

# ALGORİTMA VE PROGRAMLAMA I

Yrd. Doç. Dr. Deniz KILINÇ  
[deniz.kilinc@cbu.edu.tr](mailto:deniz.kilinc@cbu.edu.tr)

YZM 1101

Celal Bayar Üniversitesi Hasan Ferdi Turgutlu  
Teknoloji Fakültesi

# Genel Bakış...

2

- **1. Bölüm: Algoritmaya Giriş**
  - Problem Çözme
  - Algoritma Nedir?
  - Algoritma Gösterim Şekilleri
    - Düz yazı
    - Sözde kod
    - Akış şeması
  - Mantıksal Yapılar
  - İşlemler ve Operatörler
  - Algoritmada Kullanılan Terimler

# Genel Bakış...

3

- **2. Bölüm: Programlamaya Giriş**
  - Program
  - Programlama
  - IDE (Integrated Development Environment, Tümlüşik Geliştirme Ortamı)
  - Derleyici (Compiler)
  - Yorumlayıcı (Interpreter)
  - Bağlayıcı (Linker)
  - Çalıştırma (Execution)
  - Hata Türleri
  - Debug

# 1. BÖLÜM

4

## ALGORİTMAYA GİRİŞ

# **YAZILIM MÜHENDİSLİĞİ ÖĞRENCİLERİNİN DİKKATİNE !!!**

5

**SİZLER BİLGİSAYARCI DEĞİLSİNİZ**

**SİZLER PROGRAMCI DEĞİLSİNİZ**

**SİZLER YAZILIM MÜHENDİSLERİNSİNİZ**

**ÇEVRENİZDE MESLEĞİNİZİ NE KADAR  
SAVUNURSANIZ MEZUN OLDUĞUNUZDA  
İTİBARINIZ O KADAR YÜKSEK OLUR.**

# Problem Çözme

6

- İnsanlar sürekli düşünür ve problem çözerler. Birçok problem, az ya da hiç düşünülmeden çözülebilir.

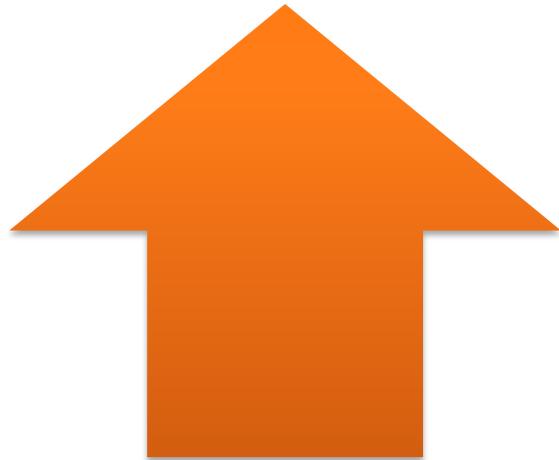
**Problem:** Bugün evden çıkarken ne giymeliyim?

**Çözüm:** Bunun için muhtemelen pencereden dışarıya bakılır. Hava yağmurlu ise mevsime göre giyinmenin yanı sıra dışarıya çıkarken bir de şemsiye alınması gereklidir. Hava güneşli ve sıcak ise daha ince giyinilerek dışarıya çıkarılır. Böylece problemin çözümü kendiliğinden oluşturulan bir kararla sağlanır.

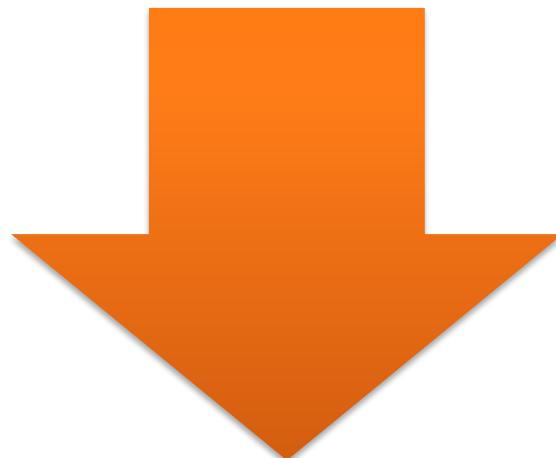


# Problemi Kim Çözecek?

7



Bilgisayar, sadece yazılımcının kendisine söylediği şeyi nasıl yapacağını bilir.



Sonuç olarak yazılımcı bilgisayara problemi nasıl çözeceğini bildirmelidir.

# Problem Çözme (devam...)

8

Bilgisayara  
nasıl iş  
yaptıracak,  
nasıl iletişim  
kuracaksınız?

Bilgisayarın dili  
makine dilidir.  
Onunla makine  
mantiği ile iletişim  
kurabiliziz.



Bir “Program” ile.  
Bilgisayarlar  
program olmadan  
çalışmazlar.

Bu da  
Algoritma  
(talimat, rutin,  
reçete) ile  
olur.

# Problem Çözme Sırası

9

1. Problemi anlama (Understanding, Analyzing),
2. Bir çözüm yolu geliştirme (Designing),
3. Algoritma ve program yazma (Writing),
4. Tekrar tekrar test etme (Reviewing)

Polya, George (1957) '**How To Solve It**',  
Princeton University Press, 2<sup>nd</sup> Edition

# Örnek: Bana bir Eczane programı lazım?

10

- Problemi anlamak için küçük parçalara ayırmamız ve soru sormamız lazım.
  - Sağlık sektöründe eczaneçilik nasıl işliyor?
  - İlaç giriş-çıkış süreçleri nasıl gerçekleşiyor?
  - Satacak ürünüm bitince Depodan nasıl mal alırım?
  - Müşterilere satış nasıl yapılıyor?
  - Reçetesiz ürünlerin satışında fark var mı?
  - Sağlık Bakanlığı e-recete entegrasyonu nasıl?
  - Barkod okuma nasıl gerçekleşiyor?
- Soruları sordukça süreçleri anlayacaksınız. Belki de işin yanlış yapıldığını göreceksiniz.
- Piyasada buna "**At Gözlüğü Takmak**" diyoruz.

# Problem Çözme Farklı Bakış

11

## Problem Çözme Aşaması

Problemin tanımlanması

Çözümün ana hatlarının ortaya konulması

Ana hatlara bağlı bir algoritma geliştirilmesi

Algoritmanın doğruluğunun sıralanması

Algoritma kodları belirli bir programlama diline dönüştürülür.

## Geçerleştirim Aşaması

Program bilgisayarda çalıştırılır.

Program belgelemesi ve bakımı yapılır.

# Problem Çözme - Descartes

12

- Problem çözmede, soruna hemen girişmek yerine, dikkatli ve sistematik yaklaşım ilke olmalıdır. Problem iyice anlaşılmalı ve mümkün olduğu kadar **küçük parçalara ayrılmalıdır**. Descartes'in "Discourse on Method" isimli kitabında problem çözme teknikleri şu dört madde ile özetlenir:
  1. Doğruluğu kesin olarak kanıtlanmadıkça, hiçbir şeyi doğru olarak kabul etmeyin; tahmin ve önyargılardan kaçının.
  2. Karşılaştığınız her güçlüğü mümkün olduğu kadar çok parçaaya bölün.
  3. Düzenli bir biçimde düşünün; anlaşılması en kolay olan şeylerle başlayıp yavaş yavaş daha zor ve karmaşık olanlara doğru ilerleyiniz.
  4. Olaya bakışınız çok genel, hazırladığınız ayrıntılı liste ise hiçbir şeyi dışarıda bırakmayacak kadar kusursuz ve eksiksiz olsun.

# Algoritma Nedir?

13

*19.yy da İranlı Musaoğlu Horzumlu Mehmet (Alharezmi adını Araplar takmıştır) problemlerin çözümü için genel kurallar oluşturdu. Algoritma Alharezmi'nin Latince okunuşudur.*

- Basit tanım: Belirli bir görevi yerine getiren sonlu sayıdaki işlemler dizisidir.
- Geniş tanım: Verilen herhangi bir sorunun çözümüne ulaşmak için uygulanması gereklili adımların hiç bir yorumya yer vermeksizin **açık**, **düzenli** ve **sıralı** bir şekilde söz ve yazı ile ifadesidir. Algoritmayı oluşturan adımlar özellikle basit ve açık olarak sıralanırılmalıdır.

