



# Python ile Programlamaya Giriş

Özcan GÜNDEŞ



<https://www.linkedin.com/in/%C3%B6zcan-g%C3%BCnde%C5%9F-7693055b/>



ozcangundes@gmail.com



<https://github.com/ozcangundes>



# ALGORİTMAYA GİRİŞ

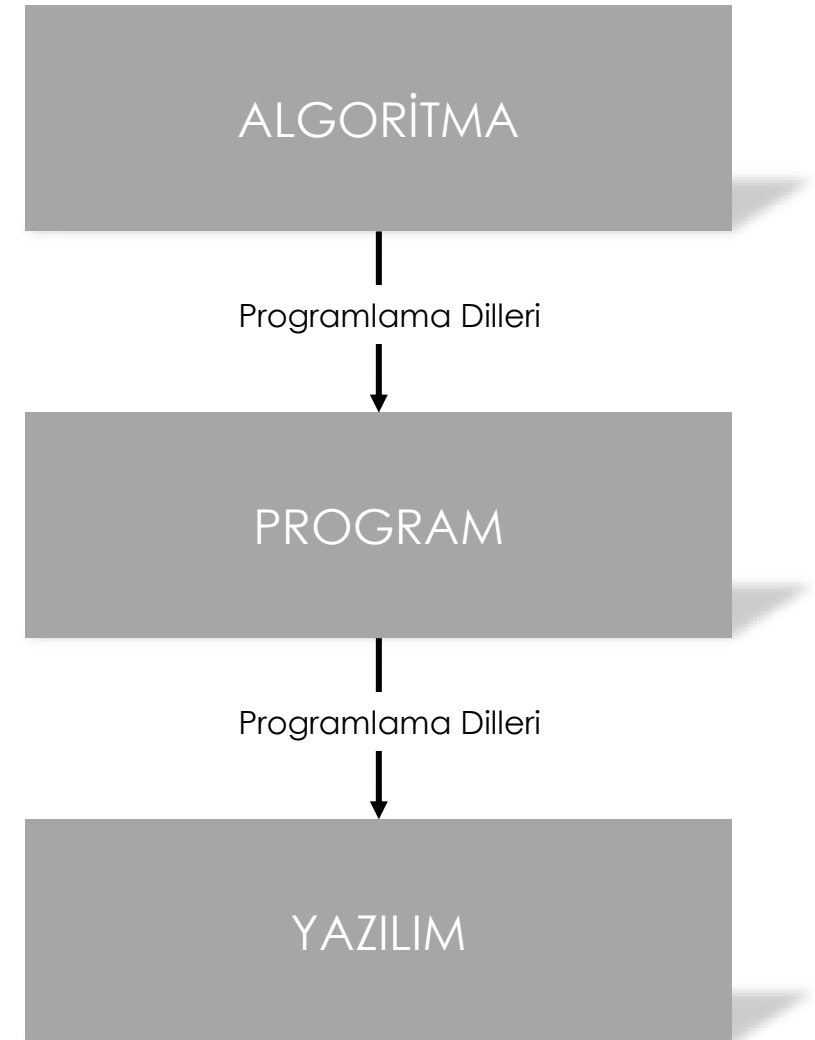
# Algoritma Temelleri

# Algoritma Nedir?

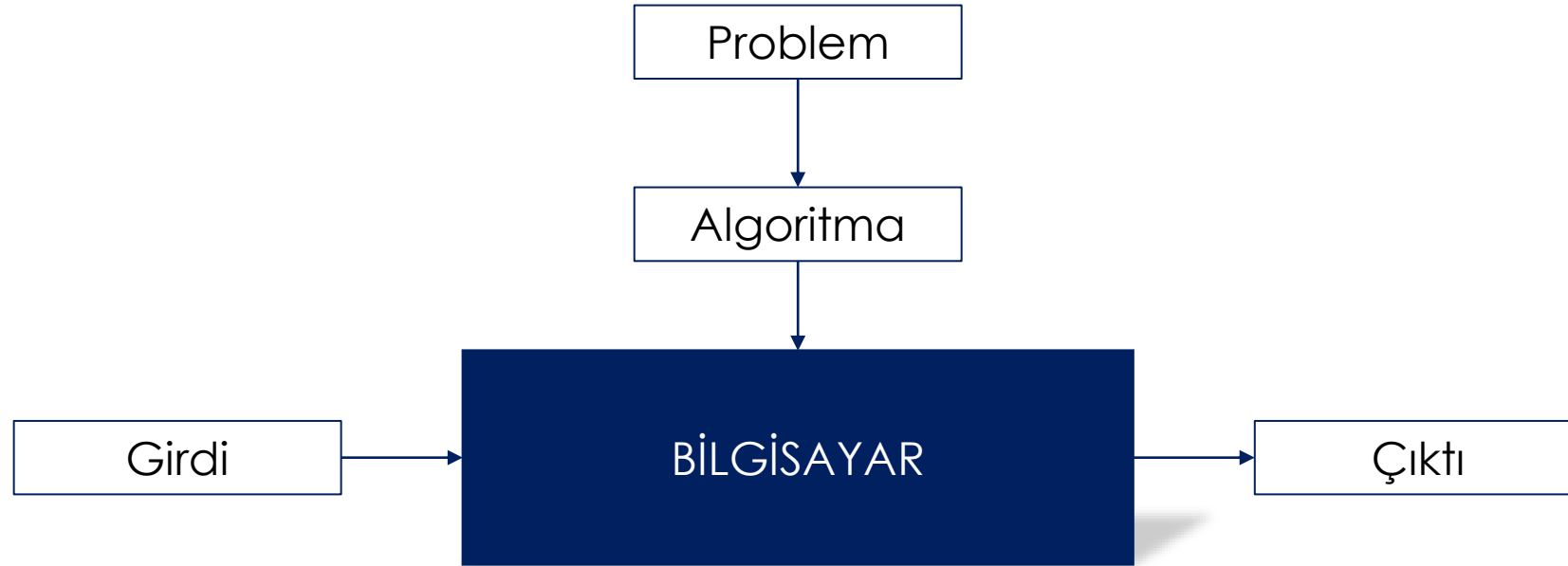
- Algoritma kavramı, ilk olarak 800'lü yıllarda yaşamış olan Acem matematikçi Muhammad ibn Musa al-Khwärizmi'nin yaptığı çalışmalarda ortaya konmuştur. 12. yüzyılda bu çalışmalar Latince'ye çevrilirken, al-Kharizmi'nin adından ötürü yaptığı bu çalışma “algorithm” olarak çevrilmiştir. Türkçe'ye ise *algoritma* olarak girmiştir.
- Görüldüğü gibi bu kavram, bilgisayar dünyasına girmeden önce, matematik alanındaki problemlerin çözümü için kullanılmaktaydı. Daha sonra bilgisayarların geliştirilmesiyle bu alandaki problemlerin çözümünde de kullanılmaya başladı.
- *Buradan anlaşılacağı üzere, algoritma, spesifik bir problemi çözme amacıyla takip edilen ve **sonlu sayıda** adımdan oluşan sözel bir çözüm yoludur.*

# Algoritma Nedir?

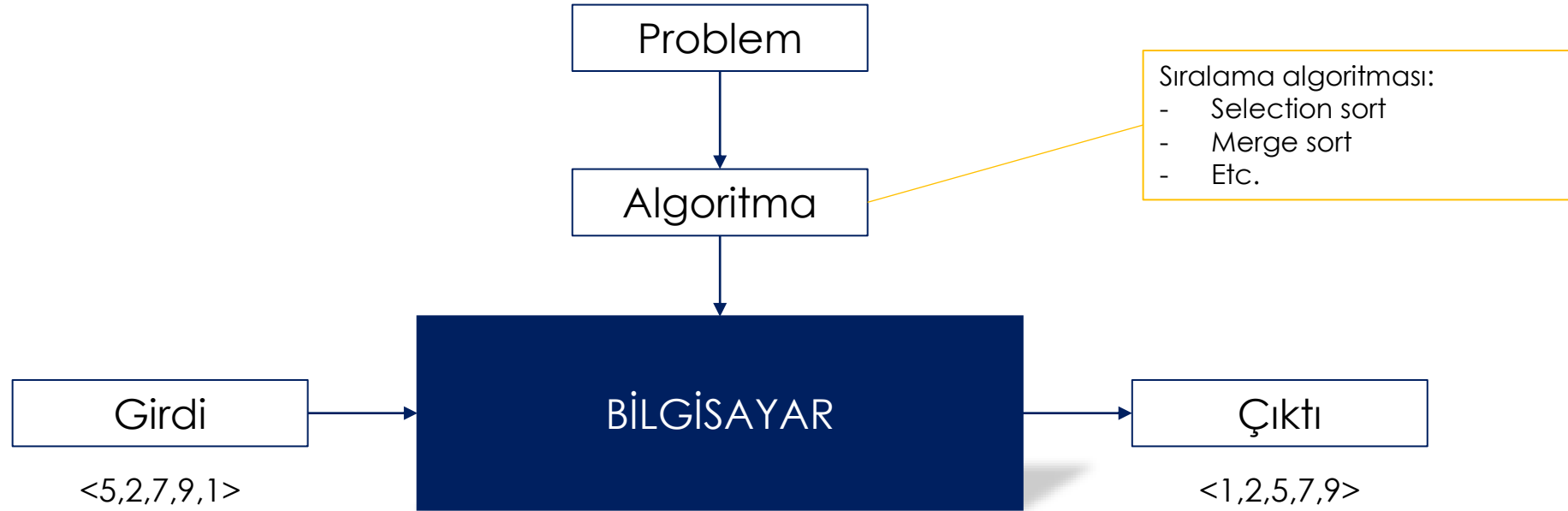
- Algoritması oluşturulmuş bir problemin çözümüyle birlikte bilgisayar ortamına aktarılmış haline **program** denir.
- Algoritmaların kodlanarak program haline getirilmesi için **programlama dilleri** kullanılır.
- **Programlama dilleri** kullanılarak yazılımlar geliştirilir.



# Algoritma Nedir?



# Algoritma Nedir? - Örnek



# Algoritma Özellikleri

## 1. Kesin Olma:

Algoritma içindeki adımlar herkes tarafından aynı şekilde anlaşılıyor olmalı, farklı anlamlara gelebilecek ifadeler içermemelidir.

## 2. Sıralı Olma:

Her algoritma için bir başlangıç durumu söz konusudur. Çözüm, bu başlangıç durumu göz önünde bulundurularak gerçekleştirilir. Adımların hangi sırada gerçekleştirileceği net bir şekilde belirtilmelidir.

## 3. Sonlu Olma:

Algoritma sonlu sayıda adımdan oluşmalı, sınırlı bir zaman diliminde tamamlanmalıdır. Her algoritmanın bir son noktası, bitişi olmalıdır.



# Algoritma - Basit Örnek

Algoritmalar problemin çözümü için gereken adımları içeren sözel ifadelerden oluşur. Telefon uygulamasından online yemek siparişi vermek için bir algoritma oluşturabiliriz.

