Bölüm 1: Giriş

JAVA ile Nesne Yönelimli Programlama

Program Geliştirme Metodolojisi: İlk Algoritma, Sonra Uygulama

- **Adım 1 - Sorunu Tanımlayın**: İlk adım, program geliştirme sürecinin temel taşıdır. Sorunu net bir şekilde tanımlamak, projenin başarısının anahtarıdır. Sorunun ne olduğunu anlamadan, bir çözüm bulmak imkansızdır.
- **Adım 2 - Bir Algoritma Belirleyin**: Sorunu tanımladıktan sonra sıradaki adım, bir çözüm yolu yaratmaktır. Bir algoritma, sorunu çözmek için adım adım yönergeler sunar. Doğru algoritmayı seçmek, programınızın verimli çalışmasını sağlar.
- **Adım 3 - Algoritmayı Kodlamaya Çevirin**: Algoritmanızı, seçtiğiniz programlama dilinde kodlamak için bu adımı kullanacaksınız. Bu aşamada dikkatli ve sistematik bir şekilde çalışmak önemlidir.
- **Adımları Sırayla İzleyin**: Sorunu tanımlamadan algoritma oluşturmadan veya kodlamadan önce, her adımı tamamlamanız gerekmektedir. Bu, program geliştirme sürecinizi daha düzenli ve etkili hale getirir.

Sorunu Tanımlama

- Program geliştirme sürecinin ilk ve en önemli adımlarından biri, sorunu doğru bir şekilde tanımlamaktır.
- Bu adım, projenin başarılı bir şekilde ilerlemesi için hayati bir öneme sahiptir.
- Doğru bir şekilde sorunu tanımlamak, programınızın başarılı bir şekilde ilerlemesine ve doğru sonuçlar üretmesine yardımcı olacaktır.
- Ayrıca, oluşturduğunuz test senaryoları, programınızı doğrulamak ve hataları belirlemek için son derece önemlidir.

Sorunu Tanımlama

- **Adım A - Sorun Tanımını Yazın**
  - Bu adımda, programınızın giriş ve çıkışlarını doğal bir dilde açıklamalısınız. Yani, hangi verilerin programınıza gireceğini ve programınızın ne tür bir çıktı üreteceğini anlatmalısınız.
- **Adım B - Tüm Program İçin Test Senaryoları Oluşturun**
  - Bu adımda, programınızın doğru çalışıp çalışmadığını doğrulamak için kullanabileceğiniz test durumlarını oluşturmalısınız.
  - Bu test senaryoları, programınıza ne tür verilerin gireceğini ve programınızın bu verilere nasıl tepki vermesi gerektiğini belirtmelidir.

Örnek Test Senaryoları

- **Sorun Tanımı**: Bir kullanıcının girdiği iki sayının toplamını hesaplamak.
  - Test Senaryosu 1: Girdi : 3, 5 - Beklenen Çıktı : 8
  - Test Senaryosu 2: Girdi : -2, 7 - Beklenen Çıktı : 5

- **Sorun Tanımı**: Bir metin belgesinin içindeki kelime sayısını bulmak.
  - Test Senaryosu 1: Girdi : "Merhaba, dünya!" - Beklenen Çıktı : 2
  - Test Senaryosu 2: Girdi : "Bu bir denemedir." - Beklenen Çıktı : 3

Bir Algoritma Belirleme

- Program geliştirmenin ikinci önemli adımı, bir algoritma oluşturmayı içerir.
- Bu adım, sorununuzu mantıklı bir şekilde çözmek için gerekli adımları belirlemeyi içerir.
- Bir algoritma oluşturmak ve test etmek, program geliştirme sürecinin önemli bir parçasıdır.
- Algoritmanızı adım adım uygulayarak ve test senaryoları kullanarak, probleminizi daha iyi anlayabilir ve algoritmanızı daha doğru hale getirebilirsiniz.

Bir Algoritma Belirleme

- **Adım A - Algoritmayı Algoritmik Bir Şekilde Uygulayın**
- Algoritma oluşturmanın ilk alt adımı, algoritmayı algoritmik bir şekilde uygulamaktır. Bu, adım adım talimatlar veya tarifler yazmayı içerir.
- **Sorun**: İki sayının toplamını hesaplayın.
- **Algoritma**:
  - İlk sayıyı al.
  - İkinci sayıyı al.
  - İki sayıyı topla.
  - Toplamı ekrana yazdır.
- Bu adımlar, probleminizi mantıklı bir şekilde çözmek için bir rehber sağlar.

Bir Algoritma Belirleme

- **Adım B - Kağıt ve Kalem Kullanarak Test Edin**
- Algoritmanızı oluşturduktan sonra, bir sonraki adım onu test etmektir. Kağıt ve kalem kullanarak test etmek, algoritmanızın mantıklı ve işlevsel olduğunu doğrulamanıza yardımcı olabilir. İşte nasıl yapılacağına dair bazı ipuçları:
  - Küçük ama anlamlı test senaryoları oluşturun.
  - Algoritmayı adım adım takip edin ve her adımın sonucunu yazın.
  - Algoritmayı uygularken düşündüklerinizi fark edin.
  - Yazdığınız adımların gerçekleştirdiklerini gözlemleyin ve gerekirse daha hassas hale getirin.
- Bu adım, algoritmanızın gerçek dünyada nasıl çalıştığını anlamanıza yardımcı olacaktır.

Koda Dönüştürme

- Algoritmanızı bir programlama dilinde kodlamak, fikirlerinizi gerçeğe dönüştürmenin heyecan verici bir kısmıdır.
- Mantıklı bir şekilde kodlama yapmak ve fonksiyonları kullanmak, programınızı daha anlaşılır, bakımı daha kolay ve genel olarak daha etkili hale getirecektir.

Koda Dönüştürme

- **Adım A - Bir Programlama Dilini Kullanarak Uygulayın**
- Algoritmanızı oluşturduktan sonra, sıradaki adım onu bir programlama dilinde kodlamaktır.
- Programlama dili, bilgisayarın anlayabileceği bir dilde algoritmanızı ifade etmenizi sağlar. Bu aşamada, bazı önemli stratejileri kullanarak ilerleyebilirsiniz:
  - Algoritmanızı mantıklı bir şekilde kodlayın, her adımı yazılı kod haline getirin.
  - Kodunuzu okunaklı ve düzenli bir şekilde yazmaya özen gösterin.
  - Programlama dilinizin özgün özelliklerini kullanarak algoritmanızı en etkili şekilde ifade edin.

Koda Dönüştürme

- **Adım B - Mantıksal Ünitelere (Fonksiyonlar) Ayırın**
- Bir programı daha yönetilebilir ve anlaşılır hale getirmenin önemli bir yolu, kodu mantıksal parçalara ayırmaktır.
- Bu parçalar, genellikle "fonksiyonlar" olarak adlandırılır ve belirli bir görevi yerine getiren kod parçalarını içerirler. Bu adımı izlerken dikkate almanız gereken bazı önemli noktalar:
  - Programınızdaki farklı işlevleri tanımlamak için fonksiyonları kullanın.
  - Her fonksiyonun ne yapacağını açıkça belirtin ve adını anlamlı bir şekilde seçin.
  - Fonksiyonlar, kodunuzu daha modüler ve bakımı daha kolay hale getirir.

