Bölüm 1.5: Hesaplama ve Hesaplamalı Düşünme

Bilgisayar ve Programlamaya Giriş

Hesaplama | Bilgisayar tarihi | Turing makinesi | Bilgi ve algoritma | Hesaplamalı düşünme

Sercan KÜLCÜ · Giresun Üniversitesi

© Sercan Külcü. Tüm hakları saklıdır. · Sunumun etkileşimli HTML sürümü:
https://sercankulcu.github.io/files/computer/slides/Bolum_01_05_Hesaplama_ve_Hesaplamali_Dusunme.html

GİRİŞ

Öğrenme Çıktıları

Hesaplama

Verinin kurallara göre işlenmesini açıklamak.

Hesaplama modelleri

Turing makinesinin çalışmasını adım adım izlemek.

Bilgi ve algoritma

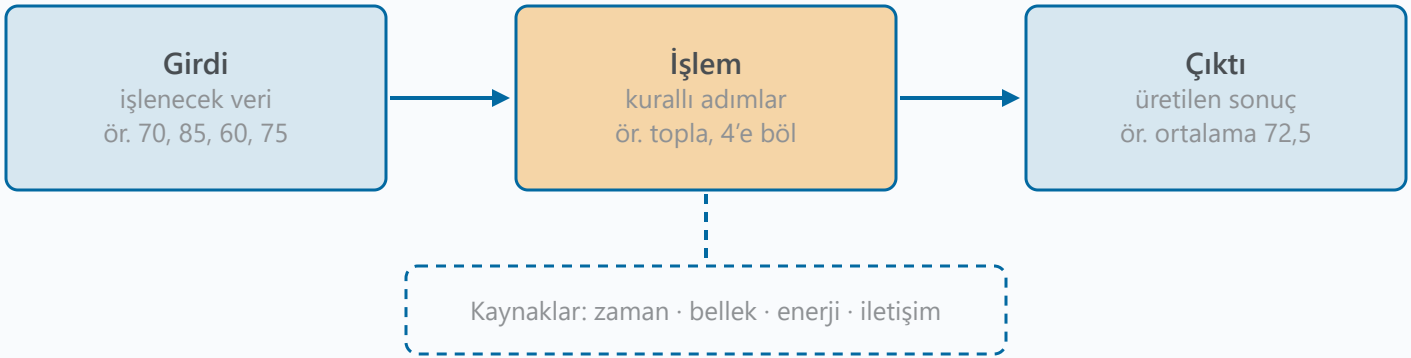
Bildirimsel ve buyurgan bilgiyi ayırt etmek; Öklid algoritmasını izlemek.

Hesaplamalı düşünme

Ayrıştırma, örüntü, soyutlama ve algoritma tasarımını uygulamak.

GİRİŞ

Hesaplama



Hesaplama yalnızca aritmetik değildir. Arama, sıralama, karar verme, dönüştürme ve iletişim de hesaplamadır.

GİRİŞ

Hesaplamanın Yapı Taşları

Matematik
Nicelikler ve ilişkiler üzerinde işlem.

Mantık
Koşulların doğru veya yanlış değerlendirilmesi.

Veri
Problemin temsil edilen girdileri ve ara sonuçları.

Algoritma
İşlemlerin sonlu ve düzenli sırası.

Donanım
Adımları fiziksel olarak yürüten sistem.

Programlama dili
Algoritmayı bilgisayara ifade etme aracı.

BİLGİSAYAR BİLİMİ

Bilgisayar Bilimi

Bilgisayar bilimi yalnızca araç kullanmayı değil, bilginin nasıl temsil edildiğini ve problemlerin nasıl çözülebileceğini inceler.

Soyutlama

Gereksiz ayrıntıları gizleyerek önemli yapıya odaklanmak.

Algoritmalar

Çözüm adımlarını tasarlamak ve analiz etmek.

Sistemler

Donanım ve yazılımın birlikte çalışmasını anlamak.

İnsan ve toplum

Teknolojinin etkilerini, güvenliğini ve etiğini değerlendirmek.

BİLGİSAYAR BİLİMİ

Teknoloji ve Sorumluluk

Erişilebilirlik

Farklı kullanıcıların teknolojiye eşit biçimde erişebilmesi.

Mahremiyet

Kişisel verilerin amaçla sınırlı ve güvenli kullanılması.

Adalet

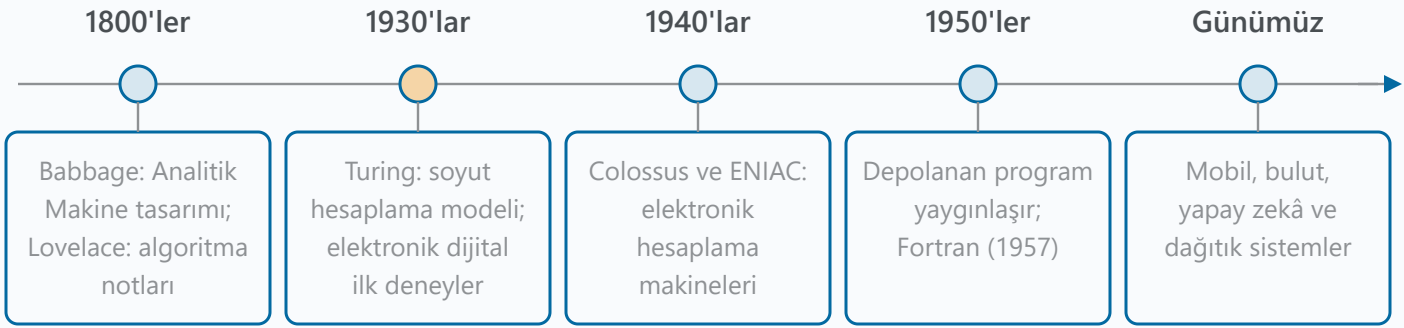
Veri ve algoritmaların ayrımcı sonuç üretmemesi.

Sürdürülebilirlik

Enerji, donanım ömrü ve elektronik atığın dikkate alınması.

TARİH

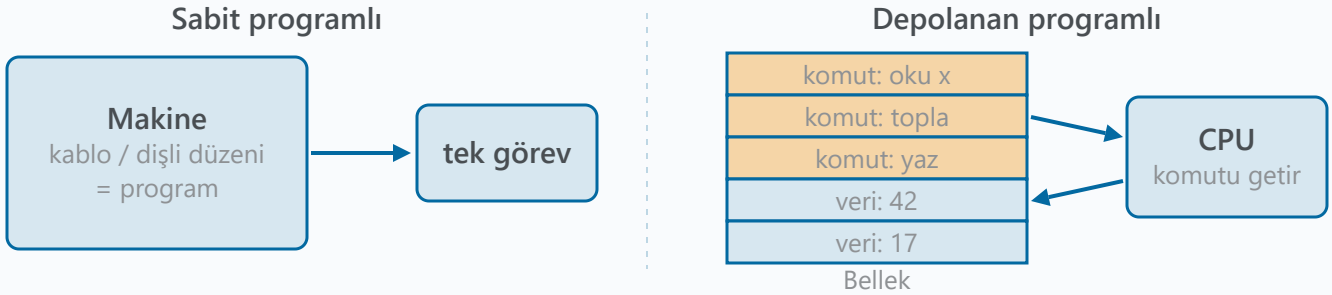
Hesaplama Tarihinde Dönüm Noktaları



Donanım hızlanıp küçüldükçe aynı temel fikirler (algoritma, depolanan program) daha geniş alanlarda uygulanabilir oldu.

