Bölüm 1.2: Hesaplama ve Hesaplamalı Düşünme

Java ile Nesne Yönelimli Programlama

Programlama ve hesaplama | Bilgisayar bilimi | Hesaplamalı düşünme

Bilgisayarın tarihi | Turing makineleri | Bilgisayarların sınırları

Bildirimsel ve buyurusal bilgi | Algoritma ve problem | Sıralı, koşullu, döngülü işlemler

Sercan KÜLCÜ · Giresun Üniversitesi · Elektrik-Elektronik Mühendisliği

GİRİŞ

Öğrenme Çıktıları

Temel kavramlar
Programlama, hesaplama ve bilgisayar bilimini tanımlamak; hesaplamalı düşünmenin dört becerisini (ayrıştırma, örüntü tanıma, soyutlama, algoritma tasarımı) açıklamak.

Tarihsel gelişim
Sabit ve depolanan programlı bilgisayarı ayırt etmek; Babbage, Lovelace, ABC, Colossus, ENIAC, EDVAC ve von Neumann mimarisini tanımak.

Turing makinesi
Bant, bant başlığı ve kuralları kullanarak basit bir Turing makinesini adım adım çalıştırmak.

Bilgi türleri
Bildirimsel (ne?) ve buyurusal (nasıl?) bilgiyi karekök ve EBOB örnekleriyle ayırt etmek.

Algoritma ve problem
Algoritmanın özelliklerini saymak; bir problemin girdi ve çıktısını belirlemek, belirsiz bir isteği sorularla netleştirmek.

İşlem türleri
Sıralı, koşullu ve döngülü işlemleri tanımak; sözde koddaki `repeat … until` ile Java'daki `do … while` arasındaki ilişkiyi açıklamak.

Programlama (Programming)

- Programlama, **bilgisayara talimat vermenin** bir yoludur.
- Programlanan bilgisayar **karmaşık görevleri** yerine getirebilir.
- **Programlama dilleri** (Java, Python, C...) bilgisayarla iletişim kurmamızı sağlayan araçlardır.
- Programlama öğrenmek **problem çözme** ve düşünme becerilerini geliştirir.

Bilgisayarlar artık her yerde

- Evden iş yerine, eğlenceden eğitime her alanda kullanılır.
- Akıllı telefon, tablet, dizüstü bilgisayar, ev aletleri, arabalar...
- İnternet sayesinde dünya "bir tık" uzaklıkta: alışveriş, haber, iletişim, eğitim dijital dünyada.

Benzetme: Bilgisayar çok hızlı ama hiçbir şeyi kendiliğinden bilmeyen bir yardımcıdır. Program, ona **ne yapacağını adım adım** anlatan yazılı talimat listesidir.

PROGRAMLAMA VE HESAPLAMA

Programlamanın Rolü

- Programlama, bilgisayarların **istediğimiz görevleri** yerine getirmesi için kullanılan bir araçtır.
- Programcılar, bilgisayarları hayatımızın her alanında daha **kullanışlı** ve **verimli** hâle getirir.

Otomasyon

Akıllı termostatlar, aydınlatma ve güvenlik sistemleri; fabrikalardaki robotlar.

Sağlık

Tıbbi cihazlar (ör. kalp pili, MR), hastane bilgi sistemleri, randevu sistemleri.

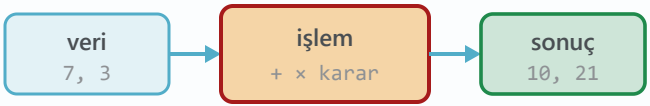
Eğitim

Çevrim içi ders platformları, öğretim yazılımları, sınav ve not sistemleri.

Elektrik-elektronik mühendisliğinde de gömülü sistemler, sinyal işleme ve kontrol yazılımları hep programlamayla yapılır.

Hesaplama (Computation)

- Bilgisayarların **temel işlevidir**.
- Bir bilgisayarın ya da hesaplama cihazının bir görevi yerine getirmek için **verileri işleme sürecidir**.
- **Matematiksel işlemleri**, **mantıksal kararları** ve veri işlemeyi içerir.
- Programlama, hesaplamayı daha **etkili** ve **verimli** hâle getirir.



Veri analizi
Ölçümlerden ortalama, eğilim, grafik.

Simülasyon
Bir devrenin ya da hava durumunun modeli.

Oyun geliştirme
Fizik, çarpışma, yapay zekâ... ve daha fazlası.

PROGRAMLAMA VE HESAPLAMA

Hesaplamanın Yapı Taşları

Matematik

Toplama, çıkarma, çarpma, bölme gibi işlemler.

Mantık

Kararlar, mantıksal ifadeler değerlendirilerek alınır: “Eğer hava güneşliyse dışarı çık.”

Hesaplama **veriyi alır, işler ve bir sonuç üretir**. Sonuç; problem çözmek, veri analiz etmek ya da görevleri otomatikleştirmek için kullanılır. Programlama, bilgisayara hesaplamayı **nasıl** yapacağını anlatır.

```java
public class Hesaplama {
    public static void main(String[] args) {
        int a = 7, b = 3;
        int toplam = a + b;      // matematik
        int carpim = a * b;      // matematik
        boolean gunesli = true; // mantık
        System.out.println("Toplam: " + toplam);
        System.out.println("Çarpım: " + carpim);
        if (gunesli) {           // karar
            System.out.println("Dışarı çık");
        }
    }
}
```

ÇIKTI

```
Toplam: 10
Çarpım: 21
Dışarı çık
```

Hesaplama Programı: Kod İzleme

```
1  public class Hesaplama {
2      public static void main(String[] args) {
▶          int a = 7, b = 3;
4          int toplam = a + b;      // matematik
5          int carpim = a * b;      // matematik
6          boolean gunesli = true; // mantık
7          System.out.println("Toplam: " + toplam);
8          System.out.println("Çarpım: " + carpim);
9          if (gunesli) {           // karar
10             System.out.println("Dışarı çık");
11         }
12     }
13 }
```



`main` başlar. Hesaplamanın girdileri `a` = 7 ve `b` = 3 olarak saklanır.

Adım 1 / 8

DEĞİŞKENLER

`a = 7` `b = 3`

ÇIKTI

Hesaplama Programı: Kod İzleme

```
1  public class Hesaplama {
2      public static void main(String[] args) {
3          int a = 7, b = 3;
▶          int toplam = a + b;      // matematik
5          int carpim = a * b;      // matematik
6          boolean gunesli = true; // mantık
7          System.out.println("Toplam: " + toplam);
8          System.out.println("Çarpım: " + carpim);
9          if (gunesli) {           // karar
10             System.out.println("Dışarı çık");
11         }
12     }
13 }
```



a + b → 7 + 3 = 10

Matematik: `a + b` = 7 + 3 = 10.

Adım 2 / 8

DEĞİŞKENLER

`a = 7` `b = 3` `toplam = 10`

ÇIKTI

Hesaplama Programı: Kod İzleme

```
1  public class Hesaplama {
2      public static void main(String[] args) {
3          int a = 7, b = 3;
4          int toplam = a + b;      // matematik
▶          int carpim = a * b;      // matematik
6          boolean gunesli = true; // mantık
7          System.out.println("Toplam: " + toplam);
8          System.out.println("Çarpım: " + carpim);
9          if (gunesli) {           // karar
10             System.out.println("Dışarı çık");
11         }
12     }
13 }
```



a * b → 7 × 3 = 21

Matematik: `a * b` = 7 × 3 = 21.

Adım 3 / 8

DEĞİŞKENLER

`a = 7` `b = 3` `toplam = 10` `carpim = 21`

ÇIKTI

Hesaplama Programı: Kod İzleme

```
1  public class Hesaplama {
2      public static void main(String[] args) {
3          int a = 7, b = 3;
4          int toplam = a + b;      // matematik
5          int carpim = a * b;      // matematik
▶          boolean gunesli = true; // mantık
7          System.out.println("Toplam: " + toplam);
8          System.out.println("Çarpım: " + carpim);
9          if (gunesli) {           // karar
10             System.out.println("Dışarı çık");
11         }
12     }
13 }
```



Mantık: `boolean` tipindeki değişken yalnızca `true` ya da `false` olabilir.

Adım 4 / 8

DEĞİŞKENLER

`a = 7` `b = 3` `toplam = 10` `carpim = 21` `gunesli = true`

ÇIKTI

Hesaplama Programı: Kod İzleme

```
1  public class Hesaplama {
2      public static void main(String[] args) {
3          int a = 7, b = 3;
4          int toplam = a + b;      // matematik
5          int carpim = a * b;      // matematik
6          boolean gunesli = true; // mantık
▶          System.out.println("Toplam: " + toplam);
8          System.out.println("Çarpım: " + carpim);
9          if (gunesli) {           // karar
10             System.out.println("Dışarı çık");
11         }
12     }
13 }
```



Sonuç dizgiyle birleştirilip yazdırılır.

Adım 5 / 8

DEĞİŞKENLER

`a = 7` `b = 3` `toplam = 10` `carpim = 21` `gunesli = true`

ÇIKTI

```
Toplam: 10
```

Hesaplama Programı: Kod İzleme

```
1  public class Hesaplama {
2      public static void main(String[] args) {
3          int a = 7, b = 3;
4          int toplam = a + b;      // matematik
5          int carpim = a * b;      // matematik
6          boolean gunesli = true; // mantık
7          System.out.println("Toplam: " + toplam);
▶          System.out.println("Çarpım: " + carpim);
9          if (gunesli) {           // karar
10             System.out.println("Dışarı çık");
11         }
12     }
13 }
```



İkinci sonuç yazdırılır.

Adım 6 / 8

DEĞİŞKENLER

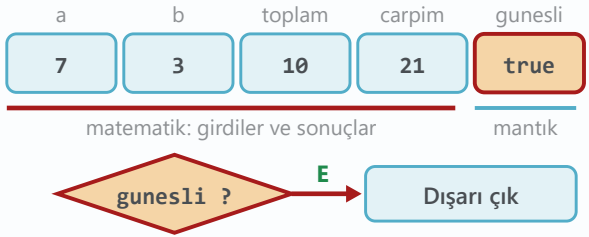
`a = 7` `b = 3` `toplam = 10` `carpim = 21` `gunesli = true`

ÇIKTI

```
Toplam: 10
Çarpım: 21
```

Hesaplama Programı: Kod İzleme

```
1  public class Hesaplama {
2      public static void main(String[] args) {
3          int a = 7, b = 3;
4          int toplam = a + b;      // matematik
5          int carpim = a * b;      // matematik
6          boolean gunesli = true; // mantık
7          System.out.println("Toplam: " + toplam);
8          System.out.println("Çarpım: " + carpim);
▶          if (gunesli) {           // karar
10             System.out.println("Dışarı çık");
11         }
12     }
13 }
```



Karar: koşul `gunesli` = `true` → Evet kolu seçilir.

