

# BİLGİSAYAR VE PROGRAMLAMAYA GİRİŞ

*Yrd. Doç. Dr. Durmuş ÖZDEMİR*  
*E-Posta: [durmus.ozdemir@dpu.edu.tr](mailto:durmus.ozdemir@dpu.edu.tr)*

## Bilgisayar ve Programlamaya Giriş

- ▶ Dersin amacı
  - ▶ Bilgisayarlara giriş, algoritma geliştirme, akış diyagramları
  - ▶ Programlamaya giriş, Java diliyle program yazma
- ▶ Sizden beklenen
  - ▶ Dersleri takip etmek (yoklama için değil, ders için)
  - ▶ Derste aynı anda sadece bir kişi konuşur
  - ▶ Çok düşünmeniz, anlatılanların dışında araştırma yapmanız
  - ▶ En önemlisi çok sayıda **PROGRAM YAZMAK**

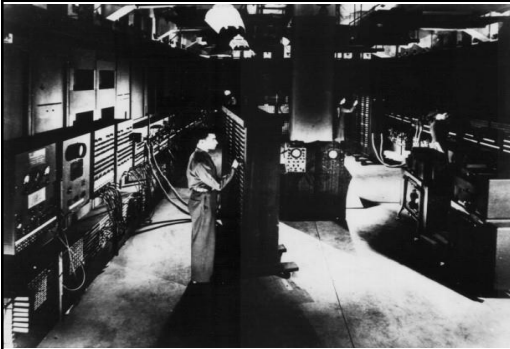
## Kaynaklar ve Ders Takibi

- Kitap:
  - Yeni Başlayanlar İçin Java (),
  - JAVA ve JAVA TEKNOLOJİLERİ (uygulamaya dönük) KODLAB YAYINCILIK
- Elektronik kaynaklar
  - Moodleden elektronik kaynak
  - İnternette İngilizce ve Türkçe kaynaklardan yararlanabilme!
- En iyi kaynağınız doğru arama yaptığınız sürece: Google, (Herşey için 1 nolu kaynağımız doğru arama yaptığınız sürece :İnternet)

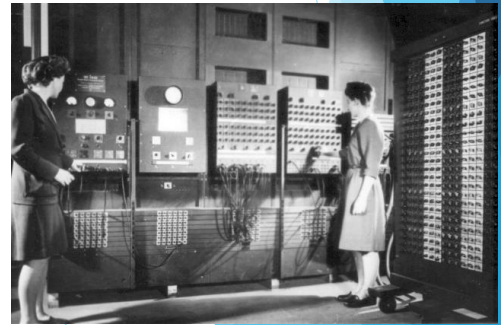
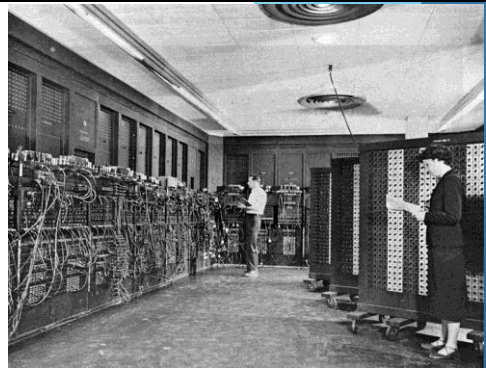


## Lab Saatleri + Ödevler + Quizler

- ▶ Lablarda pratik örneklerle programlama öğreneceksiniz
- ▶ Lablara gelmeden önce, lab notlarını (varsa), ders notlarını, ilgili örnekleri vs. Okuyarak hazırlanın
- ▶ Labda arkadaşlarınızdan yardım alabilirsiniz
- ▶ Serbest çalışma saatlerinde pratik yapabilirsiniz
- ▶ Problemleri, örnekleri arkadaşlarınızla tartışabilir, internetten örnekler arayabilirsiniz
- ▶ Ama ödevler kendi çalışmanızın ürünü olmalıdır
- ▶ Sınıfta yapılacak quizler ya da ödevler yıl sonu ortalamınıza (notunuza) %20 oranında etki edecektir