Adım 1: Başla

Adım 2: Telefon kilidini aç

Adım 3: İlgili mobil uygulamayı aç

Adım 4: Yemeğini ve restoranını seç

Adım 5: Ödeme yapmak için kart bilgilerini gir

Adım 6: Ödemeyi tamamla

Adım 7: Adres bilgilerini gir

Adım 8: Sipariş ver

Adım 9: Dur



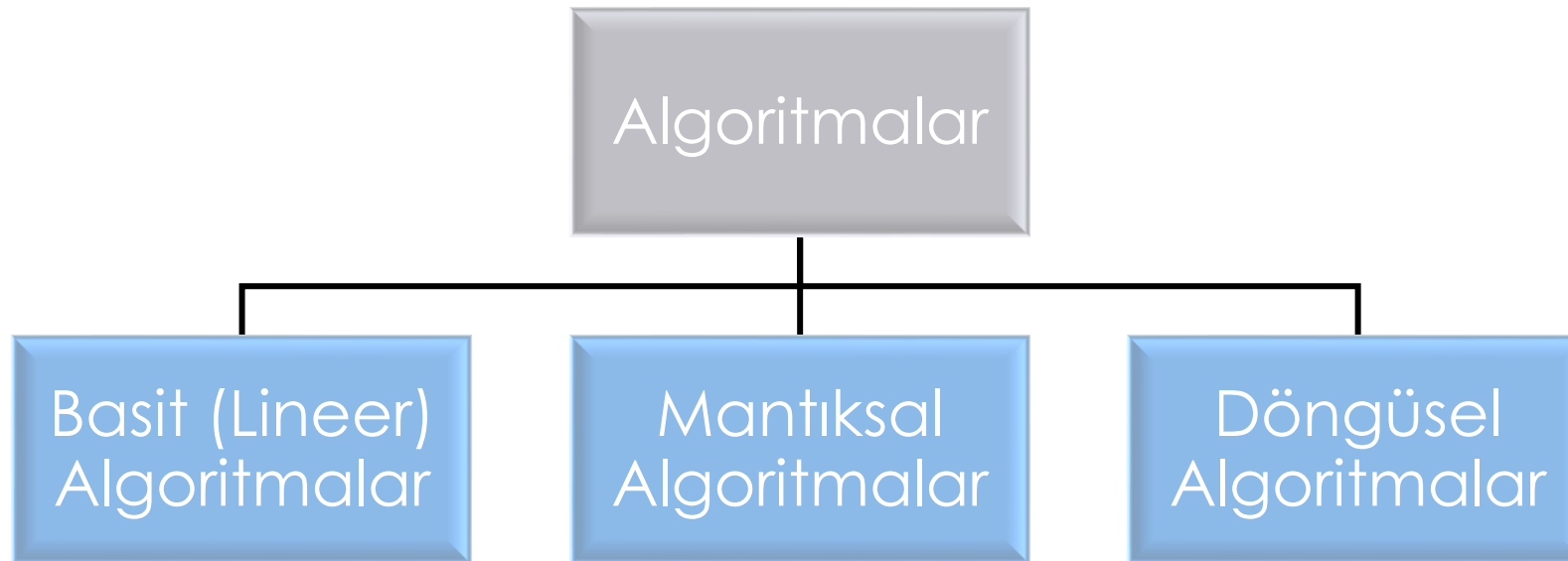
Kaynak: <https://merodomain.com/top-5-reasons-why-your-restaurant-needs-an-online-ordering-system/>

# Algoritma - Basit Örnek

Bir problemi çözmenin birden çok yolu olabilir, dolayısıyla aynı probleme yönelik farklı algoritmalar geliştirebilirsiniz. Önemli olan kesin olma, sıralı olma ve sonlu olma özelliklerine uymasındır.



Evinize bir robot aldığınızı ve ona, çoğumuzun yapmayı öğrendiği ilk yemek olan makarna yapma algoritması programlamanız gerektiğini düşünün. Algoritma adımlarınız nasıl olurdu?



# Basit (Lineer) Algoritmalar

- Basit algoritmalar içerisinde program akış dallanmaları yoktur, yani baştan sona düz bir akış olacaktır.
- Genelde küçük hesaplamaları yapmak için kullanılır ve içerisinde bir karar yapısı bulunmamaktadır.
- İkinci el online bir e-ticaret sitesinde satış yaptığınızı ve ilgili siteye %10 komisyon ödediğinizi düşünürsek, satış yaptıktan sonra elinize geçen parayı hesaplayan bir basit (lineer) algoritma yazabiliriz.

Adım 1: Başla

Adım 2: Toplam satış tutarınızı giriniz; toplam

Adım 3: Girilen değeri 0.1 komisyon değeri ile çarpınız;  $kom=0.1*toplam$

Adım 4: Girilen değerden komisyon tutarını çıkarınız;  $satis=toplam-kom$  ''

Adım 5: Hesaplanan değeri ekrana yazdırınız; satis

Adım 6: Dur



“”: Komisyon tutarını göstermek istemiyorsak 3. ve 4.adımları birleştirerek, toplam tutarı 0.9 ile çarpabiliriz. Dolayısıyla ekstra bir adım daha işlenmesinin önüne geçerek bilgisayar işlem kaynaklarını daha verimli kullanabiliriz.

# Basit (Lineer) Algoritmalar - Örnek

- Girilen 3 sayının karelerinin ortalamasının karekökünü hesaplayan algoritma:

Adım 1: Başla

Adım 2: Üç adet sayı giriniz; x,y,z

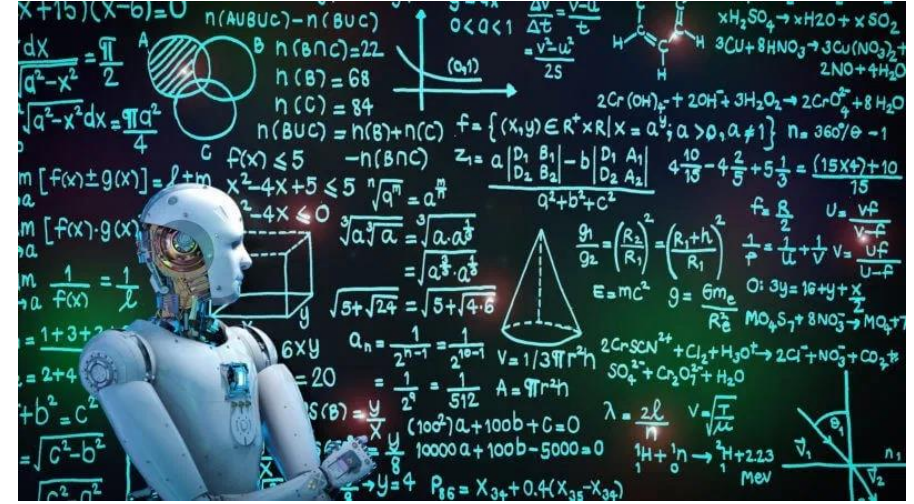
Adım 3: Sayıların karelerinin toplamını hesaplayınız;  $\text{toplam} = x^2 + y^2 + z^2$

Adım 4: Sayıların karelerinin ortalamalarını hesaplayınız;  $\text{ort} = \text{toplam} / 3$

Adım 5: Ortalamanın karekökünü hesaplayınız;  $\text{kok} = \text{ort}^{(1/2)}$

Adım 6: Hesaplanan değeri ekrana yazdırınız; kok

Adım 7: Dur



# Mantıksal Algoritmalar

- Bu tip algoritmaların içerisinde mantıksal karşılaştırmalar bulunur (“eğer” =  $,<,>$ , and, or). Bu karşılaştırmalar sonucunda algoritma akışı farklı adımlara geçecektir.
- Basit algoritmalarda hep birer adım ilerlerken, mantıksal algoritmalarda karşılaştırmalara göre birden fazla adım atlayabilir.
- Girilen 3 sayı içerisinde en küçüğünü bulan algoritma yazalım;

Adım 1: Başla

Adım 2: Üç adet sayı giriniz;  $x,y,z$

Adım 3: En küçük sayıya  $x$  değerini ver;  $kucuk=x$

Adım 4: Eğer  $y$ , en küçükten küçük ( $y < kucuk$ ) ise en küçük sayı  $y$  olsun;  $kucuk=y$

Adım 5: Eğer  $z$ , en küçükten küçük ( $z < kucuk$ ) ise en küçük sayı  $z$  olsun;  $kucuk=z$

Adım 6: En küçük sayıyı ekrana yazdırınız;  $kucuk$

Adım 7: Dur



# Mantıksal Algoritmalar - Örnek

- Girilen sayının işaretini bulmaya yarayan algoritma:

Adım 1: Başla

Adım 2: Bir sayı giriniz; x

Adım 3: Eğer x sayısı sıfırdan büyük ise ekrana “pozitif” yazdır ve adım 6'ya git

Adım 4: Eğer x sayısı sıfırdan küçük ise ekrana “negatif” yazdır ve adım 6'ya git

Adım 5: Eğer x sayısı sıfıra eşit ise ekrana “sıfır” yazdır ve adım 6'ya git

Adım 6: Dur



# Döngüsel Algoritmalar

- Algoritma içerisindeki bir adım birden fazla kez tekrarlanıyorsa döngüsel algoritmadır.
- Döngüsel algoritma içerisinde mantıksal karşılaştırma yapıları kullanılır. Fakat uygulanan mantıksal karşılaştırma sonucunda program akışı ileriye doğru değil, önceki adımlara gidiyorsa bu döngüsel algoritmadır.
- Yani mantıksal karşılaştırma sonucu program hep ileriye doğru gidiyorsa mantıksal algoritma, önceki adımlara tekrar gidiyorsa döngüsel algoritma kullanılır!

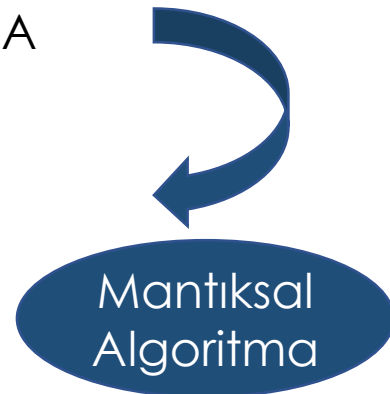
Adım 1:

Adım 2:

Adım 3: MANTIKSAL KARŞILAŞTIRMA

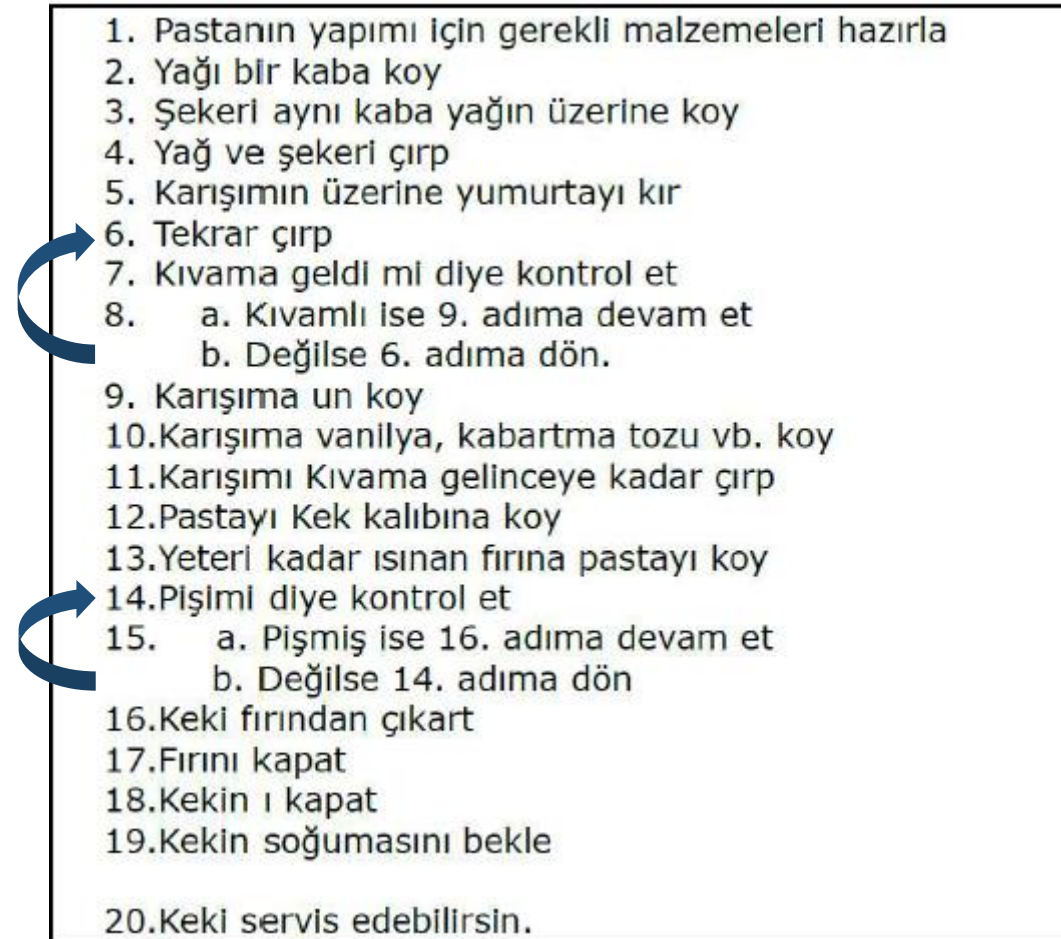
Adım 4:

Adım 5:





# Döngüsel Algoritmalar - Örnek



# Döngüsel Algoritmalar

- Girilen sayının faktöriyelini bulmaya yarayan algoritma:

Adım 1: Başla

Adım 2: Faktöriyeli hesaplanacak sayı giriniz; x

Adım 3: Faktöriyel değerini 1 yap;  $f=1$

Adım 4: İndeks değerini 1 yap;  $i=1$

Adım 5: Faktöriyel değeri ile indeks değerini çarp ve faktöriyel değerine ata;  $f=f*i$

Adım 6: İndeks değerini 1 arttır;  $i=i+1$

Adım 7: Eğer indeks değeri girilen sayıdan küçük veya eşitse Adım 5'e git

Adım 8: Faktöriyel değerini ekrana yazdır; f

Adım 9: Dur

# Döngüsel Algoritmalar

- Girilen sayının faktöriyelini bulmaya yarayan algoritma:

Adım 1: Başla

Adım 2: Faktöriyeli hesaplanacak sayı giriniz; x

Adım 3: Faktöriyel değerini 1 yap;  $f=1$

Adım 4: İndeks değerini 1 yap;  $i=1$

Adım 5: Faktöriyel değeri ile indeks değerini çarp ve faktöriyel değerine ata;  $f=f*i$

Adım 6: İndeks değerini 1 arttır;  $i=i+1$

Adım 7: Eğer indeks değeri girilen sayıdan küçük veya eşitse Adım 5'e git

Adım 8: Faktöriyel değerini ekrana yazdır; f

Adım 9: Dur

**ÖRNEK:** x değeri 5 olsun;  $x=5$

İndeks (i)	Eski F	Yeni F
1	1	$1*1=1$
2	1	$1*2=2$
3	2	$2*3=6$
4	6	$6*4=24$
5	24	$24*5=120$

# Döngüsel Algoritmalar

Girilen iki sayının en büyük ortak bölenini (EBOB) bulan algoritma yazalım.