Adım 7 / 8

DEĞİŞKENLER

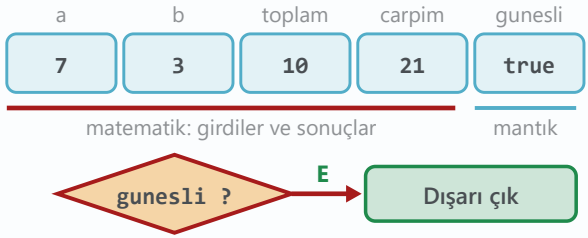
`a = 7` `b = 3` `toplam = 10` `carpim = 21` `gunesli = true`

ÇIKTI

```
Toplam: 10
Çarpım: 21
```

Hesaplama Programı: Kod İzleme

```
1  public class Hesaplama {
2      public static void main(String[] args) {
3          int a = 7, b = 3;
4          int toplam = a + b;      // matematik
5          int carpim = a * b;      // matematik
6          boolean gunesli = true; // mantık
7          System.out.println("Toplam: " + toplam);
8          System.out.println("Çarpım: " + carpim);
9          if (gunesli) {           // karar
▶              System.out.println("Dışarı çık");
11         }
12     }
13 }
```



Koşul doğru olduğu için blok çalışır. Veri alındı, işlendi, sonuç üretildi.

Adım 8 / 8

DEĞİŞKENLER

`a = 7` `b = 3` `toplam = 10` `carpim = 21` `gunesli = true`

ÇIKTI

```
Toplam: 10
Çarpım: 21
Dışarı çık
```

BİLGİSAYAR BİLİMİ

Bilgisayar Bilimi Sadece Teknoloji Öğrenmek Değildir

- Bilgisayar bilimi (computer science) yalnızca bilgisayarlarla ilgili teknik bilgi değil, aynı zamanda **problem çözme** becerileriyle ilgili bir disiplindir.
- Bilginin nasıl **işlendiğini**, **depolandığını** ve **iletildiğini** derinlemesine anlamayı gerektirir.
- Teknolojiyle sınırlı değildir: **teknoloji yalnızca bir araçtır**. Temel amaç sorunları çözmek ve bilgiyi daha iyi anlamaktır.
- Karmaşık problemleri **tanımlama**, **analiz etme** ve etkili **çözümler üretme** yeteneğini geliştirir.

Benzetme: Astronomi teleskoplarla ilgili olduğu kadar, bilgisayar bilimi de bilgisayarlarla ilgilidir. Teleskop gökyüzünü incelemenin aracıdır; bilgisayar da problem çözmenin.

BİLGİSAYAR BİLİMİ

Bilgisayar Bilimi: Üç Dayanak

Bilgisayar bilimi **mantık**, **problem çözme** ve **yaratıcılığa** dayalı bir disiplindir.

Mantık

Verileri değerlendirip doğru sonuçlar çıkarma yeteneğidir. Program yazma sürecinin temelini oluşturur.

Problem çözme

Karmaşık problemleri tanımlama ve çözme. Her problem, mantıklı bir **adım sırasıyla** çözülür.

Yaratıcılık

Yeni ve yenilikçi çözümlerin doğmasını sağlar; aynı problemin birçok farklı çözümü olabilir.

Programlama yaptıkça bu üç beceri birlikte gelişir.

BİLGİSAYAR BİLİMİ

Teknoloji ile Sosyal Sorumluluk

- Bilgisayar bilimi, teknolojinin **insanlara nasıl yardımcı olabileceğini** ve **toplumsal sorunların** çözümüne nasıl katkı sağlayabileceğini düşünmeyi teşvik eder.
- Dünyayı daha iyi anlamamıza ve geliştirmemize yardımcı olur; teknolojiyi **insanlığın hizmetine sunma sorumluluğunu** vurgular.
- Yalnızca teknolojiyle ilgilenenler için değil, problem çözme ve düşünme becerilerini geliştirmek isteyen herkes için ilginç bir alandır.

Uygulama alanlarından bazıları:

Tıp | Ekonomi | Astronomi | Biyoloji | Enerji | Ulaşım | Afet yönetimi

Yazdığımız programlar gerçek insanları etkiler: bir hastane yazılımındaki hata ya da kişisel verilerin korunmaması ciddi sonuçlar doğurabilir.

BİLGİSAYAR BİLİMİ

Bilgisayar Biliminde Neler Var?

- Bilgisayar bilimi, gerçekliği **anlama**, **modelleme** ve **analiz etme** amacı güden bir bilim dalıdır.
- Problemleri çözmek için **algoritmalar** ve **kod** kullanılır.

Soyutlama (abstraction)
Karmaşık bir sistemin gereksiz ayrıntılarını gizleyip özünü bırakmak. Metro haritası sokakları değil, yalnızca durakları ve hatları gösterir.

Problem çözme
Problemi parçalara ayırmak, her parçayı adım adım çözmek ve çözümleri birleştirmek.

Sanat ve tasarım
Mantık ve analitik düşünceyle sınırlı değildir: dijital medya, elektronik müzik, oyunlar, animasyon...

BİLGİSAYAR BİLİMİ

Hesaplamalı Düşünme (Computational Thinking)

Bir problemi, çözümünü bir insanın ya da bilgisayarın adım adım yürütebileceği biçimde ifade etme yoludur. Dört beceriye dayanır; örnek problem: **bir sınıfın not ortalaması**.

1. Ayrıştırma (decomposition)

Büyük problemi küçük alt problemlere bölmek: notları oku, topla, not sayısına böl, sonucu yaz.

2. Örüntü tanıma (pattern recognition)

Tekrar eden işi görmek: her notu okuyup toplama eklemek aynı adımdır, yani bir döngüdür.

3. Soyutlama (abstraction)

Gereksiz ayrıntıyı atmak: öğrencinin adı ve numarası ortalamayı etkilemez; yalnız notlar ve not sayısı gerekir.

4. Algoritma tasarımı (algorithm design)

Çözümü sonlu ve açık adımlarla yazmak: `toplam ← 0`; her not için `toplam ← toplam + not`; `ort ← toplam / n`.

Babil yöntemi ve Öklid algoritması da aynı yolla bulunur: tekrar eden bir adım görülür, gereksiz ayrıntı atılır, adımlar sonlu bir tarife dönüştürülür.

Bilgisayar (Computer)

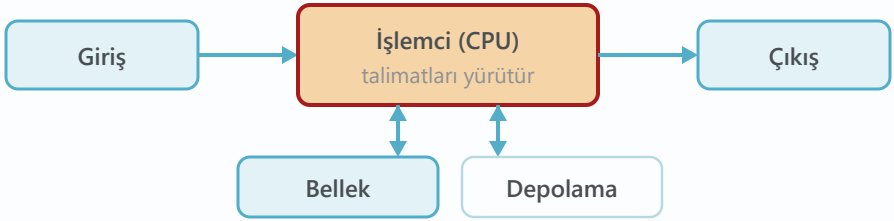
- Bir dizi **hesaplama** ve **talimatı** yürüten cihazdır; bu talimatlar bilgisayarın bir görevi yerine getirmesini sağlar.
- Verileri işlemek için **matematiksel hesaplamalar** ve **mantıksal talimatlar** kullanır.
- Günümüz bilgisayarları **elektronik** devrelerle çalışır ve sayıları **dijital** olarak temsil eder.

İşlemci (CPU)
Talimatları yürütür, hesaplar.

Bellek (RAM)
Çalışan program ve verisi; elektrik kesilince silinir.

Depolama
Disk, SSD: kalıcı saklama.

Giriş/Çıkış
Klavye, fare, ekran, yazıcı, ağ.



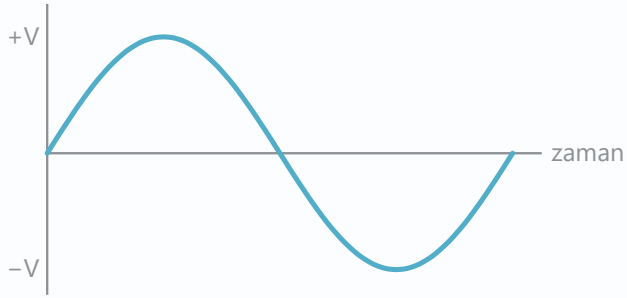
Elektronik ve Dijital

Elektronik

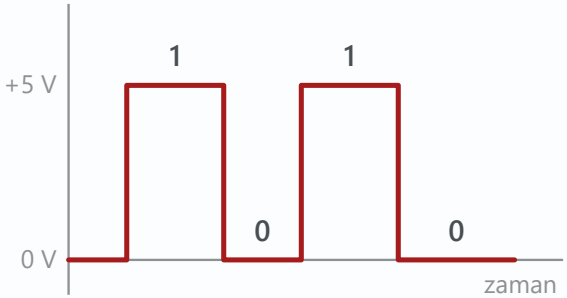
Bilgisayar, verileri **elektrik akımı** ile işler; mekanik dişliler yerine transistörler anahtarlama yapar.

Dijital (sayısal)

Veriler **0 ve 1** ile temsil edilir. Her 0/1 bir **bit**tir; 8 bit = 1 bayt.



Analog: sürekli, her değeri alabilir



Dijital: yalnızca iki düzey (0 ve 1)

İki düzeyli sinyali gürültüye rağmen ayırt etmek kolaydır: 3,9 V da 4,8 V da “1” sayılır. Dijital sistemlerin güvenilirliği buradan gelir.

Sabit Programlı Bilgisayar (Fixed Program Computer)

- **Belirli bir problemi** çözmek için tasarlanır; önceden tanımlanmış tek bir programı çalıştırır. Program, makinenin **yapısına gömülüdür**.
- Kökleri çok eskilere dayanır:
  - Abaküs, Antikythera Mekanizması (MÖ 2. yy)
  - Pascaline (Pascal, 1642), Leibniz'in basamaklı hesap makinesi (1673)
  - Babbage'ın Fark Makinesi (1822), Hollerith'in elektrikli çizelgeleme makinesi (1890)
  - Atanasoff–Berry Bilgisayarı (ABC), Turing Bombası (Enigma çözümü)

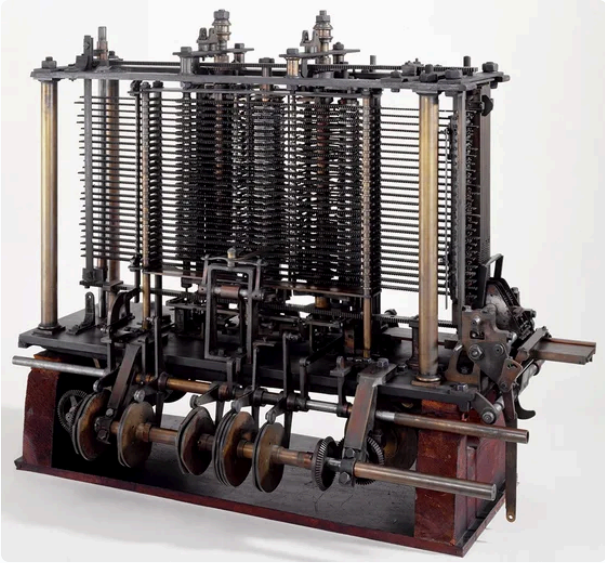
Sabit programlı makineler bilgisayar biliminin **temellerini attı**, teknolojik gelişmelere ilham verdi ve modern bilgisayarların **öncüsü** oldu.