# Algoritmaya Dair...

14

- Algoritmanın etkin bir şekilde oluşturulması Program yazma adımlından çok daha önemlidir.
- Hazırlanan algoritmanın programlama diliyle yazılması işin basit kısmıdır.
- Tasarladığınız algoritma iyi değilse, kullandığınız dilin hiçbir önemi yoktur (C, C++, C#, Java, Visual Basic vb.)
- Bir sorunun çözümü için birbirinden farklı birden fazla sayıda algoritma hazırlanabilir. Bu da gösteriyor ki herhangi bir problemin çözümü için birbirinden farklı yüzlerce bilgisayar programı yazılabılır.

# Algoritma Türlerine Örnekler

15

- Arama algoritmaları
- Bellek yönetimi algoritmaları
- Bilgisayar grafiği algoritmaları
- Evrimsel algoritmalar
- Genetik algoritmalar
- Kriptografik algoritmalar
- Optimizasyon algoritmaları
- Sıralama algoritmaları
- Veri sıkıştırma algoritmalar
- Veri Madenciliği algoritmaları
- İş Zekası algoritmaları
- Astronomi algoritmaları
- Dinamik Programlama algoritmaları
- Sağlık bilimleri algoritmaları
- Fizik algoritmaları
- Veritabanı algoritmaları
- İşletim sistemi algoritmaları
- ...

# Algoritmaların Sahip Olması Gereken Genel Özellikler

16

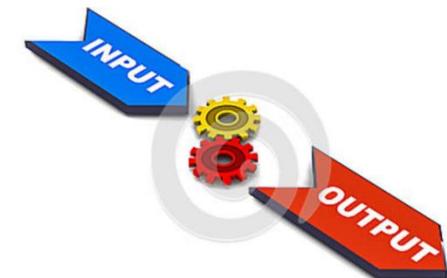
- Giriş/çıkış bilgisi,
- Sonluluk,
- Kesinlik,
- Etkinlik,
- Başarım ve performans.

# Algoritmaların Sahip Olması Gereken Genel Özellikler (Devam...)

17

## • Giriş/Cıkış Bilgisi

- Algoritmada giriş ve çıkış bilgileri olmalıdır. Dışarıdan gelen verilere giriş bilgisi denir. Bu veriler algoritmada işlenir ve çıkış bilgisini oluşturur. Çıkış bilgisi her algoritmada mutlaka vardır. Algoritmaların temel amacı giriş bilgisini işleyerek çıkış bilgisi oluşturmaktır. Ancak her durumda bir algoritmanın çıkış bilgisi istenenleri tam olarak karşılayamaz. Böyle durumlarda ilk algoritmanın ürettiği çıkış bilgisi başka bir algoritmaya giriş bilgisi olarak gönderilir ve böylece kullanıcı istediği bilgiye sahip olmuş olur.

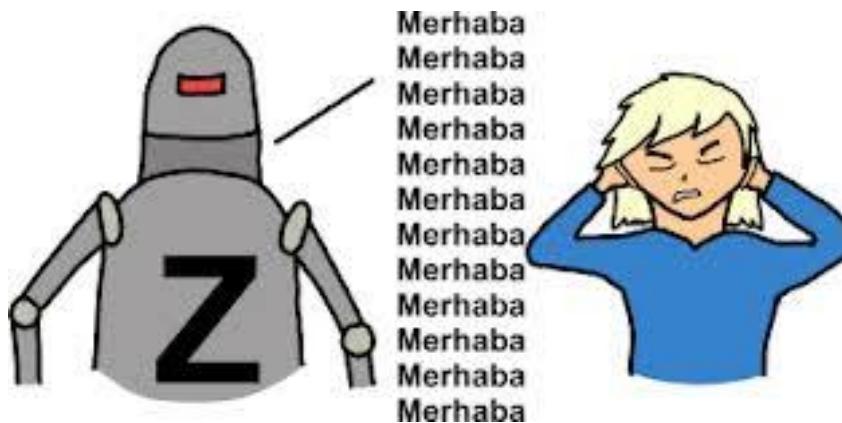


# Algoritmaların Sahip Olması Gereken Genel Özellikler (Devam...)

18

- **Sonluluk**

- Her türlü olasılık için algoritma sonlu adımda bitmelidir.
- Algoritma sonsuz döngüye girmemelidir.



# Algoritmaların Sahip Olması Gereken Genel Özellikler (Devam...)

19

- **Kesinlik**

- Her komut, kişinin kalem ve kağıt ile yürütebileceği kadar basit olmalıdır.
- Algoritmanın her adımı anlaşılır, basit ve kesin bir biçimde ifade edilmiş olmalıdır.
- Kesinlikle yorum gerektirmemeli ve belirsiz ifadelere sahip olmamalıdır.



# Algoritmaların Sahip Olması Gereken Genel Özellikler (Devam...)

20

- **Etkinlik**

- Yazılan algoritmalar etkin ve dolayısıyla gereksiz tekrarlardan uzak olısturulmalıdır. Bu algoritmanın temel özelliklerinden birisidir. Ayrıca algoritmalar genel amaçlı yazılıp yapısal bir ana algoritma ve alt algoritmalardan olısturulmalıdır. Böylece daha önce yazılmış bir algoritma daha sonra başka işlemler için de kullanılabilir.
- Buna örnek vermek gerekirse eğer elimizde, verilen  $n$  adet sayının ortalamasını bulmakta kullandığımız algoritma varsa bu algoritma, bir sınıfta öğrencilerin yaş ortalamasını bulan bir algoritma için de kullanılabilir.

# Algoritmaların Sahip Olması Gereken Genel Özellikler (Devam...)

21

- **Başarım ve Performans**

- Amaç donanım gereksinimi (bellek kullanımı gibi), çalışma süresi gibi performans kriterlerini dikkate alarak yüksek başarımlı programlar yazmak olmalıdır. Gereksiz tekrarlar ortadan kaldırılmalıdır. Bir algoritmanın performans değerlendirmesinde aşağıdaki temel kriterler göz önünde bulundurulur.
  - Birim İşlem Zamanı
  - Veri Arama ve Getirme Zamanı
  - Kıyaslama Zamanı
  - Aktarma Zamanı