TARİH

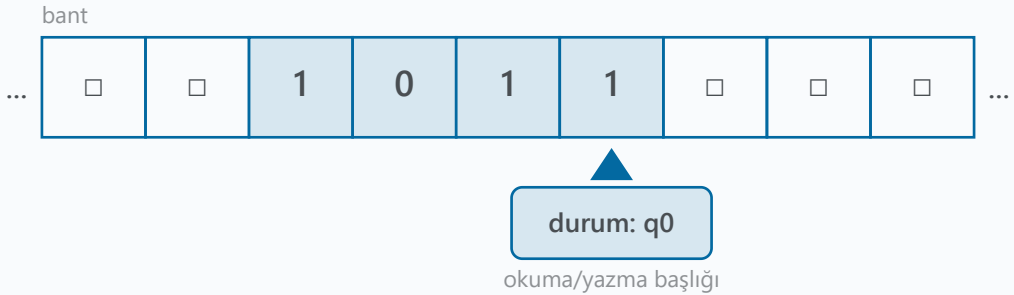
Sabit ve Depolanan Program



Yaklaşım	Program nasıl belirlenir?	Esneklik
Sabit programlı	Makinenin fiziksel düzeni veya özel mekanizması	Belirli görev
Depolanan programlı	Komutlar bellekte veri gibi saklanır	Program değiştirilerek farklı görevler
Genel amaçlı	Çok farklı programları çalıştırabilen mimari	Geniş uygulama alanı

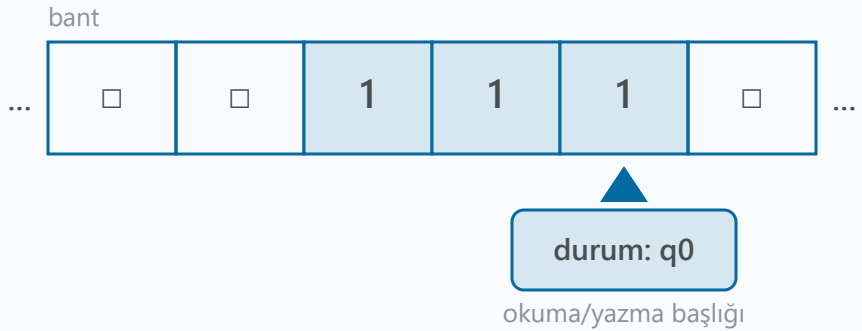
Evrensel Turing Makinesi

Turing makinesi, algoritmik hesaplamayı açıklamak için kullanılan soyut bir modeldir.



- Başlık bir simgeyi okur, yeni simge yazabilir ve sağa veya sola hareket eder.
- Geçiş kuralı mevcut durum ve simgeye göre sonraki adımı belirler.
- **Evrensel** Turing makinesi, bantta verilen başka bir makinenin kurallarını okuyup onu taklit eder: depolanan program fikrinin kuramsal temeli.

Turing Makinesi Adım Adım: 111 + 1



Geçiş kuralları

1) q0, 1 → 0, sola, q0

2) q0, 0 → 1, dur, DUR

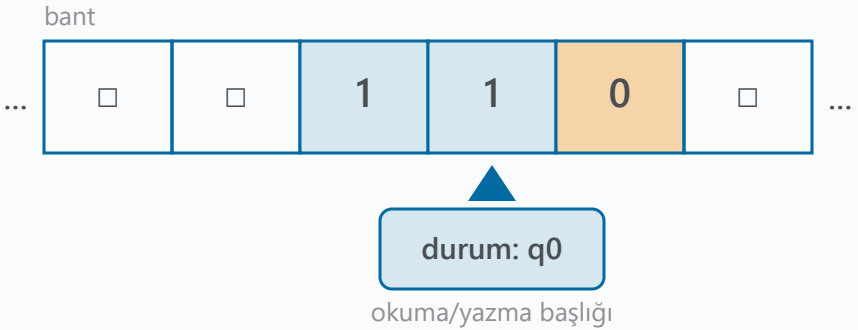
3) q0, □ → 1, dur, DUR

Bantta: 111 (ikili) = 7

(durum, okunan) → yaz, git, yeni durum

Başlangıç: bantta **111** (ikili 7) var, başlık en sağdaki basamakta, durum **q0**. Amaç: sayıya 1 eklemek.

Turing Makinesi Adım Adım: 111 + 1



Geçiş kuralları

1) q0, 1 → 0, sola, q0

2) q0, 0 → 1, dur, DUR

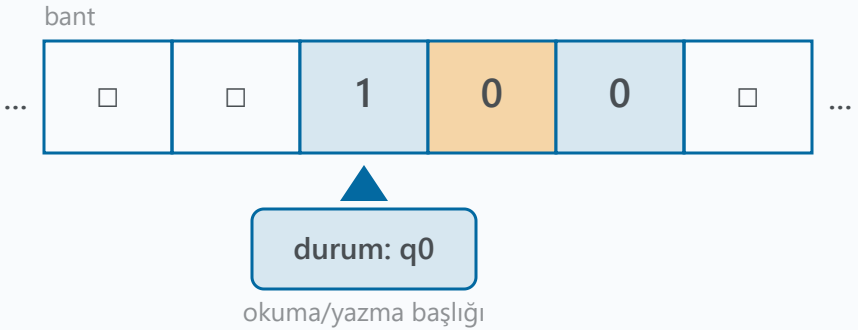
3) q0, □ → 1, dur, DUR

(durum, okunan) → yaz, git, yeni durum

Bantta: 110 (ikili) = 6

Okunan **1**, durum q0 → kural 1: **0** yaz, başlık **sola** kayar (elde 1 devam ediyor).

Turing Makinesi Adım Adım: 111 + 1



Geçiş kuralları

1) q0, 1 → 0, sola, q0

2) q0, 0 → 1, dur, DUR

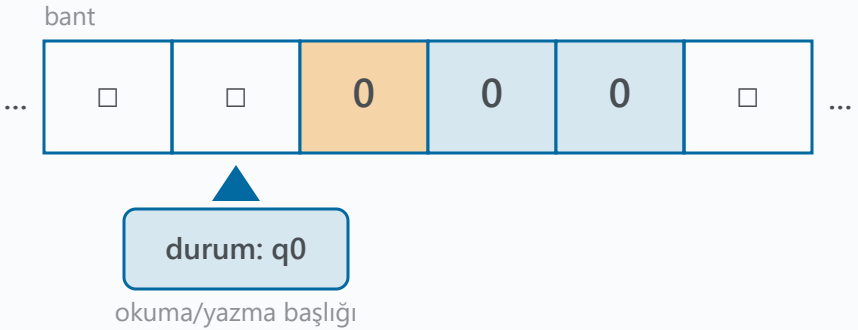
3) q0, □ → 1, dur, DUR

(durum, okunan) → yaz, git, yeni durum

Bantta: 100 (ikili) = 4

Okunan **1**, durum q0 → kural 1: **0** yaz, başlık **sola** kayar (elde 1 devam ediyor).