- Adım 1: Sayıları giriniz;  $a$ ,  $b$
- Adım 2: Eğer  $a > b$  ise  $b$ 'yi  $a$ 'dan çıkar ve tekrar  $a$ 'ya yaz ( $a = a - b$ ) ve Adım 2'ye git
- Adım 3: Eğer  $b > a$  ise  $a$ 'yı  $b$ 'den çıkar ve tekrar  $b$ 'ye yaz ( $b = b - a$ ) ve Adım 2'ye git
- Adım 4: Eğer  $a = b$  ise adım 5'e git
- Adım 5: En büyük bölen  $a$  değerini ekrana yaz;  $a$

# Döngüsel Algoritmalar

Girilen iki sayının en büyük ortak bölenini (EBOB) bulan algoritma yazalım.

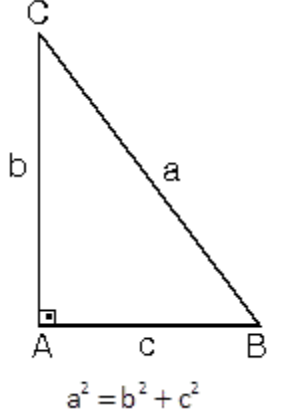
- Adım 1: Sayıları giriniz;  $a$ ,  $b$
- Adım 2: Eğer  $a > b$  ise  $b$ 'yi  $a$ 'dan çıkar ve tekrar  $a$ 'ya yaz ( $a = a - b$ ) ve Adım 2'ye git
- Adım 3: Eğer  $b > a$  ise  $a$ 'yı  $b$ 'den çıkar ve tekrar  $b$ 'ye yaz ( $b = b - a$ ) ve Adım 2'ye git
- Adım 4: Eğer  $a = b$  ise adım 5'e git
- Adım 5: En büyük bölen  $a$  değerini ekrana yaz;  $a$

ÖRNEK:  $a=30$ ,  $b=12$  olsun

<b>a</b>	<b>b</b>	<b>Yeni a</b>	<b>Yeni b</b>
30	12	$30-12=18$	12
18	12	$18-12=6$	12
6	12	6	$12-6=6$
<b>6</b>	<b>6</b>		

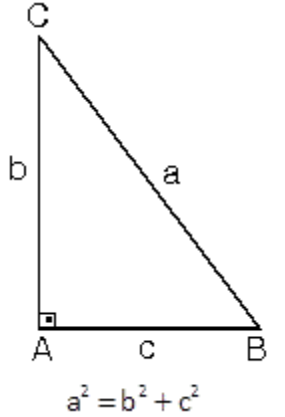
# Örnek Soru-1

- İki dik kenarını bildiğimiz bir üçgenin, Pisagor teoremine göre hipotenüsünü bulmamızı sağlayacak algoritma;
  - a) Hangi algoritma tipine girer?
  - b) Nasıl yazılabilir?



# Örnek Soru-1 Çözüm

- İki dik kenarını bildiğimiz bir üçgenin, Pisagor teoremine göre hipotenüsünü bulmamızı sağlayacak algoritma;
  - a) Hangi algoritma tipine girer? **BASİT (LINEER)**
  - b) Nasıl yazılabilir?



**Adım 1: Başla**

**Adım 2: Dik kenarları giriniz; b,c**

**Adım 3: Dik kenarların karelerinin toplamını hesaplayınız; toplam= $b^2+c^2$**

**Adım 4: Toplamın karekökünü alınız;  $a = \text{toplam}^{(1/2)}$**

**Adım 5: a değerini ekrana yazdırınız; a**

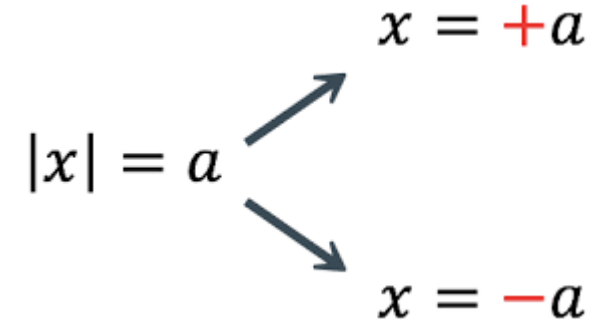
**Adım 6: Dur**

**BONUS SORU:** Neden  $a = \text{toplam}^{1/2}$  değil de  $a = \text{toplam}^{(1/2)}$ ?

# Örnek Soru-2

- Girilen bir sayının mutlak değerini hesaplatan bir algoritma:

- a) Hangi algoritma tipine girer?
- b) Nasıl yazılabilir?

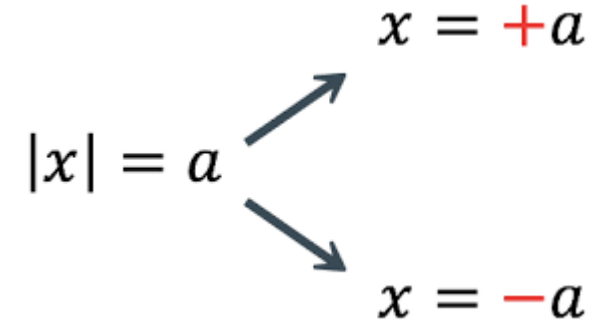




# Örnek Soru-2 Çözüm

- Girilen bir sayının mutlak değerini hesaplatan bir algoritma:

- a) Hangi algoritma tipine girer? **MANTIKSAL**
- b) Nasıl yazılabilir?



**Adım 1: Başla**

**Adım 2: Bir sayı giriniz; x**

**Adım 3: Eğer  $x < 0$  ise x'i -1 ile çarp ve a değerine yaz;  $a = -1 * x$ ;**

**Adım 4: Eğer  $x \geq 0$  ise x'i a değerine yaz;  $a = x$**

**Adım 5: a değerini ekrana yazdır; a**

**Adım 6: Dur**

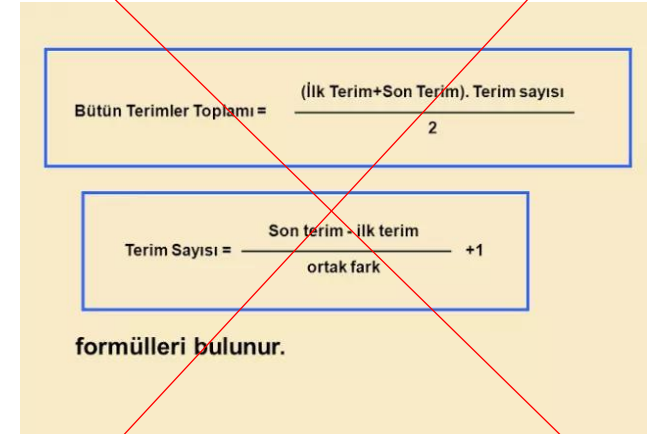
**BONUS SORU:** Neden  $\geq$ ?

# Örnek Soru-3

- Tüm rakamların toplamını hesaplayan bir algoritma;

- a) Hangi algoritma tipine girer?
- b) Nasıl yazılabilir?

\*Tek tek toplama şeklinde yazılacak olması durumu kastediliyor, terimler toplamı formülü değil ☺



Bütün Terimler Toplamı =  $\frac{(\text{İlk Terim} + \text{Son Terim}) \cdot \text{Terim sayısı}}{2}$

Terim Sayısı =  $\frac{\text{Son terim} - \text{İlk terim}}{\text{ortak fark}} + 1$

formülleri bulunur.

# Örnek Soru-3 Çözüm

- Tüm rakamların toplamını hesaplayan bir algoritma;
  - a) Hangi algoritma tipine girer? **DÖNGÜSEL (TERİMLER TOPLAMI – BASİT (LINEER))**
  - b) Nasıl yazılabilir?

**Adım 1: Başla**

**Adım 2: Toplam değerini 0 yap;  $t=0$**

**Adım 3: Rakam değerini 0 yap;  $r=0$**

**Adım 4: Toplam değeri ile rakam değerini topla ve toplam değerine yaz;  $t=t+r$**

**Adım 5: Rakam değerini 1 arttır;  $r=r+1$**

**Adım 6: Rakam değeri 9'dan küçük veya eşit ise Adım 4'e git**

**Adım 7: Toplam değerini ekrana yazdır;  $t$**

**Adım 8: Dur**

r	t	Yeni t	Yeni r
0	0	$0+0=0$	1
1	0	$1+0=1$	2
2	1	$2+1=3$	3
3	3	$3+3=6$	4
4	6	$4+6=10$	5
5	10	$5+10=15$	6
6	15	$6+15=21$	7
7	21	$7+21=28$	8
8	28	$8+28=36$	9
9	36	$9+36=45$	10

# Algoritma Geliştirmek

# Algoritma Geliştirme Kuralları

- Algoritmadaki adımlar programın sonlu sayıda işlem yapmasını sağlamalıdır.
- Adımlar sıralı ve mantıklı olmalıdır.
- Adımlar herkes tarafından anlaşılabilir şekilde ama mümkün olduğunda sade olmalıdır.
- Bellek verimliliği açısından fazladan veya gereksiz tanımlamalardan kaçınılmalıdır (tanımlanan her değişken bellekte yer tutuyor).
- Her algoritmanın bir çıktısı/sonucu olmalıdır.

# Algoritma Geliştirme Kuralları

- Algoritmadaki tüm satırlar 1 'den başlayarak numaralandırılmalıdır.
- Bütün algoritmalarda birinci satır “1. Başla” şeklindedir.
- Bütün algoritmalar “Dur” ile biter.
- Algoritmalarda kullanılan “Git” işlem akışı yönlendirme komutu satır numarasıyla birlikte kullanılmalıdır.
- Algoritmalarındaki ifadeler, herhangi bir programlama diline, donanıma, işletim sistemine vb. bağlı olmamalıdır.

# Algoritma Geliştirme

- Geliştirilen algoritmalar 3 şekilde yazılabilir;

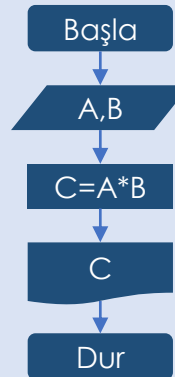
1. Satır algoritma: Problemin çözüm adımları düz metin olarak açık ve sade cümlelerle yazılır. Şu ana kadar yaptığımız örnekler bu sınıfa girer.
2. Akış diyagramları (flow-charts): Problemin çözüm adımları özel anlamlara sahip geometrik şekillerle gösterilir.
3. Sözde kod (pseudo-code): Problemin çözüm adımları, bilgisayara verilen komutlara benzer bir şekilde anlaşılır metinlerle ifade edilir.

- Girilen 2 sayının çarpımını veren algoritmanın 3 şekilde gösterimi:

## Satır algoritma:

1. Başla
2. İlk sayıyı klavyeden oku (a)
3. İkinci sayıyı klavyeden oku (b)
4. Girilen sayıları çarparak sonucu hesapla ( $c=a*b$ )
5. Sonucu ekrana yazdır (c)
6. Dur

## Akış diyagramı:



## Sözde kod:

1. Başla
2. A oku
3. B oku
4.  $C=A*B$
5. C yaz
6. Dur

# Algoritma Geliştirme

Algoritmaları geliştirirken değişken, sabit, atama, döngü (for loop), karar yapısı (if-else ) gibi öğeler kullanılır.

**Değişken (variable):** Programın akışı içinde farklı değerleri tutmak üzere ayrılmış bellek bölümüdür.  $C=A*B$  gibi bir ifadede A, B ve C tanımlayıcıları birer değişkendir.