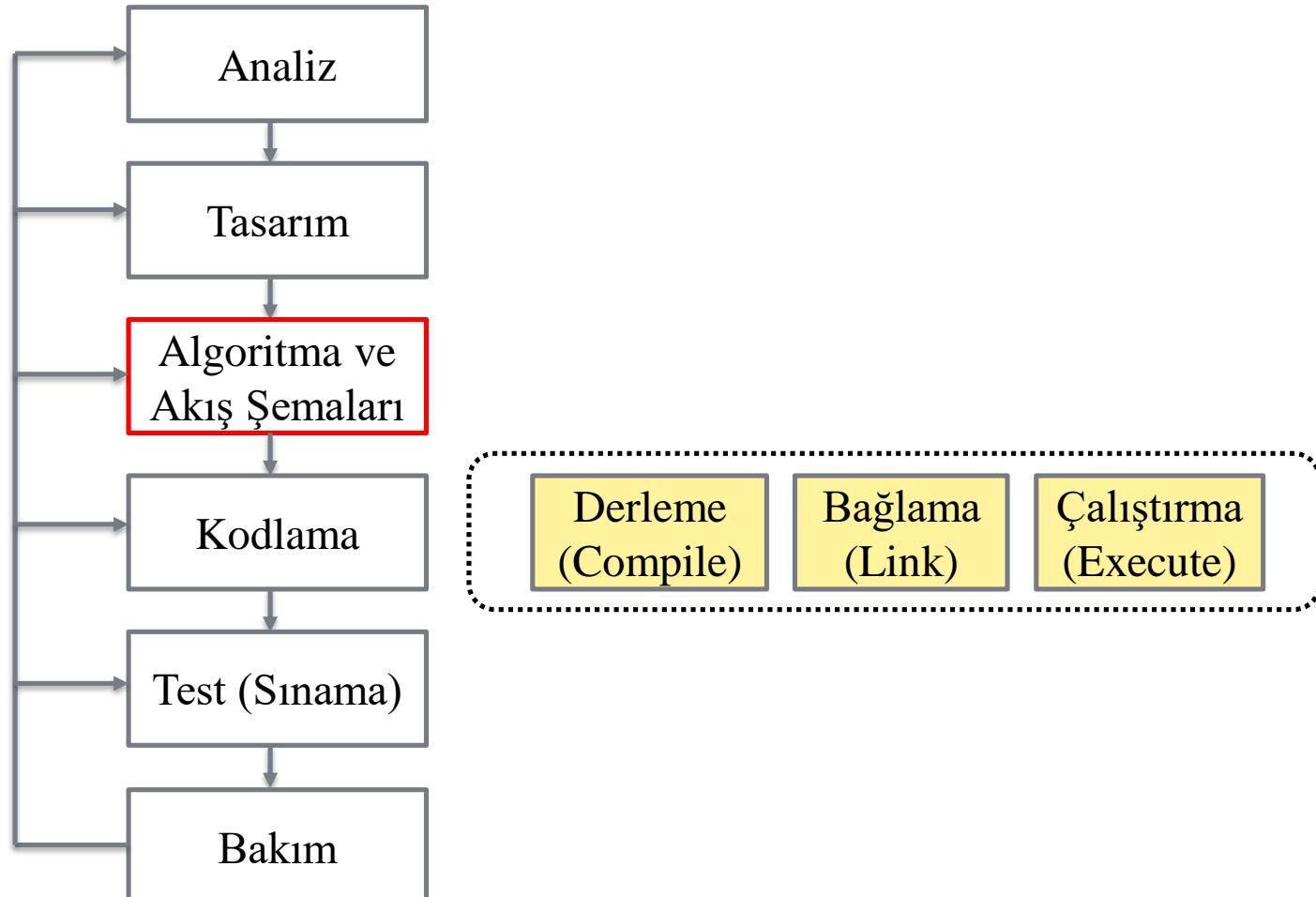
# Örnek: Çay Demleme Algoritması

22

- Mutfakta değilsen mutfağa git.
- Çayı kontrol et, çay yoksa?
- Markete git, çay al.
- Çaydanlığa bak, dolu değilse su doldur.
- Ocağı yak ve çaydanlığı ateşin üstüne koy.
- Suyun kaynamasını bekle.
- Su kaynadıktan sonra çayı bırak ve üstüne suyu dök.
- Yine demliğe biraz daha su ilave ederek bekle.
- Su kaynadığında biraz dinlendirerek ateşi kapat.
- Çay bardağını al çayını doldur.
- Çayına istediğiniz kadar şeker at (ya da atma) ve karıştır.
- Geldiğin odaya geri dön.
- Ve çayı iç.

# Yazılım Geliştirme Yaşam Döngüsünde Algoritma Nerede?

23



# Algoritma Gösterim Şekilleri

24

1. Düz yazı ile gösterim,
2. Sözde kod (pseudocode) ile gösterim,
3. Akış şeması ile gösterim.

# Düz Yazı ile Gösterim

25

- Çözülecek problem, adım adım metin olarak yazılır.
- Her satıra numara verilir.
- 'BASLA' ile başlanıp 'BİTİR' ile sonlandırılır.

# Örnek: Tahtaya Adını Yazma Algoritması

26

- 
- ```
graph TD; A[1. BAŞLA] --- B[2. Yerinden kalk]; B --- C[3. Yönü tahtaya doğru mu?]; C --- D[  └─ Hayırsa tahtaya dön]; C --- E[  └─ Evetse 4. adıma git]; C --- F[4. Tahtaya doğru yürü]; F --- G[5. Tahtaya geldin mi?]; G --- H[  └─ Hayırsa 4.adıma git]; G --- I[  └─ Evetse 6. adıma git]; G --- J[6. Kalemi al]; J --- K[7. Adını yaz]; K --- L[8. BİTİR]
```
1. BAŞLA
  2. Yerinden kalk
  3. Yönü tahtaya doğru mu?
    - └─ Hayırsa tahtaya dön
    - └─ Evetse 4. adıma git
  4. Tahtaya doğru yürü
  5. Tahtaya geldin mi?
    - └─ Hayırsa 4.adıma git
    - └─ Evetse 6. adıma git
  6. Kalemi al
  7. Adını yaz
  8. BİTİR

# Örnek: Tahtaya Adını Yazma Algoritması

27

- Örneğin amacı, adımların tutarlılığını ve mantıksal sırasını göstermektir.
- Burada emirler, belli sorgulamalar yapılarak ve mantıksal bir sıra içinde verilmiştir.
- Yerinden kalk emri verilmeden kişiden yürümesi istenemez.
- Kalemi almadan adını yaz emrinin verilmesi doğru olmaz.
- Sorgulamalarla da işlemi yapıp yapmadığı kontrol edilmiştir.

# Örnek: Tahtaya Adını Yazma Algoritması

28

- Aslında bilgisayar bu tür işleri yerine getiremez.
- Kullanıcılar bilgisayarlara belli girdiler verir.
- Onlar da programcının verdiği adımlara göre bu girdiler üzerinde matematiksel ve mantıksal işlemler yaparak bir çıktı üretirler.

# Sözde Kod (Pseudocode) ile Gösterim

29

- Herkesin anlayabileceği ve rahatlıkla bir programlama diline çevrilebilecek basit komutlardan oluşan bir dildir.
- Sözde kodun temel işlevi program geliştirmeye geçmeden algoritmayı oluşturmak ve üzerinde tartışabilmektir.
- Sözde kodlar, doğrudan konuşma dilinde ve programlama mantığı altında, eğer, iken gibi koşul kelimeleri ve  $> = <$  gibi ifadeler ile beraber yazılır.
- Programda kullanılacak elemanları temsil etmek üzere uygun isimler veya değişkenler seçilir.
- Cebirsel notasyon ve kararlar kullanarak aritmetik işlemler gerçekleştirir.

# Örnek: İki Sayının Toplamı Algoritması

30

1. BAŞLA
2. Birinci sayıyı gir
3. İkinci sayıyı gir
4. İki sayıyı topla
5. Sayıların toplam değerini yaz
6. BİTİR

Sözde koda  
nasıl çeviririz?

# Örnek: İki Sayının Toplamı Algoritması

31

## Düz Yazılı

1. BAŞLA
2. Birinci sayıyı gir
3. İkinci sayıyı gir
4. İki sayıyı topla
5. Sayıların toplam değerini yaz
6. BİTİR

## Sözde Kod

- Toplam için T, birinci sayı için X, ikinci sayı için Y seç
1. BAŞLA
  2. X değerini OKU
  3. Y değerini OKU
  4.  $T = X + Y$
  5. T değerini YAZ
  6. BİTİR

# Örnek: Üçgenin Alanını Hesaplayan Algoritma

32

1. BAŞLA
2. Taban değerini gir
3. Yükseklik değerini gir
4. Taban ile yüksekliği çarp ve sonucu ikiye böl
5. Çıkan sonucu yaz
6. BITİR

Sözde koda  
nasıl çeviririz?

# Örnek: Üçgenin Alanını Hesaplayan Algoritma

33

## Düz Yazı

1. BAŞLA
2. Taban değerini gir
3. Yükseklik değerini gir
4. Taban ile yüksekliği çarp ve sonucu ikiye böl
5. Çıkan sonucu yaz
6. BİTİR

## Sözde Kod

Taban için t, yükseklik için y, alan için A seç

1. BAŞLA
2. t değerini OKU
3. y değerini OKU
4.  $A = (t * y) / 2$
5. A değerini YAZ
6. BİTİR

# Akış Şemaları ile Gösterim

34

- Bir algoritmanın görsel şekiller ve sembollerle ifade edilmiş haline «*Akış Şemaları*» adı verilir.
- Akış şeması sembollerini **ANSI** (American National Standards Institute) standartı olarak belirlenmiş ve tüm dünyada kullanılmaktadır.
- Algoritma doğal dile yazıldığı için herkes tarafından anlaşılamayabilir ya da başka anamlar çıkarılabilir.
- Ancak akış şemalarında her bir şekil standart bir anlam taşıdığı için farklı yorumlanması mümkün değildir.

# Akış Şeması Şekilleri

35

BAŞLA

BİTİR



Akış şemasının **başlangıç** ve **bitiş** yerlerini gösterir. Başlangıç simgesinden çıkış oku vardır. Bitiş simgesinde giriş oku vardır.

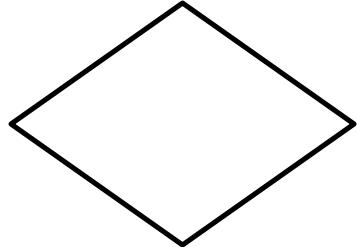
**Aritmetik işlemler** ve değişik atama işlemlerinin temsil edilmesi için kullanılır.