## ENIAC

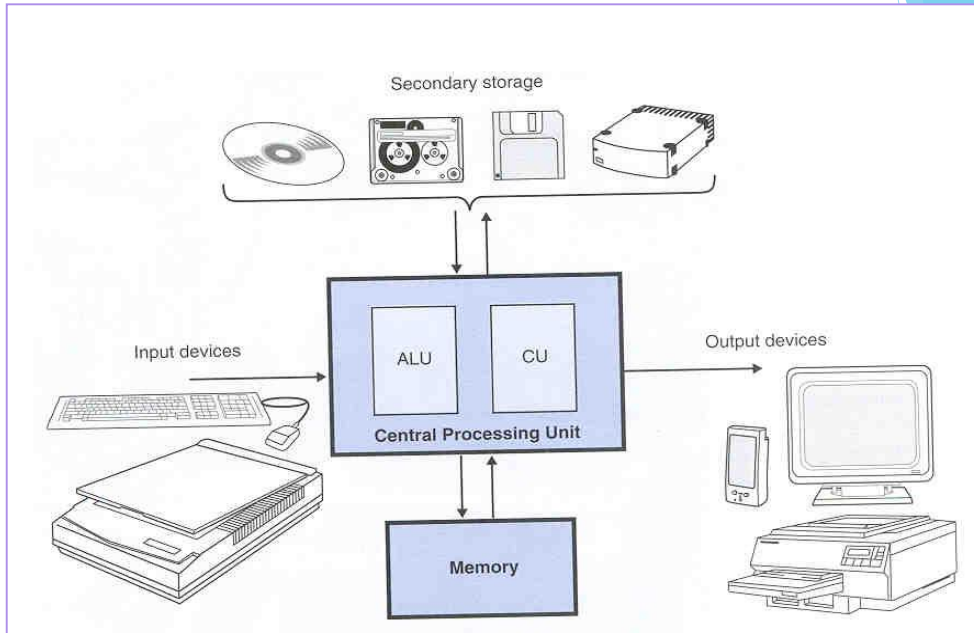


## ENIAC

- ▶ Besides its speed, the most remarkable thing about ENIAC was its size and complexity. ENIAC contained 17,468 [vacuum tubes](#), 7,200 crystal [diodes](#), 1,500 [relays](#), 70,000 [resistors](#), 10,000 [capacitors](#) and around 5 million hand-soldered joints. It weighed 30 [short tons](#) (27 t), was roughly 8.5 by 3 by 80 feet ( $2.6 \text{ m} \times 0.9 \text{ m} \times 26 \text{ m}$ ), took up 680 square feet ( $63 \text{ m}^2$ ), and consumed 150 [kW](#) of power.<sup>[8]</sup> Input was possible from an IBM card reader, and an IBM card punch was used for output. These cards could be used to produce printed output offline using an [IBM](#) accounting machine, an example of which would be the [IBM 405](#).
- ▶ ENIAC used ten-position [ring counters](#) to store digits; each digit used 36 vacuum tubes, 10 of which were the dual triodes making up the [flip-flops](#) of the ring counter. Arithmetic was performed by "counting" pulses with the ring counters and generating carry pulses if the counter "wrapped around", the idea being to emulate in electronics the operation of the digit wheels of a mechanical [adding machine](#). ENIAC had twenty ten-digit signed [accumulators](#) which used [ten's complement](#) representation and could perform 5,000 simple addition or subtraction operations between any of them and a source (e.g., another accumulator, or a constant transmitter) every second. It was possible to connect several accumulators to run simultaneously, so the peak speed of operation was potentially much higher due to parallel operation.

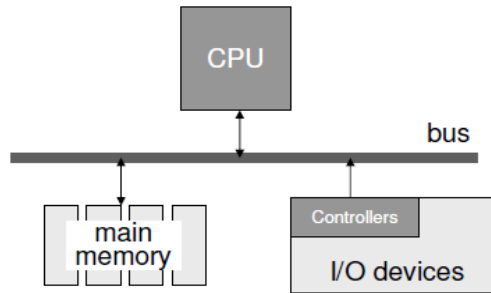
## Giriş

- ❖ Bilgisayarlar « **Bilgi işleyen makinalar** » verilen giriş değerlerini (veriler, bilgiler) istenen işlemlerle (kayıt tutma, hesaplama, karşılaştırma, karar verme, kontrol etme vs. ) işleyerek sonuçları çıkış birimlerinden kullanıcıya gösteren elektronik sistemlerdir.
- ❖ Burada verilerin ya da bilgilerin insanlar tarafından bilgisayara iletilmesi bir takım programlar (**komutlar**) ile birlikte yapılır.
- ❖ İnsanla bilgisayar arasındaki iletişim aracı olan **«program»** bilgisayara iletilen komutlar dizisidir.
- ❖ Bu amaç doğrultusunda kullanılan her türlü kodlar bütününe yani programlara **<<yazılım>>** denir.

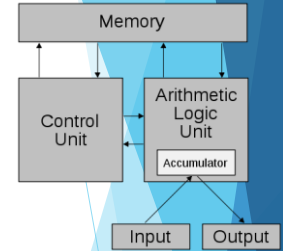


# Computer Architecture

- ❑ Central Processing Unit (CPU) contains
  - Arithmetic/Logic Unit (ALU)
  - Control Unit
  - Registers
  - Cache Memory
- ❑ Bus
- ❑ Main Memory
- ❑ I/O devices



Von Neumann Mimarisi



## Programlama Dili Nedir ?

- Yazılımlar **programlama dilleri** kullanılarak oluşturulurlar. Programlama dilleri, **yazılımcının bilgisayara neyi nasıl yapacağını** hangi koşullarda hangi işlemlerin yapılacağını tam olarak anlatmasını sağlar.
- Her programlama dili diğerinden farklıdır. Her birinin **farklı komutları ve kuralları** mevcuttur. En iyi programlama dili budur **diye bir şey söylemeyiz**. Çünkü her programlama dilinin bir diğerine göre **üstün ve zayıf** tarafları vardır. Günümüzde **en yaygın** kullanılan programlama dilleri şunlardır: Java ,C ,C++ ,C# ,PHP ,Objective-C ,(Visual) Basic ,Python ,Perl ,JavaScript ,Lisp ,Ada


## Programlamaya Giriş

- ❖ Yazılan programlar kullanım amaçlarına göre birbirinden ayrılır. Genel olarak 3 gruba ayrılabilir.
- i. Bilgisayarın çalışmasını kontrol eden ve kullanıcı ile donanım arasındaki iletişimi sağlayan «**işletim sistemleri**»
- ii. Kullanıcının program yazmasını ve çeşitli uygulamalar geliştirmesini sağlayan «**programlama dilleri**»
- iii. Program geliştirme araçları kullanılarak yazılan «**paket programlar**»

## Programlama Dilleri

- ❖ Bilgisayar sayısal (dijital) bir sistemdir ve «makine dili» olarak adlandırılan 0 ve 1'lerden oluşan 2'lik sayı sistemine göre çalışır.
- ❖ 0 ve 1 bilgisayardaki en küçük depolama birimi olup her birine bir bit denir.
- ❖ Sayısal elektronikte 0 gerilim/akım yok, 1 gerilim/akım var anlamına gelir.