**Sabit (constant):** Program her çalıştığında ve programın içinde herhangi bir anda hep aynı değeri döndüren tanımlayıcılara sabit denir.  $PI=3.14$  gibi bir ifadeden sonra PI sabiti programın her yerinde 3.14 değerini ifade eder.

**Tanımlayıcı (identifier):** Değişken, sabit, fonksiyon gibi programlama birimlerine programcı tarafından verilen isimlerdir.

## İsimlendirme Kuralları:

1. İngiliz alfabesindeki A-Z veya a-z arası 26 harf kullanılabilir.
2. 0-9 arası rakamlar kullanılabilir.
3. Simgelerden sadece alt çizgi ( \_ ) kullanılabilir.
4. Harf veya alt çizgi ile başlayabilir, ancak rakamla başlayamaz veya sadece rakamlardan oluşamaz.
5. İlgili programlama dilinin komutu veya saklı/anahtar kelimesi olamaz (Python için with, True vs.)

Örnek uygun gösterimler:

- ✓ ogrenciNo
- ✓ tcNo
- ✓ AdSoyad
- ✓ KitapListesi
- ✓ kitap



# Algoritma Geliştirme

Öğrencinin numarasını, vize ve final notunu aldıktan sonra ortalamasını hesaplayıp numarasını ve not ortalamasını yazdıran program.

1. Başla
2. Öğrencinin numarasını (ogrenciNo) al
3. Öğrencinin adını ve soyadını (AdSoyad) al
4. Öğrencinin vize notunu (VizeNot) al
5. Öğrencinin final notunu (FinalNot) al
6.  $\text{Ort} = 0.4 * \text{VizeNot} + 0.6 * \text{FinalNot}$
7. Numara (ogrenciNo) ve ortalamayı (Ort) yaz
8. Dur

# Algoritma Geliştirme – Döngü Oluşturma

- Döngü değişkenine başlangıç değeri verilir (**başlangıç**).
- Döngünün artma veya azaltma değeri belirlenir (**sayaç**).
- Döngünün bitiş değeri belirlenir (**bitiş**).
- Eğer döngü karar ifadeleri ile oluşturuluyorsa, karar işleminden önce döngü değişkenine başlangıç değeri verilmiş olmalı ve döngü içinde adım miktarı kadar artırılmalıdır/azaltılmalıdır.

Programınızda bazı işlemlerin belirli sayıda yapılması gerekebilir. Bunlar için döngülere ve işlem sayılarını control etmeye ihtiyaç vardır. Bunun için **sayaç yapıları** kullanılır.

**Sayaç yapısı:**  $\text{sayac} = \text{sayac} + \text{ilerleme\_adimi}$

**Toplam sayacı:**  $\text{toplama} = \text{toplama} + \text{yeni\_sayi}$  (toplama en başta 0 değeri atanır.)

**Çarpım sayacı:**  $\text{carpim} = \text{carpim} * \text{yeni\_sayi}$  (çarpıma en başta 1 değeri atanır.)

# Algoritma Geliştirme – Döngü Örnekleri

1. Başla
2.  $T=0$
3.  $J=1$
4. Eğer  $J>10$  ise git 8
5.  $T=T+J$
6.  $J=J+2$
7. Git 4
8. Yaz  $T$
9. Dur



1. Başla
2.  $T=0$
3. DÖNGÜ ( $J=1$  TO 10 STEP 2)
4.  $T=T+J$
5. DÖNGÜSONU
6. Yaz  $T$
7. Dur

İki program aynı görevi yerine getirmesine rağmen, soldaki örnekte döngü komutları kullanılmamış fakat sağdakinde kullanılmış ve akış 2 adım kısalmıştır.

J: Başlangıç  
10: Bitiş  
2: Artış değeri

**SORU:** Programın çıktısı ne olacaktır?

Klavyeden girilen 5 adet sayının ortalamasını bulan program:

1. Başla
2.  $N=5$
3.  $T=0$
4.  $S=0$
5. Eğer  $S>N-1$  ise git 10
6.  $S=S+1$
7. Sayıyı (A) gir
8.  $T=T+A$
9. Git 5
10.  $\text{Ortalama}=T/N$
11. Yaz Ortalama
12. Dur



**Toplam sayacı**

Klavyeden girilen N sayısının faktöriyelini hesaplanması:

1. Başla
2. N sayısını al
3.  $F=1$
4.  $S=0$
5. Eğer  $S>N-1$  ise git 9
6.  $S=S+1$
7.  $F=F*S$
8. Git 5
9. Yaz  $F$
10. Dur



**Çarpım sayacı**

# Algoritma Geliştirme – Döngü Örnek Soru

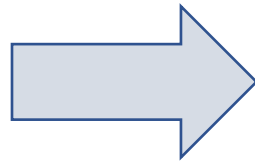
1. Başla
2.  $T=0$
3.  $J=1$
4. Eğer  $J>10$  ise git 8
5.  $T=T+J$
6.  $J=J+2$
7. Git 4
8. Yaz T
9. Dur



1. Başla
2.  $T=0$
3. DÖNGÜ ( $J=1$  TO 10 STEP 2)
4.  $T=T+J$
5. DÖNGÜSONU
6. Yaz T
7. Dur

Klavyeden girilen N sayısının faktöriyelinin hesaplanması:

1. Başla
2. N sayısını al
3.  $F=1$
4.  $S=0$
5. Eğer  $S>N-1$  ise git 9
6.  $S=S+1$
7.  $F=F*S$
8. Git 5
9. Yaz F
10. Dur



**Bu örneği döngü olarak yazmak istesek nasıl yazabilirdik?**

*İpucu: S'ye atanan ilk değerin 0 olduğunu unutmayın!*

# Algoritma Geliştirme – Döngü Örnek Soru

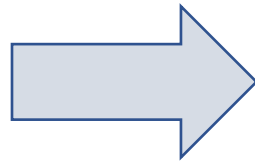
1. Başla
2.  $T=0$
3.  $J=1$
4. Eğer  $J>10$  ise git 8
5.  $T=T+J$
6.  $J=J+2$
7. Git 4
8. Yaz T
9. Dur



1. Başla
2.  $T=0$
3. DÖNGÜ ( $J=1$  TO 10 STEP 2)
4.  $T=T+J$
5. DÖNGÜSONU
6. Yaz T
7. Dur

Klavyeden girilen N sayısının faktöriyelinin hesaplanması:

1. Başla
2. N sayısını al
3.  $F=1$
4.  $S=0$
5. Eğer  $S>N-1$  ise git 9
6.  $S=S+1$
7.  $F=F*S$
8. Git 5
9. Yaz F
10. Dur



**Klavyeden girilen N sayısının faktöriyelinin hesaplanması:**

- 1. Başla**
- 2. N sayısını al**
- 3.  $F=1$**
- 4. DÖNGÜ( $S=0$  TO  $N-1$  STEP 1)**
- 5.  $F=F*(S+1)$**
- 6. DÖNGÜSONU**
- 7. Yaz F**
- 8. Dur**

# Akış Diyagramları

# Akış Diyagramları – Şekiller ve Anlamları

Akış diyagramları (flow-charts): Problemin çözüm adımları özel anlamlara sahip geometrik şekillerle gösterilir.



**Başla/Dur:** Algoritmayı başlatmak ve bitirmek için kullanılır.



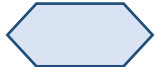
**Veri Girişi:** Algoritmaya dışarıdan veri girişini gösterir.



**İşlem:** Program çalışırken yapılacak işlemleri gösterir.



**Veri Yazma:** Ekranı veri yazdırmak için kullanılır. Sözel bilgi yazdırmak isteniyorsa “ kullanılır.



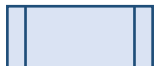
**Döngü:** Döngü işlemleri için kullanılır. İçine başlangıç, bitiş ve artış değerleri yazılır.



**Karar:** Mantıksal karşılaştırmaları gösteren şekildir. Koşullar içerisine yazılır.



**Bağlantı:** İşlem akışlarını birleştiren şekildir.



**Önceden tanımlı fonksiyon/işlem:** Büyük boyutlu programların içerisindeki alt fonksiyonları gösterir.



**Akış Yönleri:** İşlem akışlarının yönlerini gösterir.

# Akış Diyagramları – Çok Alternatifli Koşul Yapıları

Bazı problemler için basit koşul yapıları yetmeyebilir ve akışın 2 veya daha fazla kola ayrılması gerekebilir. Böyle durumlarda çok alternatifli koşul yapılarına başvururuz

**ÖRNEK:** Girilen sayının işaretini bulmaya yarayan algoritma

1. Başla
2. Yaz "Bir Sayı Girin"
3. Oku Sayı
4. Eğer Sayı>0 ise Yaz "Girdiğiniz Sayı Pozitifdir"
5. Eğer Sayı<0 ise Yaz "Girdiğiniz Sayı Negatiftir"
6. Eğer Sayı=0 ise Yaz "Girdiğiniz Sayı Sıfırdır"
7. Dur

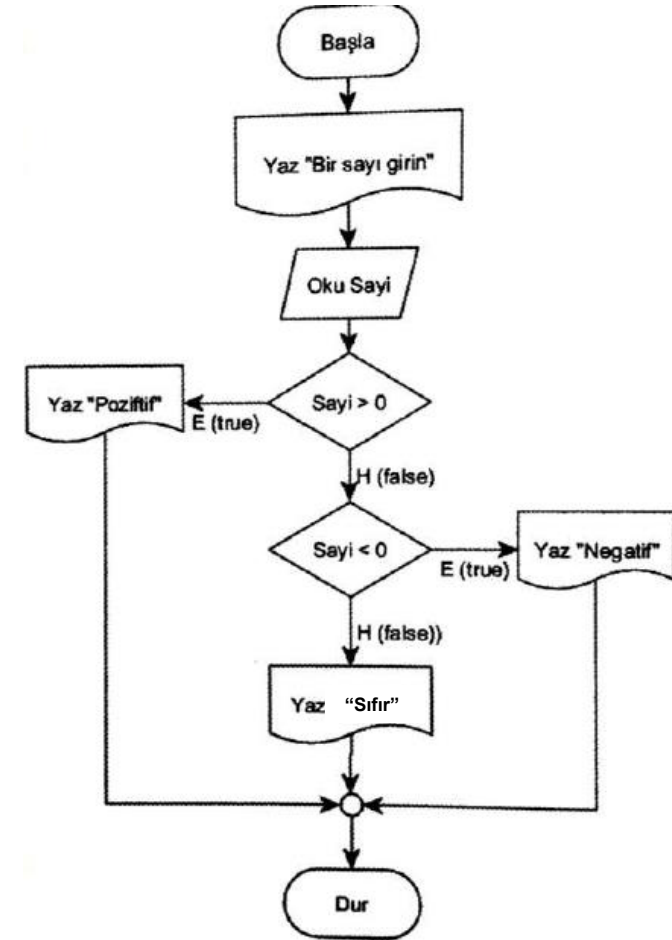


1. Başla
2. Yaz "Bir Sayı Girin"
3. Oku Sayı
4. Eğer
  - 4.1. Sayı>0 ise Yaz "Girdiğiniz Sayı Pozitifdir"
  - 4.2. Değilse Eğer
    - 4.2.1. Sayı<0 ise Yaz "Girdiğiniz Sayı Negatiftir"
    - 4.2.2. Değilse Yaz "Girdiğiniz Sayı Sıfırdır"
5. Dur



# Akış Diyagramları – Çok Alternatifli Koşul Yapıları - Örnek

1. Başla
2. Yaz "Bir Sayı Girin"
3. Oku Sayı
4. Eğer
  - 4.1. Sayı>0 ise Yaz "Girdiğiniz Sayı Pozitifdir"
  - 4.2. Değilse Eğer
    - 4.2.1. Sayı<0 ise Yaz "Girdiğiniz Sayı Negatiftir"
    - 4.2.2. Değilse Yaz "Girdiğiniz Sayı Sıfırdır"
5. Dur