Dışarıdan bilgi **giriş** ve **çıkışı** için kullanılır.

Belgeye, yazıcıya, **ekrana çıktı** için kullanılır.

# Akış Şeması Şekilleri

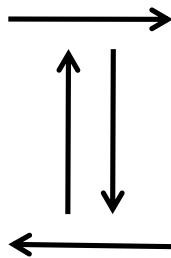
36



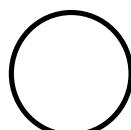
Kontrol ve karar verme işlemleri temsil eder.



Döngü olduğunu gösterir.



Oklar şemanın akış yönünü belirler.



Bağlantı işlemleri temsil eder.

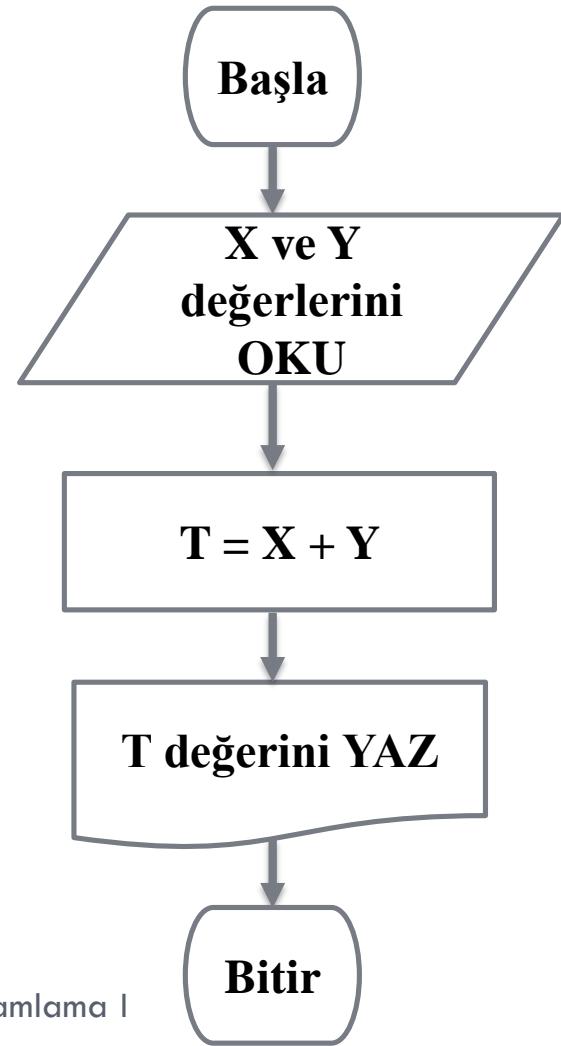
# Örnek: İki Sayının Toplamı Akış Şeması

37

## Sözde Kod

Toplam için T, birinci sayı için X, ikinci sayı için Y seç

1. BAŞLA
2. X değerini OKU
3. Y değerini OKU
4.  $T = X + Y$
5. T değerini YAZ
6. BİTİR



# Mantıksal Yapılar

38

- Bir bilgisayar programının geliştirilmesinde kullanılan programlama dili ne olursa olsun bu programların akış şemalarında genel olarak üç basit mantıksal yapı kullanılır.
  1. Sıralı Yapı
  2. Karar Verme Yapısı
  3. Tekrarlı Yapı

# Mantıksal Yapılar: Sıralı Yapı

39

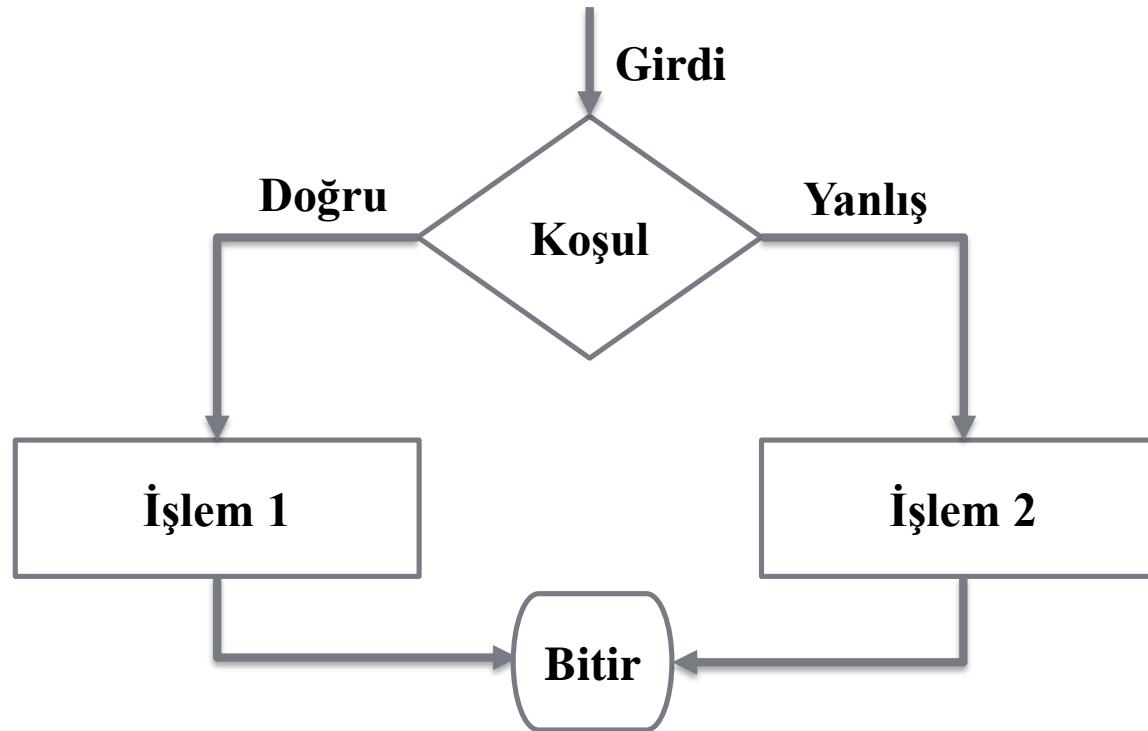
- Sıralı yapı, hazırlanacak programdaki her işlemin mantık sırasına göre nerede yer olması gerektiğini vurgular. Bu yapı sona erinceye kadar ikinci bir işlem başlayamaz.



# Mantıksal Yapılar: Karar Verme Yapısı

40

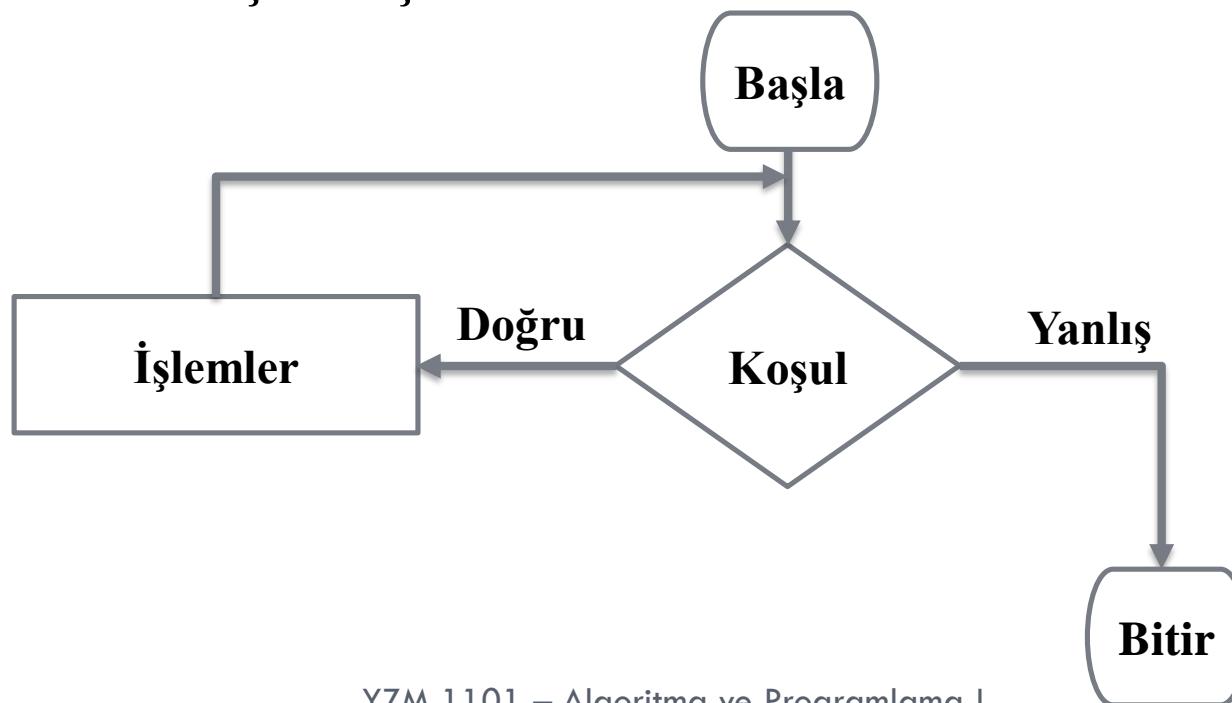
- Birden fazla sıralı yapı seçenekini kapsayan modüllerde, hangi şartlarda hangi sıralı yapının seçileceğini belirler.