Turing Makinesi Adım Adım: 111 + 1



Geçiş kuralları

1) q0, 1 → 0, sola, q0

2) q0, 0 → 1, dur, DUR

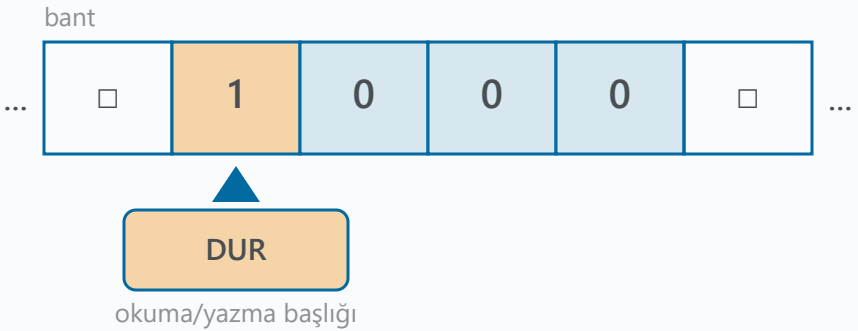
3) q0, □ → 1, dur, DUR

(durum, okunan) → yaz, git, yeni durum

Bantta: 000 (ikili) = 0

Okunan **1**, durum q0 → kural 1: **0** yaz, başlık **sola** kayar (elde 1 devam ediyor).

Turing Makinesi Adım Adım: 111 + 1



Geçiş kuralları

1) q0, 1 → 0, sola, q0

2) q0, 0 → 1, dur, DUR

3) q0, □ → 1, dur, DUR

(durum, okunan) → yaz, git, yeni durum

Bantta: 1000 (ikili) = 8

Okunan □, durum q0 → kural 3: **1** yaz, makine **durur**. Bant: **1000** = 8. Her adımı yalnızca (durum, okunan simge) belirledi.

Bilgisayarların Sınırları

Zaman
Büyük girdiler bazı algoritmaları pratik olmaktan çıkarabilir.

Bellek
Veri ve ara sonuçlar sonlu depolama kullanır.

Enerji
Her işlem ve veri aktarımı enerji tüketir.

Hesaplanabilirlik
Bazı problemler için her durumda sonuç veren algoritma yoktur (ör. durma problemi).

Veri kalitesi
Eksik veya hatalı girdi doğru sonucu engeller.

Model sınırı
Program gerçek dünyanın yalnızca seçilmiş özelliklerini temsil eder.

Bilgi Türleri

Bildirimsel bilgi

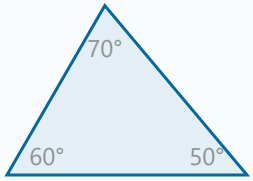
Neyin doğru olduğunu veya bir kavramın ne olduğunu açıklar.

```
Bir üçgenin iç açıları toplamı 180°'dir.
```

Buyurgan (emirsel) bilgi

Bir sonuca nasıl ulaşılacağını adımlarla açıklar.

```
Bir sayının karekökünü yaklaşık hesapla.
```

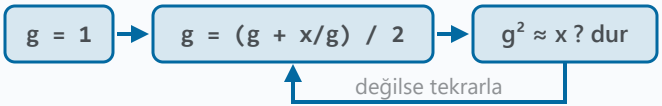


Bildirimsel: ne doğru?

60° + 70° + 50° = 180°

olgu bildirir; yöntem vermez

Buyurgan: nasıl bulunur?



x = 25: 1 → 13 → 7,46 → 5,41 → 5,02 → 5,00

Algoritmalar buyurgan bilgiyi açık, uygulanabilir ve sonlu adımlara dönüştürür.

Buyurgan Bilgi: Karekökü Yaklaşık Bulma (x = 25)



Soluk noktalar: önceki tahminler. Her tur tahmini √x'e yaklaştırır.

Karekökü aranan sayı **x = 25**.

Adım 1 / 14

```
x = 25
tahmin = 1
while |tahmin * tahmin - x| > 0.01
    tahmin = (tahmin + x / tahmin) / 2
yaz(tahmin)
```

DEĞİŞKENLER

x = **25**

ÇIKTI

Buyurgan Bilgi: Kareköku Yaklaşık Bulma (x = 25)



Soluk noktalar: önceki tahminler. Her tur tahmini √x'e yaklaştırır.

Kaba bir başlangıç tahmini: **tahmin = 1**.

Adım 2 / 14

```
1 x = 25
▶ tahmin = 1
3 while |tahmin * tahmin - x| > 0.01
4     tahmin = (tahmin + x / tahmin) / 2
5 yaz(tahmin)
```

DEĞİŞKENLER

`x = 25` `tahmin = 1`

ÇIKTI

Buyurgan Bilgi: Kareköku Yaklaşık Bulma (x = 25)



Soluk noktalar: önceki tahminler. Her tur tahmini √x'e yaklaştırır.

Hata $|1^2 - 25| = 24 \rightarrow$ **> 0,01: devam**.

Adım 3 / 14

```
1 x = 25
2 tahmin = 1
▶ while |tahmin * tahmin - x| > 0.01
4     tahmin = (tahmin + x / tahmin) / 2
5 yaz(tahmin)
```

DEĞİŞKENLER

`x = 25` `tahmin = 1`

ÇIKTI

Buyurgan Bilgi: Kareköku Yaklaşık Bulma (x = 25)



Soluk noktalar: önceki tahminler. Her tur tahmini √x'e yaklaştırır.

1. tur: tahmin = (1 + 25 / 1) / 2 = **13**. Tahmin ile x/tahmin'in ortalaması alınır.

Adım 4 / 14

```
1 x = 25
2 tahmin = 1
3 while |tahmin * tahmin - x| > 0.01
▶     tahmin = (tahmin + x / tahmin) / 2
5 yaz(tahmin)
```

DEĞİŞKENLER

`x = 25` `tahmin = 13`

ÇIKTI

Buyurgan Bilgi: Kareköku Yaklaşık Bulma (x = 25)



Soluk noktalar: önceki tahminler. Her tur tahmini √x'e yaklaştırır.

Hata $|13^2 - 25| = 144 \rightarrow$ **> 0,01: devam**.

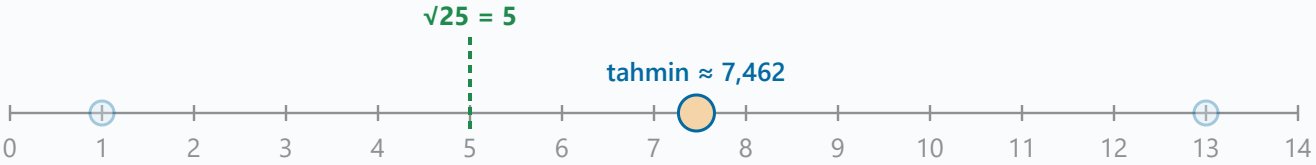

```
1 x = 25
2 tahmin = 1
▶ while |tahmin * tahmin - x| > 0.01
4     tahmin = (tahmin + x / tahmin) / 2
5 yaz(tahmin)
```

DEĞİŞKENLER

`x = 25` `tahmin = 13`

ÇIKTI

Buyurgan Bilgi: Kareköku Yaklaşık Bulma (x = 25)



Soluk noktalar: önceki tahminler. Her tur tahmini √x'e yaklaştırır.

2. tur: tahmin = (13 + 25 / 13) / 2 ≈ **7,462**. Tahmin ile x/tahmin'in ortalaması alınır.

