ALGORİTMA NEDİR ?

Algoritma, belli bir problemi çözmek veya belirli bir amaca ulaşmak için tasarlanan yol. [Matematikte](https://tr.wikipedia.org/wiki/Matematik) ve [bilgisayar biliminde](https://tr.wikipedia.org/wiki/Bilgisayar_bilimi) bir işi yapmak için tanımlanan, bir başlangıç durumundan başladığında, açıkça belirlenmiş bir son durumunda sonlanan, sonlu işlemler kümesidir. Genellikle [bilgisayar programlamada](https://tr.wikipedia.org/wiki/Programlama) kullanılır ve tüm [programlama dillerinin](https://tr.wikipedia.org/wiki/Programlama_dili) temeli algoritmaya dayanır.

ÖNCE ALGORİTMA MI AKIŞ DİYAGRAMI MI ?

Program geliştirmenin başlangıcında, algoritmanın hazırlanması ve ardından bir akış diyagramı çizilmesi iyi bir uygulamadır. İkisi arasındaki tercih, genellikle kişisel veya kurumsal tercihlere, projenin karmaşıklığına ve geliştirme sürecine bağlıdır.

Algoritma hazırlarken dikkat edilmesi gereken birçok önemli husus vardır. İşte bu hususlardan bazıları:

1. **Problem Tanımı:** Algoritma geliştirmeden önce, çözmek istediğiniz problemi açık ve net bir şekilde tanımlamalısınız. Problemin gereksinimlerini ve kısıtlamalarını anlamak, doğru bir algoritma oluşturmanın temelidir.
2. **Giriş ve Çıkışlar:** Algoritmanızın hangi verileri alacağını ve hangi sonuçları üreteceğini belirlemelisiniz. Giriş ve çıkışların formatı ve türleri de önemlidir.
3. **Veri Yapıları:** Algoritmanızın verileri nasıl saklayacağınızı ve işleyeceğinizi belirlemelisiniz. Veri yapıları, algoritmanın performansını büyük ölçüde etkileyebilir.
4. **Karmaşıklık ve Verimlilik:** Algoritmanızın zaman ve hafıza karmaşıklığını dikkate almalısınız. En iyi durumda, algoritmanız sorunu en az işlem ve hafıza kullanımıyla çözmelidir.
5. **Mantık ve Akış:** Adım adım algoritmanın mantığını düşünmelisiniz. Hangi adımların hangi sırayla çalışacağını belirlemek önemlidir. Koşullu ifadeler, döngüler ve işlem sırası bu bölümde düşünülmelidir.
6. **Hata Kontrolü:** Algoritmanızın hata ve istisna durumlarına nasıl yanıt vereceğini düşünmelisiniz. Hata kontrolleri, algoritmanızın sağlam ve güvenilir olmasını sağlar.
7. **Optimizasyon:** Algoritmanızın daha hızlı veya daha az kaynak kullanacak şekilde nasıl optimize edilebileceğini düşünmelisiniz. Bu, büyük veri kümeleri veya yüksek işlem gücü gerektiren uygulamalarda önemlidir.
8. **Test ve Sorun Giderme:** Algoritmanızı geliştirdikten sonra, çeşitli senaryoları test etmeli ve olası hataları gidermelisiniz. Test aşaması, algoritmanın doğru çalıştığını doğrulamanızı sağlar.
9. **Belgelendirme:** Algoritmanızın kullanımını ve nasıl çalıştığını açıklayan belgelendirme oluşturmalısınız. Bu, başkalarının algoritmanızı anlamalarına ve kullanmalarına yardımcı olur.
10. **İsimlendirme ve Temiz Kod:** Değişkenlerin, fonksiyonların ve adımların açık ve anlaşılır bir şekilde isimlendirilmesine dikkat etmelisiniz. Temiz, düzenli ve okunabilir bir kod yazmak önemlidir.

Algoritma geliştirme süreci, problemi iyi anlamayı, tasarım yapmayı ve kod yazmayı içerir. Bu adımları dikkatli bir şekilde takip ederek daha etkili ve güvenilir algoritmalar oluşturabilirsiniz.