- ✓ 1 byte=8 bit
- ✓ 1 KB=1024 byte
- ✓ 1 MB=1024 KB
- ✓ 1 GB=1024 MB
- ✓ 1 TB= 1024 GB
- ✓ 1 PB=1024 TB
- ✓ 1 EB=1024 PB
- ✓ 1 ZB=1024 EB
- ✓ 1 YB=1024 ZB


 KB:Kilobyte  
 MB:Megabyte  
 GB:Gigabyte  
 TB:Terabyte  
 PB: Petabyte  
 EB: Exabyte  
 ZB: Zettabyte  
 YB: Yettabyte

## Bit - Byte Kavramları

### ► Byte = 8 bit

- Binary  $00000000_2$  -  $11111111_2$
- Decimal:  $0_{10}$  -  $255_{10}$
- Hexadecimal :  $00_{16}$  -  $FF_{16}$

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

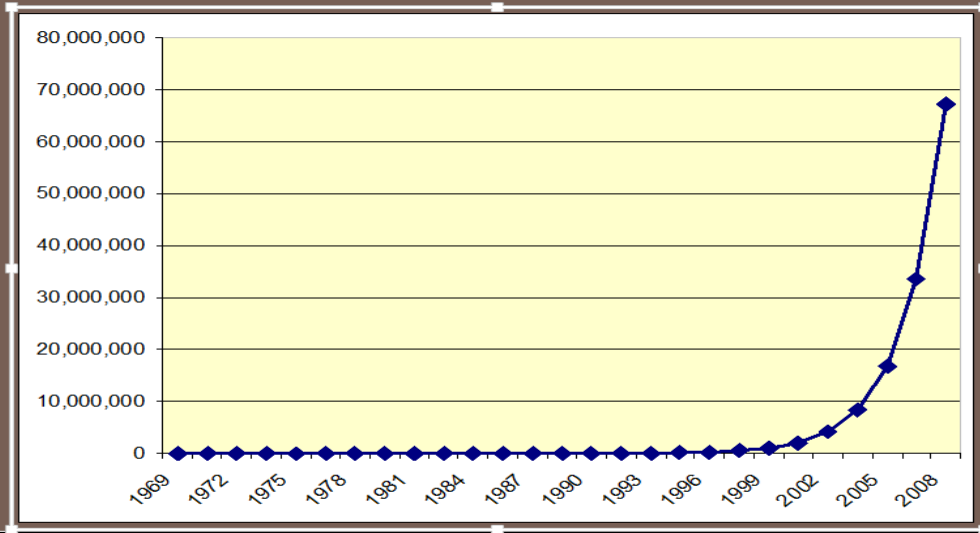
## Apollo yönlendirme bilgisayarı, 1969

- Apollo Computer: 30720 bit hafıza
- Lab bilgisayarları 1 GB (RAM)
  - 1 Gigabyte = 1024 Megabyte,
  - 1 Megabyte = 1024 Kilobyte,
  - 1 Kilobyte = 1024 Byte,
  - 1 Byte = 8 bits
- > ( $* 1024 1024 1024 8$ )
- 8589934592** ~ 8.6 Milyar bit
- > (yuvarla/ ( $* 1024 1024 1024 8$ ) 30720))
- 279620**



Apollo bilgisayarından 105 404 kat daha güçlü bilgisayarlarımız var

# HESAPLAMA GÜCÜ 1969-2008



**Moore “Kanunu”:** Hesaplama gücü her 18 ayda bir yaklaşık 2 katına çıkar!

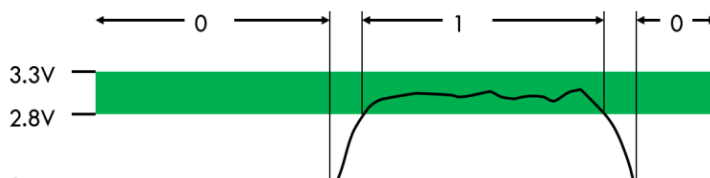
## Neden 10 tabanlı sistem değil

- 10 tabanlı sistem
  - ▣ Her bir parmak bir sayıyı temsil eder
  - ▣ Finansal işlemler için doğal temsil yöntemi
  - ▣ Bilimsel notasyonda da kullanılır
    - $1.5213 \times 10^4$
- Elektronik olarak gerçekleştirmek
  - ▣ Depolaması zor
    - ENIAC (ilk elektronik bilgisayar) her rakam için 10 vakum tüp kullanıyordu
  - ▣ İletimi zor
    - Tek bir kablo üzerinde 10 farklı sinyal seviyesi kodlamak için yüksek hassasiyet gerekir
  - ▣ Dijital mantık işlemlerini gerçekleştirmek zor
    - Toplama, çarpma, vs.



## Neden İkili Sayı Sistemi

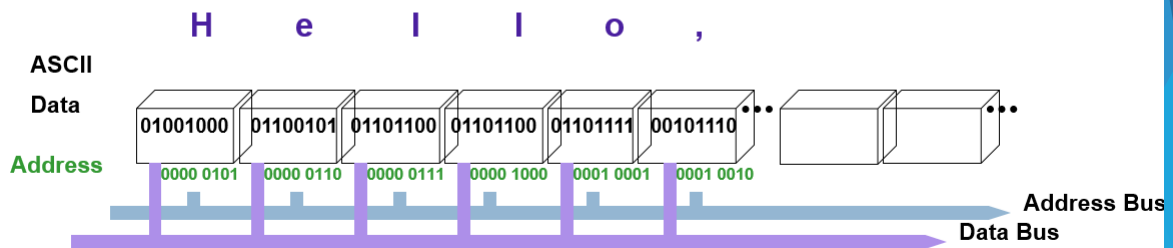
- 2 tabanlı sayılar
  - ▣  $15213_{10} \rightarrow 11101101101101_2$
  - ▣  $1.20_{10} \rightarrow 1.0011001100110011[0011]..._2$
  - ▣  $1.5213 \times 10^4$  as  $1.1101101101101_2 \times 2^{13}$
- Elektronik olarak
  - ▣ İki durumlu (bistable) elemanlarla depolanması kolay
  - ▣ Gürültülü ve güvensiz kablolarda güvenli taşınabilir