Örnek Kod Parçacıkları

```java
import java.util.Scanner;

public class SumCalculator {
    public static void main(String[] args) {
        // Create a Scanner object to read user input
        Scanner scanner = new Scanner(System.in);
        // Prompt the user to enter the number
        System.out.print("Enter the first number: ");
        double num1 = scanner.nextDouble();
        System.out.print("Enter the second number: ");
        double num2 = scanner.nextDouble();
        // Calculate the sum of the two numbers
        double sum = num1 + num2;
        // Display the result
        System.out.println("The sum of " + num1 + " and " + num2 + " is: " + sum);
        // Close the Scanner
        scanner.close();
    }
}
```

Örnek Kod Parçacıkları

```
import java.util.Scanner;

public class ToplamaHesaplama {
    public static void main(String[] args) {
        // Kullanıcı girdisini okumak için bir Scanner nesnesi oluşturun
        Scanner tarayici = new Scanner(System.in);
        // Kullanıcıdan sayıyı girmesini isteyin
        System.out.print("Birinci sayıyı giriniz: ");
        double sayi1 = tarayici.nextDouble();
        System.out.print("İkinci sayıyı giriniz: ");
        double sayi2 = tarayici.nextDouble();
        // İki sayının toplamını hesaplayın
        double toplam = sayi1 + sayi2;
        // Sonucu görüntüleyin
        System.out.println(sayi1 + " ile " + sayi2 + " toplamı: " + toplam);
        // Tarayıcıyı kapatın
        tarayici.close();
    }
}
```

Neden Fonksiyonlar?

- Fonksiyonlar, programlarınızı daha anlaşılır, daha sürdürülebilir ve daha verimli hale getiren güçlü bir araçtır.
- Kodunuzu daha iyi düzenlemek, daha hızlı hata ayıklamak ve kodunuzu başka projelerde yeniden kullanmak için fonksiyonları etkili bir şekilde kullanmamız gerekir.
- **Neden 1 - Kod Parçalarını İsimlendirmek**
  - Yeni bir fonksiyon oluşturmak, bir grup ifadeyi (statements) adlandırma fırsatı sunar. Bu, programınızı daha okunabilir ve hata ayıklamasını daha kolay hale getirir. İsimlendirmek, kodun amacını ve işlevini daha açık bir şekilde tanımlar.

Neden Fonksiyonlar?

- **Neden 2 - Tekrarlı Kodları Ortadan Kaldırmak**
  - Fonksiyonlar, tekrarlı kodları ortadan kaldırarak programınızı daha küçük ve düzenli hale getirebilir. Daha sonra bir değişiklik yapmanız gerektiğinde, bu değişikliği sadece bir kez yapmanız yeterli olur. Bu, bakımı kolaylaştırır ve hataları azaltır.
- **Neden 3 - Uzun Programları Parçalara Ayırmak**
  - Uzun bir programı fonksiyonlara bölmek, her parçayı ayrı ayrı hata ayıklamanıza ve daha sonra çalışabilir bir bütün haline getirmenize olanak tanır. Bu, karmaşık projeleri daha yönetilebilir hale getirir ve hata izleme sürecini basitleştirir.
- **Neden 4 - İyi Tasarlanmış Fonksiyonların Yeniden Kullanılabilirliği**
  - İyi tasarlanmış fonksiyonlar genellikle birçok farklı program için kullanışlıdır. Bir kez yazıp hata ayıkladığınızda, bunları başka projelerde yeniden kullanabilirsiniz. Bu, kodunuzu verimli bir şekilde kullanmanıza ve zaman tasarrufu yapmanıza yardımcı olur.

Su Düşüşü Geliştirme Stratejisi

- Su Düşüşü Geliştirme Stratejisi, projeleri aşamalı bir şekilde yönetmek ve denetlemek için güçlü bir araçtır. Bu strateji, projeyi beş ana aşamaya böler ve her aşamanın tamamlanmasının ardından bir sonraki aşamaya geçilir.
- Su Düşüşü Geliştirme Stratejisi beş ana aşamadan oluşur:
  - **İhtiyaç Analizi**: Proje gereksinimleri ve hedefleri belirlenir. Müşteri veya kullanıcı ihtiyaçları anlaşılır.
  - **Tasarım**: Proje tasarlanır. Bu aşamada, yazılımın nasıl çalışacağı ve nasıl görüneceği planlanır.
  - **Geliştirme**: Kodlama aşamasıdır. Yazılımın aslı oluşturulur ve test edilir.
  - **Test ve Kalite Kontrol**: Yazılım, hataları düzeltmek ve istenen işlevselliği sağlamak için test edilir.
  - **Dağıtım ve Bakım**: Yazılım kullanıma sunulur ve kullanıcıların geri bildirimlerine yanıt vermek için bakım yapılır.

Su Düşüşü Geliştirme Stratejisi

- **Avantajlar**
  - Proje ilerlemesi belirli aşamalara ayrıldığı için daha iyi kontrol edilebilir.
  - Her aşama tamamlandığında, sonraki aşamaya geçilmeden önce gereksinimler ve tasarım doğrulanabilir.
  - İyi belirlenmiş gereksinimler, tasarım ve kodlama aşamaları hataların daha erken tanımlanmasını sağlar.
- **Dezavantajlar**
  - Geri dönüşüm zor olabilir. Bir aşama tamamlandığında, değişiklikler maliyetli ve karmaşık olabilir.
  - Proje gereksinimleri başlangıçta eksik veya yanlış anlaşılabilir, bu da projeyi etkileyebilir.
  - Müşteri gereksinimlerinde değişikliklerle başa çıkmak zor olabilir.

İteratif Geliştirme Stratejisi

- İteratif Geliştirme Stratejisi, yazılım projelerini küçük parçalara böler ve her parçayı ayrı ayrı geliştirir.
- Her aşama tamamlandığında, geri bildirim alınır ve projenin bir sonraki aşamasına geçilir.
- Bu strateji, projenin daha esnek ve değişken gereksinimlere yanıt verebilir olmasını sağlar.

Ana Prensipler

- İteratif Geliştirme Stratejisi, şu ana prensiplere dayanır:
  - **Küçük İterasyonlar**: Projeyi küçük ve yönetilebilir parçalara bölme. Her bir iterasyon, belirli bir işlevselliği veya özelliği ele alır.
  - **Geri Bildirim Odaklı**: Her iterasyon sonrası geri bildirim alınır ve bu geri bildirim, projenin gelişimini şekillendirir.
  - **Esneklik**: Değişen gereksinimlere hızlı ve etkili bir şekilde uyum sağlama yeteneği.

İteratif Geliştirme Süreci

- İteratif Geliştirme Stratejisi genellikle aşağıdaki süreci takip eder:
  - **İlk İterasyon**: Temel gereksinimler belirlenir ve ilk prototip oluşturulur.
  - **İterasyonlar**: Küçük geliştirme döngüleri sırasında, yazılımın işlevselliği artırılır ve iyileştirilir.
  - **Geri Bildirim**: Her iterasyon sonrası, kullanıcılar ve paydaşlarla geri bildirim alınır.
  - **Yeniden Değerlendirme ve İterasyon**: Geri bildirimlere dayalı olarak, projenin hedefleri ve gereksinimleri yeniden değerlendirilir ve yeni iterasyonlar planlanır.

İteratif Geliştirme Stratejisi

- **Avantajlar**
  - Değişen gereksinimlere kolayca uyum sağlayabilir.
  - Kullanıcılar ve paydaşlar sürekli olarak projeye dahil edilebilir.
  - Hatalar ve sorunlar daha erken tespit edilir ve düzeltilir.
  - İyileştirmeler ve yeni özellikler daha hızlı eklenir.
- **Dezavantajlar**
  - İterasyonlar arasındaki geçişler zaman ve kaynak gerektirebilir.
  - İterasyonlar çok sayıda olabilir ve yönetim karmaşıklığı yaratabilir.
  - Proje süreci daha esnek olduğu için bazen belirsizlik yaratabilir.

