

# BİLGİSAYAR BİLİMLERİNE GİRİŞ I

-3-

# Bilgisayarlar ve Programlamaya Giriş

- Verilerin Bilgisayarda Nasıl Saklandığını Anlattık. O Zaman Programlar Nasıl Çalışır?
  - Bilgisayar CPU'ları sadece makine dilinde yazılmış talimatları anlar. İnsanlar bu şekilde bir program yazmayı zor bulduğu için diğer programlama dilleri gelişmiştir.
  - Bazen CPU'lar bilgisayarın beyini olarak ifade edilir ki buda CPU'ların zeki olduğu izlenimi uyandırır. Bunun bir metafor olduğunu aklınızdan çıkarmayın. Aslında CPU'lar zeki değildir, yalnızca belirli spesifik şeyleri yapabilen elektronik cihazlardır. Örneğin;
    - Bellekten bir veri parçasını okuma
    - İki sayıyı toplama
    - Bir veri bellekte başka bir lokasyona gönderme
    - vb...
  - Göreceğiniz gibi CPU'lar yalnızca belirli basit işlemleri veri parçaları üzerinde gerçekleyebilirler.
  - Öyleyse, bir programın nasıl çalıştığını kavrayabilmek için programlama dillerinin tarihine bir göz atalım.

# Bilgisayarlar ve Programlamaya Giriş

3

- Henüz Babbage'ın analitik makinesi proje halindeyken ilk programcımız Ada Lovelace bu makinenin verilen komutları anlayabilmesi gerektiğini öngörmüştür. (Analytical Engine Order Code, 1837)
- Zuse'nin Plankalkül'ü (1945) *makine kodu* cinsinden gerçek bir programlama dili idi.
- ENIAC ve diğer 1. nesil bilgisayarlar da makine kodlarıyla programlanmaktaydı. (C10 dili)
- Makine koduyla programlama yapabilmek için çalışılan makinenin çok iyi tanınması gerekiyordu.
- Makine kodu kullanılarak yazılmış bir program aslında bir *ikili sayılar dizisidir*.
  - Toplama işleminin makine kodu : 100110
- Makine dilinde programlama yükünü azaltan Assembly (1947) dili de düşük seviye (low level) bir programlama dilidir ve makinenin iyi tanınmasını gerektirir. Yine de makineye direkt müdahale edebildiği için ve hızlı çalışması nedeniyle birçok uygulamada kullanılmaktadır.
- Bilgisayarın Assembly'yi anlaması için Assembler isimli bir çeviriciden geçerek makine diline dönüştürülmesi gerekir.
  - Assembly'de toplama işleminin kodu: ADD
- 2. nesil bilgisayarların gelişimiyle G.M. Hopper tarafından kısa kod temelinde A0 programlama dili oluşturuldu (1952). Ancak bu dilin ömrü çok kısa oldu ve hızla gelişen bilgisayarların nimetlerinden yararlanmak üzere yüksek seviye (high level) diller ortaya çıkmaya başladı.

# Bilgisayarlar ve Programlamaya Giriş

- IBM 1954-1956 yılları arasında makineden bağımsız (yani her makine tarafından anlaşılabilir!) olan FORTRAN-0 (Formula Translator) dilinin deneme sürümlerini yaptı. Nihayet 1957'de FORTRAN-1 uygulamalarda kullanılmaya başlandı.
- Hemen ardından 1958'de bilgisayarları daha da akıllı yapma hevesiyle, mantıksal bir dil olan LISP (List Processing) ortaya çıktı.
- 1960'da birçok güncel programlama dilinin atası olan ve önceki dillere göre çok daha algoritmik yapıdaki ALGOL dili oluşturuldu.
- Diğer koldan ise FORTRAN-2 ve sonra da 1964'te PL/1 (Programming Language) ile Basic (Beginner's All Purpose Symbolic Instruction Code) dilleri IBM tarafından geliştirildi.
- Basic dili, ismi ile müsemma, çok kolay bir dil olduğundan hemen popülerleşmiş ve yaygın bir şekilde kullanılmaya başlamıştır.
- 1971 yılında Pascal (ismini *Blaise Pascal*'dan alır), 1972'de ise PROLOG (Programming in Logic) dillerinin oluşturulması algoritmik hesaplama ve yapay zeka çalışmalarına yeni bir kan kazandırdı.
- 1972 yılında çok daha önemli bir dil de ortaya çıktı. Bell Labs'ın programlama dili olan B'nin yetersiz olacağını düşünen (Ekim 2011'de vefat eden) *Dennis Ritchie* tarafından UNIX işletim sistemi için geliştirildi.