# Mantıksal Yapılar: Tekrarlı Yapı

41

- Algoritma içinde, bazı satırlar tekrarlı şekilde işlem görüyorsa, bir döngü söz konusudur. Döngülere belirli bir koşul geçerli olduğu sürece devam eden eylemleri tanımlamak için başvurulur.



# İşlemler ve Operatörler

42

- İşlemler 3'e ayrılır:

## 1. Matematiksel İşlemler

- Temel Aritmetik İşlemler: Toplama, çıkarma, çarpma, bölme.
- Matematiksel Fonksiyonlar: Üstel, logaritmik, trigonometrik, hiperbolik vb.

## 2. Karşılaştırma İşlemleri

## 3. Mantıksal (Logic) İşlemler

# Matematiksel İşlemler

43

| İşlem   | Gösterim | Matematiksel Yazım                      | Bilgisayar Gösterim             |
|---------|----------|-----------------------------------------|---------------------------------|
| Toplama | $a + b$  | $a+b-c+2abc-7$                          | $a+b-c+2*a*b*c-7$               |
| Çıkarma | $a - b$  | $a+b^2-c^3$                             | $a+b^2-c^3$                     |
| Carpma  | $a * b$  | $a - \frac{b}{c} + 2ac - \frac{2}{a+b}$ | $a-b/c+2*a*c-2/(a+b)$           |
| Bölme   | $a / b$  | $\sqrt{a+b} - \frac{2ab}{b^2 - 4ac}$    | $(a+b)^(1/2)-2*a*b/(b^2-4*a*c)$ |
| Üs alma | $a ^ b$  | $\frac{a^2 + b^2}{2ab}$                 | $(a^2+b^2)/(2*a*b)$             |

# Karşılaştırma İşlemleri

44

- Değişkenlerin büyük olma, küçük olma ve eşit olma durumlarını kontrol eden işlemlerdir.

| İşlem Sembolü | Anlamı        |
|---------------|---------------|
| =             | Eşittir       |
| < >           | Eşit değildir |
| >             | Büyüktür      |
| <             | Küçüktür      |
| >=            | Büyük eşittir |
| <=            | Küçük eşittir |

# Mantıksal İşlemler

45

- «Ve, Veya, Değil» operatörleri hem matematiksel işlemlerde hem de karar ifadelerinde kullanılır.

| Mantıksal İşlem | Komut |
|-----------------|-------|
| Ve              | And   |
| Veya            | Or    |
| Değil           | Not   |

- VE** bağlacı ile söylemek istenen her iki koşulun da sağlanmasıdır. VE bağlacı ile bağlanmış önermelerden en az birinin yanlış olması sonucu yanlış yapar.
- VEYA** bağlacı ile bağlanan koşullardan bir tanesinin doğru olması sonucu doğru yapar.

# Mantıksal İşlemler (devam...)

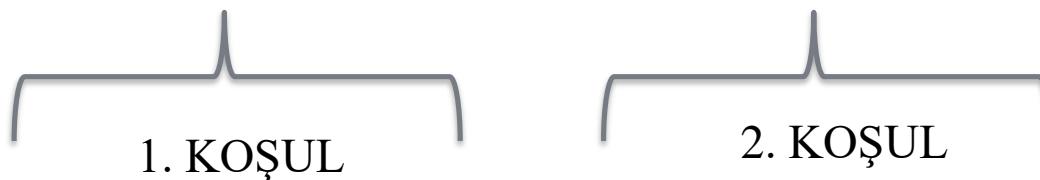
46

- **DEĞİL** bağlacı; doğruluğu yanlış, yanlışlığını doğru yapar.

**Örnek:** Yazılım departmanında çalışan erkek personellerden yaşının 30'un üzerinde olanları ekrana yazdır.

Eğer;

- $(\text{perCinsiyet} = \text{Erkek}) \text{ VE } (\text{perYas} > 30)$  ise ekrana yazdır.



# Algoritmada Kullanılan Terimler

47

1. Tanımlayıcı
2. Değişken
3. Atama
4. Sayaç
5. Döngü

# Algoritmada Kullanılan Terimler: Tanımlayıcı

48

- Programcı tarafından oluşturulur.
- Programdaki değişkenleri, sabitleri, kayıt alanlarını, özel bilgi tiplerini vb. adlandırmak için kullanılan kelimelerdir.
- Tanımlayıcılar, yerini tuttukları ifadelere çağrıım yapacak şekilde seçilmelidir.
- İngiliz alfabetesindeki A-Z veya a-z arasındaki 26 harf ile 0-9 arası rakamlar kullanılabilir.
- Sembollerden sadece alt çizgi (\_) kullanılabilir.
- Tanımlayıcı isimleri harfle veya alt çizgiyle başlayabilir.
- Tanımlayıcı ismi, rakamla başlayamaz veya sadece rakamlardan oluşamaz.

# Algoritmada Kullanılan Terimler: Değişken

49

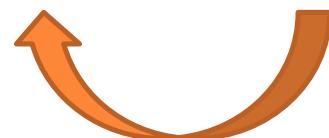
- Programın her çalıştırılmasında, farklı değerler alan bilgi/bellek alanlarıdır.
- Değişken isimlendirilmeleri, tanımlayıcı kurallarına uygun biçimde yapılmalıdır.
- Örnekler:
  - Dikdörtgenin uzun kenarının aktarıldığı değişken:
    - **uzun\_kenar**,
    - **UzunKenar**,
    - **uzunKenar**
  - Bir öğrenciye ait ismin aktarıldığı değişken:
    - **isim**,
    - **ogrenci\_isim**,
    - **ogrenciIsim**

# Algoritmada Kullanılan Terimler: Atama

50

- Değişkenlere değer aktarma işlemidir. Değişkenlere atanmış bu değerler daha sonra tekrar kullanılabilirler.

**Değişken = Değer**



**i = 0**

Sağdaki **Değer** sonucu **Değişken**'e aktarılır. Bu durumda Değişken'in bir önceki değeri varsa silinir.

# Algoritmada Kullanılan Terimler: Sayaç

51

- Bazı işlemlerin belirli sayıda yaptırılması ve üretilen değerlerin sayılması gerekebilir.
- Bu tür sayma işlemlerine algoritmada Sayaç adı veriler.
- Sayaçlar da birer değişkendir.

$$\text{Sayac} = \text{Sayac} + 1$$

Bu işlemde **Sayac** değişkenine **1** eklenmekte ve oluşan sonuç yine kendisine yani **Sayac** değişkenine aktarılmaktadır.