Adım 6 / 14

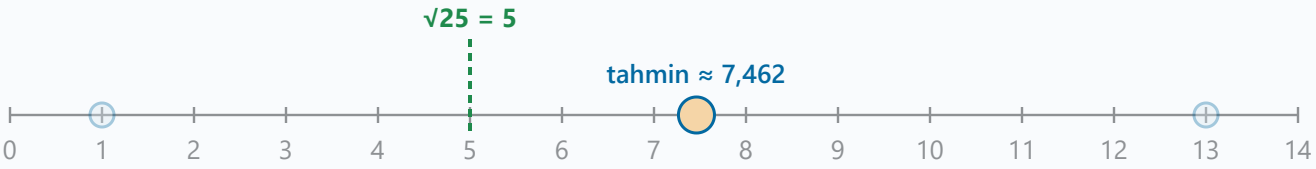
```
1 x = 25
2 tahmin = 1
3 while |tahmin * tahmin - x| > 0.01
▶     tahmin = (tahmin + x / tahmin) / 2
5 yaz(tahmin)
```

DEĞİŞKENLER

`x = 25` `tahmin = 7.4615`

ÇIKTI

Buyurgan Bilgi: Kareköku Yaklaşık Bulma (x = 25)



Soluk noktalar: önceki tahminler. Her tur tahmini √x'e yaklaştırır.

Hata $|7{,}462^2 - 25| \approx 30{,}67$ → **> 0,01: devam**.

Adım 7 / 14

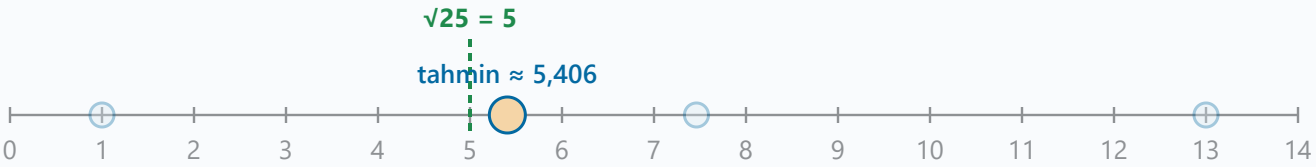
```
1 x = 25
2 tahmin = 1
▶ while |tahmin * tahmin - x| > 0.01
4     tahmin = (tahmin + x / tahmin) / 2
5 yaz(tahmin)
```

DEĞİŞKENLER

`x = 25` `tahmin = 7.4615`

ÇIKTI

Buyurgan Bilgi: Kareköku Yaklaşık Bulma (x = 25)



Soluk noktalar: önceki tahminler. Her tur tahmini √x'e yaklaştırır.

3. tur: tahmin = (7,462 + 25 / 7,462) / 2 ≈ **5,406**. Tahmin ile x/tahmin'in ortalaması alınır.

Adım 8 / 14

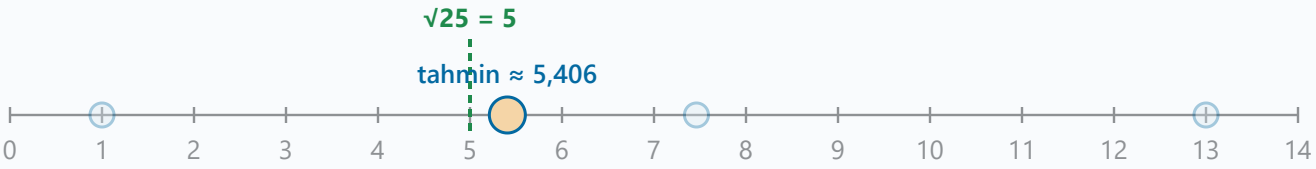
```
1 x = 25
2 tahmin = 1
3 while |tahmin * tahmin - x| > 0.01
▶     tahmin = (tahmin + x / tahmin) / 2
5 yaz(tahmin)
```

DEĞİŞKENLER

`x = 25` `tahmin = 5.406`

ÇIKTI

Buyurgan Bilgi: Kareköku Yaklaşık Bulma (x = 25)



Soluk noktalar: önceki tahminler. Her tur tahmini √x'e yaklaştırır.

Hata $|5{,}406^2 - 25| \approx 4{,}225 \rightarrow$ **> 0,01: devam**.

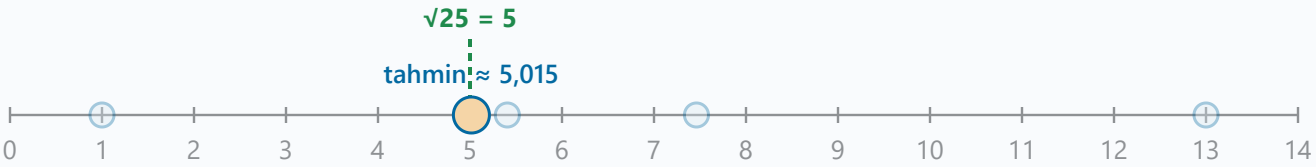

```
1 x = 25
2 tahmin = 1
▶ while |tahmin * tahmin - x| > 0.01
4     tahmin = (tahmin + x / tahmin) / 2
5 yaz(tahmin)
```

DEĞİŞKENLER

`x = 25` `tahmin = 5.406`

ÇIKTI

Buyurgan Bilgi: Kareköku Yaklaşık Bulma (x = 25)



Soluk noktalar: önceki tahminler. Her tur tahmini √x'e yaklaştırır.

4. tur: tahmin = (5,406 + 25 / 5,406) / 2 ≈ **5,015**. Tahmin ile x/tahmin'in ortalaması alınır.

Adım 10 / 14

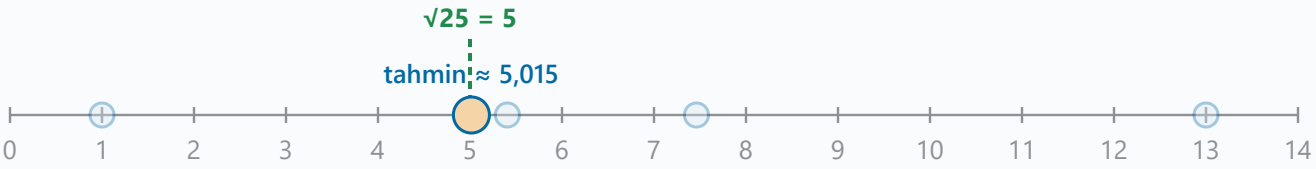
```
1 x = 25
2 tahmin = 1
3 while |tahmin * tahmin - x| > 0.01
▶     tahmin = (tahmin + x / tahmin) / 2
5 yaz(tahmin)
```

DEĞİŞKENLER

`x = 25` `tahmin = 5.0152`

ÇIKTI

Buyurgan Bilgi: Kareköku Yaklaşık Bulma (x = 25)



Soluk noktalar: önceki tahminler. Her tur tahmini √x'e yaklaştırır.

Hata $|5{,}015^2 - 25| \approx 0{,}1527$ → **> 0,01: devam**.