Örnek Algoritmalar

- İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma
- Faktöriyel Bulma
- Fibonacci Serisi Bulma
- Palindrom Kontrolü
- Asal Sayı Kontrolü
- Üs Alma

İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
Başla
    Yaz "Birinci tamsayıyı girin: "
    Oku birinciTamsayı

    Yaz "İkinci tamsayıyı girin: "
    Oku ikinciTamsayı

    Eğer birinciTamsayı < ikinciTamsayı ise
        Yaz birinciTamsayı, " ", ikinciTamsayı
    Değilse
        Yaz ikinciTamsayı, " ", birinciTamsayı
Bitir
```

İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
Başla
    Yaz "Birinci tamsayıyı girin: "
    Oku birinciTamsayı

    Yaz "İkinci tamsayıyı girin: "
    Oku ikinciTamsayı

    Eğer birinciTamsayı < ikinciTamsayı ise
        Yaz birinciTamsayı, " ", ikinciTamsayı
    Değilse
        Yaz ikinciTamsayı, " ", birinciTamsayı
Bitir
```

birinci	ikinci
4	

İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
Başla
    Yaz "Birinci tamsayıyı girin: "
    Oku birinciTamsayı

    Yaz "İkinci tamsayıyı girin: "
    Oku ikinciTamsayı

    Eğer birinciTamsayı < ikinciTamsayı ise
        Yaz birinciTamsayı, " ", ikinciTamsayı
    Değilse
        Yaz ikinciTamsayı, " ", birinciTamsayı
Bitir
```

birinci	ikinci
4	11

İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
Başla
    Yaz "Birinci tamsayıyı girin: "
    Oku birinciTamsayı

    Yaz "İkinci tamsayıyı girin: "
    Oku ikinciTamsayı

    Eğer birinciTamsayı < ikinciTamsayı ise
        Yaz birinciTamsayı, " ", ikinciTamsayı
    Değilse
        Yaz ikinciTamsayı, " ", birinciTamsayı
Bitir
```

birinci	ikinci
4	11

İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
Başla
    Yaz "Birinci tamsayıyı girin: "
    Oku birinciTamsayı

    Yaz "İkinci tamsayıyı girin: "
    Oku ikinciTamsayı

    Eğer birinciTamsayı < ikinciTamsayı ise
        Yaz birinciTamsayı, " ", ikinciTamsayı
    Değilse
        Yaz ikinciTamsayı, " ", birinciTamsayı
Bitir
```

birinci	ikinci
4	11
4 11	

İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
Başla
    Yaz "Birinci tamsayıyı girin: "
    Oku birinciTamsayı

    Yaz "İkinci tamsayıyı girin: "
    Oku ikinciTamsayı

    Eğer birinciTamsayı < ikinciTamsayı ise
        Yaz birinciTamsayı, " ", ikinciTamsayı
    Değilse
        Yaz ikinciTamsayı, " ", birinciTamsayı
Bitir
```

birinci	ikinci
13	

İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
Başla
    Yaz "Birinci tamsayıyı girin: "
    Oku birinciTamsayı

    Yaz "İkinci tamsayıyı girin: "
    Oku ikinciTamsayı

    Eğer birinciTamsayı < ikinciTamsayı ise
        Yaz birinciTamsayı, " ", ikinciTamsayı
    Değilse
        Yaz ikinciTamsayı, " ", birinciTamsayı
Bitir
```

birinci	ikinci
13	11

İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
Başla
    Yaz "Birinci tamsayıyı girin: "
    Oku birinciTamsayı

    Yaz "İkinci tamsayıyı girin: "
    Oku ikinciTamsayı

    Eğer birinciTamsayı < ikinciTamsayı ise
        Yaz birinciTamsayı, " ", ikinciTamsayı
    Değilse
        Yaz ikinciTamsayı, " ", birinciTamsayı
Bitir
```

birinci	ikinci
13	11

İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
Başla
    Yaz "Birinci tamsayıyı girin: "
    Oku birinciTamsayı

    Yaz "İkinci tamsayıyı girin: "
    Oku ikinciTamsayı

    Eğer birinciTamsayı < ikinciTamsayı ise
        Yaz birinciTamsayı, " ", ikinciTamsayı
    Değilse
        Yaz ikinciTamsayı, " ", birinciTamsayı
Bitir
```

birinci	ikinci
13	11

İki Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
Başla
    Yaz "Birinci tamsayıyı girin: "
    Oku birinciTamsayı

    Yaz "İkinci tamsayıyı girin: "
    Oku ikinciTamsayı

    Eğer birinciTamsayı < ikinciTamsayı ise
        Yaz birinciTamsayı, " ", ikinciTamsayı
    Değilse
        Yaz ikinciTamsayı, " ", birinciTamsayı
Bitir
```

birinci	ikinci
13	11
11 13	

Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
Başla
    // İlk tamsayıyı kullanıcıdan al
    Yaz "Birinci tamsayıyı girin: "
    Oku birinciTamsayı

    // İkinci tamsayıyı kullanıcıdan al
    Yaz "İkinci tamsayıyı girin: "
    Oku ikinciTamsayı

    // Üçüncü tamsayıyı kullanıcıdan al
    Yaz "Üçüncü tamsayıyı girin: "
    Oku üçüncüTamsayı
```

Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
    // Tamsayıları küçükten büyüğe sırala ve yazdır
    Eğer birinciTamsayı <= ikinciTamsayı ve birinciTamsayı <= üçüncüTamsayı ise
        Yaz birinciTamsayı
        Eğer ikinciTamsayı <= üçüncüTamsayı ise
            Yaz ikinciTamsayı, " ", üçüncüTamsayı
        Değilse
            Yaz üçüncüTamsayı, " ", ikinciTamsayı
        End Eğer
```

Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
    Değilse Eğer ikinciTamsayı <= birinciTamsayı ve ikinciTamsayı
<= üçüncüTamsayı ise
        Yaz ikinciTamsayı
        Eğer birinciTamsayı <= üçüncüTamsayı ise
            Yaz birinciTamsayı, " ", üçüncüTamsayı
        Değilse
            Yaz üçüncüTamsayı, " ", birinciTamsayı
        End Eğer
```

Üç Tamsayıyı Küçükten Büyüğe Doğru Yazdırma

```
    Değilse
        Yaz üçüncüTamsayı
        Eğer birinciTamsayı <= ikinciTamsayı ise
            Yaz birinciTamsayı, " ", ikinciTamsayı
        Değilse
            Yaz ikinciTamsayı, " ", birinciTamsayı
        End Eğer
    End Eğer