# Algoritmada Kullanılan Terimler: Döngü

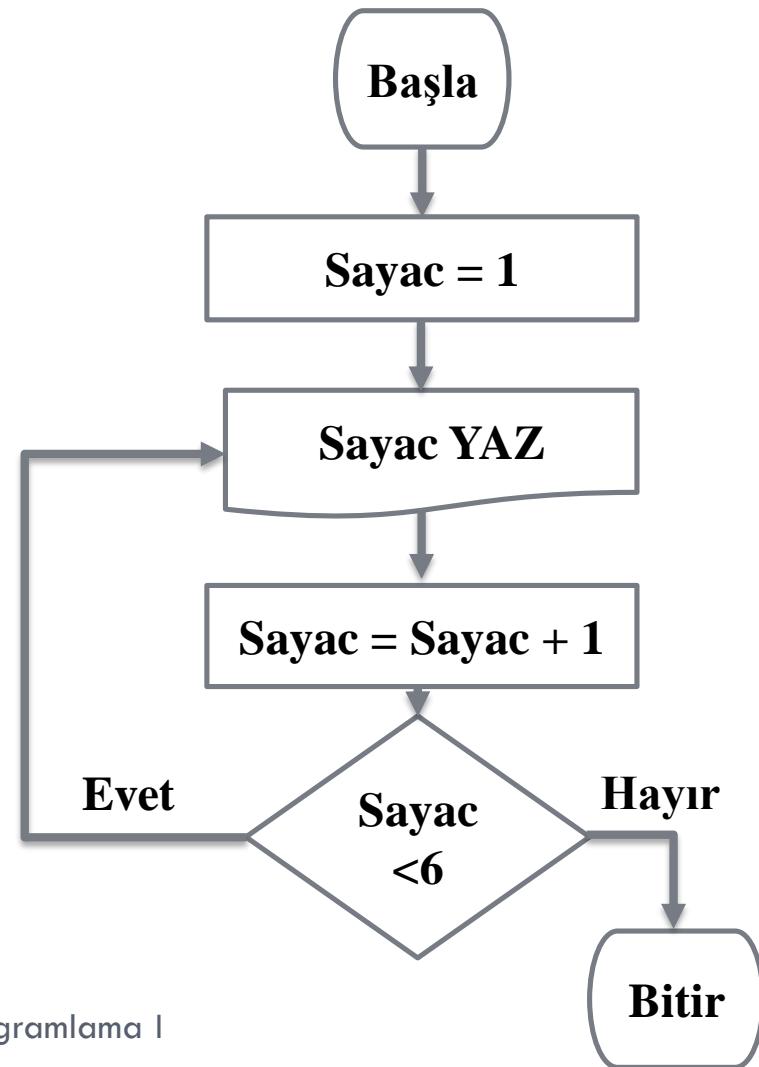
52

- Bir çok programda bazı işlemler, belirli ardışık değerlerle gerçekleştirilmekte veya belirli sayıda yaptırılmaktadır.
- Programlardaki belirli işlem bloklarını, verilen sayıda gerçekleştiren işlem akış çevrimlerine “**döngü**” denir.
- **Örneğin;** 1 ile 1000 arasındaki tek sayıların toplamını hesaplayan programda  $T=1+3+5 \dots$  yerine 1 ile 1000 arasında ikişer artan bir döngü kurulu ve döngü değişkeni ardışık toplanır.

# Örnek: 1-5 arasındaki sayıların ekrana yazdırılması

53

1. BAŞLA
2. Sayac = 1
3. Sayac değerini YAZ
4. Sayac = Sayac + 1
5. Eğer Sayac < 6, GİT 3
6. BİTİR



# Örnek: 1-5 arasındaki sayıların ekrana yazdırılması

54

1. BAŞLA

2. Sayac = 1

3. Sayac değerini YAZ ←

4. Sayac = Sayac + 1

5. Eğer Sayac < 6, GİT 3 →

6. BİTİR

Değişken İzleme Tablosu

| Eski Sayac | Yeni Sayac | Ekran |
|------------|------------|-------|
| 1          | 2          | 1     |
| 2          | 3          | 2     |
| 3          | 4          | 3     |
| 4          | 5          | 4     |
| 5          | 6          | 5     |

# Örnek: 1-10 Arasındaki Tek Sayıların Toplamı

55

1. BAŞLA
2. Sayac = 1
3. Toplam = 0
4. Eğer Sayac > 10, GİT 8
5. Toplam = Toplam + Sayac
6. Sayac = Sayac + 2
7. GİT 4
8. BİTİR

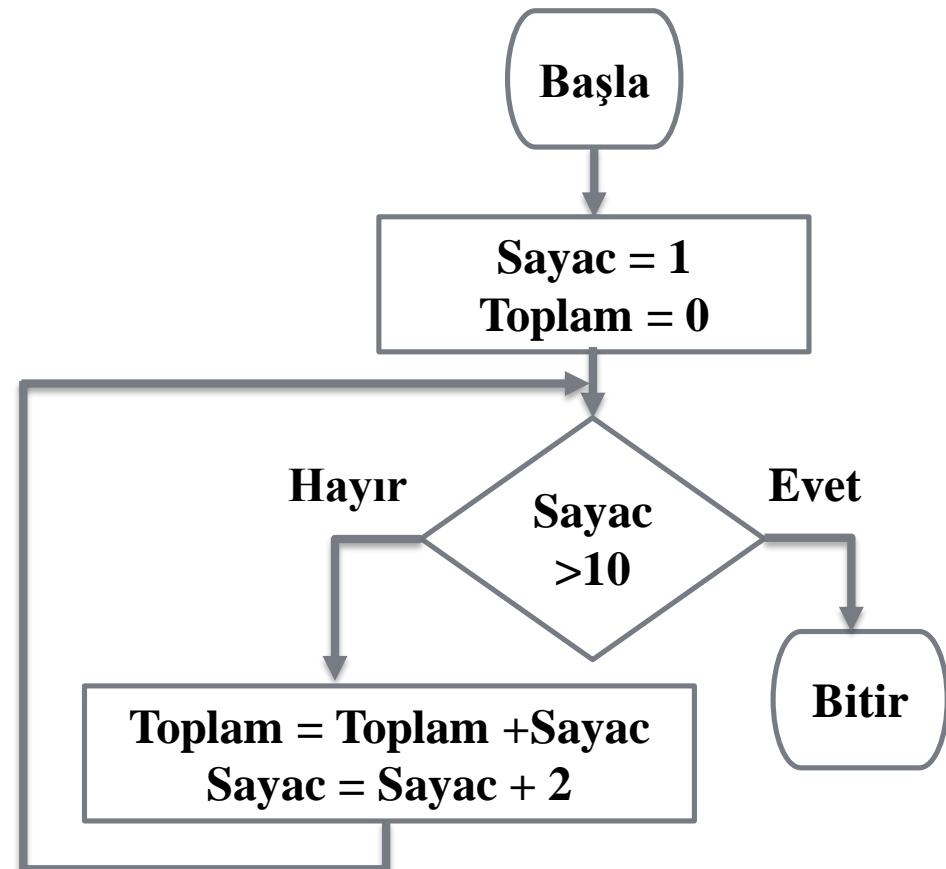
Değişken İzleme Tablosu

| Eski Sayac | Eski Toplam | Yeni Sayac | Yeni Toplam |
|------------|-------------|------------|-------------|
| 1          | 0           | 3          | 1           |
| 3          | 1           | 5          | 4           |
| 5          | 4           | 7          | 9           |
| 7          | 9           | 9          | 16          |
| 11         |             |            |             |

# Örnek: 1-10 Arasındaki Tek Sayıların Toplamı (Akış Şeması)

56

1. BAŞLA
2. Sayac = 1
3. Toplam = 0
4. Eğer Sayac > 10, GİT 8
5. Toplam = Toplam + Sayac
6. Sayac = Sayac + 2
7. GİT 4
8. BİTİR



# 2. BÖLÜM

57

## PROGRAMLAMAYA GİRİŞ

# Programlama Terimleri ve Programlama Ortamı

58

- Program
- Programlama
- IDE (Integrated Development Environment – Tümleşik Geliştirme Ortamı)
- Derleyici (Compiler)
- Yorumlayıcı (Interpreter)
- Bağlayıcı (Linker)
- Çalıştırma (Execution)
- Hata Türleri
- Debug

# Program

59

- Var olan bir problemi çözmek amacıyla bilgisayar dili kullanılarak oluşturulmuş anlatımlar (komutlar, kelimeler, aritmetik işlemler, mantıksal işlemler vb.) bütününe «*program*» denir.

```
End Sub
Private Sub tbToolBar_ButtonClick()
On Error Resume Next
timTimer.Enabled = True
Select Case Button.Key
    Case "Back"
        brwWebBrowser.Go
    Case "Forward"
        brwWebBrowser.
    Case "Refresh"
        brwWebBrows
    Case "Home"
        WebBro
```

# Programlama

60

- Bir programı oluşturabilmek için gerekli komutların belirlenmesi ve uygun biçimde kullanılmasına **programlama** denir.
- Programlama, bir programlama dili kullanılarak yapılır.
  - Bu programlama dili Java ve C# gibi yüksek seviyede bir dil olabileceği gibi C, Assembly ve bazı durumlarda makine dili de olabilir.
- Yazılan kaynak kodu genellikle bir derleyici ve bağlayıcı yardımıyla belirli bir sistemde çalıştırılabilir hale getirilir. Ayrıca kaynak kodu, bir yorumlayıcı yardımıyla derlemeye gerek duyulmadan satır satır çalıştırılabilir.