Pascaline: yalnızca toplama ve çıkarma yapabilir.

Charles Babbage ve Analitik Makine

- **Charles Babbage** (1791–1871): İngiliz matematikçi ve mucit.
- 1837'de **Analitik Makine**'yi (Analytical Engine) tasarladı: ilk **genel amaçlı, programlanabilir** bilgisayar tasarımı.
- Matematiksel hesaplama yapabilen ve sonuçları **saklayabilen** bir makineydi; farklı hesaplamalar için programlanabilirdi.
- Bugünkü bilgisayarın parçaları bu tasarımda vardı: **store** (bellek), **mill** (işlemci), delikli kartlarla **giriş** ve yazıcıyla **çıkış**.
- Delikli kart fikri, desenleri kartlarla belirlenen **Jacquard dokuma tezgâhından** (1804) alındı.
- Makine Babbage'ın yaşamında tamamlanamadı, ama **modern bilgisayarların öncüsü** kabul edilir.



Analitik Makine'nin deneme modeli (bir parçası).

Ada Lovelace: İlk Program

- **Ada Lovelace** (1815–1852): İngiliz matematikçi ve yazar.
- 1843'te Analitik Makine üzerine bir makaleyi çevirip kendi **notlarını** ekledi.
- "Not G"de makinenin **Bernoulli sayılarını** hesaplaması için adım adım bir yöntem verdi: bu, **ilk bilgisayar programı** kabul edilir.
- Matematiksel düşünceyi ve programlamayı **bilimsel bir yaklaşımla** birleştirdi; bilgisayar tarihinin ilk programcısıdır.

Öngörüsü

Makinenin yalnızca sayılarla değil, **sembollerle** de (ör. müzik notaları) çalışabileceğini ilk fark edenlerdendir.

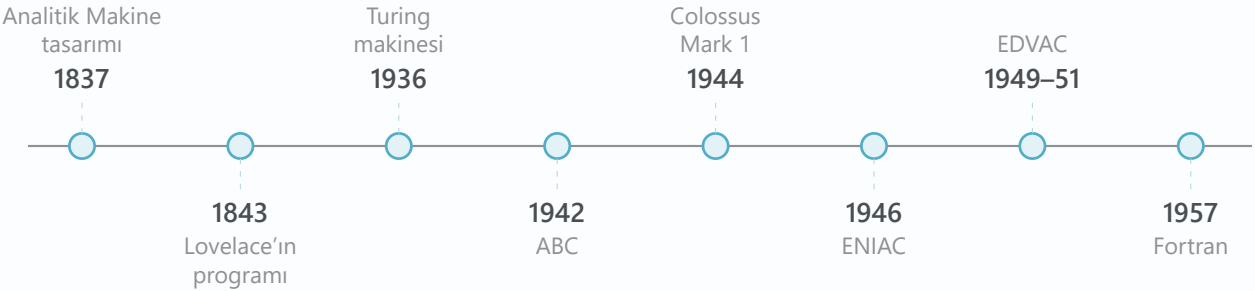
Mirası

1980'de tanıtılan **Ada** programlama dili onun adını taşır.

Program, makine henüz yokken yazıldı: algoritma **kâğıt üzerinde** tasarlanabilir.

BİLGİSAYARIN TARİHİ

Bilgisayarın Kısa Tarihi



Mekanik dönem

Dişliler ve kartlar: Pascaline, Fark ve Analitik Makine.

Kuramsal temel

Turing (1936): "Hesaplanabilir" ne demektir? Evrensel makine fikri.

Elektronik dönem

Vakum tüplü ABC, Colossus, ENIAC; depolanan programlı EDVAC.

- **1945 · Depolanan program:** program ve veri aynı bellekte durur; program da bir veri olduğundan girdi, başka bir programın tanımı olabilir.
- **Von Neumann mimarisi:** işlemci komutları bellekten sırayla okur; işlemci, bellek ve G/Ç birimleri adres, veri ve kontrol yollarıyla haberleşir.

Çizelgedeki aralıklar eşit çizilmiştir; yıllar arasındaki gerçek uzaklığı göstermez.

İlk Elektronik Bilgisayarlar: ABC ve Colossus

Atanasoff–Berry Bilgisayarı (ABC)

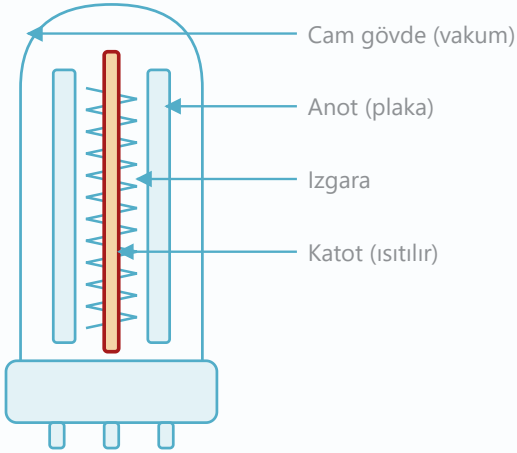
- John Atanasoff ve Clifford Berry, Iowa State Üniversitesi, 1937–1942.
- İlk **elektronik dijital** bilgisayar kabul edilir; mantık devrelerini vakum tüpleriyle kurdu.
- Sayıları **ikili** (0/1) sistemde işledi.
- Bellek: dönen tamburlar üzerindeki **kondansatörler**.
- Programlanamazdı: yalnızca doğrusal denklem sistemlerini çözüyordu (sabit programlı).

Colossus Mark 1

- Tommy Flowers ve ekibi, Bletchley Park (İngiltere), 1944.
- İlk **programlanabilir** elektronik dijital bilgisayar; anahtar ve fişlerle programlanıyordu.
- II. Dünya Savaşı'nda düşmanın **şifreli mesajlarını çözdü**.
- Yaklaşık 1600 **vakum tüpü** kullandı: mekanik cihazlardan çok daha hızlı ve esnekti.
- Bilgisayar teknolojisinin ilerlemesine büyük katkı sağladı.

Elektronik Valf: Vakum Tüpü

- Havası boşaltılmış cam tüp: ısıtılan **katot** elektron yayar, **anot** bu elektronları toplar.
- Aradaki **ızgaraya** uygulanan küçük gerilim akımı açar ya da keser: tüp bir **elektronik anahtar** gibi çalışır (açık = 1, kapalı = 0).
- Hareketli parça olmadığından mekanik rölelerden **binlerce kat hızlı** anahtarlama yapar.
- Sakıncaları: çok ısınır, çok enerji harcar ve sık arızalanır.



1947'de icat edilen **transistör** aynı anahtarlama işini çok daha küçük, serin ve güvenilir biçimde yapar; bugünkü işlemcilerde milyarlarca transistör vardır.

ENIAC ve EDVAC

ENIAC (1946)

- John Mauchly ve J. Presper Eckert, Pennsylvania Üniversitesi (ABD).
- **Genel amaçlı** ilk büyük ölçekli elektronik bilgisayar.
- Mekanik parçalar yerine yaklaşık 17 500 **vakum tüpü**: daha hızlı ve güvenilir.
- Saniyede yaklaşık 5000 toplama; topçu atış tabloları gibi büyük hesaplar.
- Sayıları **onlu** sistemde tutuyordu; programlamak için kablolar elle yeniden takılıyordu.

EDVAC (1949–1951)

- Tasarımı, John von Neumann'ın 1945 raporunda tanımlandı; 1951'de tam olarak çalıştı.
- ENIAC'tan farklı olarak **ikili** sayı sistemini kullandı.
- **Programı ve veriyi aynı bellekte** saklıyordu (depolanan program).
- Komutlar sırayla çalışıyor, koşula bağlı **atlama** yapılabiliyordu.
- Bu tasarım, bugünkü bilgisayarların temeli olan **von Neumann mimarisidir**.

Genel Amaçlı Bilgisayar (General Purpose Computer)

- Kullanıcının ihtiyacına göre **programlanabilir**; geniş bir uygulama yelpazesi için tasarlanmıştır.
- İş, eğitim, eğlence gibi birçok alanda kullanılır. **Program değiştirilerek** farklı görevlere uyarlanır: aynı dizüstü bilgisayar bir dakika önce müzik çalar, sonra Java derler.
- **İşlemci**, **bellek** ve **giriş/çıkış** birimlerinden oluşur. İşlemci verileri talimatlara göre işler ve sonuç üretir.

Otomasyon

İş süreçlerini otomatikleştirir.

Bilim

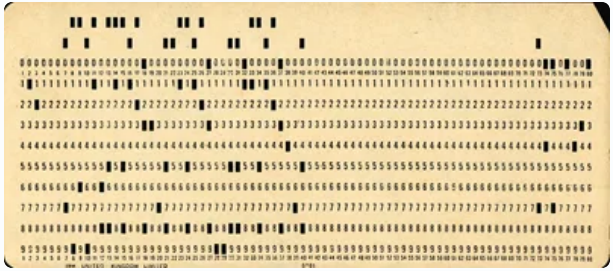
Bilimsel hesaplamaları hızlandırır.

İletişim

İletişimi kolaylaştırır.

İlk Programlar

- İlk programlar çoğunlukla **matematiksel hesaplamalar** içindi; ilk kullanımlar **bilimsel ve askerî** uygulamalara odaklanıyordu.
- Programlar ve veriler **delikli kartlarla** girilirdi: her sütundaki delik bir karakteri kodlar.
- İlk yaygın **yüksek düzeyli** programlama dili **Fortran** (Formula Translation), John Backus önderliğinde IBM'de geliştirildi ve 1957'de kullanıma sunuldu.
- Fortran bilimsel hesaplamalarda yaygın olarak kullanıldı ve hâlâ kullanılmaktadır.

