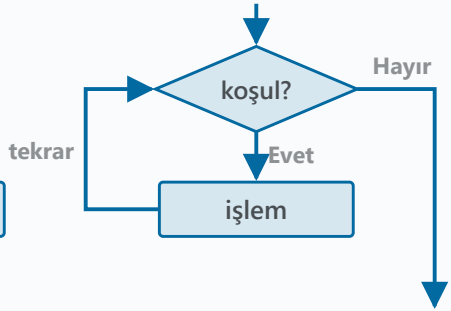
- Akış diyagramı algoritmanın yürütme sırasını standart simgelerle gösterir.
- Sıralı akış doğrusal, koşullu akış dallanan, tekrarlı akış geri dönen yol içerir.
- Karar yolları etiketlenmeli ve her yol bir sonraki adıma ulaşmalıdır.
- Diyagram, farklı test girdileriyle elle izlenerek doğrulanmalıdır.



Sıralı: doğrusal



Koşullu: dallanan



Tekrarlı: geri dönen

ÖZET

Kontrol Soruları

1. İşlem ve karar simgeleri neyi temsil eder?
2. Karar çıkışları neden etiketlenmelidir?
3. Bir döngü diyagramında sonsuz tekrar nasıl oluşur?
4. Sayaç ve biriktirici arasındaki fark nedir?
5. Bir akış diyagramını test etmek için hangi girdiler seçilir?

ÖZET

Alıştırmalar

1. Girilen sayının tek mi çift mi olduğunu yazan akış diyagramını çizin (ipucu: sayi % 2 == 0 ?).
2. Bir sınavın 0–100 arası notunu AA / BA / … / FF harfine çeviren akış diyagramını iç içe kararlarla çizin; 85, 60, 42 için izleyin.
3. 1'den n'ye kadar sayıların **çarpımını** bulan diyagramı çizin; biriktiricinin başlangıç değeri ne olmalı?
4. Kullanıcı 0 girene kadar sayı okuyup toplamı ve kaç sayı girildiğini yazan diyagramı çizin (sayaç + biriktirici).
5. Sayı tahmin oyununa "en fazla 5 deneme" kuralını ekleyin: 5 denemede bilinemezse "Kaybettin" yazılsın.
6. Aşağıdaki hatayı bulun: i = 1, i <= 10 ? Evet → i yaz, karara dön. Nasıl düzeltilir?

▶ **İpuçları**

ÖZET

Kaynaklar

- ISO 5807:1985, Information processing: Documentation symbols and conventions for data, program and system flowcharts.
- Deitel ve Deitel, Java: How to Program, Early Objects, 11. baskı, Pearson, 2018.
- Sedgewick ve Wayne, Computer Science: An Interdisciplinary Approach, Addison-Wesley, 2016: **1.3 Conditionals and Loops**.
- Oracle, The Java Tutorials: **Control Flow Statements**.
- Downey ve Mayfield, Think Java, 2. baskı, O'Reilly, 2019 (**ücretsiz çevrim içi sürüm**).

Son

Akış Diyagramları

Sonraki sunum: **Programlamaya Giriş** · **Çalışma soruları** · Simülatör: **Akış Diyagramı Yürütücü** · **Terim sözlüğü**