## Hafıza Organizasyonu



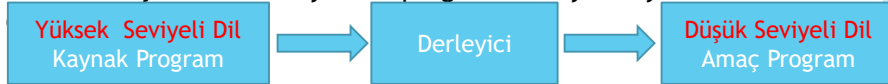
1 byte alan hafıza hücresi



## Programlama Dilleri

- ❖ Programlama dilleri 5 gruba ayrılır
  - i. Makine dili: Bilgisayarın çalışma dili 1 ve 0'lerden oluşur.
  - ii. Düşük seviyeli diller: Assembly gibi makine dilini içerirler
  - iii. Orta seviyeli diller: donanımdan bağımsızdır, Örneğin: C, C++, Java, Fortran, Pascal, Basic vs..
  - iv. Yüksek seviyeli diller:, vs. Delphi, Visual Basic, VB Net, sql, MATLAB
  - v. Çok yüksek seviyeli diller: Yapay zeka alanında kullanılan Prolog, OPS5, Mercury vs.

- ✓ Yüksek seviyeli bir dil ile yazılan program derleyici sayesinde makine



## I- Düşük Seviyeli Diller Makine dili; geliştirilen ilk programlama dilidir

- Bu dilde yazılan tüm komutlar 0 ve 1 lerden oluşur. Belirli bir işlemci/makine için yazılan kod, farklı yapıdaki başka bir makinede çalışmaz, tamamen yeniden yazılması gerekir. Örneğin 1011101100010001 farklı yapıdaki işlemcilerde farklı işlemleri yapar. Bu yüzden makine dili donanıma bağlıdır. Makine dili biraz daha kolay okunabilmesi için 16'lık sayı sistemi ile yazılır fakat derlenme sonrası kod ikilik sisteme çevrilir. İşlemci ikilik tabandaki kodu okur ve uygular.
- Makine dili örneği

Bu program ekrana "Hello world" yazısını yazar.

```

1011101100010001 0000000110111001 0000110100000000 1011010000001110
1000101000000111 0100001111001101 0001000011100010 1111100111001101
0010000001001000 011001010101100 0110110001101111 0010110000100000
0101011101101111 0111001001101100 0110010000100001
  
```

## II-Düşük Seviyeli Diller Assembly

- ▶ Assembly programlama dili örneği
- ▶ Bu program ekrana "merhaba dünya" yazısını yazar.

format mz

org 100h

mov ah,09

mov dx,yazi

int 21h

mov ah,00

int 16h

int 20h

yazi db "merhaba dünya\$"

### Examples of "Programs"

□ Task: compute the sum of one, two, and three.

- Program 1 (written in the English language):

```
Compute the sum of one, two, and three.
```

- Program 2 (written in the C language):

```
for (idx = 1, sum = 0; idx <= 3; idx++) sum += idx;
```

- Program 3 (written in the x86 assembly language):

```
mov ax, 1
mov bx, 2
add ax, bx
inc bx
add ax, bx
```

Hexadecimal hali ise

Bu program ekrana "Hello world" yazısını yazar.

```
BB 11 01 B9 0D 00 B4 0E 8A 07
43 CD 10 E2 F9 CD 20 48 65 6C
6C 6F 2C 20 57 6F 72 6C 64 21
```

Makine diline oranla daha anlaşılırdır. (Human Readable)

- BAL (Basic Assembler)
- COMPASS (COMPhensive ASSEMBler)
- TASM (Turbo Assembler, Borland)

## III- Orta seviyeli diller (Üçüncü Nesil)

- ▶ İnsanların kullandığı dillere yakınlaşmaya başlamış ve makine diline bağımlılıktan uzaklaşmıştır.
- ▶ Üçüncü nesil (orta seviye) dillerin yüklendikleri bilgisayarda çalışması için
  - ▶ DERLEYİCİ (compiler)
  - ▶ YORUMLAYICI (interpreter)
 İhtiyaç duyulur.
- ▶ C, C++, Java, Fortran, Pascal, Basic vb.

## IV-Yüksek Seviye Diller (Dördüncü Nesi)

- ▶ Ticaret ve iş yaşamındaki ihtiyaçlara yönelik daha hızlı çözümler üretebilmek için dördüncü nesil (yüksek seviye) diller ortaya çıkmıştır.
  - ▶ Kullanımı kolay
  - ▶ Daha az kod
  - ▶ Hazır Şablonlar ve Sihirbazlar
  - ▶ Rapor üreticiler
  - ▶ İstatistiksel Analizler
- ▶ MATLAB, SQL, Oracle Forms/Reports, .NET platformları

## V-ÇOK YÜKSEK SEVİYE (BEŞİNCİ NESİL)

- ▶ Koşulları ve kısıtları bilgisayara verdiğimizde, bilgisayarın çözümü bulmasına yönelik tasarlanmaktadır.
- ▶ Kodlamanının yerine
- ▶ imperative(kısıtlar)
- ▶ Declarative(bildirimsel)
- ▶ YAPAY ZEKA alanındaki araştırmalar için kullanılır.
- ▶ PROLOG, OPS5, Mercury
- ▶ Örn; Hepsiburada müşteri asistanı vs.