# Bilgisayarlar ve Programlamaya Giriş

5

- Programlama dilleri tarihinde büyük sıçrama yaratan C dili artık bir standart haline gelmiştir.
- C ailesinden diğer diller C++ (nesne tabanlı programlama, 1980), C-- (1997), C# (2001) dilleridir.
- Öte yandan C üzerine geliştirilen dillerin sayısı bunlardan çok daha fazladır. Başlıcaları:
  - AMPL, AWK, C shell, C++, C--, C#, Objective-C, BitC, D, Go, Java, JavaScript, Limbo, LPC, Perl, PHP, Pike, Processing, Seed7...
- Neden ismi C ?
  - Daha önceki dillerin isimlerinin A ve B olduğunu hatırlayın.
- 1995'de *James Gosling* tarafından Sun Microsystems'de geliştirilen Java, açık kaynaklı (open source), yalnızca bilgisayarlarda değil hemen hemen her platformda çalışabilen ve tam anlamıyla nesne tabanlı bir programlama dilidir. Gosling ve ekibinin en sevdiği kahvenin markası da 'Java'dır. (Bkz. Java logosu)
- Bir diğeri ise Java'nın hızla yayılışına seyirci kalmayan Microsoft'un ona rakip olarak geliştirdiği C# (C sharp) dilidir. (2001)
  - Bu iki nesne tabanlı yüksek seviye programlama dili günümüzün en popüler programlama dillerindendir.

# Bilgisayarlar ve Programlamaya Giriş

6

- Yakın zamanda bilgisayarların donanımsal olarak güçlenmesiyle görsel diller ağırlık kazanmaya başlamıştır.
- İnternetin gücünü kullanan web tabanlı diller (HTML, JavaScript, ASP vb.) de günümüzde oldukça gündemdedir.
- Veri tabanlarında sorgu yapma amaçlı ortaya çıkan ancak gitgide gelişerek programlama dili yapısı kazanmaya başlayan veri tabanı dillerinin (Clipper, LINQ, SQL vb.) sayısının 200'ün üzerinde olduğu bilinmektedir.

■ Ayrıca...

- Yüksek seviye dillerde toplama işleminin kodu:  
+
- Günümüz yüksek seviye dillerini, makine diline çevirmeye yarayan programlara derleyici (compiler) ve yorumlayıcı (interpreter) denir. Aralarında ki fark nedir?