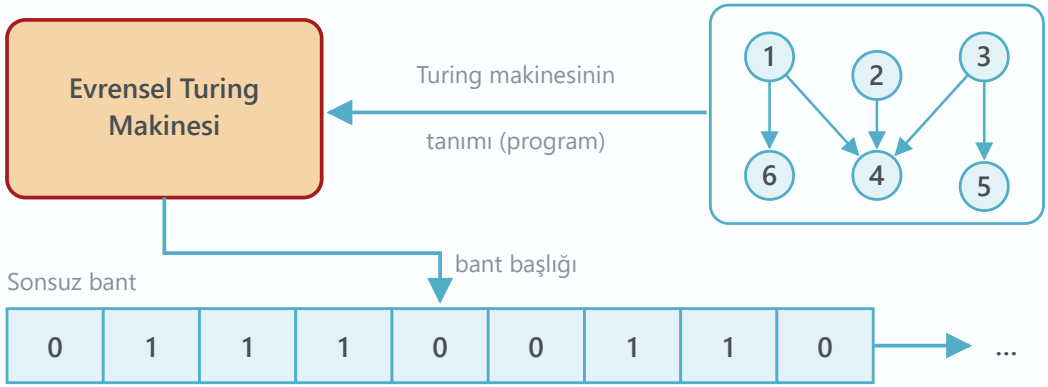


80 sütunlu delikli kart: bir kart ≈ bir satır program.

Yüksek düzeyli dil, programcının `K = I + J` gibi matematiğe yakın ifadeler yazmasını sağlar; **derleyici** bunu makine komutlarına çevirir. Java da (1995) yüksek düzeyli bir dildir.

Evrensel Turing Makinesi (Universal Turing Machine)

- Alan Turing'in 1936'da tanımladığı **soyut** ve **genel amaçlı** bir bilgisayar modelidir; bilgisayar biliminin en temel modellerinden biridir.
- Girdi olarak **başka bir Turing makinesinin tanımını** alır ve o makineyi **taklit eder** (simüle eder).
- Yani **hesaplanabilir her problemi** çözebilir. Modern bilgisayarların evrensel olarak programlanabilmesinin kuramsal temelidir.



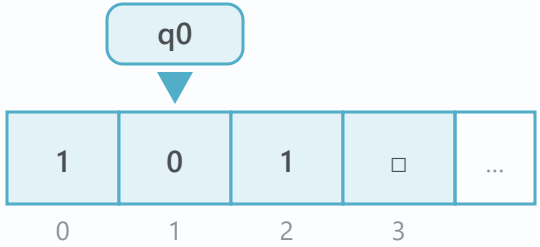
Depolanan programlı bilgisayar ve evrensel Turing makinesi, bilgisayar teknolojisinin iki önemli dönüm noktasıdır.

Turing Makinesinin Parçaları

Bant (tape)
Hücrelere bölünmüş, sonsuz uzunlukta bir şerit. Girdi, çıktı ve ara sonuçları saklar; her hücrede bir sembol (0, 1, boş □...) bulunur.

Bant başlığı (tape head)
Her an tek bir hücreyi gösterir. O hücredeki sembolü **okur**, yerine yeni sembol **yazar** ve bir hücre **sola ya da sağa** gider.

Durum ve kurallar
Makinenin o anki **durumu** (q0, q1...) ile **okunan sembol** birlikte ne yazılacağını, nereye gidileceğini ve sonraki durumu belirler.



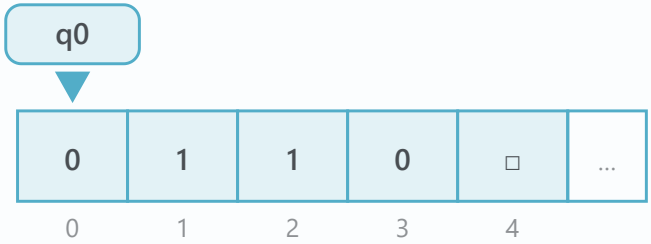
başlık 1 numaralı hücredeki 0'ı okuyor

Kural biçimi: (durum, okunan) → (yazılacak, yön, yeni durum). Örnek: (q0, 0) → (1, sağ, q0).

Örnek: Bitleri Ters Çeviren Turing Makinesi

#	Durum, okunan	Yaz, git, durum
1	q0, 0	1, sağ, q0
2	q0, 1	0, sağ, q0
3	q0, □	dur

Yalnızca üç kural: her 0'ı 1, her 1'i 0 yapar; boş hücreye gelince durur.



Başlangıç: durum q0, başlık 0. hücrede. Bantta 0110 yazılı.

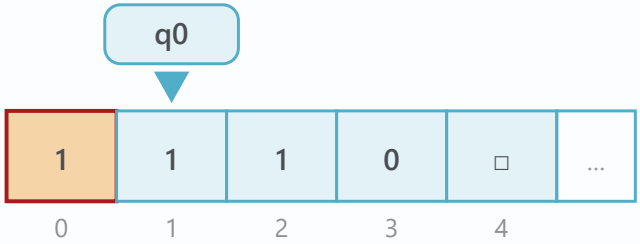
Adım 1 / 6

Bu kural tablosu, makinenin "programı"dır. Evrensel makine böyle bir tabloyu bantta okuyup çalıştırabilir.

Örnek: Bitleri Ters Çeviren Turing Makinesi

#	Durum, okunan	Yaz, git, durum
1	q0, 0	1, sağ, q0
2	q0, 1	0, sağ, q0
3	q0, □	dur

Yalnızca üç kural: her 0'ı 1, her 1'i 0 yapar; boş hücreye gelince durur.



0. hücrede 0 okundu → kural 1: 1 yazıldı, başlık sağa geçti.

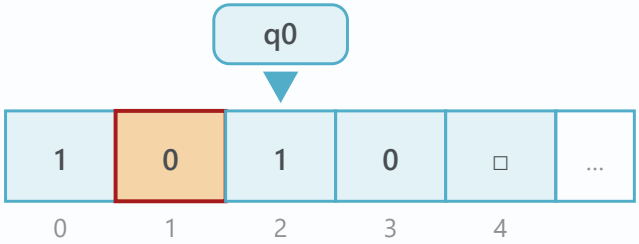
Adım 2 / 6

Bu kural tablosu, makinenin "programı"dır. Evrensel makine böyle bir tabloyu bantta okuyup çalıştırabilir.

Örnek: Bitleri Ters Çeviren Turing Makinesi

#	Durum, okunan	Yaz, git, durum
1	q0, 0	1, sağ, q0
2	q0, 1	0, sağ, q0
3	q0, □	dur

Yalnızca üç kural: her 0'ı 1, her 1'i 0 yapar; boş hücreye gelince durur.



1. hücrede 1 okundu → kural 2: 0 yazıldı, başlık sağa geçti.

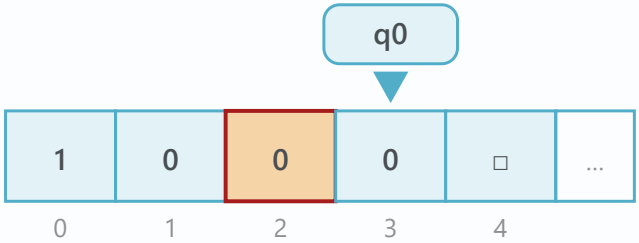
Adım 3 / 6

Bu kural tablosu, makinenin "programı"dır. Evrensel makine böyle bir tabloyu bantta okuyup çalıştırabilir.

Örnek: Bitleri Ters Çeviren Turing Makinesi

#	Durum, okunan	Yaz, git, durum
1	q0, 0	1, sağ, q0
2	q0, 1	0, sağ, q0
3	q0, □	dur

Yalnızca üç kural: her 0'ı 1, her 1'i 0 yapar; boş hücreye gelince durur.



2. hücrede 1 okundu → kural 2: 0 yazıldı, başlık sağa geçti.

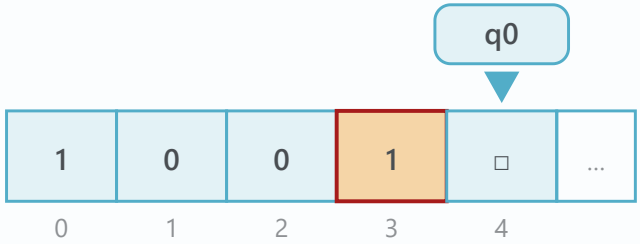
Adım 4 / 6

Bu kural tablosu, makinenin "programı"dır. Evrensel makine böyle bir tabloyu bantta okuyup çalıştırabilir.

Örnek: Bitleri Ters Çeviren Turing Makinesi

#	Durum, okunan	Yaz, git, durum
1	q0, 0	1, sağ, q0
2	q0, 1	0, sağ, q0
3	q0, □	dur

Yalnızca üç kural: her 0'ı 1, her 1'i 0 yapar; boş hücreye gelince durur.

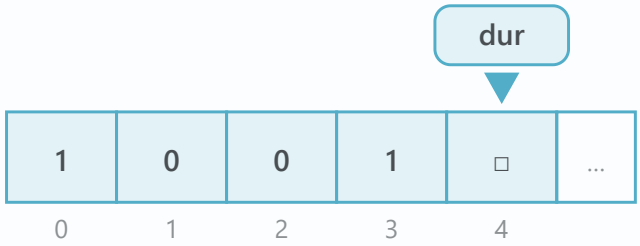


3. hücrede 0 okundu → kural 1: 1 yazıldı, başlık sağa geçti.

Bu kural tablosu, makinenin "programı"dır. Evrensel makine böyle bir tabloyu bantta okuyup çalıştırabilir.

Örnek: Bitleri Ters Çeviren Turing Makinesi

#	Durum, okunan	Yaz, git, durum
1	q0, 0	1, sağ, q0
2	q0, 1	0, sağ, q0
3	q0, □	dur



Yalnızca üç kural: her 0'ı 1, her 1'i 0 yapar; boş hücreye gelince durur.

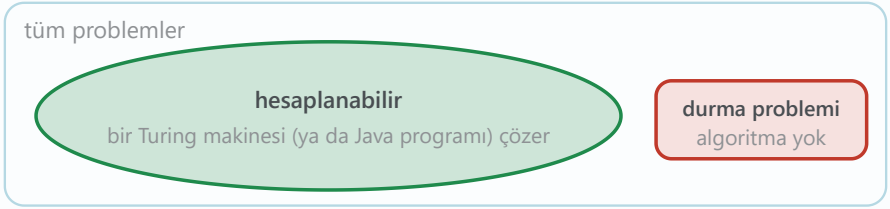
□ okundu → kural 3: makine **durur**. Sonuç bantta: **1001**.

Adım 6 / 6

Bu kural tablosu, makinenin "programı"dır. Evrensel makine böyle bir tabloyu bantta okuyup çalıştırabilir.