## Programlama Hataları (Yanlışları)

- ❖ Programlar yazılırken genelde 3 tür hata yapılır:
- ❖ **Yazım (Sözdizim) Hatası (Syntax hatası):** Programın derlenmesi sırasında ortaya çıkan kullanıcıdan kaynaklanan yazım (grammer) hatalarıdır. Bunlar düzeltilmeden program çalışmaz. DERLEYİCİ sayesinde bu hatalar programcıya iletilir.
- ❖ **Mantık Hatası (Logical Error):** Mantıksal olarak yanlış işlem yapıldığında yazım hatası yoksa program çalışacaktır fakat istenen doğru değeri üretmeyecektir.
- ❖ **Çalıştırma Zamanı Hataları (Run Time Error):** Bir sayıyı sifıra bölme, ya da boş parametre değerleri



```

1 import lotus.domino.*;
2
3 public class JavaAgent extends AgentBase {
4
5     public void NotesMain() {
6
7         try {
8             Session session = getSession();
9             AgentContext agentContext = session.getAgentContext();
10
11             Map<String, String> test;
12
13             // (Your code goes here)
14
15         } catch (Exception e) {
16             e.printStackTrace();
17         }
18     }
19 }
20
21

```

## Programlama İşlemleri

- ❖ Bilgisayardaki işlemler temel olarak 3 gruba ayrılır.
- I. **Matematiksel (aritmetik) işlemler:** toplama , çıkarma, çarpma, bölme, üs alma vs. işlemler. Örneğin:  $Y=A*B/C$
- II. **Karşılaştırma (karar) işlemleri:** Büyüktür, küçüktür, eşittir, eşit değildir vs. Örneğin:  $A>B$  ise.. ,  $A=B$  ise...,  $A<>B$  ise...
- III. **Mantıksal (lojik) işlemler:** VE, VEYA, DEĞİL Mantıksal işlem operatörleri hem karar ifadelerinde hem de matematiksel işlemlerde kullanılır.

Karşılaştırma ve matematiksel işlemlerde birden fazla koşulun belirli bir düzeyde (Her durumda, durumlardan sadece birinde ya da hepsinde, ya da hiçbirinde gibi..) sağlanması istendiğinde kullanılırlar.

- ✓ VE: AND
- ✓ VEYA: OR
- ✓ DEĞİL: NOT

# Matematiksel İşlemler

## Matematiksel İşlemlerin Bilgisayar Dilindeki Karşılıkları

İşlem	Matematik	Bilgisayar
Toplama	$a + b$	$a + b$
Çıkarma	$a - b$	$a - b$
Çarpma	$a \cdot b$	$a * b$
Bölme	$a \div b$	$a / b$
Üs Alma	$a^b$	$a ^ b$

## ➤ Matematiksel İşlemlerde Öncelik Sıraları

Sıra	İşlem	Bilgisayar
1	Sayıların Negatifliği	$-...$
2	Parantezler	$( ..... )$
3	Matematiksel Fonksiyonlar	$\cos, \sin, \log, ...$
4	Üs alma	$a ^ b, \text{pow}, ...$
5	Çarpma ve Bölme	$a * b$ ve $a/b$
6	Toplama ve Çıkarma	$a + b$ ve $a - b$

# Matematiksel İşlemler

## ➤ Örnek:

### ❑ Matematiksel ifade :

$$x = a \cdot b / c + d \cdot e^f - g$$

### ❑ Bilgisayar ifadesi:

$$x = a * b / c + d * e ^ f - g$$

- **Not:** İşlem önceliklerine dikkat edilmez ise aynı gibi görünen ifadeler, farklı sonuçlar verebilir.

## KARŞILAŞTIRMA İŞLEMLERİ

### ➤ Karşılaştırma İşlemlerinin Bilgisayar Dilindeki Karşılıkları

Sembol	Anlamı
= veya ==	Eşittir
<> Veya !=	Eşit Değildir
>	Büyüktür
<	Küçüktür
>= veya =>	Büyük eşittir
<= veya =<	Küçük eşittir

### ➤ Örnek:

Eğer **A > B** ise Yaz “ A sayısı B’den büyüktür ”

## KARŞILAŞTIRMA İŞLEMLERİ

### ➤ Sayısal karşılaştırmalarda doğrudan değerler karşılaştırılır.

➤ Örnek:  $25 > 15$  “25 sayısı 15 ten büyüktür”

### ➤ Karakter olarak karşılaştırmalarda ise karşılaştırma işlemine ilk karakterlerden başlanılarak sıra ile karşılaştırılır.

➤ Örnek:  $a > c$  “1. karakter alfabetik olarak daha önde”

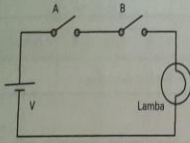
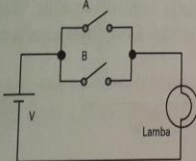
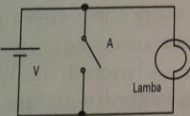
➤ **Not:** Karakter karşılaştırma işlemlerinde, karşılaştırma karakterler arasında değil, karakterlerin ASCII kodları arasında yapılır. **Örneğin,** **A** nın ASCII kod karşılığı 65 tir. **a** nın ise ASCII kod karşılığı 97 tir. Büyük ve küçük harfler arasındaki ASCII kod farkı ise 32’dir.