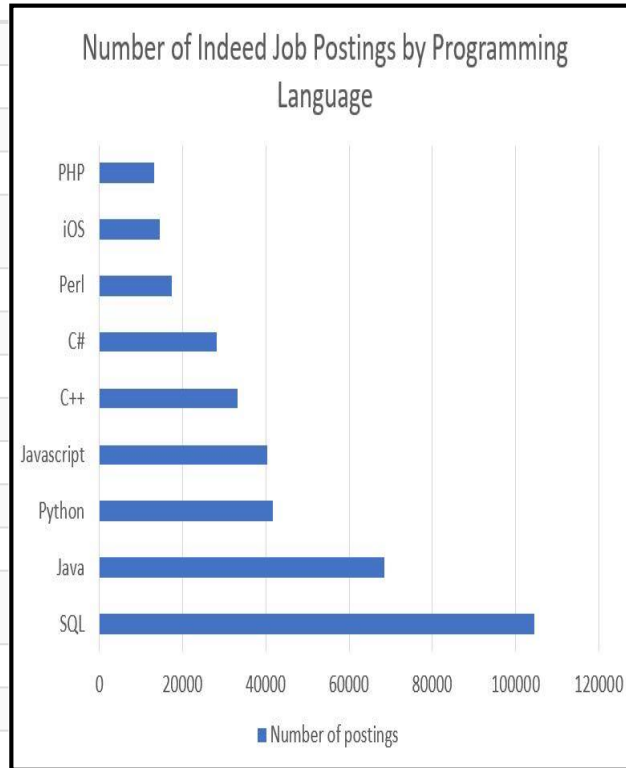
\*\*\* Biraz araştırma zamanı



# Bilgisayarlar ve Programlamaya Giriş

7

Oct 2018	Oct 2017	Programming Language	Ratings	Change
1	1	Java	17.801%	+5.37%
2	2	C	15.376%	+7.00%
3	3	C++	7.593%	+2.59%
4	5	Python	7.156%	+3.35%
5	8	Visual Basic .NET	5.884%	+3.15%
6	4	C#	3.485%	-0.37%
7	7	PHP	2.794%	+0.00%
8	6	JavaScript	2.280%	-0.73%
9	-	SQL	2.038%	+2.04%
10	16	Swift	1.500%	-0.17%
11	13	MATLAB	1.317%	-0.56%
12	20	Go	1.253%	-0.10%
13	9	Assembly language	1.245%	-1.13%
14	15	R	1.214%	-0.47%
15	17	Objective-C	1.202%	-0.31%
16	12	Perl	1.168%	-0.80%
17	11	Delphi/Object Pascal	1.154%	-1.03%
18	10	Ruby	1.108%	-1.22%
19	19	PL/SQL	0.779%	-0.63%
20	18	Visual Basic	0.652%	-0.77%



Jun 2020	Jun 2019	Change	Programming Language	Ratings	Change
1	2	▲	C	17.19%	+3.89%
2	1	▼	Java	16.10%	+1.10%
3	3		Python	8.36%	-0.16%
4	4		C++	5.95%	-1.43%
5	6	▲	C#	4.73%	+0.24%
6	5	▼	Visual Basic	4.69%	+0.07%
7	7		JavaScript	2.27%	-0.44%
8	8		PHP	2.26%	-0.30%
9	22	▲	R	2.19%	+1.27%
10	9	▼	SQL	1.73%	-0.50%
11	11		Swift	1.46%	+0.04%
12	15	▲	Go	1.02%	-0.24%
13	13		Ruby	0.98%	-0.41%
14	10	▼	Assembly language	0.97%	-0.51%
15	18	▲	MATLAB	0.90%	-0.18%
16	16		Perl	0.82%	-0.36%
17	20	▲	PL/SQL	0.74%	-0.19%
18	26	▲	Scratch	0.73%	+0.20%
19	19		Classic Visual Basic	0.65%	-0.42%
20	38	▲	Rust	0.64%	+0.38%

# Bilgisayarlar ve Programlamaya Giriş (2020)

8

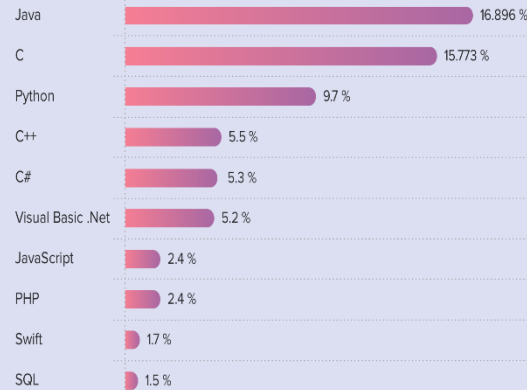
## Language Types

Web Mobile Enterprise Embedded

## IEEE's rankings

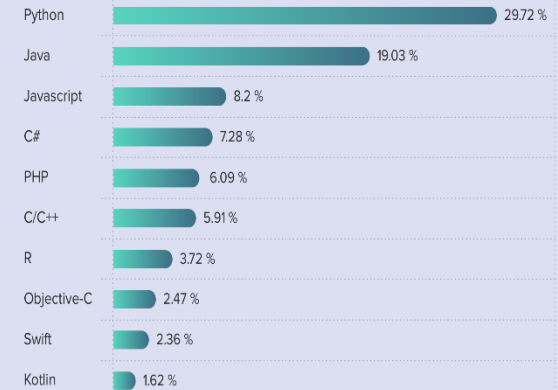
Rank	Language	Type	Score
1	Python	Web, Mobile, Enterprise	100.0
2	Java	Web, Mobile, Enterprise	95.3
3	C	Mobile, Enterprise	94.6
4	C++	Mobile, Enterprise	87.0
5	JavaScript	Web	79.5
6	R	Enterprise	78.6
7	Arduino	Enterprise	73.2
8	Go	Web, Mobile	73.1
9	Swift	Mobile	70.5
10	Matlab	Enterprise	68.4
11	Ruby	Web, Mobile	66.8