Hesaplamanın Sınırlarını Tanımlamak

- Turing makinesi, **hesaplamanın ne olduğunu** ve **sınırlarını** kesin olarak tanımlar.
- **Church–Turing tezi:** Adım adım bir yöntemle hesaplanabilen her şey bir Turing makinesiyle de hesaplanabilir.
- Ama her problem hesaplanabilir **değildir**. Turing, "Verilen bir program bir gün durur mu?" sorusunu (**durma problemi**, halting problem) her program için doğru yanıtlayan bir algoritmanın **olamayacağını** kanıtladı.
- Bu sonuçlar 20. yüzyılın en önemli bilimsel sonuçlarından biri kabul edilir.



Java ile bağlantı: Java, Python, C gibi diller **Turing-tam**dır (Turing complete): bir Turing makinesinin yapabildiği her hesabı yapabilirler. Diller hız ve kolaylık bakımından farklıdır, **güç** bakımından değil.

BİLGİSAYARLARIN SINIRLARI

Bilgisayarların Sınırları

Bilgisayarlar karmaşık hesaplamaları çok hızlı yapar; bu güç teknolojik gelişmelere dayanır. Yine de her kaynak **sınırlıdır**.

İşlemci hızı

Hesapların ne kadar hızlı yapılacağını belirler; ancak işlemci hızı sınırlıdır.

Bellek

Veri saklama kapasitesidir. Daha fazla bellek daha karmaşık görevler demektir; ancak bellek de sınırlıdır.

Veri işleme kapasitesi

Büyük veri kümelerini işlemek zaman ve kaynak ister.

Enerji

Bilgisayarlar sınırlı bir kaynak olan enerjiyi tüketir; daha güçlü bilgisayar daha fazla enerji demektir.

Programcılar bu sınırları **kaynakları etkin kullanarak** aşmaya çalışır: aynı işi daha az adımda ve daha az bellekle yapan algoritmalar seçerler.

Sınırlar Java'da da Görünür

```java
public class Sinirlar {
    public static void main(String[] args) {
        int enBuyuk = Integer.MAX_VALUE;
        System.out.println(enBuyuk);
        System.out.println(enBuyuk + 1);   // taşma
        System.out.println(0.1 + 0.2);     // duyarlık
        System.out.println(5 / 2);         // tamsayı bölmesi
    }
}
```

ÇIKTI

```
2147483647
-2147483648
0.30000000000000004
2
```

- `int` 32 bitlik bellek kullanır: en büyük değeri $2^{31} - 1$ = 2 147 483 647'dir. Bir eklenince **taşma** (overflow) olur, en küçük değere döner.
- `double` sayıları ikili tabanda yaklaşık 15–16 basamak **duyarlıkla** saklar; 0,1 ikili tabanda tam yazılamaz.
- İki `int`'in bölümü yine `int`'tir: 5 / 2 = 2 (kesir atılır).

BİLGİSAYARLARIN SINIRLARI

Makineler ve Teknoloji

- Teknoloji insanların yaşamını etkileyen bir güçtür; makineler bu gelişmenin önemli bir parçasıdır.
- Makineler işleri hızlı ve verimli yapar; bu güç sanayi devriminden bugüne sürekli arttı. Yine de **inşa edebileceğimiz makinelerin sınırları vardır**.

Fiziksel sınırlar

Boyut, ağırlık ve dayanıklılık fizik yasalarına bağlıdır: bir transistör birkaç atomdan küçük yapılamaz, sinyal ışıktan hızlı gidemez.

Enerji sınırları

Makineler enerjiye ihtiyaç duyar ve enerji kaynakları sınırlıdır; daha güçlü makine daha fazla enerji ve soğutma ister.

Teknolojik sınırlar

Yeni malzemeler, tasarım fikirleri ve **algoritmalar** sınırları zorlayabilir.

Sınırları anlamak, daha iyi ve verimli makineler tasarlamayı sağlar. Yazılımda bunun karşılığı, problemi **daha az kaynakla** çözen algoritmalardır.

BİLGİ TÜRLERİ

Bilgi: Bildirimsel ve Buyurusal

Bilgi, insanların ya da makinelerin dünyayı anlamasına ve sorunları çözmesine yardım eden bilinçli kavrayıştır. İki biçimde ifade edilebilir:

	Bildirimsel bilgi (declarative)	**Buyurusal bilgi (imperative)**
Soru	Bir şey **nedir**?	Bir şey **nasıl yapılır**?
Biçim	Tanım, aksiyom, gerçek	Sıralı talimatlar, tarif
Sınanması	Doğru ya da yanlış olarak doğrulanır	Adımlar izlenir, sonuç elde edilir
Karekök	$y \cdot y = x$ ise y, x'in karekökü dür	Babil yöntemi: tahmin et, iyileştir, tekrarla
Günlük örnek	"Kek, un, şeker ve yumurtadan yapılan bir tatlıdır."	Kek tarifi: "Yumurtaları çırp, unu ekle, 180 °C'de 35 dk pişir."

Bilgisayara **buyurusal** bilgi gerekir: programlar "nasıl yapılır" bilgisini yazar. Bildirimsel tanım ise sonucun **doğru olup olmadığını** sınamaya yarar.

Bildirimsel Bilgi (Declarative Knowledge)

- Açıkça ifade edilen bilgi türüdür; **doğru veya yanlış** olarak doğrulanabilir.
- **Aksiyomlar** ya da **tanımlar** yoluyla ifade edilir: bir şeyin ne olduğunu, nasıl tanımlandığını belirtir.
- **Gerçeklerin** ifadesini de içerebilir: gerçekler, dünya hakkında doğru ve doğrulanabilir ifadelerdir (“Su 100 °C’de kaynar”, deniz düzeyinde).

Örnek tanım

y, x’in karekköküdür $\Leftrightarrow y \cdot y = x$ ve $y \geq 0$.

Bu tanımla bir tahmini **sınayabiliriz**: $4 \cdot 4 = 16$ olduğundan 4, 16’nın karekköküdür. Ama tanım, karekökün **nasıl hesaplanacağını** söylemez: $\sqrt{2}$’yi bu tanımdan doğrudan bulamayız.

Buyurusal Bilgi: Babil Yöntemi ile Karekök

Buyurusal bilgi, bir şeyin **nasıl yapılacağına** dair talimatlar içerir. Babil yöntemi bir sayının karekökünü adım adım bulur:

1. x değerini al.
2. Bir başlangıç tahmini y_0 seç (ör. $y_0 = 1$).
3. $y_n^2 \approx x$ ise dur; cevap y_n.
4. Değilse yeni tahmini hesapla: $y_{n+1} = (y_n + x / y_n) / 2$.
5. Adım 3'e dön.

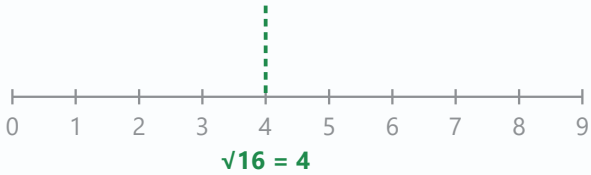
Neden çalışır? y çok büyükse x / y küçüktür (ya da tersi); ikisinin ortalaması gerçek köke daha yakındır. Her tekrarda tahmin köke yaklaşır.

n	y_n (x = 16)	y_n^2
0	1,000000	1,000000
1	8,500000	72,250000
2	5,191176	26,948313
3	4,136665	17,111995
4	4,002258	16,018065
5	4,000001	16,000005

Beş tekrarda 4'e çok yakın bir değer bulundu.

Babil Yöntemi: Java ile Adım Adım

```
1 double x = 16;    // karekökü aranan sayı
2 double y = 1;     // başlangıç tahmini y0
3 while (Math.abs(y * y - x) > 0.001) {
4     y = (y + x / y) / 2;    // yeni tahmin
5     System.out.println(y);
6 }
7 System.out.println("Karekök ≈ " + y);
```



x = 16 için karekök aranıyor (Java `double` 16'yı 16.0 olarak gösterir).

Adım 1 / 14

DEĞİŞKENLER

x = **16.0**

ÇIKTI

Babil Yöntemi: Java ile Adım Adım

```
1 double x = 16;   // karekökü aranan sayı
▶ double y = 1;    // başlangıç tahmini y0
3 while (Math.abs(y * y - x) > 0.001) {
4     y = (y + x / y) / 2;   // yeni tahmin
5     System.out.println(y);
6 }
7 System.out.println("Karekök ≈ " + y);
```



Başlangıç tahmini $y_0 = 1$.

Adım 2 / 14

DEĞİŞKENLER

`x = 16.0` `y = 1.0`

ÇIKTI

Babil Yöntemi: Java ile Adım Adım

```
1 double x = 16;    // karekökü aranan sayı
2 double y = 1;     // başlangıç tahmini y0
▶ while (Math.abs(y * y - x) > 0.001) {
4     y = (y + x / y) / 2;    // yeni tahmin
5     System.out.println(y);
6 }
7 System.out.println("Karekök ≈ " + y);
```



Koşul: |y·y − x| = 15 > 0,001 → **doğru**, döngüye girilir.

Adım 3 / 14

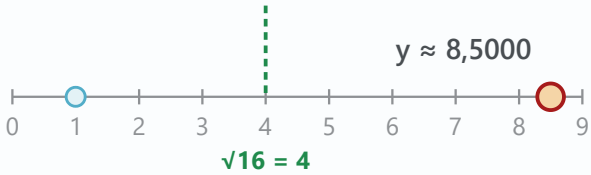
DEĞİŞKENLER

`x = 16.0` `y = 1.0`

ÇIKTI

Babil Yöntemi: Java ile Adım Adım

```
1 double x = 16;    // karekökü aranan sayı
2 double y = 1;     // başlangıç tahmini y0
3 while (Math.abs(y * y - x) > 0.001) {
      y = (y + x / y) / 2;    // yeni tahmin
      System.out.println(y);
6 }
7 System.out.println("Karekök ≈ " + y);
```



Yeni tahmin hesaplanır ve yazdırılır: y = 8,5.

Adım 4 / 14

DEĞİŞKENLER

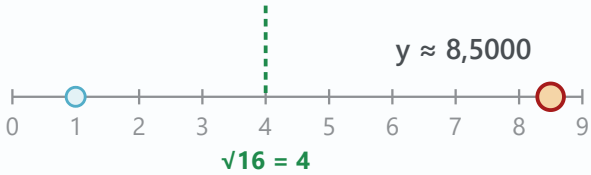
x = 16.0 y = 8.5

ÇIKTI

```
8.5
```

Babil Yöntemi: Java ile Adım Adım

```
1 double x = 16;    // karekökü aranan sayı
2 double y = 1;     // başlangıç tahmini y0
▶ while (Math.abs(y * y - x) > 0.001) {
4     y = (y + x / y) / 2;    // yeni tahmin
5     System.out.println(y);
6 }
7 System.out.println("Karekök ≈ " + y);
```



Koşul: |y·y − x| = 56,25 > 0,001 → **doğru**, döngüye girilir.