Bitir
```

Faktöriyel Bulma

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sonucu saklamak için bir değişkeni başlatın
    faktoriyel = 1

    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngüyü Bitir

    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

Faktöriyel Bulma

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sonucu saklamak için bir değişkeni başlatın
    faktoriyel = 1

    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngüyü Bitir

    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

sayı	faktoriyel	i
5		

Faktöriyel Bulma

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sonucu saklamak için bir değişkeni başlatın
    faktoriyel = 1

    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngüyü Bitir

    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

sayı	faktoriyel	i
5	1	

Faktöriyel Bulma

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sonucu saklamak için bir değişkeni başlatın
    faktoriyel = 1

    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngüyü Bitir

    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

sayı	faktoriyel	i
5	1	1

Faktöriyel Bulma

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sonucu saklamak için bir değişkeni başlatın
    faktoriyel = 1

    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngüyü Bitir

    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

sayı	faktoriyel	i
5	1	1

Faktöriyel Bulma

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sonucu saklamak için bir değişkeni başlatın
    faktoriyel = 1

    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngüyü Bitir

    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

sayı	faktoriyel	i
5	1	2

Faktöriyel Bulma

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sonucu saklamak için bir değişkeni başlatın
    faktoriyel = 1

    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngüyü Bitir

    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

sayı	faktoriyel	i
5	2	2

Faktöriyel Bulma

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sonucu saklamak için bir değişkeni başlatın
    faktoriyel = 1

    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngüyü Bitir

    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

sayı	faktoriyel	i
5	2	3

Faktöriyel Bulma

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sonucu saklamak için bir değişkeni başlatın
    faktoriyel = 1

    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngüyü Bitir

    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

sayı	faktoriyel	i
5	6	3

Faktöriyel Bulma

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sonucu saklamak için bir değişkeni başlatın
    faktoriyel = 1

    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngüyü Bitir

    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

sayı	faktoriyel	i
5	6	4

Faktöriyel Bulma

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sonucu saklamak için bir değişkeni başlatın
    faktoriyel = 1

    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngüyü Bitir

    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

sayı	faktoriyel	i
5	24	4

Faktöriyel Bulma

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sonucu saklamak için bir değişkeni başlatın
    faktoriyel = 1

    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngüyü Bitir

    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

sayı	faktoriyel	i
5	24	5

Faktöriyel Bulma

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sonucu saklamak için bir değişkeni başlatın
    faktoriyel = 1

    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngüyü Bitir

    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

sayı	faktoriyel	i
5	120	5

Faktöriyel Bulma

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sonucu saklamak için bir değişkeni başlatın
    faktoriyel = 1

    // Faktöriyel hesabı için döngü
    Döngü i = 1'den başlayarak sayı'ya kadar
        faktoriyel = faktoriyel * i
    Döngüyü Bitir

    Yaz "Girilen sayının faktöriyeli: ", faktoriyel
Bitir
```

sayı	faktoriyel	i
5	120	5

Fibonacci Serisi Bulma

```
Başla
    // Bir sayıyı girin
    Input: n

    // İlk iki Fibonacci sayısını başlatın
    a = 0
    b = 1

    // Sonucu saklamak için bir değişkeni başlatın
    sonuc = 0

    Eğer n = 0 ise
        Yaz 0
    Değilse
```

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	0	1		

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	0	1		
0				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	0	1		
0 1				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	0	1	2	
0 1				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	0	1	2	1
0 1				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	1	1	2	1
0 1				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	1	1	2	1
0 1				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	1	1	2	1
0 1 1				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	1	1	3	1
0 1 1				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	1	1	3	2
0 1 1				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	1	1	3	2
0 1 1				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	1	2	3	2
0 1 1				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	1	2	3	2
0 1 1 2				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	1	2	4	2
0 1 1 2				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	1	2	4	3
0 1 1 2				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	2	2	4	3
0 1 1 2				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	2	3	4	3
0 1 1 2				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	2	3	4	3
0 1 1 2 3				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	2	3	5	3
0 1 1 2 3				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	2	3	5	5
0 1 1 2 3				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	3	3	5	5
0 1 1 2 3				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	3	5	5	5
0 1 1 2 3				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	3	5	5	5
0 1 1 2 3 5				

Fibonacci Serisi Bulma

```
    Değilse
        // İlk n Fibonacci sayısını hesaplayın
        Yaz a
        Yaz b
        Döngü i = 2'den başlayarak n - 1'e kadar
            sonuc = a + b
            a = b
            b = sonuc
            Yaz sonuc
        Döngüyü Bitir
    End Eğer

Bitir
```

n	a	b	i	sonuc
5	3	5	5	5
0 1 1 2 3 5				

Palindrom Kontrolü

```
Başla
    // Bir kelime veya cümlenin girilmesini isteyin
    Input: girdiMetin

    // Temizlenmiş girdiyi ters çevirin
    tersGirdi = TersÇevir(girdiMetin)

    // Temizlenmiş girdi ile ters girdiyi karşılaştırın
    Eğer girdiMetin = tersGirdi ise
        Yaz "Girilen metin bir palindromdur."
    Değilse
        Yaz "Girilen metin bir palindrom değildir."
    End Eğer
Bitir
```

Palindrom Kontrolü

```
Başla
    // Bir kelime veya cümlenin girilmesini isteyin
    Input: girdiMetin

    // Temizlenmiş girdiyi ters çevirin
    tersGirdi = TersÇevir(girdiMetin)

    // Temizlenmiş girdi ile ters girdiyi karşılaştırın
    Eğer girdiMetin = tersGirdi ise
        Yaz "Girilen metin bir palindromdur."
    Değilse
        Yaz "Girilen metin bir palindrom değildir."
    End Eğer
Bitir
```

girdiMetin	tersGirdi

Palindrom Kontrolü

```
Başla
    // Bir kelime veya cümlenin girilmesini isteyin
    Input: girdiMetin

    // Temizlenmiş girdiyi ters çevirin
    tersGirdi = TersÇevir(girdiMetin)

    // Temizlenmiş girdi ile ters girdiyi karşılaştırın
    Eğer girdiMetin = tersGirdi ise
        Yaz "Girilen metin bir palindromdur."
    Değilse
        Yaz "Girilen metin bir palindrom değildir."
    End Eğer
Bitir
```

girdiMetin	tersGirdi
abcba	

Palindrom Kontrolü

```
Başla
    // Bir kelime veya cümlenin girilmesini isteyin
    Input: girdiMetin

    // Temizlenmiş girdiyi ters çevirin
    tersGirdi = TersÇevir(girdiMetin)

    // Temizlenmiş girdi ile ters girdiyi karşılaştırın
    Eğer girdiMetin = tersGirdi ise
        Yaz "Girilen metin bir palindromdur."
    Değilse
        Yaz "Girilen metin bir palindrom değildir."
    End Eğer
Bitir
```

girdiMetin	tersGirdi
abcba	abcba

Palindrom Kontrolü

```
Başla
    // Bir kelime veya cümlenin girilmesini isteyin
    Input: girdiMetin

    // Temizlenmiş girdiyi ters çevirin
    tersGirdi = TersÇevir(girdiMetin)

    // Temizlenmiş girdi ile ters girdiyi karşılaştırın
    Eğer girdiMetin = tersGirdi ise
        Yaz "Girilen metin bir palindromdur."
    Değilse
        Yaz "Girilen metin bir palindrom değildir."
    End Eğer
Bitir
```

girdiMetin	tersGirdi
abcba	abcba

Palindrom Kontrolü

```
Başla
    // Bir kelime veya cümlenin girilmesini isteyin
    Input: girdiMetin

    // Temizlenmiş girdiyi ters çevirin
    tersGirdi = TersÇevir(girdiMetin)

    // Temizlenmiş girdi ile ters girdiyi karşılaştırın
    Eğer girdiMetin = tersGirdi ise
        Yaz "Girilen metin bir palindromdur."
    Değilse
        Yaz "Girilen metin bir palindrom değildir."
    End Eğer
Bitir
```

girdiMetin	tersGirdi
abcba	abcba
Girilen metin bir palindromdur	

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

Asal Sayı Kontrolü

```
    // Bayrağa (flag) göre sonucu yazdırın
    Eğer asalMi = True ise
        Yaz sayı, " bir asal sayıdır."
    Değilse
        Yaz sayı, " bir asal sayı değildir."
    End Eğer

Bitir
```