Adım 11 / 14

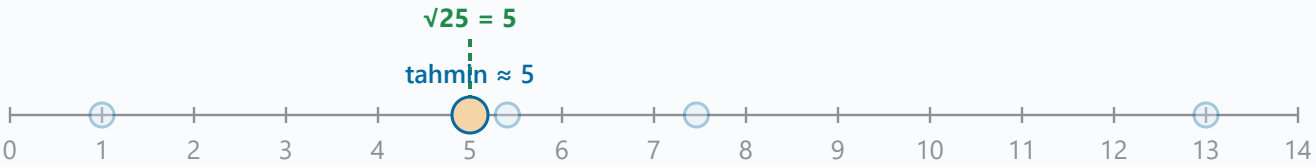
```
1 x = 25
2 tahmin = 1
▶ while |tahmin * tahmin - x| > 0.01
4     tahmin = (tahmin + x / tahmin) / 2
5 yaz(tahmin)
```

DEĞİŞKENLER

`x = 25` `tahmin = 5.0152`

ÇIKTI

Buyurgan Bilgi: Kareköku Yaklaşık Bulma (x = 25)



Soluk noktalar: önceki tahminler. Her tur tahmini √x'e yaklaştırır.

5. tur: tahmin = (5,015 + 25 / 5,015) / 2 ≈ **5,00002**. Tahmin ile x/tahmin'in ortalaması alınır.

Adım 12 / 14

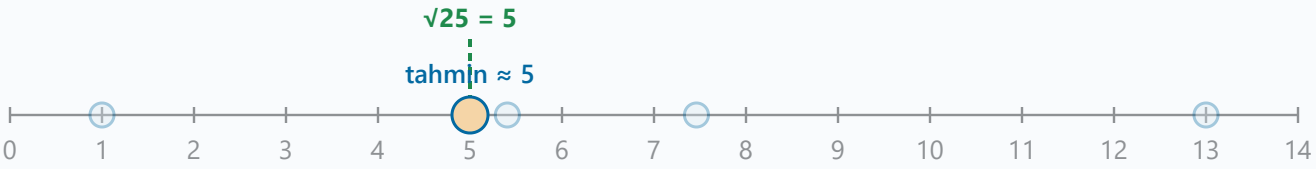
```
1 x = 25
2 tahmin = 1
3 while |tahmin * tahmin - x| > 0.01
▶     tahmin = (tahmin + x / tahmin) / 2
5 yaz(tahmin)
```

DEĞİŞKENLER

`x = 25` `tahmin = 5`

ÇIKTI

Buyurgan Bilgi: Kareköku Yaklaşık Bulma (x = 25)



Soluk noktalar: önceki tahminler. Her tur tahmini √x'e yaklaştırır.

Hata $|5{,}00002^2 - 25| \approx 0{,}00023 \rightarrow$ **≤ 0,01: yeterince yakın, dur**.

Adım 13 / 14

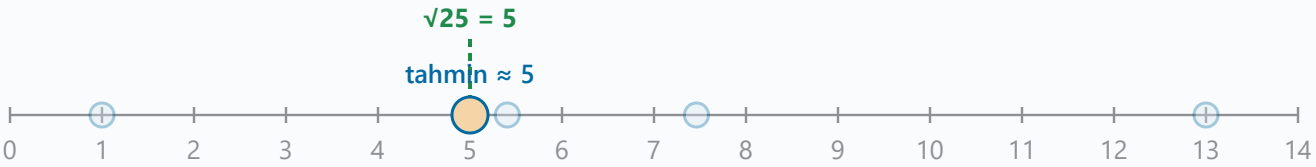
```
1 x = 25
2 tahmin = 1
▶ while |tahmin * tahmin - x| > 0.01
4     tahmin = (tahmin + x / tahmin) / 2
5 yaz(tahmin)
```

DEĞİŞKENLER

`x = 25` `tahmin = 5`

ÇIKTI

Buyurgan Bilgi: Kareköku Yaklaşık Bulma (x = 25)



Soluk noktalar: önceki tahminler. Her tur tahmini √x'e yaklaştırır.

Sonuç yazılır: ≈ **5,0000**. 5 turda gerçek değere (5) çok yaklaştı: "nasıl bulunur" bilgisi = **buyurgan bilgi**.

Adım 14 / 14

```
1 x = 25
2 tahmin = 1
3 while |tahmin * tahmin - x| > 0.01
4     tahmin = (tahmin + x / tahmin) / 2
▶ yaz(tahmin)
```

DEĞİŞKENLER

`x = 25` `tahmin = 5`

ÇIKTI

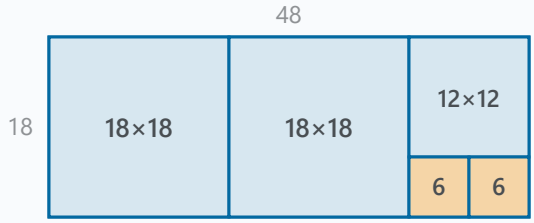
```
5.0000
```

BİLGİ VE ALGORİTMA

Öklid Algoritması

Algoritma, bir problemi sonlu ve açık adımlarla çözen yöntemdir. Bilinen en eski örneklerden biri Öklid'in (MÖ 300 dolayları) algoritmasıdır: iki pozitif tam sayının en büyük ortak bölenini tekrar eden kalan işlemleriyle bulur.

```
GCD(a, b)
while b ≠ 0
    r = a mod b
    a = b
    b = r
return a
```



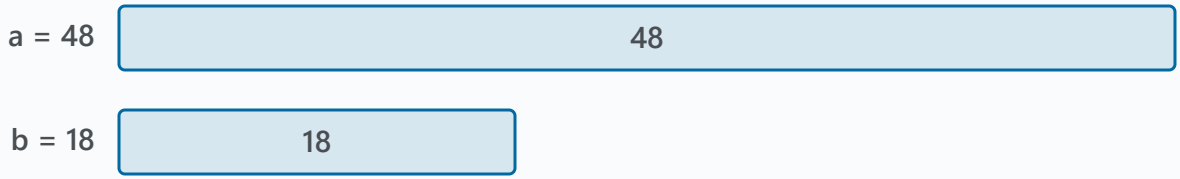
48 = 2×18 + 12 ; 18 = 1×12 + 6 ; 12 = 2×6

Her şeyi kalansız döşeyen kare: 6×6

GCD(48, 18): 48 mod 18 = 12 · 18 mod 12 = 6 · 12 mod 6 = 0 · sonuç 6

BİLGİ VE ALGORİTMA · ADIM ADIM

Öklid Adım Adım: GCD(48, 18)



GCD(48, 18) çağrılır: a = 48, b = 18. Çubuklar sayıların uzunluğudur.

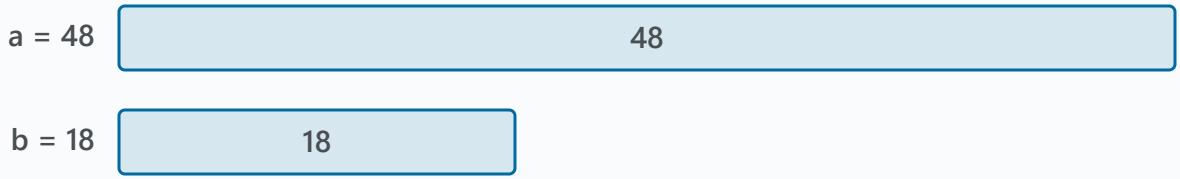
Adım 1 / 15

```
▶ GCD(a, b)
2 while b ≠ 0
3     r = a mod b
4     a = b
5     b = r
6 return a
```

DEĞİŞKENLER

a = 48 b = 18

Öklid Adım Adım: GCD(48, 18)



b ≠ 0 → 18 ≠ 0 → **doğru**: 1. tur başlar.