## Top programming languages, TIOBE



SHARE

## Top programming languages, PYPL



SHARE



# Bilgisayarlar ve Programlamaya Giriş (2020)

9

## Top Paying and Most Popular Programming Languages in 2020

### Rank by Average Salary

1. Python	\$119,000
2. JavaScript	\$117,000
3. Java	\$104,000
4. C	\$103,000
5. C++	\$102,000
6. C#	\$97,000
7. PHP	\$94,000
8. SQL	\$92,000

### Rank by Volume of Job Openings

1. Python	50,000
2. SQL	50,000
3. Java	45,000
4. JavaScript	38,000
5. C++	29,000
6. C#	21,000
7. PHP	13,000
8. C	9,000

# Program Tasarımı

10

- Daha öncede bahsettiğimiz gibi, günümüzde bir programcının, program yazmak için tipik olarak yüksek seviyeli diller kullanması gerekmektedir. Ancak, hangi profesyonel programcıya sorarsanız sorun program yazmadan önce dikkatlice tasarlanması gerektiğini size söyler. Başlanacak yeni bir projede her zaman ilk adım doğrudan kodlamanın içine girmek değil, programı tasarlamaktır.
- Program tasarlandıktan sonra bir yüksek seviye dilde kod yazılmaya başlanır. Unutulmamalıdır ki nasıl, insanların birbirleri ile iletişimi kurarken kullandıkları dillerin bazı sözdizimi (syntax) kuralları varsa, programlama dillerinin de vardır. Bu kurallar operatörler, anahtar kelimeler gibi şeyleri içerir. Bir programcı bu kurallara dikkat etmezse, sözdizimi hatası (syntax error) ortaya çıkar.
- Eğer program bir sözdizimi hatası içeriyorsa derleyici (compiler) (veya yorumlayıcı (interpreter)) hatanın ne olduğunu içeren bir hata mesajı verir. Gerçekte, çoğu projede kodlama ilk yapıldığında sözdizimi hataları olabilir. Ancak programcı kolaylık ve birazda vakit harcayarak hepsini düzeltir. Programımız artık derlenip, makine diline dönüşmeye hazırdır.



# Program Tasarımı

11

- ❑ Kod gerçekleştirilebilir (executable) forma geldiğinde mantıksal hataların (logic error) varlığının tespiti için test edilmelidir. Mantıksal hatalar programın çalışmasına engel olmayan fakat yanlış sonuç vermesine neden olan hatalardır. (Genelde matematiksel yanlışlar mantıksal hataların ortak nedenidir.)
- ❑ Eğer programda mantıksal hatalar var ise, programcı kodu onarma (debug) işlemine geçmelidir. Bunun anlamı programcının koddaki hataları tespit edip , düzeltmesidir. Bazen bu aşamada programcı, programın orijinal tasarımında hata olduğunu ve tasarımı değiştirmesi gerektiğini fark eder. Böylece program geliştirme döngüsü (program developmant cycle)'üne girilir ve bu döngü programda hiçbir hata olmayıncaya kadar sürer.

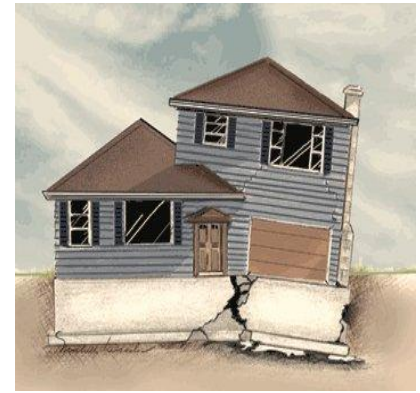


Program Geliştirme Döngüsü

# Program Tasarımı

12

- Bu ders özellikle bu döngünün ilk aşaması olan program tasarlama kısmına odaklanmıştır. Bu aşama bir program için tartışmasız en önemli aşamadır.
- Program tasarımını bir temel olarak düşünebilirsiniz.
  - Eğer evinizi kötü bir temel üzerine inşa ederseniz, ilerleyen zamanlarda evi tamir etmek için büyük bir çaba ve emek sarf etmek zorunda kalabilir ve hatta buna rağmen tamir edemeyebilirsiniz.
  - Aynı şey program içinde geçerlidir. Kötü temeli bir program, program geliştirme döngüsü içinde sıkışıp kalmanıza ve programı düzeltebilmek için çok uğraşmanıza neden olur.