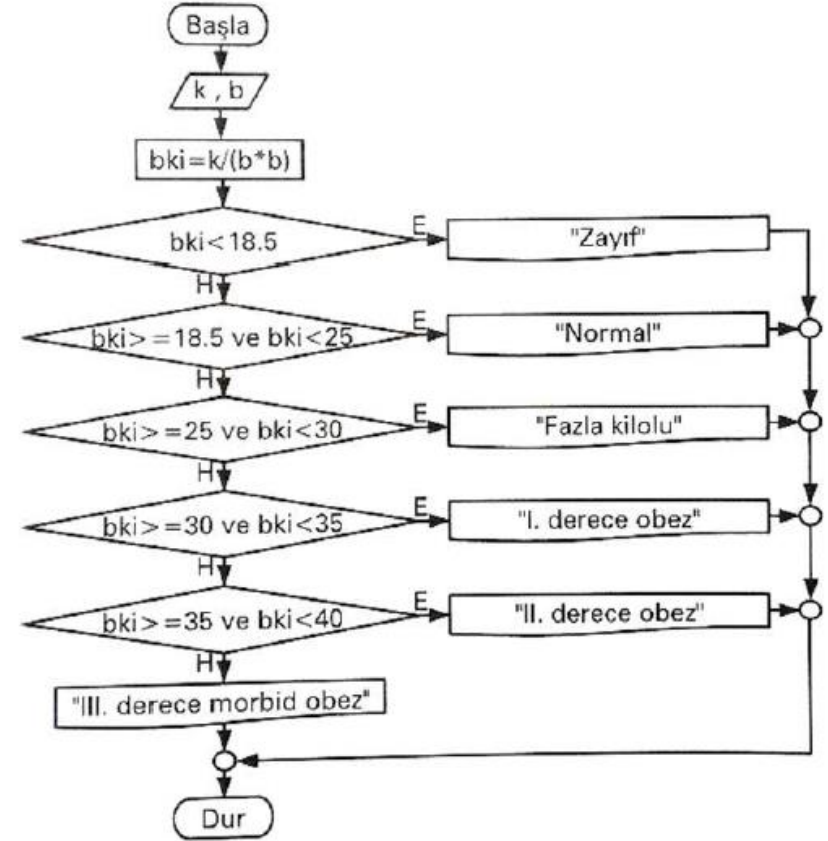


# Akış Diyagramları – Çok Alternatifli Koşul Yapıları - Örnek

Kullanıcıdan boy (m) ve kilo (kg) bilgisini alarak, beden kitle endeksini hesaplatan ve kilo durumunu ekrana yazdıran programın akış diyagramı

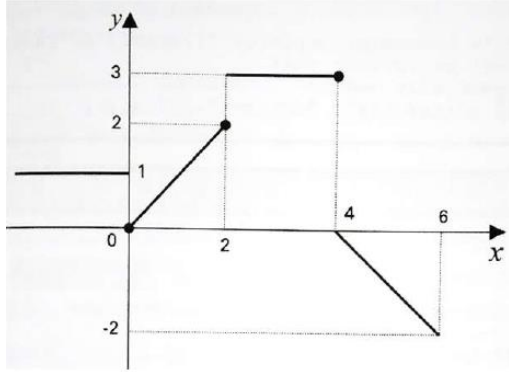
$$VKİ = \frac{\text{vücut ağırlığı (kg)}}{(\text{boy(m)})^2}$$

	BKI (kg/m <sup>2</sup> )
Zayıf	18,5 altında
Normal	18,5 - 24,9
Fazla kilolu	25 - 29,9
I. derece obez	30 - 34,9
II. derece obez	35 - 39,9
III. derece morbid obez	40 ve üzerinde



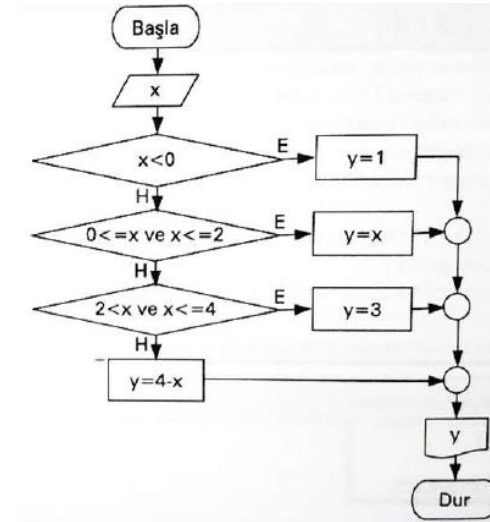
# Akış Diyagramları – Çok Alternatifli Koşul Yapıları - Örnek

Aşağıdaki  $y=f(x)$  fonksiyonuna göre, klavyeden girilen  $x$  değerine bakarak  $y$ 'yi hesaplayıp ekrana yazdıran program:



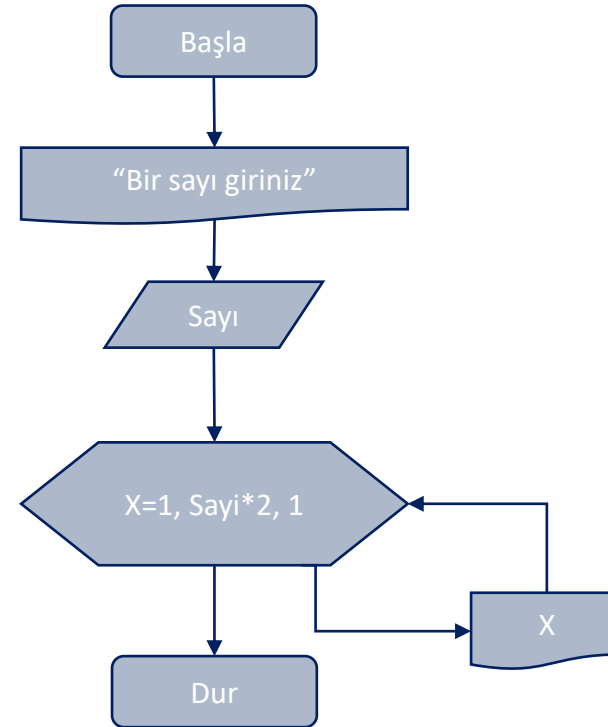
$$f(x) = \begin{cases} 1 & , \quad x < 0 \\ x & , \quad 0 \leq x \leq 2 \\ 3 & , \quad 2 < x \leq 4 \\ 4-x & , \quad 4 < x \end{cases}$$

1. Başla
2. Oku x
3. Eğer  $x < 0$  ise  $y = 1$
4. Eğer  $x \geq 0$  ve  $x \leq 2$  ,se  $y = x$
5. Eğer  $x > 2$  ve  $x \leq 4$  ise  $y = 3$
6. Eğer  $x > 4$  ise  $y = 4 - x$
7. Yaz y
5. Dur

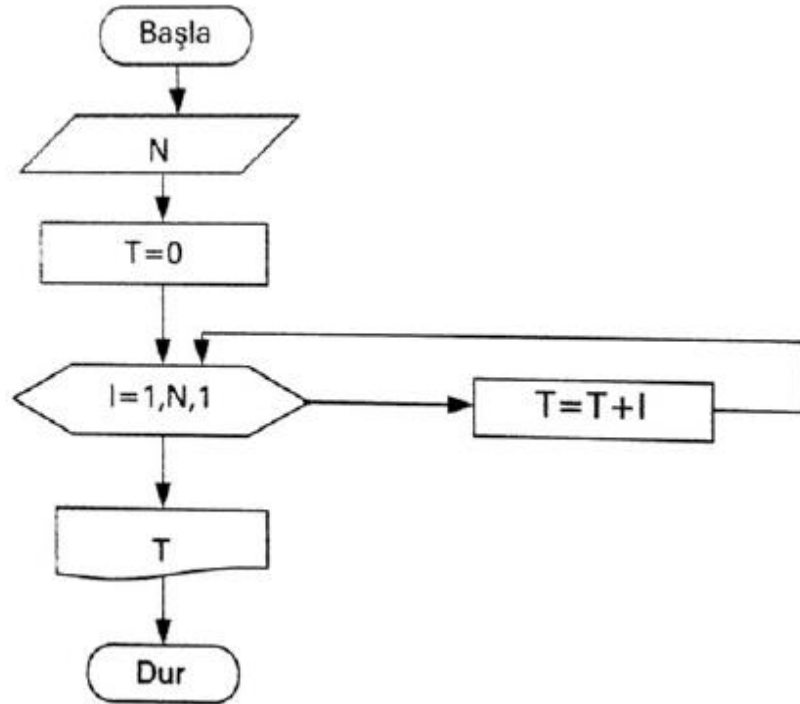


# Akış Diyagramları – Örnekler

1. Başla
2. Yaz "Bir sayı giriniz"
3. Oku Sayı
4. Döngü (X=1 TO Sayı\*2 STEP 1)
5. Yaz X
6. DöngüSonu
7. Dur



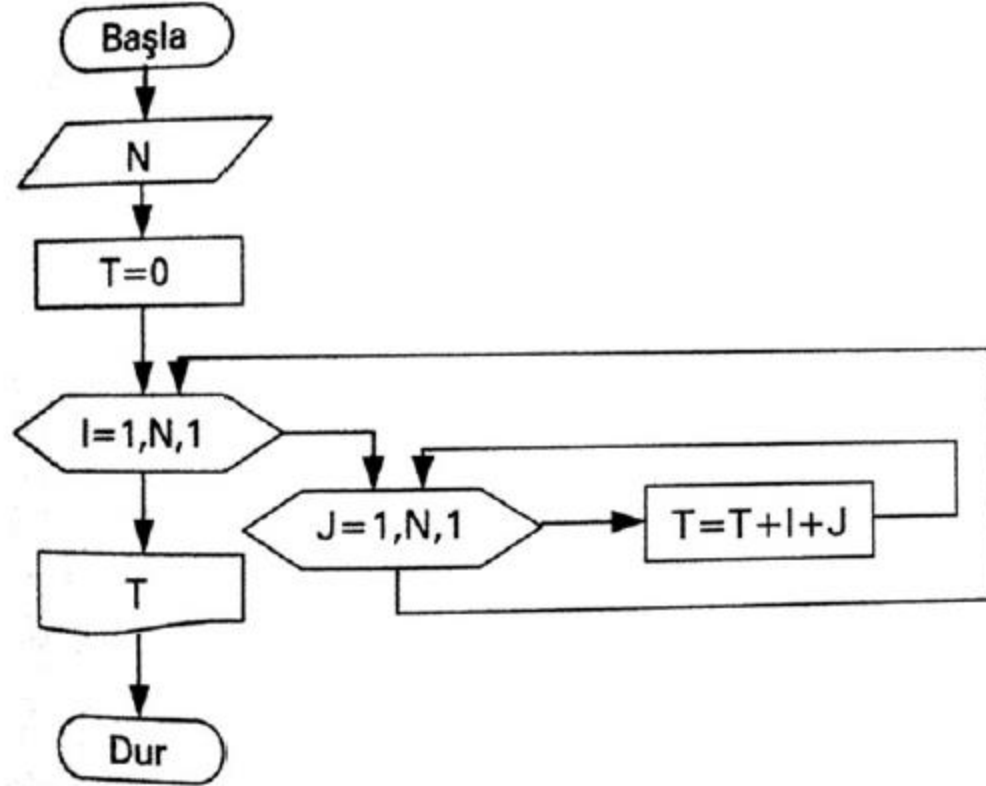
# Akış Diyagramları – Örnekler



Girilen sayı 5 olsun.  $N=5$ ;

İndeks (I)	Eski T	Yeni T
1	0	$1+0=1$
2	1	$1+2=3$
3	3	$3+3=6$
4	6	$6+4=10$
5	10	$10+5=15$

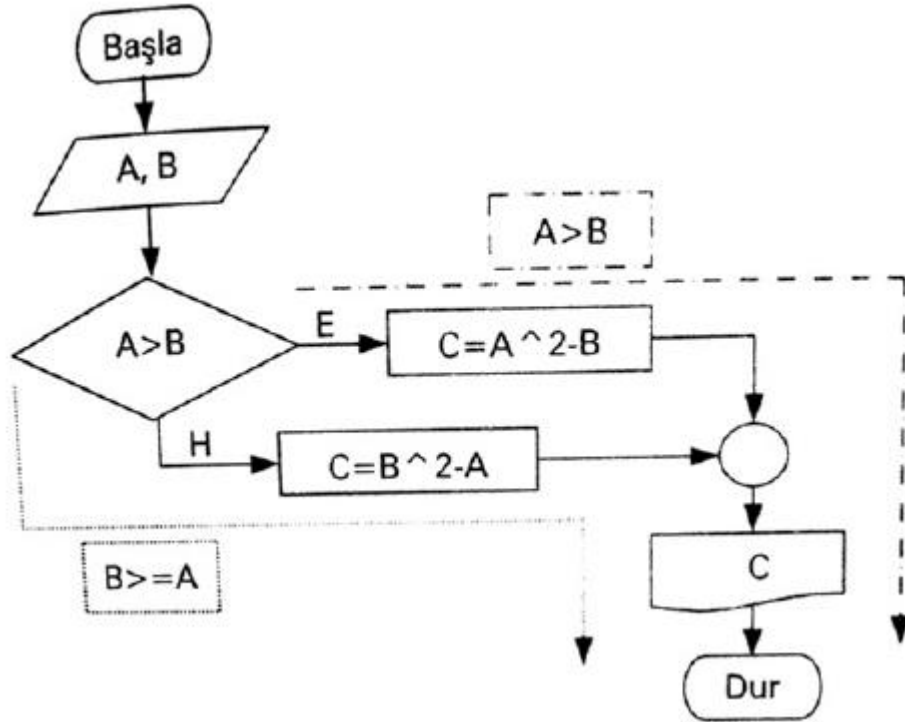
# Akış Diyagramları – Örnekler



Girilen sayı 2 olsun.  $N=2$ ;