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
7		

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
7	True	

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
7	True	2

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
7	True	2

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
7	True	3

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
7	True	3

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
7	True	3

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
21		

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
21	True	

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
21	True	2

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
21	True	2

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
21	True	3

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
21	True	3

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
21	False	3

Asal Sayı Kontrolü

```
Başla
    // Bir sayıyı girin
    Input: sayı

    // Sayının asal olup olmadığını kontrol etmek için bir bayrak (flag) başlatın
    asalMi = True

    // 2'den başlayarak sayının yarısına kadar olan tüm bölenleri kontrol et
    Döngü bölen = 2'den başlayarak sayı / 2'e kadar
        Eğer sayı % bölen = 0 ise
            // Bölünüyorsa, sayı asal değildir
            asalMi = False
            Döngüyü Kır
        End Eğer
    Döngüyü Bitir
```

sayı	asalMi	bölen
7	False	3

Üs Alma

```
Başla
    // Bir sayıyı girin
    Input: taban
    // Bir üssü girin
    Input: üs

    // Sonucu saklamak için bir değişkeni başlatın
    sonuc = 1

    // Üssü kullanarak sonucu hesaplayın
    Döngü i = 1'den başlayarak üs kadar
        sonuc = sonuc * taban

    Yaz "Sonuç: ", sonuc
Bitir
```

Üs Alma

```
Başla
    // Bir sayıyı girin
    Input: taban
    // Bir üssü girin
    Input: üs

    // Sonucu saklamak için bir değişkeni başlatın
    sonuc = 1

    // Üssü kullanarak sonucu hesaplayın
    Döngü i = 1'den başlayarak üs kadar
        sonuc = sonuc * taban

    Yaz "Sonuç: ", sonuc
Bitir
```

taban	üs	sonuc	i
3			

Üs Alma

```
Başla
    // Bir sayıyı girin
    Input: taban
    // Bir üssü girin
    Input: üs

    // Sonucu saklamak için bir değişkeni başlatın
    sonuc = 1

    // Üssü kullanarak sonucu hesaplayın
    Döngü i = 1'den başlayarak üs kadar
        sonuc = sonuc * taban

    Yaz "Sonuç: ", sonuc
Bitir
```

taban	üs	sonuc	i
3	4		

Üs Alma

```
Başla
    // Bir sayıyı girin
    Input: taban
    // Bir üssü girin
    Input: üs

    // Sonucu saklamak için bir değişkeni başlatın
    sonuc = 1

    // Üssü kullanarak sonucu hesaplayın
    Döngü i = 1'den başlayarak üs kadar
        sonuc = sonuc * taban

    Yaz "Sonuç: ", sonuc
Bitir
```

taban	üs	sonuc	i
3	4	1	

Üs Alma

```
Başla
    // Bir sayıyı girin
    Input: taban
    // Bir üssü girin
    Input: üs

    // Sonucu saklamak için bir değişkeni başlatın
    sonuc = 1

    // Üssü kullanarak sonucu hesaplayın
    Döngü i = 1'den başlayarak üs kadar
        sonuc = sonuc * taban

    Yaz "Sonuç: ", sonuc
Bitir
```

taban	üs	sonuc	i
3	4	1	1

Üs Alma

```
Başla
    // Bir sayıyı girin
    Input: taban
    // Bir üssü girin
    Input: üs

    // Sonucu saklamak için bir değişkeni başlatın
    sonuc = 1

    // Üssü kullanarak sonucu hesaplayın
    Döngü i = 1'den başlayarak üs kadar
        sonuc = sonuc * taban

    Yaz "Sonuç: ", sonuc
Bitir
```

taban	üs	sonuc	i
3	4	3	1

Üs Alma

```
Başla
    // Bir sayıyı girin
    Input: taban
    // Bir üssü girin
    Input: üs

    // Sonucu saklamak için bir değişkeni başlatın
    sonuc = 1

    // Üssü kullanarak sonucu hesaplayın
    Döngü i = 1'den başlayarak üs kadar
        sonuc = sonuc * taban

    Yaz "Sonuç: ", sonuc
Bitir
```

taban	üs	sonuc	i
3	4	3	2

Üs Alma

```
Başla
    // Bir sayıyı girin
    Input: taban
    // Bir üssü girin
    Input: üs

    // Sonucu saklamak için bir değişkeni başlatın
    sonuc = 1

    // Üssü kullanarak sonucu hesaplayın
    Döngü i = 1'den başlayarak üs kadar
        sonuc = sonuc * taban

    Yaz "Sonuç: ", sonuc
Bitir
```

taban	üs	sonuc	i
3	4	9	2

Üs Alma

```
Başla
    // Bir sayıyı girin
    Input: taban
    // Bir üssü girin
    Input: üs

    // Sonucu saklamak için bir değişkeni başlatın
    sonuc = 1

    // Üssü kullanarak sonucu hesaplayın
    Döngü i = 1'den başlayarak üs kadar
        sonuc = sonuc * taban

    Yaz "Sonuç: ", sonuc
Bitir
```

taban	üs	sonuc	i
3	4	9	3

Üs Alma

```
Başla
    // Bir sayıyı girin
    Input: taban
    // Bir üssü girin
    Input: üs

    // Sonucu saklamak için bir değişkeni başlatın
    sonuc = 1

    // Üssü kullanarak sonucu hesaplayın
    Döngü i = 1'den başlayarak üs kadar
        sonuc = sonuc * taban

    Yaz "Sonuç: ", sonuc
Bitir
```

taban	üs	sonuc	i
3	4	27	3

Üs Alma

```
Başla
    // Bir sayıyı girin
    Input: taban
    // Bir üssü girin
    Input: üs

    // Sonucu saklamak için bir değişkeni başlatın
    sonuc = 1

    // Üssü kullanarak sonucu hesaplayın
    Döngü i = 1'den başlayarak üs kadar
        sonuc = sonuc * taban

    Yaz "Sonuç: ", sonuc
Bitir
```

taban	üs	sonuc	i
3	4	27	4

Üs Alma

```
Başla
    // Bir sayıyı girin
    Input: taban
    // Bir üssü girin
    Input: üs

    // Sonucu saklamak için bir değişkeni başlatın
    sonuc = 1

    // Üssü kullanarak sonucu hesaplayın
    Döngü i = 1'den başlayarak üs kadar
        sonuc = sonuc * taban

    Yaz "Sonuç: ", sonuc
Bitir
```

taban	üs	sonuc	i
3	4	81	4

Üs Alma

```
Başla
    // Bir sayıyı girin
    Input: taban
    // Bir üssü girin
    Input: üs

    // Sonucu saklamak için bir değişkeni başlatın
    sonuc = 1

    // Üssü kullanarak sonucu hesaplayın
    Döngü i = 1'den başlayarak üs kadar
        sonuc = sonuc * taban

    Yaz "Sonuç: ", sonuc
Bitir
```

taban	üs	sonuc	i
3	4	81	4

Arama Algoritmaları

- Arama algoritmaları, bir koleksiyon içinde belirli bir öğenin varlığını aramak veya bulmak için kullanılır.
- Örneğin, bir listede belirli bir sayının varlığını kontrol etmek için arama algoritmaları kullanılır.
- Verileri etkili bir şekilde işlemek ve değerli bilgileri çıkarmak için temel bir işlemdir.
- Programlarımızın belirli öğeleri hızlı ve doğru bir şekilde bulmasına yardımcı olur.
- Örnek Kullanım Alanları: Kitapların bir kütüphanedeki yerini bulma, İnternet üzerinde bir web sitesinde belirli bir sayfayı bulma, Bir veritabanında belirli bir kaydı arama.

Arama Algoritmaları

- Problem Tanımı

- Girdi:
  - Nesnelerin bir koleksiyonu, ona "Sepet" diyelim.
  - Belirli bir nesne, ona «Elma» diyelim.
- Çıktı:
  - Eğer «Elma», "Sepet" içindeyse "True".
  - Eğer «Elma», "Sepet" içinde değilse "False".