# Program Tasarımı

13

## □ Program Tasarımı

### ▣ Bir program tasarlama süreci iki adımda özetlenebilir;

- Programın gerçekleştireceği görevi anlamak.
- Program verilen görevleri gerçekleştirecekken yapması gereken adımların hesaplanması.

### ▣ Programın gerçekleştireceği görevi anlamak.

- Profesyonel bir programcı olduğumuzu varsayalım. Bu durumda size belirli bir programı yazmanız için para veren, kişi, grup yada organizasyon sizin müşteriniz olsun. (Patronunuz yada departman müdürünüzde olabilir.) Programcı ve müşteri peşi sıra gerçekleşen görüşmeler yaparlar. Bu sırada müşteri gerçekleştirmesini istediği görevi programcıya aktarır. Aynı zamanda programcı kafasında ki tüm soru işaretlerini detaylı olarak müşteriye sorar. Bu görüşmelerin sonunda programcı bir yazılım ihtiyaç listesi hazırlar. Bu listenin tek amacı programın, müşteri tatmin edecek biçimde görevlerini yerine getirmesidir. Sonuç olarak müşteri bu listeyi onayladığı anda bir sonraki aşama olan ‘Program verilen görevleri gerçekleştirecekken yapması gereken adımların hesaplanması.’ aşamasına geçilebilir.

# Program Tasarımı

14

- Program verilen görevleri gerçekleştirecekken yapması gereken adımların hesaplanması.
  - Burada yapılan şey, başka bir insanın takip etmesi için görevin küçük parçalara ayrılmasına benzer. Örneğin; kız kardeşinizin size suyun nasıl kaynatılacağını sorduğunu düşünün.
    1. Cezveye istenilen miktar kadar su koy.
    2. Cezveyi ocağa yerleştir.
    3. Ocağı yüksek ateşe ayarla.
    4. Sudan büyük kabarcıklar çıkana kadar suyu izle. Bu olduğunda su kaynamıştır.
  - Verilen görev iyi tanımlanmış mantıksal adımlara bölündü. Buna algoritma diyoruz. Gördüğümüz basit bir algoritma örneğidir.

# Problem Çözme Sırası

15

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

# Algoritmaya Giriş



16

- Algoritmayı daha iyi anlayabilmek önce kısaca tarihinden bahsedelim.
- **El Harezmi – Algoritma**
  - Algoritma kelimesi ismini Özbekistan'ın Harezm kentinde doğmuş olan 9. yüzyıl alimi *Musa el Khowarizmi (el Harezmi)* tarafından 825 yılında yazılmış olan *Kitap al jabr w'al muqabala (Cebir ve kıyaslama kitabı)* kitabından almıştır.
  - “Bir algoritma; Girdi olarak bir değer veya değer kümesi alan ve çıktı olarak bir değer veya değer kümesi üreten iyi tanımlanmış herhangi bir hesaplama prosedürü. Dolayısıyla bir algoritma, girdiyi çıktıya dönüştüren bir dizi hesaplama adımlarından oluşur. ”
  - Ancak algoritma kavramı çok daha öncesine dayanır...
    - Babillerin M.Ö. 1800'lü yıllarda yazdığı düşünülen tabletlerde algoritmik işlemlerin yer aldığı görülmüştür.
    - İnanması güç olsa da bu tabletlerde çarpanlara ayırma ve kök bulma algoritmaları yer almaktadır.



Musa el Harezmi



# Algoritmaya Giriş

17

- Algoritmalar teorisi, algoritmaları 3 farklı model şeklinde ele alır:
- **Birinci tür**, algoritma kavramını klasik hesaplama ve sayısal fonksiyonlar gibi matematiksel kavramlarla ilişkilendirir. Bu sınıfın en gözde modeli, en eski algoritma kavramlarını da biçimlendiren *özyinelemeli fonksiyonlar (recursive functions)* sayılabilir.
- **İkinci tür**, algoritmaların her bir ayrık (discrete) zamanda, basit işlemleri yapan bir deterministik makine üzerinde çalışması prensibini benimser. Bu modelin temeli *Turing Makinesi*'nin kullanılmasına dayanır.