İndeks (I)	İndeks (J)	Eski T	Yeni T
1	1	0	$1+1+0=2$
1	2	2	$1+2+2=5$
2	1	5	$2+1+5=8$
2	2	8	$2+2+8=12$

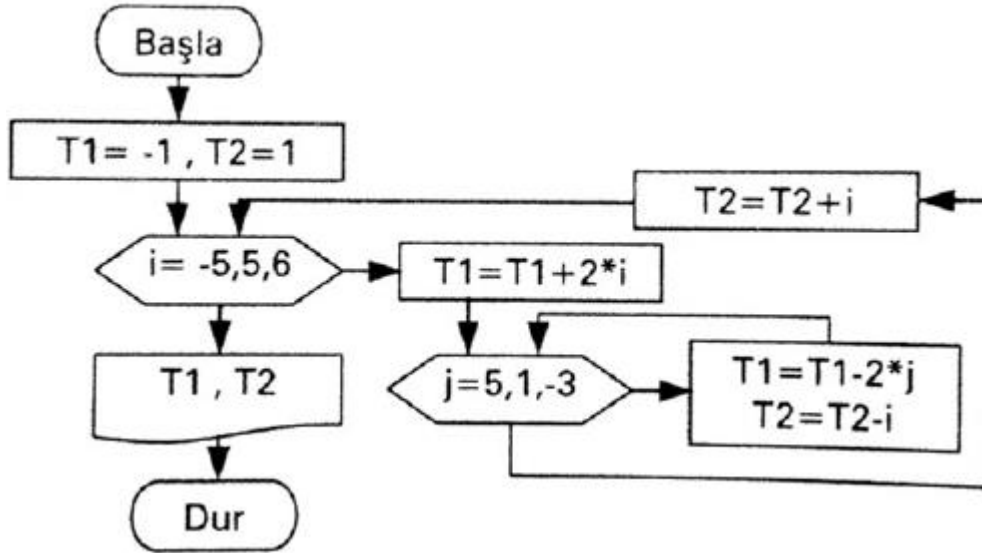
# Akış Diyagramları – Örnekler



A	B	C
10	8	$10^2 - 8 = 92$
6	6	$6^2 - 6 = 30$
4	7	$7^2 - 4 = 45$
15	8	$15^2 - 8 = 217$

# Akış Diyagramları – Örnek Soru

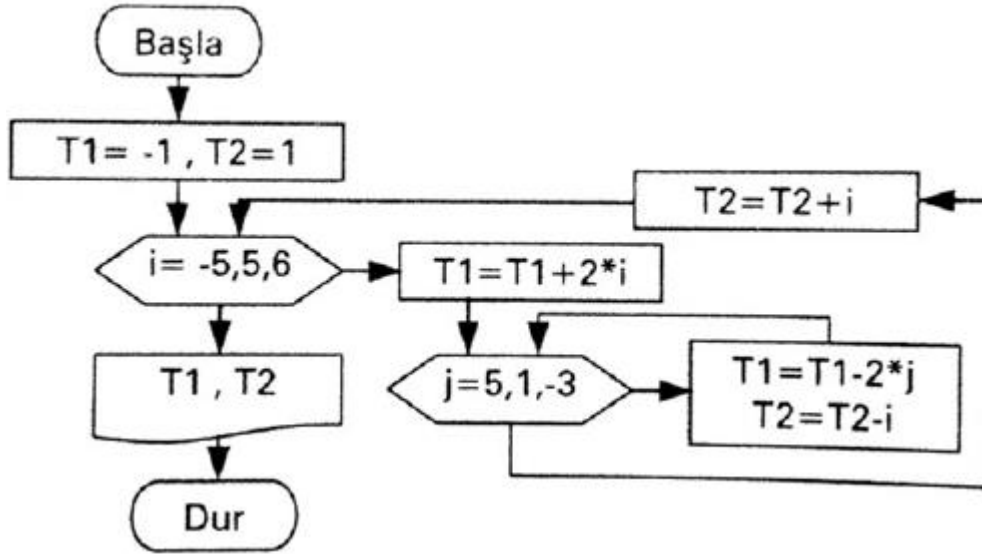
Akış diyagramındaki değişkenlerin, akış boyunca değişimlerini gösteren tabloyu oluşturalım.





# Akış Diyagramları – Örnek Soru Çözüm

Akış diyagramındaki değişkenlerin, akış boyunca değişimlerini gösteren tabloyu oluşturalım.



İndeks (I)	İndeks (J)	Eski T1	Eski T2	Yeni T1	Yeni T2	Yeni T1	Yeni T2
-5	5	-1	1	$-1+2 \cdot 5 = 9$	$1 - (-5) = 6$	$-1 - 2 \cdot 5 = -11$	
-5	2	-11	6	$-11 - 2 \cdot 2 = -15$	$6 - (-5) = 11$		$11 + (-5) = 6$
1	5	-15	6	$-15 + 2 \cdot 1 = -13$	$6 - 1 = 5$	$-13 - 2 \cdot 5 = -23$	
1	2	-23	5	$-23 - 2 \cdot 2 = -27$	$5 - 1 = 4$		$4 + 1 = 5$

# Akış Diyagramları – Örnek Soru

- Klavyeden girilen bir sayının mutlak değerini ekrana yazdıran programın akış diyagramını çizelim.

$$|x| = \begin{cases} -x, & x < 0 \\ x, & x \geq 0 \end{cases}$$

## **SATIR ALGORİTMA**

Adım 1: Başla

Adım 2: Bir sayı giriniz; x

Adım 3: Eğer  $x < 0$  ise x'i -1 ile çarp ve a değerine yaz;  $a = -1 * x$ ;

Adım 4: Eğer  $x \geq 0$  ise x'i a değerine yaz;  $a = x$

Adım 5: a değerini ekrana yazdır; a

Adım 6: Dur

# Akış Diyagramları – Örnek Soru Çözümü

- Klavyeden girilen bir sayının mutlak değerini ekrana yazdıran programın akış diyagramını çizelim.

$$|x| = \begin{cases} -x, & x < 0 \\ x, & x \geq 0 \end{cases}$$

## SATIR ALGORİTMA

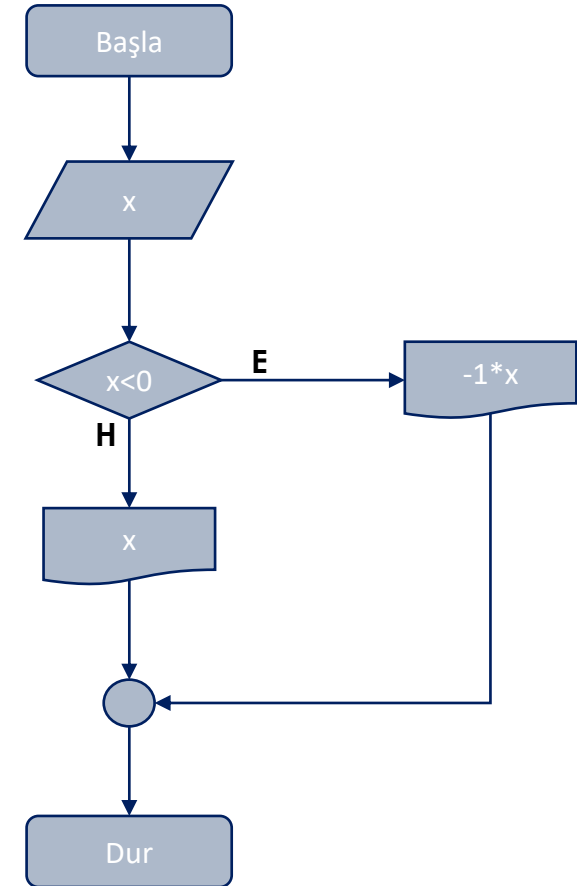
Adım 1: Başla

Adım 2: Bir sayı giriniz; x

Adım 3: Eğer  $x < 0$  ise x'i -1 ile çarp ve ekrana yazdır;  $-1 * x$

Adım 4: Eğer  $x \geq 0$  ise x'i ekrana yazdır; x

Adım 5: Dur



# Sözde Kod (Pseudocode)

# Sözde Kod Nedir?

- Bilgisayarda bir programlama dili olarak çalışmayan, ancak yazı/konuşma dilinden ziyade programlama dillerine daha yakın olan algoritma ifadelerine **sözde kod (pseudo-code)** denilir.
- Sözde kodların **İngilizce** ifadelerle belirtilmesi dünyada yaygın olarak kabul görmektedir.
- Sözde kod yapılarında özel olarak başla ve dur komutlarına gerek yoktur.

## SÖZDE KOD YAPILARI

1. **Okuma/Yazma:** Temel okuma ve yazma işlemleri. Komutlar: READ, GET, DISPLAY, WRITE
2. **İşlemler:** Sözde kod içinde gerçekleştirilen toplama, çıkartma, bölme, vb. aritmetik ve diğer işlemler ile bir değişkene değer atanması gibi olaylardır.
3. **Karar Yapıları:** Bir koşulu kontrol edip, bir veya birden fazla alternatiften hangisinin işletileceğine karar veren yapılardır.
4. **Tekrarlı Yapılar:** Program içinde bir koşula bağlı olarak ya da belirli bir sayıda tekrar edecek işlemler için kullanılır.

# Sözde Kod Karar Yapıları

**Basit Karar Yapısı:** Bir koşula bağlı olarak, bir alternatifin yapılıp yapılmayacağına karar verir.

```
IF [koşul] THEN
    Koşul doğru (true) ise gerçekleşecek işlemler
ENDIF
```

**İki Alternatifli Karar Yapısı:** Koşula uyan durumda bir alternatifi, uymayan durumda diğer alternatifi işletir.

```
IF [koşul] THEN
    Koşul doğru (true) ise gerçekleşecek işlemler
ELSE
    Koşul yanlış (false) ise gerçekleşecek işlemler
ENDIF
```

# Sözde Kod Karar Yapıları

**Çok Alternatifli Karar Yapısı:** Birden fazla koşul deyimi içeren karar yapısıdır.

```
IF [koşul1] THEN
    Koşul1 doğru ise gerçekleşecek işlemler
ELSEIF [koşul2] THEN
    Koşul1 yanlış, Koşul2 doğru ise gerçekleşecek işlemler
ELSE
    Koşul1 ve Koşul2 yanlış ise gerçekleşecek işlemler
ENDIF
```

İç içe koşullu  
karar yapısı

```
IF [koşul1] THEN
    Koşul1 doğru ise gerçekleşecek işlemler
    IF [koşul2] THEN
        Koşul 1 ve Koşul2 doğru ise gerçekleşecek işlemler
    ELSE
        Koşul1 doğru ve Koşul2 yanlış ise gerçekleşecek işlemler
    ENDIF
ENDIF
```

# Sözde Kod Tekrarlı Yapılar

**Tekrarlı yapının girişine uygulanan koşullar:** Koşul sağlanmadığı sürece işlemler gerçekleşmez.

```
LOOP [koşul]
    ...Tekrarlanacak işlemler
ENDLOOP
```

**Tekrarlı yapının sonuna uygulanan koşullar:** Koşul sağlanmasa bile, tekrarlı yapının içindeki işlem en az bir defa gerçekleştirilecektir.

```
LOOP
    ...Tekrarlanacak işlemler
ENDLOOP [koşul]
```

**Sayaç tipli tekrarlı yapılar:** Bu yapılarda, belirli bir sayıdan başlanarak, belirli bir hedefe kadar sayılır. Sayma işleminde artışın ne kadar olacağını STEP deyimi belirtir

```
FOR Sayac = [başlangıç değeri] TO [Hedef Sayı Sayı] STEP [artış]
    ...Tekrarlanacak işlemler
ENDFOR
```

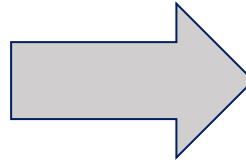


# Örnekler – Satır Algoritmadaki Sözcük Kod Üretmek

İki sayıyı alıp çarpımını ekrana yazdıran algoritma

## Satır algoritma:

1. Başla
2. Oku (A,B)
3.  $C=A*B$
4. Yaz C
5. Dur



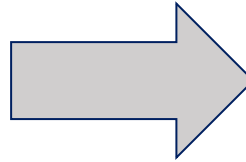
## Sözcük kod:

```
GET A
GET B
C=A*B
DISPLAY C
```