**Araştırınız !! ASCII nedir?**  
(American Standard Code  
for Information Interchange)

# MANTIKSAL İŞLEMLER

## ➤ Temel Mantıksal İşlem Karşılıkları

İşlem	Komut	Matematiksel Sembol	Anlamı
VE	AND	.	Koşulların <b>hepsi</b> doğru ise sonuç doğrudur
VEYA	OR	+	Koşullardan <b>en az biri</b> doğru ise sonuç doğrudur
DEĞİL	NOT	'	Sonuç koşulun <b>tersidir</b> . 1 ise 0 dır

İşlem	Elektriksel eşdeğeri	Doğruluk tablosu															
VE (AND)		<table><tr><th>A</th><th>B</th><th>A VE B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	A VE B	0	0	0	0	1	0	1	0	0	1	1	1
A	B	A VE B															
0	0	0															
0	1	0															
1	0	0															
1	1	1															
VEYA (OR)		<table><tr><th>A</th><th>B</th><th>A VEYA B</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	A VEYA B	0	0	0	0	1	1	1	0	1	1	1	1
A	B	A VEYA B															
0	0	0															
0	1	1															
1	0	1															
1	1	1															
DEĞİL (NOT)		<table><tr><th>A</th><th>DEĞİL A</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	DEĞİL A	0	1	1	0									
A	DEĞİL A																
0	1																
1	0																

## ➤ Mantıksal İşlemlerde Öncelik Sıraları

Sıra	İşlem	Komut
1	Parantez içindeki işlemler	( ..... )
2	DEĞİL	NOT
3	VE	AND
4	VEYA	OR



## Temel Kavramlar

### ➤ Program

- ❑ Basit olarak bilgisayar ile kullanıcı arasındaki iletişim aracıdır.
- ❑ Giriş değerlerini kullanarak, çıkış değerlerinin elde edilmesi için bilgisayara iletilen komutlar dizisidir.
- ❑ Belirli bir görevi yerine getiren **algoritmik** ifadedir.

### ➤ Yazılım

- ❑ Bir yada birden fazla programın bir araya gelmesinden oluşur.

### ➤ Programlama Dilleri

- ❑ Programcı ile bilgisayar arasındaki iletişimi sağlayan bir araçtır (komut kümesidir).

## ALGORİTMA

- ❖ Bilgisayardaki işlemlerin gerçekleştirilmesinde izlenecek adımlara «**algoritma**» denir. Yani algoritma işlemleri yaptırabilmek ( problem çözümü, kontrol, karşılaştırma vs. ) için bilgisayara iletilen işlem basamaklarıdır.

- ✓ Bilgisayarın hangi ve neredeki giriş değerlerini alacağı
- ✓ Giriş değerlerini işlerken ne tür yöntemler kullanacağı
- ✓ Ne tür sonuçlar üreteceği
- ✓ Sonuçların nerede gösterileceği ya da saklanacağı

Gibi adımların hepsi hazırlanan algoritmanın herhangi bir programla dilinin kurallarına uygun olarak yazılmış komutlarla bilgisayara iletilir.

Algoritmanın özel geometrik şekillerle çizilmiş haline «**akış diyagramı**» denir.

## İyi Bir Algoritmada Neler Olmalıdır?

- Etkinlik
  - ❑ Gereksiz tekrarlarla bulunmayan diğer algoritmalar içerisinde de kullanılabilir olmalıdır.
- Sonluluk
  - ❑ Her algoritmanın bir başlangıçtan oluşmalı, belirli işlem adımı içermeli ve bir bitiş noktasına sahip olmalıdır. Kısır bir döngüye girmemelidir.
- Kesinlik
  - ❑ İşlem sonucu kesin olmalı, her yeni çalıştırmada aynı sonucu üretmelidir.
- Giriş/Çıkış
  - ❑ Algoritma giriş (üzerinde işlem yapılacak değerler) ve çıkış (yapılan işlemler neticesinde üretilen sonuç değerler) değerlerine sahip olmalıdır.
- Başarım/Performans
  - ❑ Amaç donanım gereksinimi (bellek kullanımı gibi), çalışma süresi gibi performans kriterlerini dikkate alarak yüksek başarımlı programlar yazmak olmalıdır.

## Algoritmalar Kaç Farklı Şekilde İfade Edilir?

- ① Algoritma nın **metin** olarak yazılması
  - ❑ Çözülecek problem, adım adım metin olarak yazılır.
- ② Pseudo Code (Kaba Kod)
  - ❑ Algoritmanın sözde kodlarla yazılmasıdır.
  - ❑ Problemin çözüm adımları komut benzeri anlaşılır metinlerle ifade edilir.
  - ❑ Yarı kod yarı metin olarak ta adlandırılır.
- ③ Akış Diyagramı
  - ❑ Problemin çözüm adımları geometrik şekiller ile ifade edilir.

## AKIŞ DİYAGRAMINDA Kullanılan Genel Şekiller

Başla/Dur	Algoritmanın başlangıcını ve bitişini gösterir
Bilgi Girişi	Kullanıcıdan bilgi alınacağı zaman kullanılır. (Veri/Bilgi Girişi)
İşlem	Aritmatiksel, mantıksal vb. işlemler gösterilir
Döngü	Tekrarlı işlemleri ifade etmek için kullanılır
Karar	Belirli bir koşula göre algoritmanın dallanmasını sağlamak için kullanılır
Bilgi Çıkışı	Kullanıcıya bilgi gösterileceği zaman kullanılır
Alt program	Önceden tanımlı işlem ya da fonksiyon (Alt prog.)
Bağ	Aynı sayfaya sığmayan algoritmanın devamını göstermek için kullanılır.
	İşlem Akış Yönü

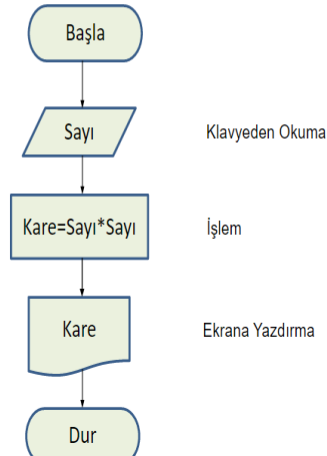
## Örnek Algoritma

➤ **Problem:** Klavyeden girilen sayının karesini hesaplayarak ekrana yazdıran programın algoritmasını yazınız?

- 1 Başla
- 2 Sayıyı (A) gir
- 3 Sayının karesini hesapla ( $Kare = A * A$  işlemini yap)
- 4 Sonucu (Kare) yaz
- 5 Dur

## Örnek Akış Diyagramı

➤ **Problem:** Klavyeden girilen sayının karesini hesaplayarak ekrana yazdıran programın akış diyagramını çiziniz?