Arama Algoritmaları

- Rastgele (Random)
- Doğrusal (Lineer)
- İkili (Binary)

Rastgele Arama

- Rastgele arama, bir koleksiyon içindeki bir öğeyi bulmak için rastgele bir şekilde öğeleri kontrol eden bir arama algoritmasıdır.
- Her adımda bir öğeyi rastgele seçer ve hedefi bulana veya arama alanı tükenene kadar devam eder.
- Büyük veri koleksiyonlarında hızlı bir şekilde arama yapmak için kullanılabilir.
- Avantajlar: Basit ve hızlıdır, veri koleksiyonunun sırası önemli değildir.
- Dezavantajlar: En kötü durumda veri koleksiyonunun tamamını incelemek gerekebilir, etkili olmayabilir.

Rastgele Arama

- Nasıl Çalışır?
  - Rastgele bir öğe seç.
  - Seçilen öğeyi hedefle karşılaştır.
  - Eğer hedef bulunduysa işlemi sonlandır.
  - Eğer hedef bulunamazsa adımları tekrarla.

Rastgele Arama

- Soru: Rastgele Arama, Arama Problemini Çözer mi?
  - Rastgele arama her zaman etkili bir çözüm sağlamaz.
  - Rastgele arama, öğeyi bulma olasılığını artırabilir, ancak kesin bir sonuç garantisi vermez.
- Soru: Eğer Öğe "Sepet" İçerisinde Değilse Ne Olur?
  - Eğer «Elma» öğesi "Sepet" içinde değilse, rastgele arama işlemi sona erebilir ve «Elma» bulunamayabilir.
  - Bu nedenle, rastgele arama kullanırken öğenin koleksiyon içinde olup olmadığını doğrulamak önemlidir.

Rastgele Arama

- Elimizde rastgele bir dizi olsun: [77, 42, 64, 12, **23**, 55, 7, 48, 31]
- Aramak istediğimiz öğe: 48
- Adım 1:
- Dizi üzerinde rastgele bir eleman seçin ve aranan öğe (48) ile karşılaştırın.
- Örneğin, ilk seçilen eleman 23'tür, bu nedenle 23 ile 48'i karşılaştırın.

Rastgele Arama

- Elimizde rastgele bir dizi olsun: [77, **42**, 64, 12, 23, 55, 7, 48, 31]
- Aramak istediğimiz öğe: 48
- Adım 2:
- Aranan öğe (48) henüz bulunamadı, çünkü seçilen eleman 23'tür.
- Dizi üzerinde bir başka rastgele eleman seçin ve tekrar karşılaştırın.
- Bu sefer seçilen eleman 42'dir.

Rastgele Arama

- Elimizde rastgele bir dizi olsun: [77, 42, 64, 12, 23, 55, **7**, 48, 31]
- Aramak istediğimiz öğe: 48
- Adım 3:
- 42 ile 48'i karşılaştırın.
- Aranan öğe (48) henüz bulunamadı, çünkü seçilen eleman 42'dir.
- Yeni bir rastgele eleman seçin.

Rastgele Arama

- Elimizde rastgele bir dizi olsun: [77, 42, 64, 12, 23, 55, 7, **48**, 31]
- Aramak istediğimiz öğe: 48
- Adım 4:
- Bu adımları tekrarlayarak aranan öğeyi (48) bulana kadar devam edin.
- Örneğin, bir sonraki seçilen eleman 48 olduğunda aranan öğeyi bulduk.

Doğrusal Arama

- Doğrusal arama, bir koleksiyon içinde bir öğeyi bulmanın en temel yoludur.
- Adından da anlaşılacağı gibi, öğeleri sırayla kontrol ederek aradığınız öğeyi bulmaya çalışır.
- Küçük koleksiyonlarda etkili bir çözümdür.
- Öğelerin sıralı olması gerekmez, herhangi bir sırayla bulunabilirler.
- Avantajlar: Basit ve anlaşılır, herhangi bir sırayla çalışabilir.
- Dezavantajlar: Büyük koleksiyonlarda yavaş olabilir, her öğeyi kontrol etmesi gerekebilir.

Doğrusal Arama

- Nasıl Çalışır?
  - Öğeleri bir listeye koyun
  - İlk öğeden başlayarak sırayla hedef öğe ile karşılaştırın.
  - Eğer hedef öğe bulunursa işlemi sonlandırın ve bulunan öğeyi dönün.
  - Eğer hedef öğe bulunmazsa bir sonraki öğeye geçin.
  - Listenin sonuna kadar devam edin.

Doğrusal Arama

- Soru: Doğrusal Arama, Arama Problemini Çözer mi?
  - Evet, Doğrusal Arama herhangi bir liste için ve herhangi bir öğe için Arama problemini çözecektir!
  - Doğrusal Arama, belirli bir öğeyi bulmak için her öğeyi sırayla kontrol eder.
  - Hedef öğeyi bulana veya tüm öğeleri kontrol edene kadar devam eder.

Doğrusal Arama

- Elimizde sıralanmamış bir dizi olsun: [**42**, 17, 8, 23, 31]

- Adım 1:

- Aranacak öğeyi belirleyin, örneğin 23'ü arıyoruz.
- Dizinin ilk elemanından başlayın (42) ve aranan öğe ile karşılaştırın.

Doğrusal Arama

- Elimizde sıralanmamış bir dizi olsun: [42, **17**, 8, 23, 31]

- Adım 2:

- İlk eleman (42) aradığımız öğe (23) ile eşleşmez.
- Bir sonraki elemana geçin ve 17 ile karşılaştırın.

Doğrusal Arama

- Elimizde sıralanmamış bir dizi olsun: [42, 17, **8**, 23, 31]

- Adım 3:

- İkinci eleman (17) de aradığımız öğe ile eşleşmez.
- Bir sonraki elemana geçin ve 8 ile karşılaştırın.

Doğrusal Arama

- Elimizde sıralanmamış bir dizi olsun: [42, 17, 8, **23**, 31]

- Adım 4:

- Üçüncü eleman (8) de aradığımız öğe ile eşleşmez.
- Bir sonraki elemana geçin ve 23 ile karşılaştırın.

Doğrusal Arama

- Elimizde sıralanmamış bir dizi olsun: [42, 17, 8, **23**, 31]

- Adım 5:

- Dördüncü eleman (23), aradığımız öğe ile eşleşir.
- Aradığımız öğeyi bulduk ve bu adımda arama işlemi sona erer.

İkili Arama

- İkili arama, bir koleksiyon içindeki bir öğeyi bulmanın hızlı ve etkili bir yoludur.
- Ancak, verilerin sıralı (küçükten büyüğe veya büyükten küçüğe) olması gerekmektedir.
- Büyük veri koleksiyonlarında hızlı bir şekilde arama yapmak için idealdir.
- Veriler sıralı olduğunda daha etkilidir, çünkü her adımda yarıya kadar öğeleri elemeyi sağlar.
- Avantajlar: Hızlı, büyük veri koleksiyonlarında etkilidir.
- Dezavantajlar: Veriler sıralı olmalıdır, koleksiyonun yeniden düzenlenmesi gerekebilir.

İkili Arama

- Nasıl Çalışır?
  - Ortadaki öğeyi seçin ve hedefle karşılaştırın.
  - Eğer hedef öğe bulunursa işlemi sonlandırın.
  - Eğer hedef öğe, ortadaki öğeden küçükse, sol yarıya odaklanın ve adımları tekrarlayın.
  - Eğer hedef öğe, ortadaki öğeden büyükse, sağ yarıya odaklanın ve adımları tekrarlayın.
  - Hedef öğe bulunana veya arama alanı tükenene kadar bu adımları tekrarlayın.

İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, **42**, 48, 55, 64, 77]
- Aramak istediğimiz öğe: 48
- Adım 1:
- Dizinin ortasındaki elemanı (42) alın ve aranan öğe (48) ile karşılaştırın.

İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, 42, 48, 55, 64, 77]
- Aramak istediğimiz öğe: 48
- Adım 2:
- 48, ortadaki elemandan (42) büyüktür.
- Bu nedenle, aranan öğenin dizinin sağ yarısında olduğunu anlarız.
- Sol yarı artık arama için kullanılmayacak.

İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, 42, 48, 55, 64, 77]
- Aramak istediğimiz öğe: 48
- Adım 3:
- Sağ yarıyı ele alın: [48, 55, 64, 77]

İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, 42, 48, 55, **64**, 77]
- Aramak istediğimiz öğe: 48
- Adım 4:
- Sağ yarının ortasındaki elemanı (64) alın ve aranan öğe (48) ile karşılaştırın.

İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, 42, 48, 55, 64, 77]
- Aramak istediğimiz öğe: 48
- Adım 5:
- 48, ortadaki elemandan (64) küçüktür.
- Bu nedenle, aranan öğenin dizinin sol yarısında olduğunu anlarız.
- Sağ yarı artık arama için kullanılmayacak.

İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, 42, 48, 55, 64, 77]
- Aramak istediğimiz öğe: 48
- Adım 6:
- Sol yarıyı ele alın: [48, 55]

İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, 42, **48**, 55, 64, 77]
- Aramak istediğimiz öğe: 48
- Adım 7:
- Sol yarının ortasındaki elemanı (48) alın ve aranan öğe (48) ile karşılaştırın.

İkili Arama

- Elimizde sıralanmış bir dizi olsun: [7, 12, 23, 31, 42, **48**, 55, 64, 77]
- Aramak istediğimiz öğe: 48
- Adım 8:
- Aranan öğe (48) bulundu.
- Arama işlemi başarıyla tamamlandı.

İkili Arama vs. Lineer Arama: Karşılaştırma

- Lineer arama, bir koleksiyon içinde bir öğeyi sırayla kontrol ederek aramaya çalışır. Her adımda bir öğeyi kontrol eder ve hedefi bulana veya koleksiyon sonuna kadar devam eder.
- İkili arama, sıralı verilerde hedef öğeyi daha hızlı bulmak için kullanılır. Verilerin sıralı (küçükten büyüğe veya büyükten küçüğe) olması gerekir.
- Lineer Arama: Büyük koleksiyonlarda yavaş olabilir, her öğeyi kontrol etmesi gerekebilir. İkili Arama: Büyük koleksiyonlarda hızlıdır, her adımda yarıya kadar öğeleri eleme avantajı vardır.
- Örnek Kullanım:
  - Lineer Arama: Telefon rehberinde bir kişiyi bulma.
  - İkili Arama: Sözlük içinde bir kelimeyi bulma.

Sıralama Algoritmaları

- Sıralama algoritmaları, bir veri koleksiyonundaki öğeleri belirli bir düzene göre düzenlemek için kullanılır.
- Bu düzenleme genellikle öğeleri küçükten büyüğe veya büyükten küçüğe sıralamak şeklinde olur.
- Verileri düzenlemek ve sıralamak, veri analizi ve arama işlemleri için temel bir adımdır.
- Sıralı veriler, arama işlemlerini hızlandırabilir.

Sıralama Algoritmaları

- Rastgele Sıralama (Random)
- Seçmeli Sıralama (Selection Sort)
- Kabarcık Sıralama (Bubble Sort)

Rastgele Sıralama

- Rastgele sıralama, bir veri koleksiyonunu rastgele bir düzende sıralama işlemidir.
- Bu sıralama yöntemi, verilerin rastgele karıştırılmasını sağlar.
- Rastgele sıralama, veri koleksiyonunu rastgele hale getirerek verileri çeşitlendirmek ve analiz etmek için kullanılır.
- Bazı algoritmaların başlangıç verilerini rastgele sıralama işlemi ile iyileştirmesinde kullanılır.
- Nasıl Çalışır?
  - Veri koleksiyonundaki öğeleri rastgele bir şekilde karıştırır.
  - Öğelerin sıralamasını tamamen rastgele yapar, önceki sıralamayla ilgilenmez.
- Avantajlar: Verileri rastgele sıralayarak daha çeşitli sonuçlar elde edebilirsiniz.
- Dezavantajlar: Verilerin doğru bir şekilde sıralanmasını sağlamaz, bazı durumlarda kullanışsız olabilir.

Rastgele Sıralama

- Elimizde sıralanmamış bir dizi olsun: [42, 17, 8, 23, 31]

- Adım 1:

- Dizi elemanlarını rastgele bir sırayla düzenleyin.
- Her düzenleme farklı bir sonuç üretecektir. Örneğin, dizi aşağıdaki gibi bir düzende olabilir: [31, 17, 42, 8, 23]

Rastgele Sıralama

- Elimizde sıralanmamış bir dizi olsun: [31, 17, 42, 8, 23]

- Adım 2:

- Dizi elemanlarını tekrar rastgele bir sırayla düzenleyin.
- Bu sefer farklı bir düzenleme olabilir: [8, 23, 31, 42, 17]

Rastgele Sıralama

- Elimizde sıralanmamış bir dizi olsun: [8, 23, 31, 42, 17]

- Adım 3:

- Dizi elemanlarını tekrar rastgele bir sırayla düzenleyin.
- Bu sefer farklı bir düzenleme olabilir: [8, 23, 17, 31, 42]

Seçmeli Sıralama

- Seçmeli sıralama, bir veri koleksiyonundaki öğeleri sırayla karşılaştırarak sıralamak için kullanılan basit bir sıralama algoritmasıdır.
- Bu algoritma, en küçük veya en büyük öğeyi bulup sırayla yer değiştirerek sıralamayı gerçekleştirir.
- Avantajlar: Basit ve anlaşılır, küçük veri koleksiyonlarında etkili.
- Dezavantajlar: Büyük veri koleksiyonlarında yavaş, daha verimli algoritmalar mevcut.

Seçmeli Sıralama

- Nasıl Çalışır?
  - Veri koleksiyonundaki en küçük (veya en büyük) öğeyi bulun.
  - Bu öğeyi sıralı koleksiyonun başına taşıyın.
  - Koleksiyonun bir sonraki bölümünü ele alın ve en küçük (veya en büyük) öğeyi bulun.
  - Bu öğeyi sıralı koleksiyonun ikinci pozisyonuna taşıyın.
  - Bu adımları koleksiyonun sonuna kadar tekrarlayın.