Adım 5 / 14

DEĞİŞKENLER

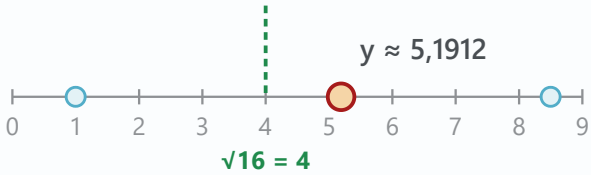
x = **16.0** y = **8.5**

ÇIKTI

```
8.5
```

Babil Yöntemi: Java ile Adım Adım

```
1 double x = 16;    // karekökü aranan sayı
2 double y = 1;     // başlangıç tahmini y0
3 while (Math.abs(y * y - x) > 0.001) {
      y = (y + x / y) / 2;    // yeni tahmin
      System.out.println(y);
6 }
7 System.out.println("Karekök ≈ " + y);
```



Yeni tahmin hesaplanır ve yazdırılır: y = 5,1911764705882355.

Adım 6 / 14

DEĞİŞKENLER

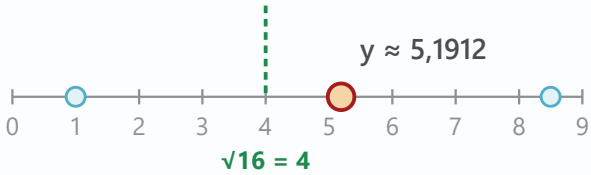
x = 16.0 | y = 5.1911764705882355

ÇIKTI

```
8.5
5.1911764705882355
```

Babil Yöntemi: Java ile Adım Adım

```
1 double x = 16;    // karekökü aranan sayı
2 double y = 1;     // başlangıç tahmini y0
▶ while (Math.abs(y * y - x) > 0.001) {
4     y = (y + x / y) / 2;    // yeni tahmin
5     System.out.println(y);
6 }
7 System.out.println("Karekök ≈ " + y);
```



Koşul: |y·y − x| = 10,9483 > 0,001 → **doğru**, döngüye girilir.

Adım 7 / 14

DEĞİŞKENLER

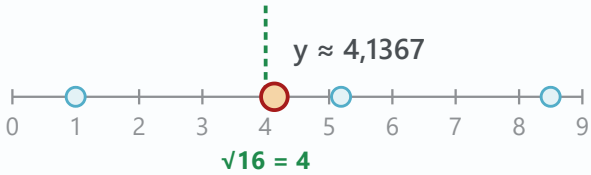
`x = 16.0` `y = 5.1911764705882355`

ÇIKTI

```
8.5
5.1911764705882355
```

Babil Yöntemi: Java ile Adım Adım

```
1 double x = 16;    // karekökü aranan sayı
2 double y = 1;     // başlangıç tahmini y0
3 while (Math.abs(y * y - x) > 0.001) {
      y = (y + x / y) / 2;    // yeni tahmin
      System.out.println(y);
6 }
7 System.out.println("Karekök ≈ " + y);
```



Yeni tahmin hesaplanır ve yazdırılır: y = 4,136664722546242.

Adım 8 / 14

DEĞİŞKENLER

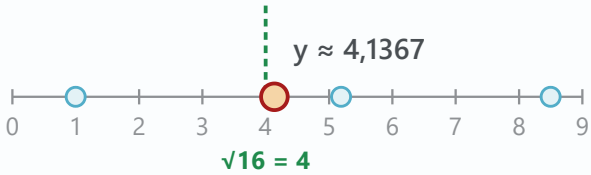
`x = 16.0` `y = 4.136664722546242`

ÇIKTI

```
8.5
5.1911764705882355
4.136664722546242
```

Babil Yöntemi: Java ile Adım Adım

```
1 double x = 16;    // karekökü aranan sayı
2 double y = 1;     // başlangıç tahmini y0
▶ while (Math.abs(y * y - x) > 0.001) {
4     y = (y + x / y) / 2;    // yeni tahmin
5     System.out.println(y);
6 }
7 System.out.println("Karekök ≈ " + y);
```



Koşul: |y·y − x| = 1,112 > 0,001 → **doğru**, döngüye girilir.

Adım 9 / 14

DEĞİŞKENLER

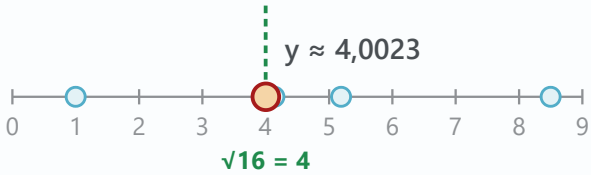
`x = 16.0` `y = 4.136664722546242`

ÇIKTI

```
8.5
5.1911764705882355
4.136664722546242
```

Babil Yöntemi: Java ile Adım Adım

```
1 double x = 16;    // karekökü aranan sayı
2 double y = 1;     // başlangıç tahmini y0
3 while (Math.abs(y * y - x) > 0.001) {
      y = (y + x / y) / 2;    // yeni tahmin
      System.out.println(y);
6 }
7 System.out.println("Karekök ≈ " + y);
```



Yeni tahmin hesaplanır ve yazdırılır: y = 4,002257524798522.

Adım 10 / 14

DEĞİŞKENLER

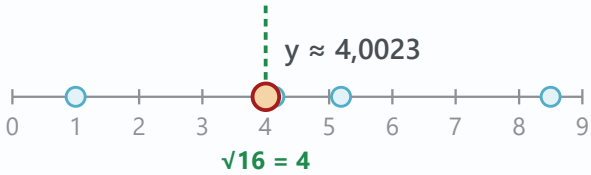
`x = 16.0` `y = 4.002257524798522`

ÇIKTI

```
8.5
5.1911764705882355
4.136664722546242
4.002257524798522
```

Babil Yöntemi: Java ile Adım Adım

```
1 double x = 16;    // karekökü aranan sayı
2 double y = 1;     // başlangıç tahmini y0
▶ while (Math.abs(y * y - x) > 0.001) {
4     y = (y + x / y) / 2;    // yeni tahmin
5     System.out.println(y);
6 }
7 System.out.println("Karekök ≈ " + y);
```



Koşul: |y·y − x| = 0,0180653 > 0,001 → **doğru**, döngüye girilir.

Adım 11 / 14

DEĞİŞKENLER

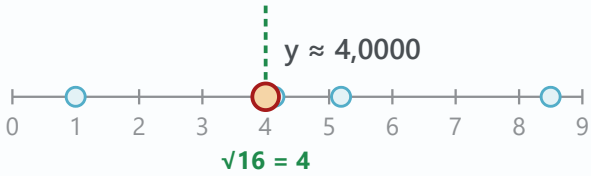
`x = 16.0` `y = 4.002257524798522`

ÇIKTI

```
8.5
5.1911764705882355
4.136664722546242
4.002257524798522
```

Babil Yöntemi: Java ile Adım Adım

```
1 double x = 16;    // karekökü aranan sayı
2 double y = 1;     // başlangıç tahmini y0
3 while (Math.abs(y * y - x) > 0.001) {
      y = (y + x / y) / 2;    // yeni tahmin
      System.out.println(y);
6 }
7 System.out.println("Karekök ≈ " + y);
```



Yeni tahmin hesaplanır ve yazdırılır: y = 4,000000636692939.

Adım 12 / 14

DEĞİŞKENLER

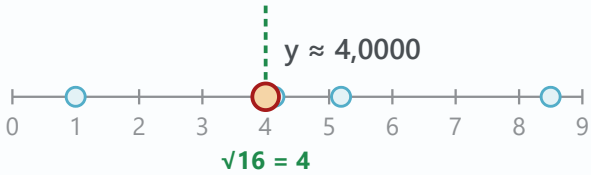
x = 16.0 y = 4.000000636692939

ÇIKTI

```
8.5
5.1911764705882355
4.136664722546242
4.002257524798522
4.000000636692939
```

Babil Yöntemi: Java ile Adım Adım

```
1 double x = 16;    // karekökü aranan sayı
2 double y = 1;     // başlangıç tahmini y0
▶ while (Math.abs(y * y - x) > 0.001) {
4     y = (y + x / y) / 2;    // yeni tahmin
5     System.out.println(y);
6 }
7 System.out.println("Karekök ≈ " + y);
```



Koşul: $|y \cdot y - x| \approx 5{,}1 \cdot 10^{-6} \leq 0{,}001$ → **yanlış**, döngü biter.

Adım 13 / 14

DEĞİŞKENLER

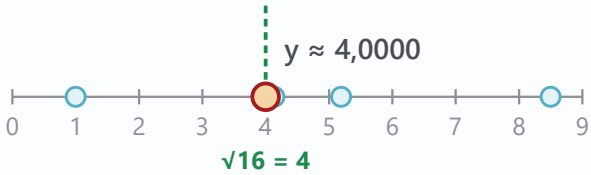
`x = 16.0` `y = 4.000000636692939`

ÇIKTI

```
8.5
5.1911764705882355
4.136664722546242
4.002257524798522
4.000000636692939
```

Babil Yöntemi: Java ile Adım Adım

```
1 double x = 16;    // karekökü aranan sayı
2 double y = 1;     // başlangıç tahmini y0
3 while (Math.abs(y * y - x) > 0.001) {
4     y = (y + x / y) / 2;    // yeni tahmin
5     System.out.println(y);
6 }
▶ System.out.println("Karekök ≈ " + y);
```



Sonuç yazdırılır. Tam 4 değil, 4'e çok yakın bir **yaklaşık** değer.

Adım 14 / 14

DEĞİŞKENLER

`x = 16.0` `y = 4.000000636692939`

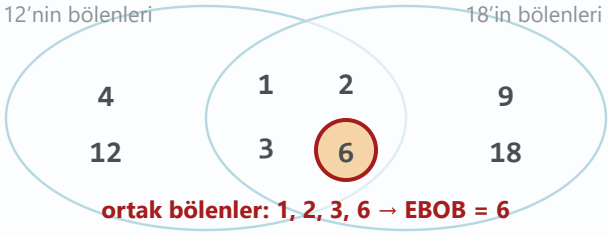
ÇIKTI

```
8.5
5.1911764705882355
4.136664722546242
4.002257524798522
4.000000636692939
Karekök ≈ 4.000000636692939
```

BİLGİ TÜRLERİ

En Büyük Ortak Bölen (EBOB): İki Tanım

İki sayının EBOB'u (greatest common divisor, GCD), ikisini de tam bölen **en büyük pozitif tam sayıdır**. Örnek: 12'nin bölenleri 1, 2, 3, 4, 6, 12; 18'in bölenleri 1, 2, 3, 6, 9, 18 → EBOB(12, 18) = **6**.