## Pseudo Kodlar Örnekler (Sözde Kodlar)

- OkumaYazma İşlemleri: READ, GET, WRITE, DISPLAY gibi komutlar kullanılır.
- Karar Yapıları ifade edilirken IF-THEN-ELSE-ENDIF kullanılır.
- Döngüler ise LOOP-ENDLOOP olarak ifade edilir.

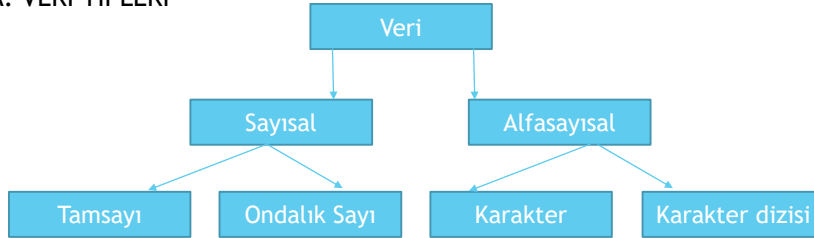
Satır Algoritmada	Pseudo Kod
1. Başla 2. Yaz 1 3. Yaz 2 4. Yaz 3 5. Dur	DISPLAY 1 DISPLAY 2 DISPLAY 3  Pseudo kodda başla-dur yazılmıyor
1. Başla 2. Oku(K) 3. $C = K * 4$ 4. Yaz C 5. Dur	GET (K) $C = K * 4$ DISPLAY C
1. Başla 2. Oku "Yaşınız ?" (Y) 3. Eğer 4. ( $Y \geq 18$ ) yaz "Reşitsiniz" 5. (Değilse) yaz "Değilsiniz" 6. Dur	DISPLAY "Yaşınız?" GET Y IF $Y \geq 18$ THEN DISPLAY " Reşitsiniz" ELSE DISPLAY "deĞİLSİİNİZ" ENDIF

### Program Yazma Adımları

- I. Problem/iş iyice irdelenir. (Analiz edilir)
- II. Çözüm yolları ortaya konularak programlamaya en uygun (en az komutla, en kısa sürede, en doğru sonuç veren) çözüm yolu belirlenir.
- III. Programın algoritması hazırlanır veya akış şeması çizilir.
- IV. Algoritma ve ya akış diyagramı bir programlama diliyle kodlanır.
- V. Program kullanıldığı editörde çalıştırılarak yazım hataları düzeltilir.
- VI. Bilinen giriş/çıkış değerleri ile programın doğru çalışıp çalışmadığı test edilir (doğrulama yapılır)

## Algoritmada Kullanılan Terimler

### A. VERİ TİPLERİ



SAYISAL VERİ/BİLGİ	ALFASAYISAL VERİ/BİLGİ
2004	BURSA
-555	a
6.78	;
1999	"1999"

## Algoritmada Kullanılan Terimler

### ► B.

#### ► Tanımlayıcı

- ❑ Programcı tarafından oluşturulan/verilen ve programdaki değişkenleri, sabitleri, paragrafları, kayıt alanlarını, özel bilgi tiplerini, alt programları vb. adlandırmak için kullanılan kelimelerdir.
- ❑ Tanımlayıcı, yerini tutacağı ifadeye çağrışım yapmalıdır.
- ❑ **Tanımlayıcı kelimeler oluşturulurken uyulması gereken kurallar**
  - ❶ İngiliz alfabesindeki A-Z veya a-z arası 26 harf kullanılabilir
  - ❷ 0-9 arası rakamlar kullanılabilir
  - ❸ Simgelerden sadece alt çizgi ( \_ ) kullanılabilir
  - ❹ Tanımlayıcı isimleri, harf veya alt çizgi ile başlayabilir
  - ❺ Rakam ile başlayamaz veya sadece rakamlardan oluşamaz
  - ❻ Kullanılan programlama dilinin komutu ya da saklı kelimelerinden olamaz

## Algoritmada Kullanılan Terimler

### ► C. DEĞİŞKEN

- ❑ Verilerin saklandığı bellek alanlarına verilen simgesel isimlerdir.
- ❑ Programın her çalıştırılmasında, farklı değerler alabilen/aktarılabilen bilgi/bellek alanlarıdır.
- ❑ Bir programda birbirinden farklı kaç tane veri tutulacak ise o kadar değişken tanımlanmalıdır.
- ❑ Değişkenleri adlandırma, tamamen programcının isteğine bağlıdır.
- ❑ Değişken adlandırmada tanımlayıcı isimlendirme kuralları geçerlidir.
- ❑ Değişken adının, yerini aldığı ifadeyi çağrışım yapması programın anlaşılabilirliği açısından önemlidir.
- ❑ **Örnek :** Bir kişinin adını tutmak için **Ad** , telefonunu tutmak için **Tel**

Değişken isimlerinde harfler, rakamlar ve alt çizgi kullanılabilir.  
Örneğin; Ad\_Soyad, Adres, C=A+B (A, B ve C birer değişkendir)

## Algoritmada Kullanılan Terimler

### ► D. SABİT

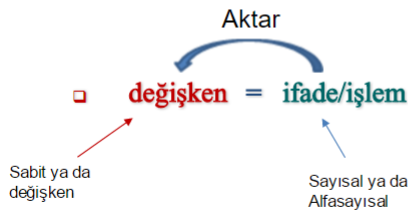
- ❑ Programlardaki değeri değişmeyen ifadelere sabit denir.
- ❑ Sabit adlandırmalarında da tanımlayıcı kuralları geçerlidir.
- ❑ Sabitlere değer aktarma
  - ❑ Sayısal veriler doğrudan aktarılır.
    - ❑ **Örnek:** SabitKomutu  $\pi = 3.14$
  - ❑ Alfayısal (karakter) veriler ise tek/çift tırnak içerisinde aktarılır.
    - ❑ **Örnek:** SabitKomutu ilkharf = 'A'
    - ❑ **Örnek:** SabitKomutu OkulAdi = "Sakarya "