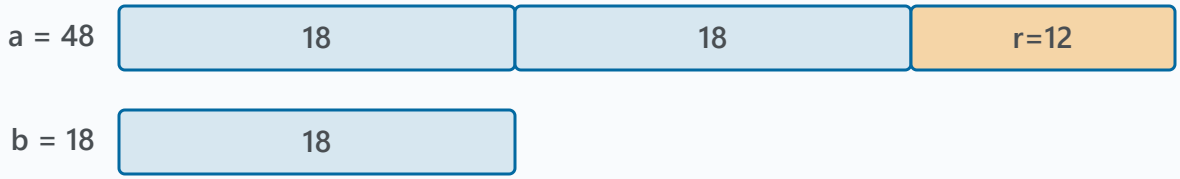
Adım 2 / 15

```
1 GCD(a, b)
▶ while b ≠ 0
3     r = a mod b
4     a = b
5     b = r
6 return a
```

DEĞİŞKENLER

a = 48 b = 18

Öklid Adım Adım: GCD(48, 18)



48 = 2 × 18 + 12 → kalan r = 12

r = a mod b = 48 mod 18 = **12**: a'dan b uzunluğunda 2 parça çıkar, 12 kalır.

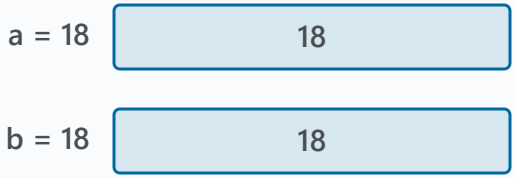
Adım 3 / 15

```
1 GCD(a, b)
2 while b ≠ 0
▶     r = a mod b
4     a = b
5     b = r
6 return a
```

DEĞİŞKENLER

`a = 48` `b = 18` `r = 12`

Öklid Adım Adım: GCD(48, 18)



a = b → a = 18. Artık küçük sayı bölünen olur.

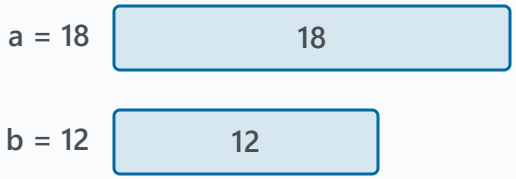
Adım 4 / 15

```
1 GCD(a, b)
2 while b ≠ 0
3     r = a mod b
▶     a = b
5     b = r
6 return a
```

DEĞİŞKENLER

a = 18 | b = 18 | r = 12

Öklid Adım Adım: GCD(48, 18)



b = r → b = 12. Ortak bölen, kalanı da böler; problem küçüldü.

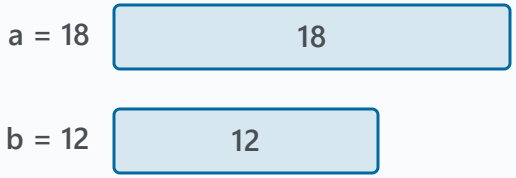
Adım 5 / 15

```
1 GCD(a, b)
2 while b ≠ 0
3     r = a mod b
4     a = b
▶     b = r
6 return a
```

DEĞİŞKENLER

a = 18 | b = 12 | r = 12

Öklid Adım Adım: GCD(48, 18)



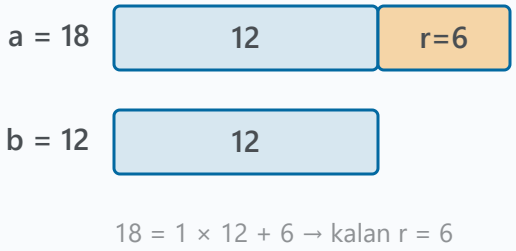
b ≠ 0 → 12 ≠ 0 → **doğru**: 2. tur başlar.


```
1 GCD(a, b)
▶ while b ≠ 0
3     r = a mod b
4     a = b
5     b = r
6 return a
```

DEĞİŞKENLER

`a = 18` `b = 12` `r = 12`

Öklid Adım Adım: GCD(48, 18)



18 = 1 × 12 + 6 → kalan r = 6

r = a mod b = 18 mod 12 = **6**: a'dan b uzunluğunda 1 parça çıkar, 6 kalır.

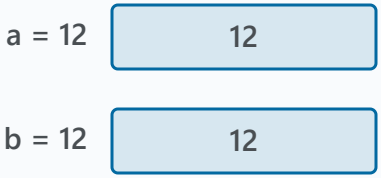
Adım 7 / 15

```
1 GCD(a, b)
2 while b ≠ 0
▶     r = a mod b
4     a = b
5     b = r
6 return a
```

DEĞİŞKENLER

a = 18 | b = 12 | r = 6

Öklid Adım Adım: GCD(48, 18)



a = b → a = 12. Artık küçük sayı bölünen olur.

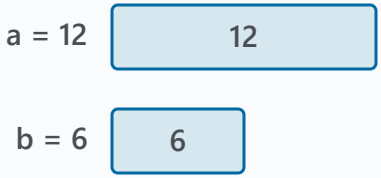
Adım 8 / 15

```
1 GCD(a, b)
2 while b ≠ 0
3     r = a mod b
▶     a = b
5     b = r
6 return a
```

DEĞİŞKENLER

a = **12** b = **12** r = **6**

Öklid Adım Adım: GCD(48, 18)



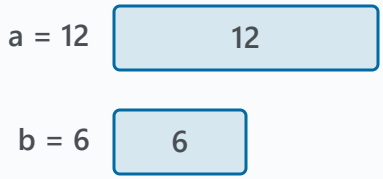
b = r → b = 6. Ortak bölen, kalanı da böler; problem küçüldü.


```
1 GCD(a, b)
2 while b ≠ 0
3     r = a mod b
4     a = b
▶     b = r
6 return a
```

DEĞİŞKENLER

a = 12 | b = 6 | r = 6

Öklid Adım Adım: GCD(48, 18)



b ≠ 0 → 6 ≠ 0 → **doğru**: 3. tur başlar.

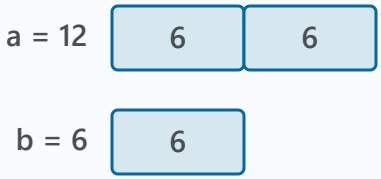
Adım 10 / 15

```
1 GCD(a, b)
▶ while b ≠ 0
3     r = a mod b
4     a = b
5     b = r
6 return a
```

DEĞİŞKENLER

a = 12 | b = 6 | r = 6

Öklid Adım Adım: GCD(48, 18)



12 = 2 × 6 + 0 → kalan r = 0

r = a mod b = 12 mod 6 = **0**: a'dan b uzunluğunda 2 parça çıkar, 0 kalır.

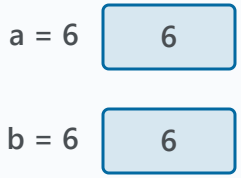
Adım 11 / 15

```
1 GCD(a, b)
2 while b ≠ 0
▶     r = a mod b
4     a = b
5     b = r
6 return a
```

DEĞİŞKENLER

`a = 12` `b = 6` `r = 0`

Öklid Adım Adım: GCD(48, 18)



a = b → a = 6. Artık küçük sayı bölünen olur.