Bildirimsel tanım

EBOB'un **ne olduğunu** söyler: d, a ve b'nin ortak bölenidir (bir x, y tam sayısı için a = d·x ve b = d·y) ve bu koşulu sağlayan sayıların **en büyüğüdür**. Bir adayı sınamaya yarar, ama EBOB'u bulma yolunu vermez.

Buyurusal tanım: Öklid algoritması

EBOB'u **nasıl bulacağımızı** söyler: bölme ve kalan almayı, kalan 0 olana kadar tekrarla. Yaklaşık MÖ 300'de Öklid'in *Elemanlar* kitabında yer alan, bilinen en eski algoritmalardan biridir.

Öklid Algoritması (Euclidean Algorithm)

Algoritmanın adımları

1. İki pozitif tam sayı a ve b'yi al (a ≥ b).
2. a'yı b'ye böl, kalana r de: r = a mod b.
3. r = 0 ise dur: **EBOB = b**.
4. Değilse a'ya b'yi, b'ye r'yi ata.
5. Adım 2'ye dön.

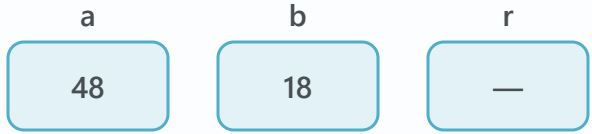
Neden doğru? a'yı ve b'yi bölen her sayı, kalan r = a − k·b'yi de böler; tersine, b'yi ve r'yi bölen her sayı a = k·b + r'yi de böler. İki çiftin ortak bölenleri aynıdır, bu yüzden EBOB(a, b) = EBOB(b, r); sayılar küçülür ama EBOB değişmez.

a	b	r = a mod b
48	18	12
18	12	6
12	**6**	0

48 = 2·18 + 12, 18 = 1·12 + 6, 12 = 2·6 + 0 → EBOB(48, 18) = **6**.

Öklid Algoritması: Java ile Adım Adım

```
▶ int a = 48, b = 18;   // a >= b
2 int r = a % b;        // kalan
3 while (r != 0) {
4     a = b;
5     b = r;
6     r = a % b;
7 }
8 System.out.println("EBOB = " + b);
```



a = 48, b = 18 (a ≥ b).

Adım 1 / 12

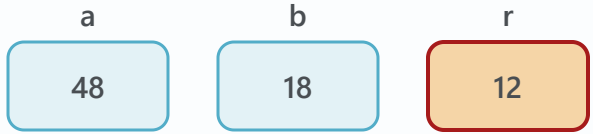
DEĞİŞKENLER

`a = 48` `b = 18` `r = –`

ÇIKTI

Öklid Algoritması: Java ile Adım Adım

```
1 int a = 48, b = 18;    // a >= b
2 int r = a % b;         // kalan
3 while (r != 0) {
4     a = b;
5     b = r;
6     r = a % b;
7 }
8 System.out.println("EBOB = " + b);
```



r ← a % b = 48 % 18

Kalan: 48 % 18 = 12 (48 = 2·18 + 12).

Adım 2 / 12

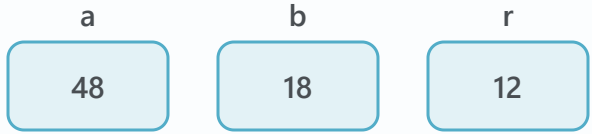
DEĞİŞKENLER

a = 48 | b = 18 | r = 12

ÇIKTI

Öklid Algoritması: Java ile Adım Adım

```
1 int a = 48, b = 18;   // a >= b
2 int r = a % b;        // kalan
▶ while (r != 0) {
4     a = b;
5     b = r;
6     r = a % b;
7 }
8 System.out.println("EBOB = " + b);
```



r = 12 ≠ 0 → döngüye girilir.

Adım 3 / 12

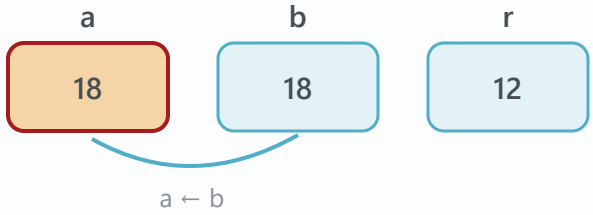
DEĞİŞKENLER

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

ÇIKTI

Öklid Algoritması: Java ile Adım Adım

```
1 int a = 48, b = 18;   // a >= b
2 int r = a % b;        // kalan
3 while (r != 0) {
▶     a = b;
5     b = r;
6     r = a % b;
7 }
8 System.out.println("EBOB = " + b);
```



a ← b: a = 18.

Adım 4 / 12

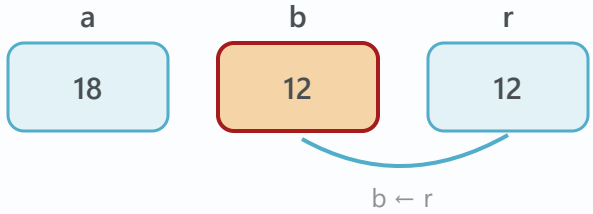
DEĞİŞKENLER

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

ÇIKTI

Öklid Algoritması: Java ile Adım Adım

```
1 int a = 48, b = 18;   // a >= b
2 int r = a % b;        // kalan
3 while (r != 0) {
4     a = b;
      b = r;
6     r = a % b;
7 }
8 System.out.println("EBOB = " + b);
```



b ← r: b = 12.

Adım 5 / 12

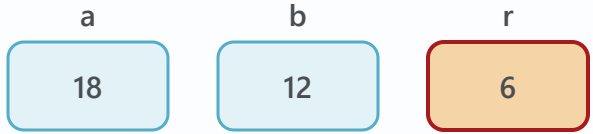
DEĞİŞKENLER

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

ÇIKTI

Öklid Algoritması: Java ile Adım Adım

```
1 int a = 48, b = 18;   // a >= b
2 int r = a % b;        // kalan
3 while (r != 0) {
4     a = b;
5     b = r;
▶     r = a % b;
7 }
8 System.out.println("EBOB = " + b);
```



r ← a % b = 18 % 12

Yeni kalan: 18 % 12 = 6.

Adım 6 / 12

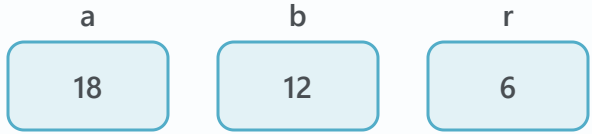
DEĞİŞKENLER

a = 18 | b = 12 | r = 6

ÇIKTI

Öklid Algoritması: Java ile Adım Adım

```
1 int a = 48, b = 18;   // a >= b
2 int r = a % b;        // kalan
▶ while (r != 0) {
4     a = b;
5     b = r;
6     r = a % b;
7 }
8 System.out.println("EBOB = " + b);
```



r = 6 ≠ 0 → döngüye girilir.

Adım 7 / 12

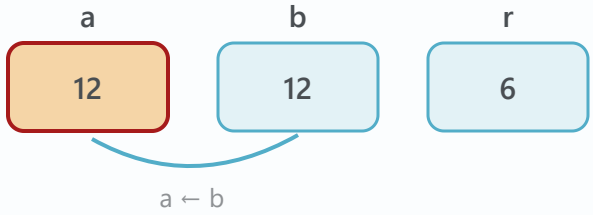
DEĞİŞKENLER

`a = 18` `b = 12` `r = 6`

ÇIKTI

Öklid Algoritması: Java ile Adım Adım

```
1 int a = 48, b = 18;   // a >= b
2 int r = a % b;        // kalan
3 while (r != 0) {
▶     a = b;
5     b = r;
6     r = a % b;
7 }
8 System.out.println("EBOB = " + b);
```



a ← b: a = 12.

Adım 8 / 12

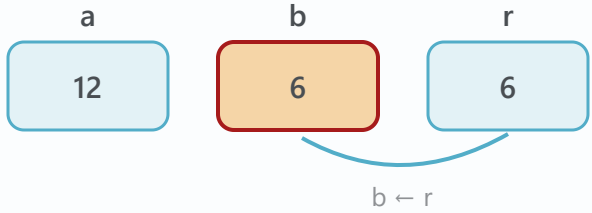
DEĞİŞKENLER

a = 12 | b = 12 | r = 6

ÇIKTI

Öklid Algoritması: Java ile Adım Adım

```
1 int a = 48, b = 18;   // a >= b
2 int r = a % b;        // kalan
3 while (r != 0) {
4     a = b;
5     b = r;
6     r = a % b;
7 }
8 System.out.println("EBOB = " + b);
```



b ← r: b = 6.

Adım 9 / 12

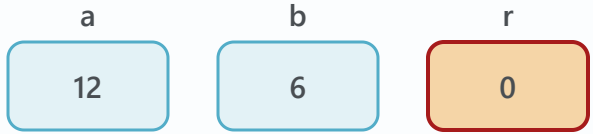
DEĞİŞKENLER

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

ÇIKTI

Öklid Algoritması: Java ile Adım Adım

```
1 int a = 48, b = 18;   // a >= b
2 int r = a % b;        // kalan
3 while (r != 0) {
4     a = b;
5     b = r;
▶     r = a % b;
7 }
8 System.out.println("EBOB = " + b);
```



r ← a % b = 12 % 6

Yeni kalan: 12 % 6 = 0.

Adım 10 / 12

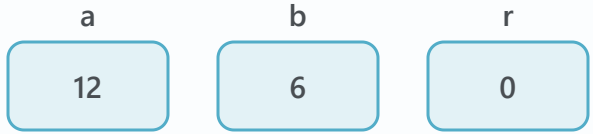
DEĞİŞKENLER

a = 12 | b = 6 | r = 0

ÇIKTI

Öklid Algoritması: Java ile Adım Adım

```
1 int a = 48, b = 18;   // a >= b
2 int r = a % b;        // kalan
▶ while (r != 0) {
4     a = b;
5     b = r;
6     r = a % b;
7 }
8 System.out.println("EBOB = " + b);
```



r = 0 → koşul yanlış, döngü biter.

Adım 11 / 12

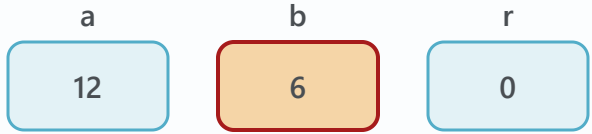
DEĞİŞKENLER

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

ÇIKTI

Öklid Algoritması: Java ile Adım Adım

```
1 int a = 48, b = 18;    // a >= b
2 int r = a % b;         // kalan
3 while (r != 0) {
4     a = b;
5     b = r;
6     r = a % b;
7 }
▶ System.out.println("EBOB = " + b);
```



Son bölen b = 6 sonuçtur: EBOB(48, 18) = 6.