## Algoritmada Kullanılan Terimler

### ► E. AKTARMA

#### ➤ Aktarma

- ❑ Bir bilgi alanına, veri yazma;
- ❑ Bir ifadenin sonucunu başka bir değişkende gösterme vb. görevlerde “aktarma =” operatörü kullanılır.
- ❑ Kullanım şekli;



#### Örnek:

- Matematiksel ifade olarak
  - ❑  $A=2, B=3$
  - ❑  $C=A+B$        $C=5$
- Alfaisayısal ifade olarak
  - ❑  $A=\text{"Sak"}, B=\text{"arya"}$
  - ❑  $C=A+B$        $C=\text{Sakarya}$

## Algoritmada Kullanılan Terimler

### F. SAYAÇ YA DA SAYICI

- ❑ Belirli sayıda yapılması istenen işlemleri takip etmek için;
- ❑ İşlenen ya da üretilen değerlerin sayılması gerektiği durumlarda kullanılır.
- ❑ **Örnek:** Rasgele oluşturulmuş bir dizideki tek sayıların tespitinde “tek sayıların adedini belirlemek için sayaç kullanılır.
- ❑ Kullanım şekli;



## Algoritmada Kullanılan Terimler

### F. SAYAÇ DEVAMI

Bazı işlemlerin belirli sayıda yapılması veya işlenen/üretilen değerlerin sayılması gerekebilir.

Sayaç değişkeni=sayaç değişkeni + adım

Örneğin;  $X=X+3$  olsun  $Y=Y-5$

Örneğin;

$S=0$  olsun

$S>4$  oluncaya kadar  $S=S+1$  olsun

Eski S	Yeni S	Ekran Çıktısı
0	$0+1=1$	1
1	$1+1=2$	2
2	$2+1=3$	3
3	$3+1=4$	4
4	$4+1=5$	5

## Algoritmada Kullanılan Terimler

### G. DÖNGÜ

- ❑ Bazı işlemleri belirli sayıda tekrar etmede,
- ❑ Belirli bir aralıktaki **ardışık değerler** ile işlem yapmada döngü kullanılır.
- ❑ Diğer bir deyişle, programdaki belirli işlem bloklarını, verilen sayıda gerçekleştiren işlem akış çevrimlerine **DÖNGÜ** denir.
- ❑ **Örnek:**
  - ❑ 1'den 5'e kadar sayıların toplamı
  - ❑ Ard arda ekrana 4 defa SAKARYA yazdırma
  - ❑ 1 ile 100 arasındaki çift sayıların ya da tek sayıların toplamı



## SIK KULLANILAN ARDIŞIK DÖNGÜ YAPILARI

### ➤ Ardışık Toplama

$$\square \text{ Toplam değişkeni} = \text{Toplam değişkeni} + \text{Sayı}$$

### ➤ Ardışık Çarpma

$$\square \text{ Çarpım değişkeni} = \text{Çarpım değişkeni} * \text{Sayı}$$

## Algoritmada Kullanılan Terimler

### G. Döngü

Programlardaki belirli işlem bloklarını aynı veya farklı değerlerle, verilen sayıda gerçekleştiren çevrim yapılarına «döngü» denir.

Örneğin; 1 ile 1000 arasındaki tek sayıların toplamının hesaplanmasında  $T=1+3+5+7+\dots$  gibi uzun uzun yazılamayacağından 1 ile 1000 arasında ikişer ikişer artan bir döngü açılır ve döngü değişkeni ardışık olarak toplanır.

Örneğin;

$T=0$  ve  $J=1$  olsun

$J>10$  oluncaya kadar

$T=T+J$

$J=J+2$

olsun

Eski J	Eski T	Yeni T	Yeni J
1	0	$0+1=1$	3
3	1	$1+3=4$	5
5	4	$4+5=9$	7
7	9	$9+7=16$	9
9	16	$16+9=25$	11

## Algoritmada Kullanılan Terimler

### ➤ Operatör

- ❑ İşlemleri belirten yani veriler üzerinde işlem özelliği olan simgelerdir.
- ❑ **Örnek:** + = >

## Algoritma Hazırlama Kuralları

- I. Problem iyice irdelenir tüm olasılıklar gözden geçirilir.(Analiz edilir)
- II. Çözüm yolları ortaya konularak programlamaya en uygun (en az komutla, en kısa sürede, en doğru sonuç veren) çözüm yolu belirlenir.
- III. Tanımlayıcı isimleri belirlenir.
- IV. Algoritmada her işlem adımına bir numara verilir.
- V. Problem çözümü için gerekli olan veriler girilir ve ya başka bir ortamdan alınır.
- VI. Yapılacak işlem/kullanılacak yöntemler açık bir şekilde verilir
- VII. Bulunan sonuçlar görüntülenir ve ya başka bir ortamda saklanır.

### Algoritma ve Akış Diyagramının Avantajları

- I. Program yazmayı kolaylaştırır.
- II. Hatalı kodlama oranını azaltır.
- III. Program yazımı için geçen süreyi kısaltır.
- IV. İşlem akışını açık bir şekilde gösterdiğinden program kontrolünü ve hata takibini kolaylaştırır.
- V. Sonradan yapılacak düzenlemelerde kolaylık sağlar.
- VI. Kullanılan ortak yapı sebebiyle birden fazla kişinin aynı programa müdahale etmesini kolaylaştırır.