# Örnekler – Satır Algoritmadan Sözde Kod Üretmek

## Satır algoritma:

1. Başla
2. Yaz "Yaşınızı Giriniz"
3. Oku (Yas)
4. Eğer
  - 4.1.  $Yas \geq 18$  ise Yaz "Kredi kartı çıkartabilirsiniz."
  - 4.2. Değilse ise Yaz "Kredi kartı çıkartamazsınız."
5. Dur



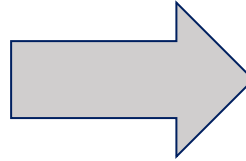
## Sözde kod:

```
DISPLAY "Yaşınızı Giriniz"
GET Yas
IF Yas >= 18 THEN
    DISPLAY "Kredi kartı çıkartabilirsiniz."
ELSE
    DISPLAY "Kredi kartı çıkartamazsınız."
ENDIF
```

# Örnekler – Satır Algoritmadan Söзде Kod Üretmek

## Satır algoritma:

1. Başla
2. Toplam=0
3. DÖNGÜ (X=1 TO 100 STEP 1)
4. Yaz "Bir Sayı Girin"
5. Oku Sayı
6. Toplam=Toplam+Sayı
7. DÖNGÜSONU
8. Yaz Toplam
5. Dur

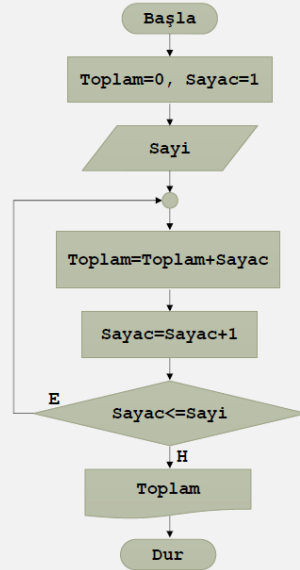


## Sözde kod:

```
Toplam=0
FOR(X=1 TO 100 STEP 1)
    DISPLAY "Bir Sayı Girin"
    GET Sayı
    Toplam = Toplam + Sayı
ENDFOR
DISPLAY Toplam
```

# Örnekler – Akış Diyagramından Sözde Kod Üretmek

## Akış Diyagramı:



## Sözde kod:

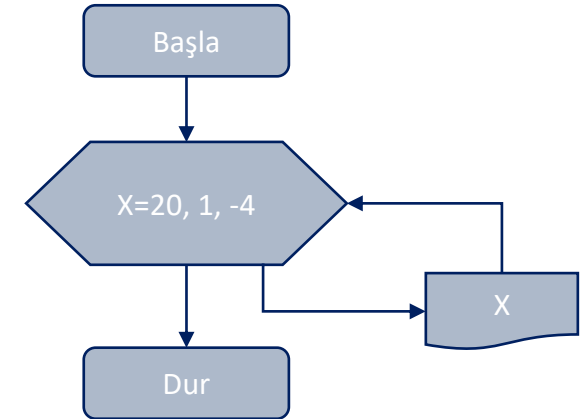
```
Toplam=0
GET Sayi
FOR Sayac=1 TO Sayi STEP 1
    Toplam=Toplam+Sayac
ENDFOR
DISPLAY Toplam
```

# Genel Örnekler

20'den başlayıp 1'e kadar, dörder dörder geriye doğru sayıp ekrana yazdıran algoritmayı 3 gösterim şekliyle de gösteriniz.

1. Başla
2. Döngü (X=20 TO 1 STEP -4)
3. Yaz X
4. DöngüSonu
5. Dur

```
FOR X=20 TO 1 STEP -4  
    DISPLAY X  
ENDFOR
```

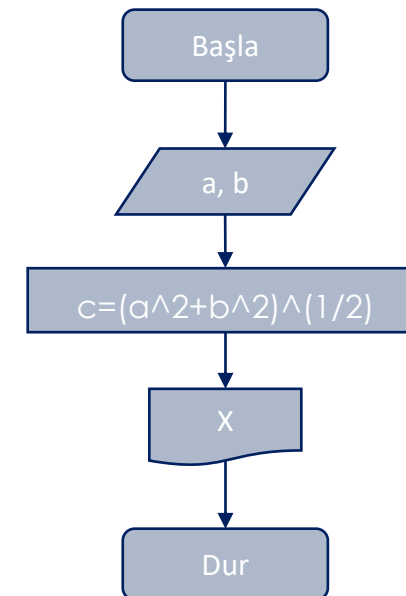


# Genel Örnekler

İki dik kenarını bildiğimiz bir üçgenin, Pisagor teoremine göre hipotenüsünü bulmamızı sağlayacak algoritmayı 3 gösterim şekliyle de gösteriniz ;

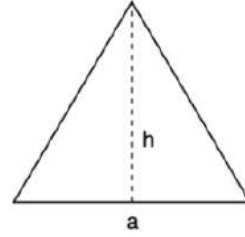
1. Başla
2. İlk dik kenarı (a) gir
3. İkinci dik kenarı (b) gir
4.  $c = (a^2 + b^2)^{1/2}$
5. Yaz c
6. Dur

```
GET a
GET B
 $c = (a^2 + b^2)^{1/2}$ 
DISPLAY c
```



# Genel Örnekler Soru

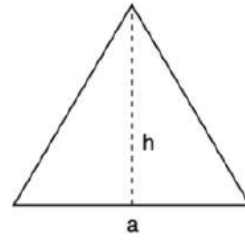
Klavyeden bir kenar uzunluğu ve o kenara ait yüksekliği girilen üçgenin alanını hesaplayan program 3 gösterim şekliyle de gösteriniz ;



$$\text{Alan} = (a.h) / 2$$

# Genel Örnekler Soru - Çözüm

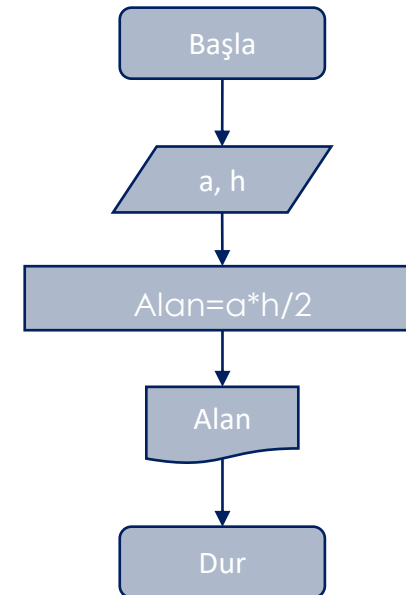
Klavyeden bir kenar uzunluğu ve o kenara ait yüksekliği girilen üçgenin alanını hesaplayan program 3 gösterim şekliyle de gösteriniz ;



$$\text{Alan} = (a \cdot h) / 2$$

1. Başla
2. Kenar uzunluğunu (a) gir
3. Yüksekliği (h) gir
4.  $\text{Alan} = a \cdot h / 2$
5. Yaz Alan
6. Dur

```
GET a
GET h
Alan = a * h / 2
DISPLAY Alan
```

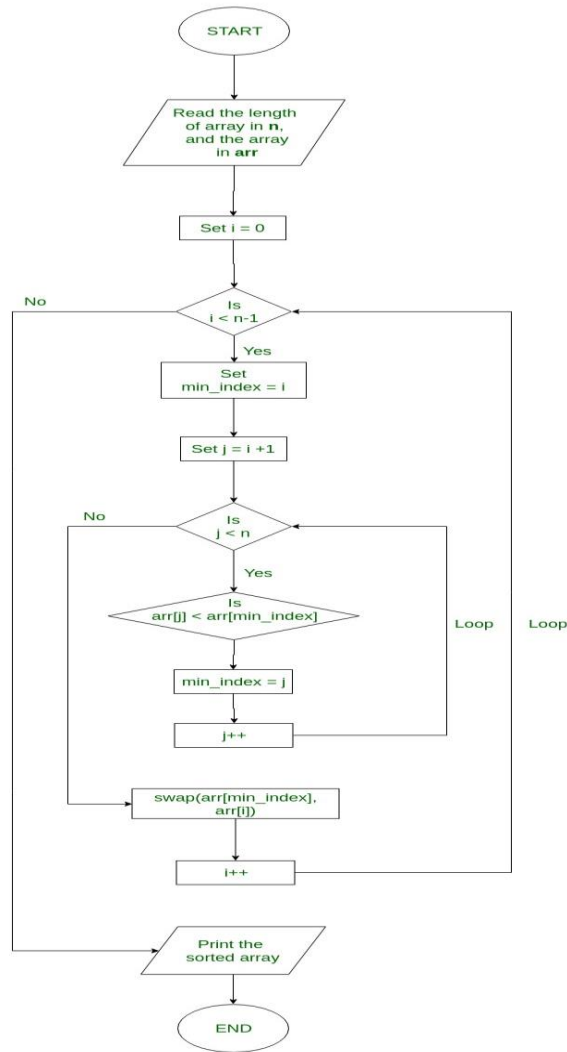




# BİLGİSAYAR DÜNYASINDAKİ ÖNEMLİ ALGORİTMALAR

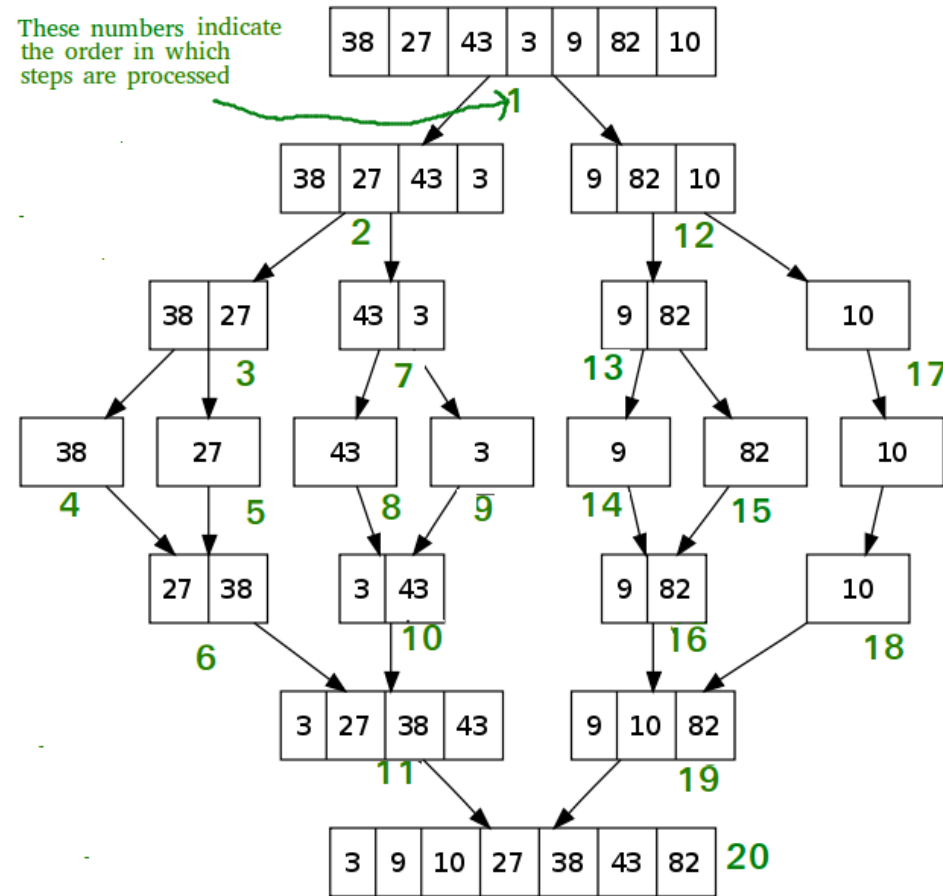
- [Insertion sort, Selection sort, Bubble sort](#)
- [Merge Sort, Quicksort](#)
- [Binary Search](#)
- [Breadth First Search \(BFS\)](#)
- [Depth First Search \(DFS\)](#)
- [Lee algorithm | Shortest path in a Maze](#)
- [Flood fill Algorithm](#)
- [Floyd's Cycle Detection Algorithm](#)
- [Kadane's algorithm](#)
- [Longest Increasing Subsequence](#)
- [Inorder, Preorder, Postorder](#) Tree Traversals
- [Heap Sort](#)
- [Topological Sorting in a DAG](#)
- [Disjoint-Set Data Structure \(Union-Find Algorithm\)](#)
- [Kruskal's Algorithm for finding Minimum Spanning Tree](#)
- [Single-Source Shortest Paths — Dijkstra's Algorithm](#)
- [All-Pairs Shortest Paths — Floyd Warshall Algorithm](#)