Adım 12 / 12

DEĞİŞKENLER

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

ÇIKTI

```
EBOB = 6
```

Algoritma (Algorithm)

- Bir problemi **adım adım çözme yöntemi**dir; izlenmesi gereken adımları belirleyen bir **tarif**tir.
- Bir görevi nasıl yapacağımızı belirler; karmaşık problemleri basitleştirmeye yarar.
- Bilgisayar programlarının temelidir: **programlama**, algoritmaları bir programlama diliyle ifade etme sürecidir.
- Bilgisayar bilimi büyük ölçüde algoritmaların tasarımı ve analizidir.
- İyi programcı, problem için **uygun algoritmayı** seçip uygulayabilir.

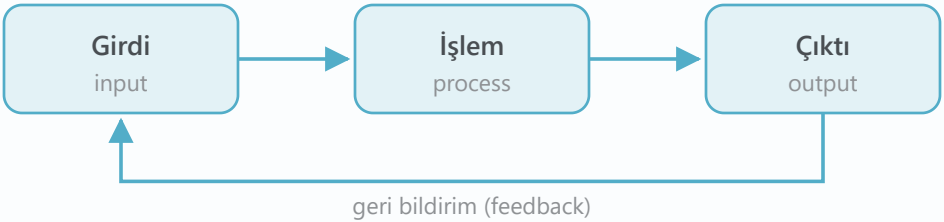
Bir algoritma…

- adımları açıkça belirler,
- adımları mantıklı bir **sıraya** koyar,
- sonlu sayıda adımda **biter**,
- problemi çözme amacı güder.

İyi tasarlanmış algoritma **doğru** ve **verimli** olmalıdır. Ad, 9. yüzyıl matematikçisi el-Hârizmî'den gelir.

Girdi, İşlem, Çıktı

- Programın ya da algoritmanın **ne yapması gerektiği** açıkça tanımlanır; gerekli adımlar ve çıktı belirlenir.
- Problem belirlemenin iki bileşeni vardır:
  - **Girdi** (input): problemin çözülmesi için gereken bilgi ya da veri.
  - **Çıktı** (output): problemin sonucu ya da istenen bilgi.



Problemi tanımlarken **Ne? Kim? Nerede? Ne zaman? Nasıl? Neden?** soruları sorulur. "Not ortalaması hesapla" belirsizdir: kaç not olduğu, ağırlıkların eşit olup olmadığı ve sonucun tam sayı mı ondalıklı mı olacağı yanıtlanmadan yazılan program yanlış problemi çözebilir.

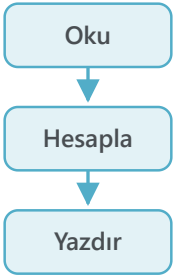
Geri bildirim: çıktıya bakarak girdiyi ya da işlemi düzeltmek (ör. termostat sıcaklığı ölçer, ısıtıcıyı açar ya da kapatır).

Algoritmik İşlem Türleri

Her algoritma üç tür işlemden kurulur. Bu türler **her programlama dilinde** bulunur, algoritmaların temel yapısını oluşturur ve karmaşıklığı denetlemeye yarar.

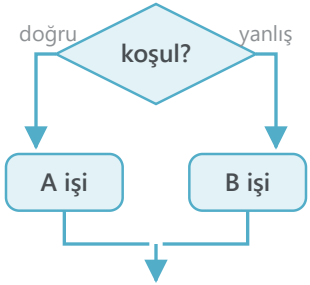
Sıralı (sequential)

Her işlem bir öncekinden sonra, sırayla çalışır. Ör. veriyi oku, işle, sonucu üret.



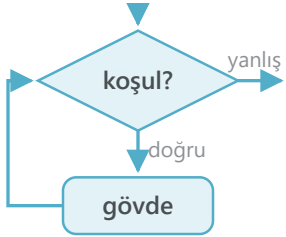
Koşullu (conditional)

Koşul sağlanıyorsa bir işlem, sağlanmıyorsa başka bir işlem yapılır. Ör. sayı pozitif mi?



Döngülü (iterative)

Bir işlem, koşul sağlandığı sürece tekrarlanır. Ör. bir listedeki tüm sayıları toplamak.



Döngülü İşlemler: Tekrar Eden Eylemler

- **Devam koşulu** yanlış olana kadar belirli eylemleri tekrar eder; listeleri gezmek, verileri işlemek için kullanışlıdır.

`while`: önce koşul, sonra gövde

i > 0 olduğu sürece s'yi 1 artır, i'yi 1 azalt.

```
while i > 0 do
    set s to s + 1
    set i to i - 1
```

```
int s = 0, i = 3;
while (i > 0) {
    s = s + 1;
    i = i - 1;
}
```

`repeat … until`: önce gövde

k, n'den büyük olana kadar k'yi yazdır ve artır.

```
repeat
    print k
    set k to k + 1
until k > n
```

```
int k = 1, n = 3;
do {
    System.out.println(k);
    k = k + 1;
} while (k <= n);
```

Java'da `repeat … until` yoktur; karşılığı `do … while`'dır ve koşul **tersine çevrilir**: "k > n olana kadar" = "k ≤ n olduğu sürece". k = 1, n = 3 için çıktı: 1, 2, 3. Gövde koşulu sonunda yanlış yapmazsa döngü sonsuza dek sürer; takılan program IDE'deki durdur düğmesiyle ya da konsolda Ctrl+C ile durdurulur.

Özet

Kavram	Akılda kalacak nokta
Programlama / hesaplama	Bilgisayara talimat vermek / veriyi matematik ve mantıkla işleyip sonuç üretmek.
Bilgisayar bilimi	Mantık, problem çözme ve yaratıcılık; teknoloji yalnızca araçtır.
Sabit / depolanan program	İşlev donanıma gömülü / program bellekte, değiştirilebilir (EDVAC, von Neumann).
Turing makinesi	Bant + başlık + kurallar; hesaplanabilir her şeyi hesaplar, ama her problem hesaplanabilir değildir.
Bildirimsel / buyurusal	"Nedir?" (tanım, sınama) / "Nasıl yapılır?" (adımlar: Babil, Öklid).
Algoritma, sözde kod	Sonlu, sıralı, açık adımlar; önce girdi ve çıktıyı belirle, sonra adımları yaz.
İşlem türleri	**Sıralı** (sırayla), **koşullu** (`if/else`), **döngülü** (`while` , `do-while` , `for`).
Döngü	Devam koşulu + gövde; gövde koşulu sonunda yanlış yapmalı, yoksa sonsuz döngü.

ÖZET

Sık Yapılan Hatalar

Noktalı virgülü unutmak

`int a = 5` → `error: ';' expected`. Sözde kodda serbestiz, Java'da her deyim `;` ile biter.

`=` ile `==`'yi karıştırmak

`if (x = 5)` → `error: incompatible types: int cannot be converted to boolean`. Karşılaştırma `==`'dir.

İlk değer vermemek

`int toplam; System.out.println(toplam);` → `error: variable toplam might not have been initialized`.

Tam sayı bölmesi

`5 / 2` = 2, `9 / 5` = 1. Ondalıklı sonuç için `5 / 2.0` ya da `5.0 / 2` yazılır.

Ondalık ayırıcı

Türkçe ayarlı bilgisayarda `nextDouble()` "1,75" bekler; "1.75" yazılırsa `InputMismatchException` fırlatılır.

Döngü mantık hataları

En büyüğü 0 ile başlatmak (negatif listede yanlış); koşul değişkenini güncellemeyi unutmak (sonsuz döngü). Derleyici ikisini de yakalamaz.

ÖZET

Örnek Problemler

1. Hangisi bildirimsel, hangisi buyurusal? (a) “Bir sayı 2’ye kalansız bölünüyorsa çifttir.” (b) “Sayıyı 2’ye böl; kalan 0 ise ‘çift’ yaz.”
2. Öklid algoritmasıyla EBOB(84, 36)’yı bulun. Kaç bölme yapılır?
3. Babil yöntemiyle $\sqrt{10}$’u $y_0 = 3$ alarak iki adım hesaplayın.
4. Üç notun ortalamasını bulan programın girdi, çıktı ve sözde kodunu yazın. Notlar 40, 75, 90 ise çıktı?
5. `while` örneğinde s = 0, i = 5 ile başlanırsa kaç tur döner, sonunda s ve i kaçtır?
6. Bitleri ters çeviren Turing makinesi 1010 bandında kaç adım çalışır, bantta ne kalır?

▶ **Cevapları göster**

ÖZET

Alıştırmalar

1. Santigratı Fahrenhayta çeviren (F = C · 9/5 + 32) programı yazın. `int f = 9 / 5 * c + 32;` c = 25 için ne verir?
2. Üç sayıdan en büyüğünü yazdıran sözde kodu yazın.
3. 1'den n'e kadar sayıların toplamını `while` ile hesaplayın. n = 10 için sonuç?
4. EBOB(1071, 462)'yi Öklid algoritmasıyla tablo hâlinde izleyin.
5. Bantta tekli (unary) bir sayı (111 = 3) varken sonuna bir 1 ekleyen Turing makinesinin kurallarını yazın.
6. Bir listedeki en küçük öğeyi bulan sözde kodu yazın. Başlangıç değeri neden 0 değil, listenin ilk öğesi olmalıdır?

▶ **Cevaplar / ipuçları**

Kaynaklar

- J. V. Guttag, *Introduction to Computation and Programming Using Python*, 3. baskı, MIT Press, 2021 (Bölüm 1: bildirimsel ve buyurusal bilgi).
- A. B. Downey ve C. Mayfield, *Think Java*, 2. baskı, O'Reilly, 2019 (Bölüm 1: programlama ve problem çözme).
- R. Sedgewick ve K. Wayne, *Computer Science: An Interdisciplinary Approach*, Addison-Wesley, 2016 (Turing makineleri, hesaplanabilirlik).
- C. Petzold, *Code: The Hidden Language of Computer Hardware and Software*, 2. baskı, 2022.
- A. M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem", *Proc. London Math. Society*, 1936.
- H. M. Deitel ve P. J. Deitel, *Java: How to Program*, 11. baskı, Pearson, 2017.
- Oracle, *The Java Tutorials*: "Language Basics", docs.oracle.com/javase/tutorial.

Son

Bölüm 1.2: Hesaplama ve Hesaplamalı Düşünme

Sonraki bölüm: Bölüm 1.3: Algoritmalar · Terim sözlüğü