Adım 12 / 15

```
1 GCD(a, b)
2 while b ≠ 0
3     r = a mod b
▶     a = b
5     b = r
6 return a
```

DEĞİŞKENLER

a = 6 | b = 6 | r = 0

Öklid Adım Adım: GCD(48, 18)

a = 6 **6**

b = 0 (0: bölünecek bir şey kalmadı)

b = r → b = 0. Ortak bölen, kalanı da böler; problem küçüldü.


```
1 GCD(a, b)
2 while b ≠ 0
3     r = a mod b
4     a = b
▶     b = r
6 return a
```

DEĞİŞKENLER

a = 6 | b = 0 | r = 0

Öklid Adım Adım: GCD(48, 18)

a = 6 **6**

b = 0 (0: bölünecek bir şey kalmadı)

b ≠ 0 → 0 ≠ 0 → **yanlış**: döngü biter.

Adım 14 / 15

```
1 GCD(a, b)
▶ while b ≠ 0
3     r = a mod b
4     a = b
5     b = r
6 return a
```

DEĞİŞKENLER

a = 6 | b = 0 | r = 0

BİLGİ VE ALGORİTMA · ADIM ADIM

Öklid Adım Adım: GCD(48, 18)

a = 6 **6**

b = 0 (0: bölünecek bir şey kalmadı)

return a → 6. GCD(48, 18) = 6: 48 ve 18, 6 uzunluğundaki parçalarla tam ölçülür. 3 turda bitti.

Adım 15 / 15

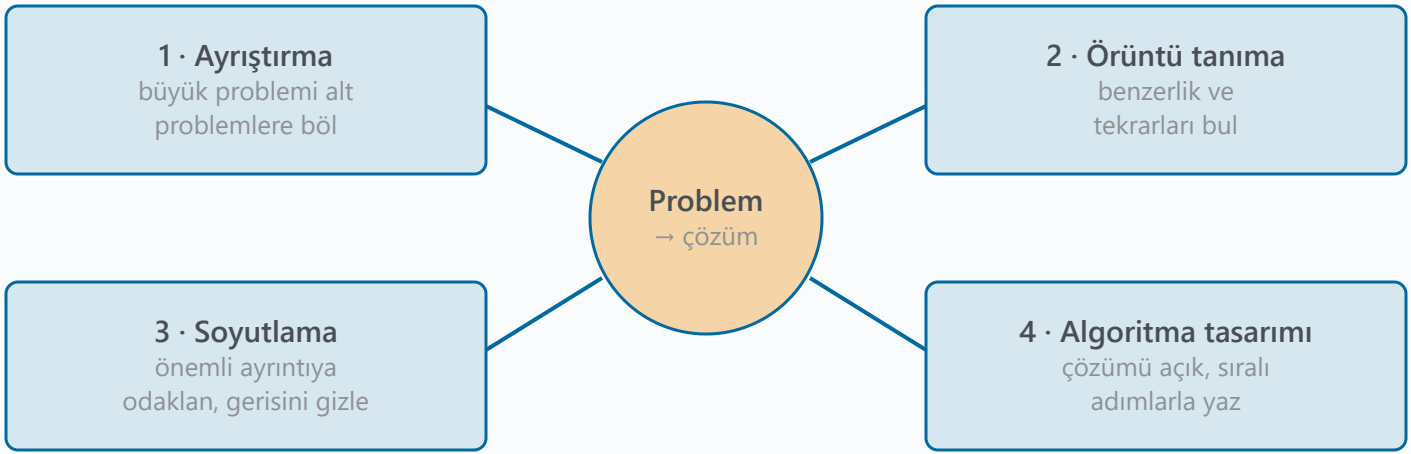
```
1 GCD(a, b)
2 while b ≠ 0
3     r = a mod b
4     a = b
5     b = r
▶ return a
```

DEĞİŞKENLER

a = 6 | b = 0 | r = 0

Hesaplamalı Düşünme

Hesaplamalı düşünme, problemleri bilgisayarın veya insanın sistemli biçimde çözebileceği yapılara dönüştürme yaklaşımıdır.



Ayrıştırma

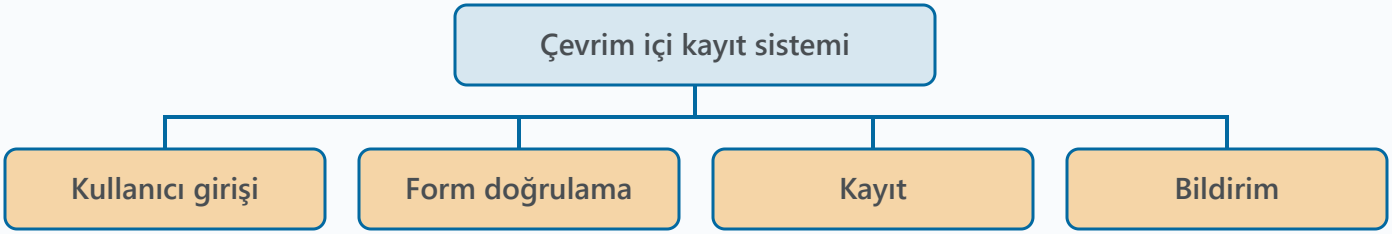
Çevrim içi kayıt sistemi

Büyük problem: bir **çevrim içi kayıt sistemi**. Tek parça olarak ele almak zor.

Adım 1 / 3

- Her alt problem ayrı geliştirilebilir ve test edilebilir.
- Alt problemlerin girdi ve çıktıları açıkça tanımlanır.
- Bağımlılıklar erken fark edilir.

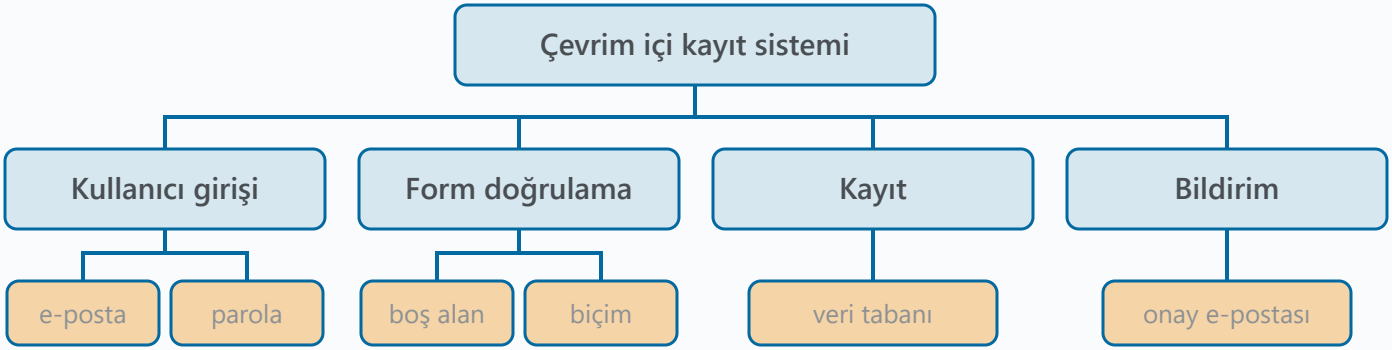
Ayrıştırma



Ana alt görevlere bölünür; her birinin girdisi ve çıktısı ayrı tanımlanır.

- Her alt problem ayrı geliştirilebilir ve test edilebilir.
- Alt problemlerin girdi ve çıktıları açıkça tanımlanır.
- Bağımlılıklar erken fark edilir.

Ayrıştırma



Gerekirse daha da bölünür. Her yaprak küçük, tek başına yazılıp test edilebilir bir iştir.