DEĞİŞKEN NEDİR ?

Programlama bağlamında, bir değişken, bir değeri saklayabilen bir veri saklama birimidir. Bu değer, sayı, metin, dizi, nesne gibi farklı türlerde olabilir. Değişkenler, program içinde bilgileri depolamak, manipüle etmek ve kullanmak için kullanılır.

NEDEN DEĞİŞKENLERE İHTİYAÇ DUYULUR ?

1. **Veri Saklama:** Değişkenler, program içinde çeşitli veri türlerini (sayılar, metinler, listeler, vb.) saklamak için kullanılır. Örneğin, kullanıcı tarafından girilen bir sayıyı ya da bir metni bir değişkende saklayabiliriz.
2. **Veri İşleme:** Programlar, genellikle verileri işler. Bu işlemler sırasında geçici değerlerin saklanması ve manipüle edilmesi gerekebilir. Değişkenler, bu tür geçici değerleri saklamak için kullanılır.
3. **Esneklik:** Değişkenler, programın çalışma zamanında değerlerini değiştirebilme yeteneği sağlar. Bu, programın kullanıcı girişine veya diğer çeşitli koşullara dinamik olarak uyum sağlamasına olanak tanır.
4. **Hesaplamalar:** Değişkenler, matematiksel hesaplamalarda kullanılabilir. Örneğin, bir değişken içinde saklanan sayıları kullanarak matematiksel operasyonlar gerçekleştirebiliriz.
5. **Kodun Anlaşılabilirliği:** Değişkenler, kodu daha okunabilir ve anlaşılır hale getirebilir. Değişken isimleri, programın hangi amaçla kullanıldığını açıkça ifade edebilir.
6. **Kodun Tekrar Kullanılabilirliği:** Değişkenler, aynı değerleri veya hesaplamaları tekrar tekrar kullanma ihtiyacını ortadan kaldırarak kodun tekrar kullanılabilirliğini artırır.
7. **Bellek Yönetimi:** Değişkenler, programın bellek yönetimini sağlar. Program çalışma sırasında bellekte yer ayırır ve kullanır.

SAYAÇLARA NERELERDE VE NİÇİN İHTİYAÇ DUYULUR ?

Sayaçlar (veya counter'lar), programlarda belirli olayları saymak veya izlemek için kullanılan değişkenlerdir. Sayaçlar, çeşitli durumları, işlemleri veya olayları saymak, izlemek veya kontrol etmek amacıyla kullanılır. İşte sayaçların kullanıldığı bazı yaygın senaryolar:

1. **Döngülerde Kullanım:** Sayaçlar, döngülerde belirli bir durumun veya işlemin kaç kez tekrarlandığını izlemek için kullanılır. Örneğin, bir döngüyü belirli bir sayıda tekrar etmek için sayaç kullanılabilir.
2. **Veri Toplama ve İstatistiksel Analizlerde Kullanım:** Programlar veri toplarken, belirli olayları izlerken veya kullanıcı davranışlarını analiz ederken sayaçlar kullanılabilir. Örneğin, bir web sitesinde belirli bir sayfayı ziyaret eden kullanıcı sayısını izlemek için bir sayaç kullanılabilir.
3. **Performans Ölçümlerinde Kullanım:** Programların çalışma süresini veya performansını ölçmek için sayaçlar kullanılabilir. Belirli bir işlemin ne kadar sürede tamamlandığını ölçmek için zamanlayıcı olarak kullanılabilir.
4. **Koşullu İfadelerde Kullanım:** Sayaçlar, belirli bir koşul gerçekleştiğinde kaç defa gerçekleştiğini izlemek için kullanılabilir. Örneğin, bir oyunda oyuncunun kaç kez bir seviyeyi geçtiğini saymak için sayaç kullanılabilir ve belirli bir sayıya ulaşıldığında özel bir ödül veya yetenek verilebilir.
5. **Hata Takibinde Kullanım:** Programlarda hataların ne sıklıkta gerçekleştiğini veya belirli bir hata mesajının kaç kez görüntülendiğini takip etmek için sayaçlar kullanılabilir. Bu, hata ayıklama ve programın güvenilirliği açısından önemli olabilir.
6. **Dosya ve Veritabanı İşlemlerinde Kullanım:** Dosya veya veritabanı işlemleri yapılırken, kaç kez belirli bir veri okundu, yazıldı veya güncellendiğini izlemek için sayaçlar kullanılabilir.