# Programlama (devam...)

61

- Programlama aktivitesi genelde “Merhaba Dünya” (**Hello World!**) programı yazılmasıyla başlar.

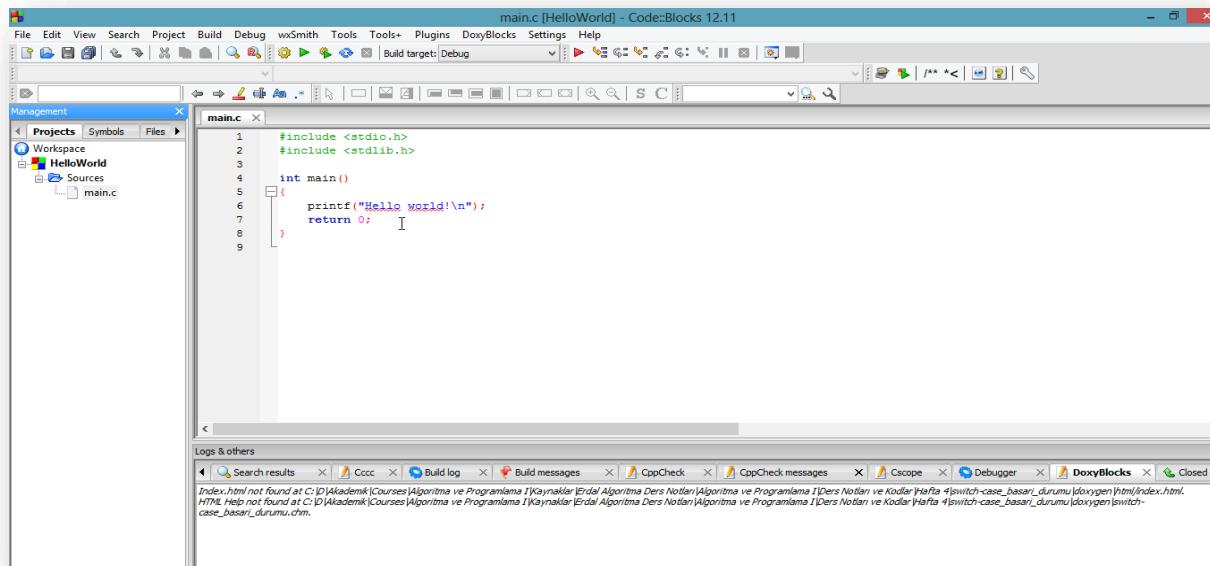
```
010000000001010001101100000010010110001  
11000101110100010001111111110100000100  
010100101100001101011101101011011001000  
1101100000101011001000100001110001001111  
01001100101101001101010011110111101111  
00110100010011010101001111010000110100  
1001001101000000000000000000000000000000  
10001001int main()  
10101001{  
1110011000    printf("Hello World");  
0100000111    return 42;  
0011010001000111010010110001101000011010  
10010011011110101110111100000001010001111  
10001001000101011100100111011101000101111  
1010100111001101010111000010101000011000  
11100110000011011111101010011111100001100  
0100000011111110101001001001101010111011
```

- Bir programlama dilini öğrenmekteki tek zorluk *programlamanın ne olduğunu öğrenmektir*. Bundan sonraki aşamalar daha basittir.

# IDE (Integrated Development Environment – Tümleşik Geliştirme Ortamı)

62

- IDE yazılımcının hızlı ve rahat bir şekilde program geliştirebilmesini amaçlayan, geliştirme sürecini organize edebilen birçok araç ile birlikte geliştirme sürecinin verimli kullanılmasına katkıda bulunan araçların tamamını içerisinde barındıran bir yazılım türüdür.



# IDE (Integrated Development Environment – Tümleşik Geliştirme Ortamı) (devam...)

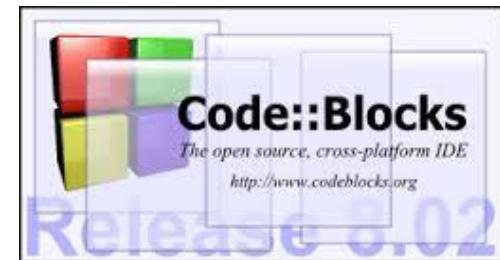
63

- Tümleşik geliştirme ortamlarında olması gereklili en temel özellikler aşağıdaki gibidir:
  - Programlama diline göre **sözdizimi** renklendirmesi yapabilen kod yazım editörü.
  - Kod dosyalarının **hiyerarşik** olarak görülebilmesi amacıyla hazırlanmış gerçek zamanlı bir dizelge.
  - Tümleşik bir derleyici, **yorumlayıcı** ve **hata ayıklayıcı**.
  - Yazılımın **derlenmesi**, **bağlanması**, çalışmaya tümüyle hazır hale gelmesi ve daha birçok ek işi otomatik olarak yapabilmek amacıyla küçük inşa araçları.

# IDE (Integrated Development Environment – Tümleşik Geliştirme Ortamı) (devam...)

64

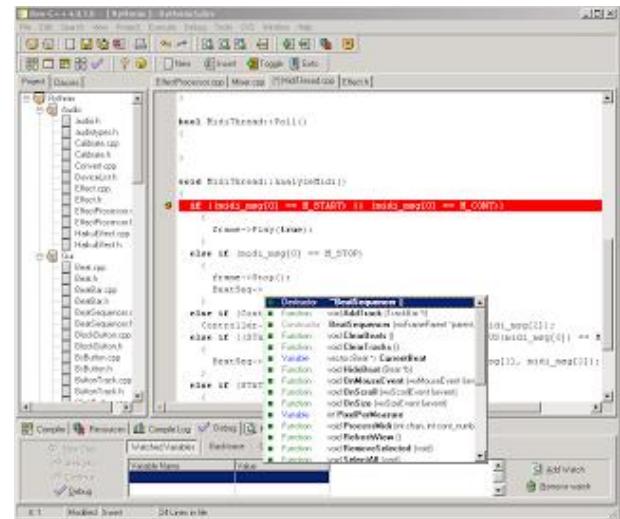
- En bilinen tümleşik geliştirme ortamları: Eclipse, Microsoft Visual Studio, Code::Blocks, Dev-C++, Anjuta, KDevelop, NetBeans...



# Derleyici (Compiler)

65

- Derleyici, yazılan programın kaynak kodunu okuyup îçerisinde mantıksal veya yazınsal hatalar olup olmadığını bulan, bulduğu hataları kullanıcıya göstererek programın düzeltilmesine yardım eden, hata yoksa programın çalıştırılması öncesinde kaynak kodu makine çeviren diline bir yazılımdır.



# Yorumlayıcı (Interpreter)

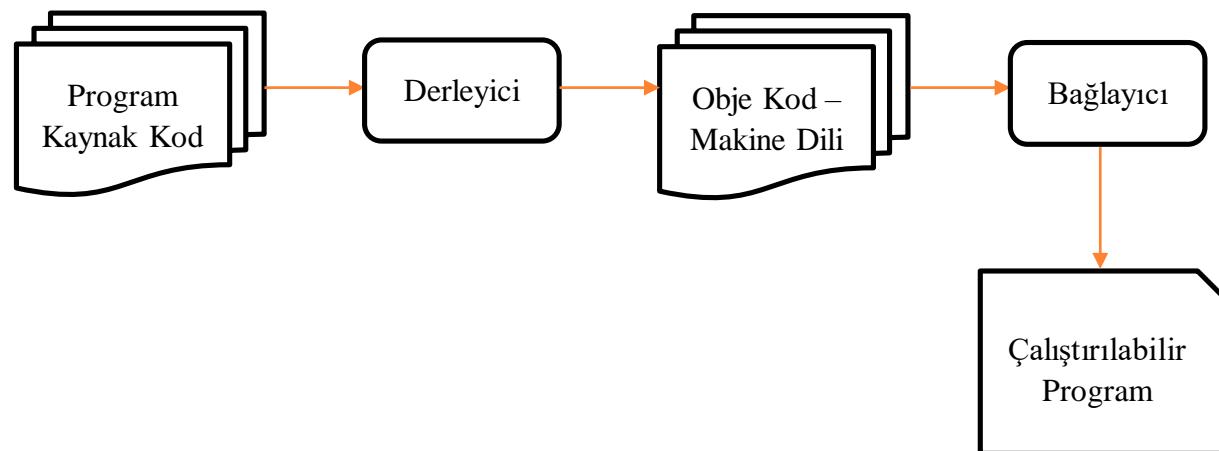
66

- Yorumlayıcı, kaynak kodu **kısim** **kısim** ele alarak doğrudan çalıştırır.
- Yorumlayıcılar **standart bir çalıştırılabilir kod** üretmezler.
- Yorumlama işlemi aşama aşama yapılmadığı için genellikle **ilk hatanın bulunduğu yerde programın çalışması kesilir**.
- Derleyicilerin tersine kodun işlenmeyen satırları üzerinden hiç geçilmez ve buralardaki hatalar ile ilgilenilmez.
- Yorumlayıcılar genelde kaynak koddan, makine diline anlık olarak dönüşüm yaptıkları için, derleyicilere göre daha yavaş çalışırlar. Ayrıca kodu iyileştirme (optimizasyon) imkanı da çoğu zaman yoktur.