- Her alt problem ayrı geliştirilebilir ve test edilebilir.
- Alt problemlerin girdi ve çıktıları açıkça tanımlanır.
- Bağımlılıklar erken fark edilir.

HESAPLAMALI DÜŞÜNME

Örüntü ve Soyutlama

Öğrenci
ad
kimlik no
e-posta
öğrenci no

Personel
ad
kimlik no
e-posta
birim

Ziyaretçi
ad
kimlik no
e-posta
ziyaret tarihi

Üç farklı kayıt türü: öğrenci, personel ve ziyaretçi.

Adım 1 / 3

İyi soyutlama, önemli farkları kaybetmeden tekrar eden yapıyı ortaklaştırır.

Örüntü ve Soyutlama

Öğrenci
ad
kimlik no
e-posta
öğrenci no

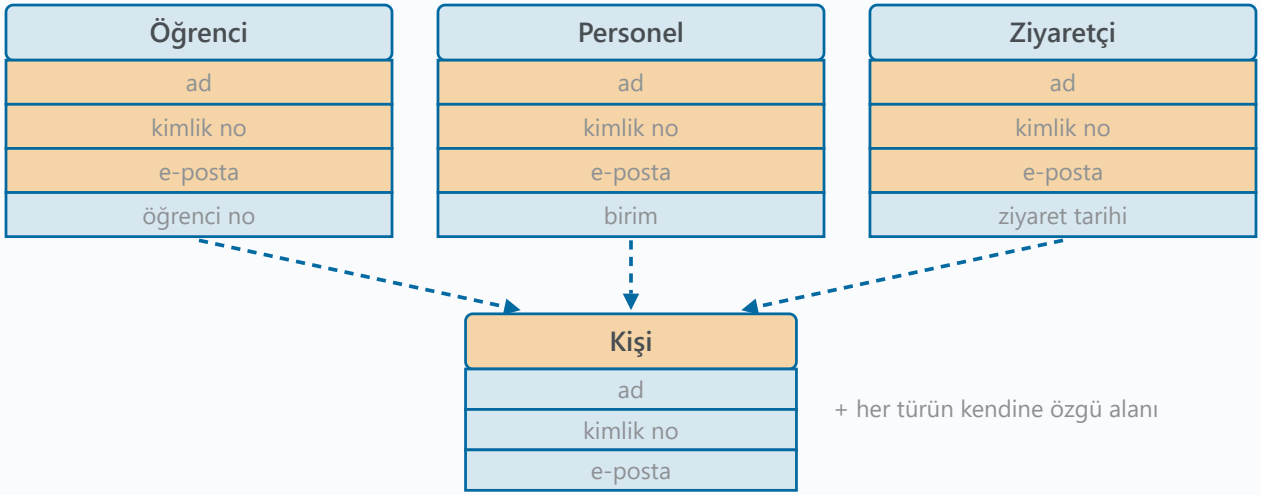
Personel
ad
kimlik no
e-posta
birim

Ziyaretçi
ad
kimlik no
e-posta
ziyaret tarihi

Örüntü tanıma: ad, kimlik no ve e-posta alanları üçünde de tekrar ediyor.

İyi soyutlama, önemli farkları kaybetmeden tekrar eden yapıyı ortaklaştırır.

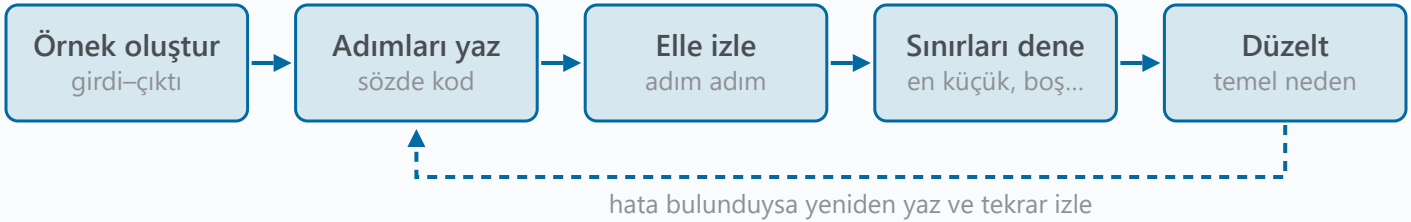
Örüntü ve Soyutlama



Soyutlama: ortak alanlar tek bir `Kişi` modelinde toplanır; farklı ayrıntılar yalnızca gereken türde kalır.

İyi soyutlama, önemli farkları kaybetmeden tekrar eden yapıyı ortaklaştırır.

Algoritma Tasarımı ve Test



- Normal, en küçük, en büyük ve geçersiz girdiler düşünülür.
- Her adımın beklenen durumu ürettiği izlenir.
- Hata bulunduğunda yalnızca belirti değil, temel neden düzeltilir.

ÖZET

Bölüm Özeti

- Hesaplama, verinin kurallara göre işlenmesidir; aritmetikle sınırlı değildir.
- Depolanan program fikri, komutların bellekte veri gibi saklanmasıyla genel amaçlı bilgisayarı mümkün kıldı.
- Turing makinesi algoritmik hesaplamanın soyut modelidir; her adımı durum ve okunan simge belirler.
- Buyurgan bilgi bir sonuca nasıl ulaşılacağını anlatır; Öklid algoritması bunun tarihsel örneğidir.
- Hesaplamalı düşünme; ayrıştırma, örüntü tanıma, soyutlama ve algoritma tasarımını birleştirir.
- Zaman, bellek, enerji ve hesaplanabilirlik bilgisayarların sınırlarını oluşturur.

Kontrol Soruları

1. Hesaplama neden yalnızca aritmetik değildir?
2. Bildirimsel ve buyurgan bilgiye birer örnek verin.
3. Sabit programlı ve depolanan programlı makine nasıl ayrılır?
4. Turing makinesinde bir sonraki adımı ne belirler?
5. Öklid algoritması neden her zaman durur?
6. Bir problemi hesaplamalı düşünmenin dört bileşeniyle nasıl ele alırsınız?

Alıştırmalar

1. **GCD(84, 36)**'yı Öklid algoritmasıyla izleyin; her turda a, b, r değerlerini tabloya yazın.
2. Hesap makinesi, çamaşır makinesi kumandası ve akıllı telefonu **sabit** ya da **depolanan programlı** olarak sınıflandırın; gerekçesini yazın.
3. Bir metro haritası gerçek şehrin hangi ayrıntılarını gizler, hangilerini korur? Bu bir soyutlama olarak neden işe yarar?
4. Sayıya 1 ekleyen Turing makinesini bantta **1011** ile (başlık en sağda) çalıştırın; bant hangi değerle durur?
5. Karekök yönteminde x = 10 ve tahmin = 1 ile üç tur yapın; tahmin hangi değere yaklaşıyor?
6. Bir **kütüphane ödünç alma sistemini** ayrıştırın: en az dört alt görev ve her birinin girdi/çıktısı; hangi alanlar üye ve personel kayıtlarında ortaktır?

Kaynaklar

- Wing, Computational Thinking, Communications of the ACM, 2006.
- Cormen ve diğerleri, Introduction to Algorithms.
- Downey, Think Java.
- Sipser, Introduction to the Theory of Computation.

Son

Hesaplama ve Hesaplamalı Düşünme

Sonraki sunum: Yazılım ve İşletim Sistemleri · Çalışma soruları · Simülatör: Turing Makinesi · Terim sözlüğü