Seçmeli Sıralama

- Elimizde sıralanmamış bir dizi olsun: [64, 25, 12, 22, **11**, 36, 48]

- Adım 1:

- Sıralanmamış kısmın içindeki en küçük elemanı bulun (bu durumda 11).
- İlk elemanla (64) yer değiştirin.
- Dizi birinci adımdan sonra: [**11**, 25, 12, 22, 64, 36, 48]

Seçmeli Sıralama

- Elimizde sıralanmamış bir dizi olsun: [**11**, 25, **12**, 22, 64, 36, 48]
- Adım 2:
- İkinci en küçük elemanı bulun (bu durumda 12) ve ikinci elemanla yer değiştirin.
- Dizi ikinci adımdan sonra: [**11, 12**, 25, 22, 64, 36, 48]

Seçmeli Sıralama

- Elimizde sıralanmamış bir dizi olsun: [**11, 12**, 25, **22**, 64, 36, 48]
- Adım 3:
- Üçüncü en küçük elemanı bulun (bu durumda 22) ve üçüncü elemanla yer değiştirin.
- Dizi üçüncü adımdan sonra: [**11, 12, 22**, 25, 64, 36, 48]

Seçmeli Sıralama

- Elimizde sıralanmamış bir dizi olsun: [**11, 12, 22**, **25**, 64, 36, 48]
- Adım 4:
- Dördüncü en küçük elemanı bulun (bu durumda 25) ve dördüncü elemanla yer değiştirin.
- Dizi dördüncü adımdan sonra: [**11, 12, 22, 25**, 64, 36, 48]

Seçmeli Sıralama

- Elimizde sıralanmamış bir dizi olsun: [**11, 12, 22, 25**, 64, **36**, 48]
- Adım 5:
- Beşinci en küçük elemanı bulun (bu durumda 36) ve beşinci elemanla yer değiştirin.
- Dizi beşinci adımdan sonra: [**11, 12, 22, 25, 36**, 64, 48]

Seçmeli Sıralama

- Elimizde sıralanmamış bir dizi olsun: [**11, 12, 22, 25, 36**, 64, **48**]
- Adım 6:
- Altıncı en küçük elemanı bulun (bu durumda 48) ve altıncı elemanla yer değiştirin.
- Dizi altıncı adımdan sonra: [**11, 12, 22, 25, 36, 48**, 64]

Seçmeli Sıralama

- Elimizde sıralanmamış bir dizi olsun: [**11, 12, 22, 25, 36**, **48**, 64]
- Adım 7:
- Dizinin sonuna ulaşıldığı için değişiklik yok.
- Dizi yedinci adımdan sonra: [**11, 12, 22, 25, 36, 48, 64**]

Kabarcık Sıralama

- Kabarcık sıralama, bir veri koleksiyonundaki öğeleri sırayla karşılaştırarak sıralamak için kullanılan basit bir sıralama algoritmasıdır.
- Bu algoritma, yan yana duran öğeleri sırayla karşılaştırarak en büyük öğeyi koleksiyonun sonuna doğru "kabarcık" gibi taşır.
- Avantajlar: Basit ve anlaşılır, küçük veri koleksiyonlarında etkili.
- Dezavantajlar: Büyük veri koleksiyonlarında çok yavaş, daha verimli algoritmalar mevcut.

Kabarcık Sıralama

- Nasıl Çalışır?
  - Koleksiyonun başından itibaren başlayın.
  - İki yan yana öğeyi karşılaştırın ve gerektiğinde yerlerini değiştirin, büyük olan öğeyi sağa taşıyın.
  - Koleksiyonun sonuna kadar bu adımları tekrarlayın.
  - En büyük öğe, koleksiyonun sonuna taşındığı için son öğeyi sıralanmış kabul edin.
  - İlk dört adımı koleksiyonun sonuna kadar tekrarlayarak sıralama işlemi tamamlanır.

Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [**64, 25**, 12, 22, 11, 36, 48]

- Adım 1:

- Dizinin ilk iki elemanını karşılaştırın (64 ve 25).
- 25, 64'ten küçük olduğu için bu iki elemanın yerini değiştirin.
- Dizi birinci adımdan sonra: [**25, 64**, 12, 22, 11, 36, 48]

Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [25, **64, 12**, 22, 11, 36, 48]

- Adım 2:

- Şimdi ikinci ve üçüncü elemanları karşılaştırın (64 ve 12).
- 12, 64'ten küçük olduğu için bu iki elemanın yerini değiştirin.
- Dizi ikinci adımdan sonra: [25, **12, 64**, 22, 11, 36, 48]

Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [25, 12, **64, 22**, 11, 36, 48]

- Adım 3:

- Üçüncü ve dördüncü elemanları karşılaştırın (64 ve 22).
- 22, 64'ten küçük olduğu için bu iki elemanın yerini değiştirin.
- Dizi üçüncü adımdan sonra: [25, 12, **22, 64**, 11, 36, 48]

Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [25, 12, 22, **64, 11**, 36, 48]

- Adım 4:

- Dördüncü ve beşinci elemanları karşılaştırın (64 ve 11).
- 11, 64'ten küçük olduğu için bu iki elemanın yerini değiştirin.
- Dizi dördüncü adımdan sonra: [25, 12, 22, **11, 64**, 36, 48]

Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [25, 12, 22, 11, **64, 36**, 48]

- Adım 5:

- Beşinci ve altıncı elemanları karşılaştırın (64 ve 36).
- 36, 64'ten küçük olduğu için bu iki elemanın yerini değiştirin.
- Dizi beşinci adımdan sonra: [25, 12, 22, 11, **36, 64**, 48]

Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [25, 12, 22, 11, 36, **64, 48**]

- Adım 6:

- Altıncı ve yedinci elemanları karşılaştırın (64 ve 48).
- 48, 64'ten küçük olduğu için bu iki elemanın yerini değiştirin.
- Dizi altıncı adımdan sonra: [25, 12, 22, 11, 36, **48, 64**]

Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [**25, 12**, 22, 11, 36, 48, 64]

- Adım 7:

- Dizinin ilk iki elemanını karşılaştırın (25 ve 12).
- 12, 25'ten küçük olduğu için bu iki elemanın yerini değiştirin.
- Dizi 7. adımdan sonra: [**12, 25**, 22, 11, 36, 48, 64]

Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [12, **25, 22**, 11, 36, 48, 64]

- Adım 8:

- Dizinin ikinci ve üçüncü elemanını karşılaştırın (25 ve 22).
- 22, 25'ten küçük olduğu için bu iki elemanın yerini değiştirin.
- Dizi 8. adımdan sonra: [12, **22, 25,** 11, 36, 48, 64]

Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [12, 22, **25, 11**, 36, 48, 64]

- Adım 9:

- Dizinin üçüncü ve dördüncü elemanını karşılaştırın (25 ve 11).
- 11, 25'ten küçük olduğu için bu iki elemanın yerini değiştirin.
- Dizi 9. adımdan sonra: [12, 22, **11, 25**, 36, 48, 64]

Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [12, 22, 11, **25, 36**, 48, 64]

- Adım 10:

- Dizinin dördüncü ve beşinci elemanını karşılaştırın (25 ve 36).
- 36, 25'ten büyük olduğu için değişiklik yok.
- Dizi 10. adımdan sonra: [12, 22, 11, **25, 36**, 48, 64]

Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [12, 22, 11, 25, **36, 48**, 64]

- Adım 11:

- Dizinin beşinci ve altıncı elemanını karşılaştırın (36 ve 48).
- 48, 36'dan büyük olduğu için değişiklik yok.
- Dizi 11. adımdan sonra: [12, 22, 11, 25, **36, 48**, 64]

Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [12, **22, 11**, 25, 36, 48, 64]

- Adım 13:

- Dizinin ikinci ve üçüncü elemanını karşılaştırın (22 ve 11).
- 11, 22'den küçük olduğu için iki elemanın yerini değiştirin.
- Dizi 13. adımdan sonra: [12, **11, 22**, 25, 36, 48, 64]

Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [**12, 11**, 22, 25, 36, 48, 64]

- Adım 16:

- Dizinin birinci ve ikinci elemanını karşılaştırın (12 ve 11).
- 11, 12'den küçük olduğu için iki elemanın yerini değiştirin.
- Dizi 16. adımdan sonra: [**11, 12,** 22, 25, 36, 48, 64]

Kabarcık Sıralama

- Elimizde sıralanmamış bir dizi olsun: [**11, 12**, 22, 25, 36, 48, 64]

- Adım 21:

- Dizinin birinci ve ikinci elemanını karşılaştırın (11 ve 12).
- 12, 11'den büyük olduğu için değişiklik yok.
- Dizi 21. adımdan sonra: [**11, 12, 22, 25, 36, 48, 64**]

SON