# Bağlayıcı (Linker) ve Çalıştırma (Execute)

67

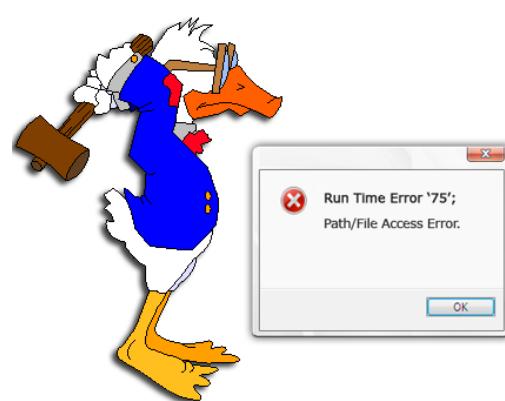
- **Bağlayıcı:** Derleyici tarafından object dosyasına çevrilen bir veya birden çok dosyanın birbirleri ile ilişkilendirmesi ve tek bir çalıştırılabilir dosyaya (Örneğin Windows exe) çevrilmesini sağlayan yazılımdır.
- **Çalıştırma:** Oluşturulan makine dili programının çalıştırılması adımıdır.



# Yazılım Hataları

68

- Yazılım geliştirme sürecinin herhangi bir aşamasında (temel olarak analiz, tasarım, kodlama, test, bakım) yapılan insanı hatalardır.
- Hiçbir yazılımcı veya programcı isteyerek hata yapmaz. Dolayısı ile programı kendi kendine test ederken yaptığı hatayı da görmemesi normaldir.



# Yazılım Hataları (devam...)

69

- Sıfır hatalı bir yazılım üretmek pratikte mümkün değildir. Ancak doğru hata yönetimi yaparak hata sayısını azaltabilir ve hata oluştuğunda müdahale için daha hızlı olabilirsiniz.
- Uygulama geliştirme aşamasında hatalar 3 grupta değerlendirilir:
  1. Syntax Error – Sözdizimi Hataları
  2. Run-time Error – Çalışma Zamanı Hataları
  3. Logic Error (Bug) – Mantıksal Hatalar (Böcek)

# **1. Syntax Error – Söz dizimi Hataları**

70

- Yazılan programda **programlama dili kurallarına aykırı** bir takım **ifadelerden** dolayı karşılaşabilecek hatalardır.
- Düzeltilmesi **basit hatalardır**.
- Hatanın bulunduğu **satır derleyici tarafından rapor** edilir.
- Günümüz IDE’lerinde bu sıkıntılar neredeyse yok denecek kadar azdır. Özellikle kod editörlerinin gelişmiş yazım denetimi sayesinde yazılımcılar söz dizimi hatalarını derlemeye gerek bile kalmadan fark edebiliyorlar.
- Eğer bir derlemede Syntax Error alındı ise obje kod üretilememiştir demektir.

## 2. Run-time Error – Çalışma Zamanı Hataları

71

- Programın çalıştırılması sırasında karşılaşılan hatalardır. Programcının ele almadığı bir takım aykırı durumlar ortaya çıktığında programın işletim sistemi tarafından kesilmesi ile ortaya çıkar. Bu tip hatalarda hata mesajı çoğunlukla çalışan işletim sisteminin dili ile verilir.
- Eğer bu tip hataları kullanıcı ele almışsa, program programcının vereceği mesajlarla ve uygun şekilde sonlandırılabilir. Bu tip hataların nerelerde ve hangi şartlarda ortaya çıkabileceğini bazen kestirmek zor olabilir.
- Örneğin **olmayan bir dosya açmaya çalışmak, var olan bir dosyanın üzerine yazmaya çalışmak**, olmayan bir bellek kaynağından bellek ayırtmaya çalışmak, olmayan bir donanıma ulaşmaya çalışmak vs.

### 3. Logic Error (Bug) – Mantıksal Hatalar (Böcek)

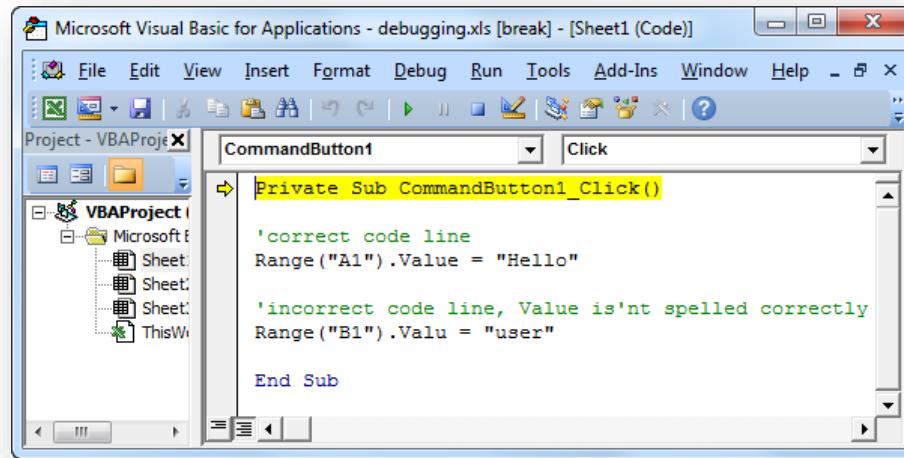
72

- Karşılaştabileceğiniz **en tehlikeli hatadır**. Programlama mantığında bir takım şeylerin yanlış düşünülmesinden kaynaklanır. Hata **test aşamasında veya müşteri kullanımı** sırasında ortaya çıkar.
- Örneğin: Hesaplanması gereken veya bulunması gereken değerlerin eksik veya yanlış hesaplanması mantıksal bir hatadır. Bu sorunun giderilebilmesi için Analiz aşamasına kadar geri dönülmesi gerekebilir. Bazen bu hatanın nereden kaynaklandığını bulabilmek çok zor olmaktadır.
- Gerek **serbest yazılım** gerek **ticari yazılımların** tümünde **bug** dediğimiz **mantıksal hatalar** bulunur.
- Günümüzde en etkin yazılım firmaları bile yazılımlarında **bug olduğunu kabul eder** ve zaman zaman bu bugları giderebilmek için ya yazılımlarına yama yazılımı (**Update, Patch**) üretirler ya da o yazılımın yeni bir versiyonunu piyasaya sürerler.

# Debug (Bugdan arındırma)

73

- Mantıksal hataları giderebilmek ve yazılımdaki hataları (bug) bulabilmek için yapılan işlemin adıdır. Genellikle yazılan programın adım adım ve denetim altında çalıştırılmasıdır.
- Programın her一步一步ında ilgili değişkenlerin hangi değere sahip olduğunu görmeyi sağlayarak anormal bir durumu daha kolay izleyip bulmanızı sağlar.



# KAYNAKLAR

74

- Okt. Tuna GÖKSU Bilgisayar ve Programlama Sunumu
- N. Ercil Çağiltay ve ark., C DERSİ PROGRAMLAMAYA GİRİŞ, Ada Matbaacılık, ANKARA; 2009.
- Milli Eğitim Bakanlığı "Programlamaya Giriş ve Algoritmalar Ders Notları", 2007
- <http://www.AlgoritmaveProgramlama.com>



# İYİ ÇALIŞMALAR...

Yrd. Doç. Dr. Deniz KILINÇ  
deniz.kilinc@cbu.edu.tr