## SELECTION SORT

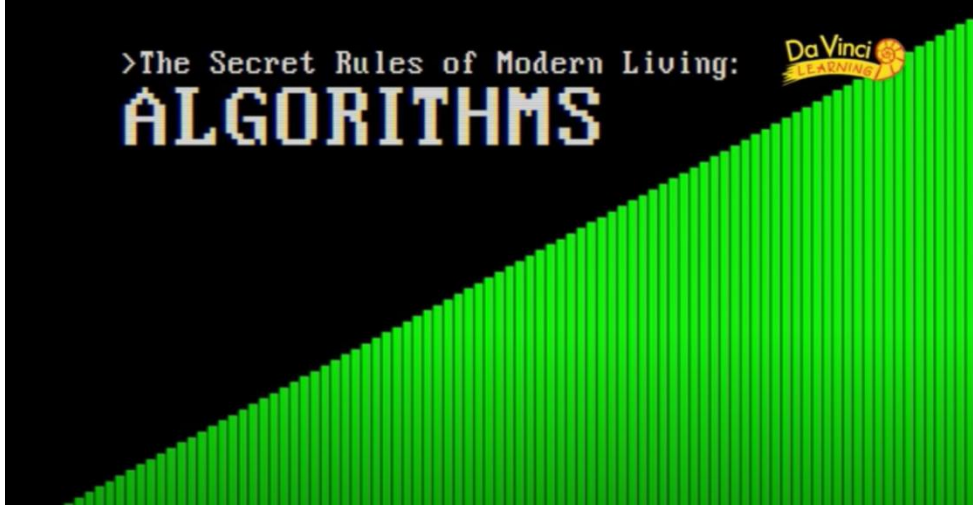


Flowchart for Selection Sort

## MERGE SORT



# BELGESEL, KİTAP ve FİLM ÖNERİSİ 😊



**BELGESEL:** Modern Yaşamın Gizli Sırları - Algoritmalar



**KİTAP**

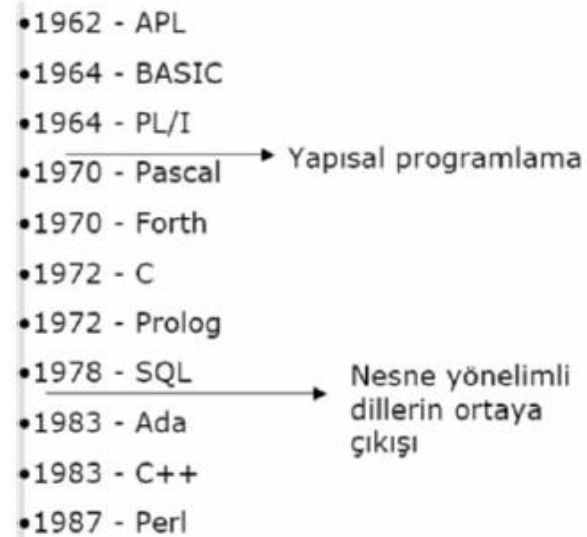


**FİLM:** The Imitation Game

# Programlama Dilleri

# Programlama Dillerinin Tarihçesi

- Programlama Dili: İstenilen işlemleri yapmak, elde edilen veriyi saklamak ve girdi/çıkı aygıtlarına (klavye/monitor gibi) veriyi gönderip alma adımlarını yapmamızı sağlayan dildir.
- Kısaca, bilgisayarlara ne yapmaları gerektiğini söylememizi sağlayan özel bir dil olarak tanımlanabilir.
- İlk programın, Ada Lovelace tarafından Charles Babbage'ın tanımlamış olduğu “Analytical Engine” i ile Bernoulli sayılarının hesaplanmasına ilişkin makalesinde olduğu söylenmektedir. Bu nedenle ilk gerçek anlamdaki programlama dillerinden birinin adı Ada Lovelace anısına ADA olarak isimlendirilmiştir.
- 1940'lerde ilk modern bilgisayar ortaya çıktığında, programcılar yalnızca assembly dili kullanarak yazılım yapabiliyorlardı.



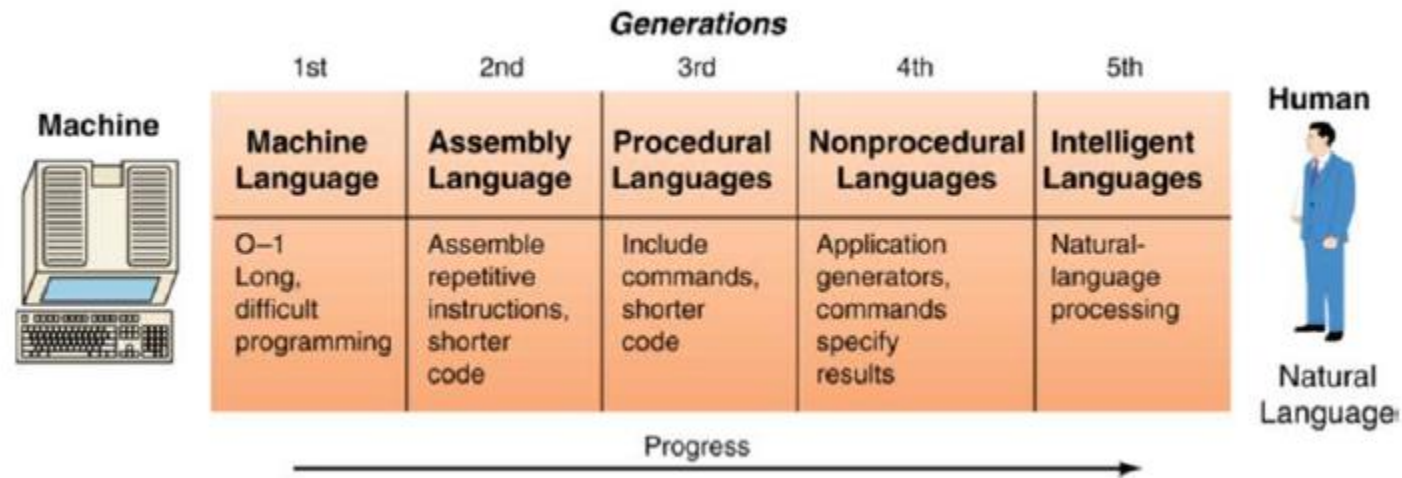
## 1990 lar, Internet

- 1991 - Python
- 1991 - Java
- 1995 - PHP
- 2000 - C#

Tamamı nesne yönelimli dillerdir. Yeni programlama kavramlarından ziyade, programlamanın kolaylaşmasını ve taşınabilirliği amaçlamaktadırlar

# Programlama Dillerinin Tarihçesi

- İnsanın en zor anlayıp öğrenebileceği, 100110 gibi makina kodlarına yakın dillere, düşük seviyeli programlama dilleri denilir.
- İnsanın rahatça anlayıp öğrenebileceği, insan diline yakın komutlara ve yazım kurallarına sahip dillere yüksek seviyeli diller denilir.



# Programlama Dillerinin Tarihçesi

- Her programlama dilinin kendine ait yazım kuralları (syntax) vardır.
- Tarihçesinde, yeni gelişen dillerin kolay okunabilir olması amacı güdülmüştür.

## Python

```
In [1]: print("Hello world!")  
Hello world!
```

## PASCAL

```
Program HelloWorld;  
uses wincrt;  
begin  
  writeln('Merhaba Dünya');  
  readln;  
end.
```

## C

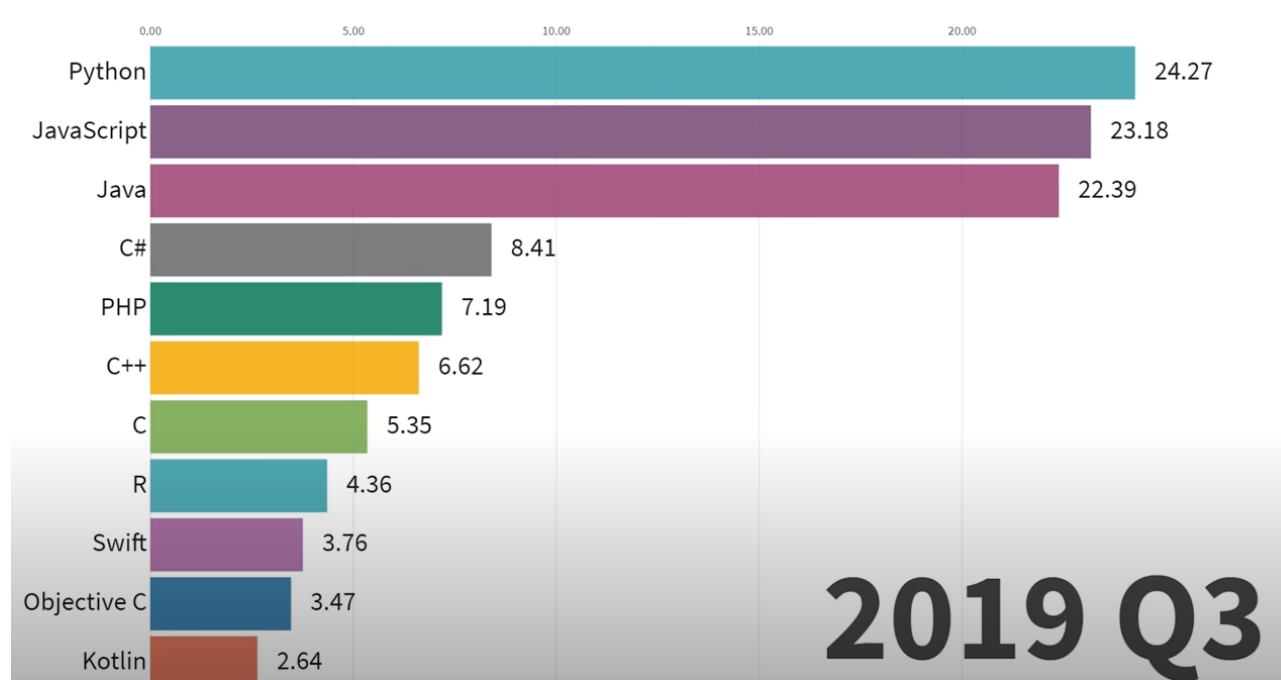
```
#include <stdio.h>  
main() {  
    printf("Merhaba  
    Dünya");  
    getchar();  
}
```

## C#

```
class MerhabaDunya  
{  
    // Programın ilk girdiği nokta  
    static void Main(/*string[] args*/)  
    {  
        System.Console.WriteLine("Merhaba  
        Dünya!");  
    }  
}
```



# Neden Python?



Programlama dillerinin kullanım oranlarının yıldan yıla değişimi

# Neden Python?

- Öncelikle hem sektörde, hem akademide videoda görüldüğü üzere sıklıkla kullanılan dillerden birisi, dolayısıyla iş fırsatı yaratıyor.
- Okuması ve yazması oldukça basit bir dil, fonksiyon isimleri rahatlıkla anlaşılabilir.
  - Dosya okumak için gereken fonksiyon `read()`, yazmak için `write()`, toplam almak için `sum()` gibi yapılacak işlemin İngilizcesi ile kodlar çok rahat anlaşılabilir.
- Tamamen ücretsiz ve açık kaynak; internetten indirip bilgisayarınıza kolayca kurabilir ve uygulama geliştirmeye başlayabilirsiniz.
- Oldukça geniş bir ekosistemi var, yapmak istediğiniz program için gereken fonksiyonları sizden önce geliştirenler varsa, erişebilirsiniz. Sıklıkla başvuracağınız siteler:
  - [Stackoverflow.com](https://stackoverflow.com)
  - [Github](https://github.com)
  - [Quora](https://quora.com)
  - [pypi.org](https://pypi.org)
- Program kütüphaneleri yardımıyla kısa kodlarla önemli işlemleri hızlıca yapabilirsiniz.

# Kaynakça

- <https://www.slideshare.net/dinakan1/01-introduction-to-algorithms>
- <https://www.slideshare.net/AshimLamichhane/algorithm-introduction>
- <http://ikucukkoc.baun.edu.tr/lectures/BIL1202/BIL1202BirlestirilmisNotlar.pdf>
- <https://medium.com/@codingfreak/top-algorithms-data-structures-concepts-every-computer-science-student-should-know-e0549c67b4ac>
- <https://www.geeksforgeeks.org/selection-sort/>
- <https://www.geeksforgeeks.org/merge-sort/>
- <https://data-flair.training/blogs/features-of-python/>
- <https://slideplayer.biz.tr/slide/9125936/>