Sayaçlar, programın belirli durumları izlemesini, saymasını ve bu durumları yönetmesini sağlayarak programların daha etkili ve kontrollü bir şekilde çalışmasına yardımcı olur. Bu, programların kullanımı ve performansı açısından önemlidir.

ALGORİTMA ÇÖZÜMLERİ

**6.Soru Çözümleri:**

T=0 S=0

T=0+(2\*0)=0 S=0+2=2

T=0+(2\*2)=4 S=2+2=4

T=4+(2\*4)=12 S=4+2=6

T=12

**7.Soru Çözümleri:**

F=1 S=20

S=20-3=17 F=17+1=18 F=18+2=20

S=17-3=14 F=20+14=34 F=34+2=36

S=14-3=11 F=36+11=47 F=47+2=49

S=11-3=8 F=49+8=57 F=57+2=59

S=8-3=5 F=59+5=64 F=64+2=66

S=5-3=2 F=66+2=68 F=68+2=70

S=2-2=0 F=70+0=70 F=70+2=72

F=72

3 SAYIDAN EN BÜYÜĞÜNÜ BULAN ALGORİTMA

DEĞİŞKENLER  
S = Birinci sayı x  
M = İkinci sayı y  
T = Üçüncü sayı z  
Max = En büyük değerdir.  
  
ALGORİTMA  
Adım 1 = Başla  
Adım 2 = S oku  
Adım 3 = M oku  
Adım 4 = T oku  
Adım 5 = s > = m ve s > = t ise s e maximun değer ver ve 8. adıma git.  
Adım 6 =  m > = s ve m > = t ise m ye maximun değer ver ve 8. adıma git.  
Adım 7 = T ye maximun değer ver ve 8. adıma git.  
Adım 8 = max değerini yazdır.  
Adım 9 =  Bitir.

3 SAYIYI BÜYÜKTEN KÜÇÜĞE SIRALAYAN

1.Başla

2. Birinci sayıyı "a" olarak tanımla

3. lkinci sayıyı "x" olarak tanimla

4. Uçüncü sayıyı "c" olarak tanımla

5, Eger "a"> "x" ise "a" ile "x" i yer değiştir

6. Eger "a" > "c" ise "a" ile "c" yi yer değiştir

7.Eger "x"> "c" ise "x" ile "c" yi yer değiştir

8. "sayi1," "sayi2" ve "sayi3" artik küçükten büyüğe sıralanmıştir.

9. Sonuçları yazdır

10, Dur.

1-99 ARASINDAKİ TEK VE ÇİFT SAYILARIN TOPLAMI

1. Basla

2.toplam\_tek=0

3.toplam.cift=0

4.carpm\_tek=1

5. carpim\_çift= 1

6. Sayi=1

7. Döngü başlatı Sayı 1'den 99'a kadar artarken

8. Eger Sayı çiftse o zaman sonraki adıma git

9. toplam\_çift= toplam\_çift-+ Sayı

10. çarpim\_çift = carpim\_cift \* Sayı

11.Değilse

12. toplam\_tek = toplam\_lek+Say

13. çarpim\_tek= çarpim\_tek\* Sayı

14. Döngüyü sonlandır

15. Sonuçlan yazdır.

16, "Tek sayilarin toplamı:" toplam\_tek

17. "Çift sayiların toplamı" toplam\_çift

18. Tek sayiların çarpımı: carpim\_tek

19. "Çift sayilann çarpımı" çarpim\_çift

20.Dur.