# Algoritmaya Giriş

18

- Algoritma tam olarak ne demektir?
  - ▣ Bir problemi çözmeye yarayan, sonlu, iyi tanımlanmış, sıralı işlemler kümesidir.
  - ▣ Adım adım ilerleyen bir hesaplama prosedürüdür.
- Aşağıdaki algoritma örneklerine bakalım.

## *Fen Fakültesi Binasından Yemekhaneye Gitme Algoritması*

1. BAŞLA.
2. Binanın ana kapısından dışarı çık.
3. Önündeki yola girip sağa dön ve döner kavşağa gelene kadar yürü.
4. Sol ön çaprazında gördüğün 4 katlı binaya gir.
5. 2. kata çık. *// Yemekhaneye vardın!!!*
6. BİTİR.

## *Sınıfta Soru Sorma Algoritması*

1. BAŞLA.
2. Hoca konuşuyor mu? (*Evet/Hayır*)
3. 'Evet' ise Adım 4'e git, 'Hayır' ise Adım 5'e git.
4. Hocanın lafını bitirmesini bekle.
5. Elini kaldır ve söz iste.
6. Hoca sana söz verdi mi? (*Evet/Hayır*)
7. 'Evet' ise sorunu sor, 'Hayır' ise Adım 2'ye git.
8. BİTİR.

# Algoritmaya Giriş

19

## □ Fen Fakültesi Binasından Yemekhaneye Gitme Algoritması

Bitiş: DEÜ Kaynaklar Yerleşkesi, Kuruçeşme Mahallesi,...

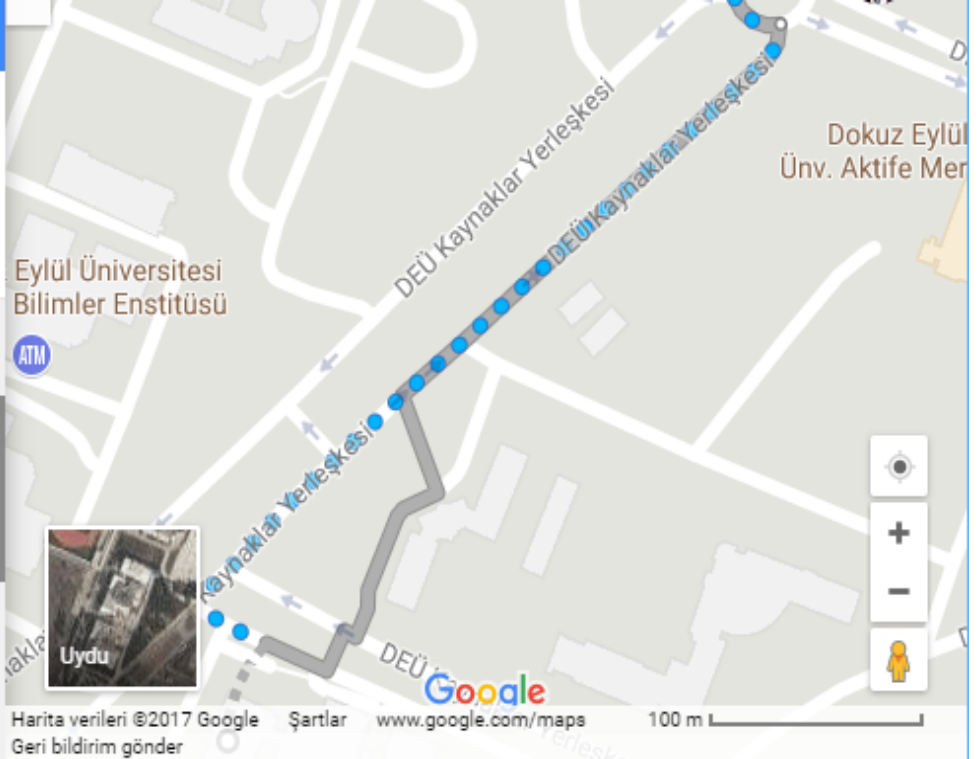
6 dk. (500 m)

DEÜ Kaynaklar Yerleşkesi üzerinden  
Çoğunlukla düz

↑ DEÜ Kaynaklar Yerleşkesi adlı yerden kuzeybatı  
yönünde ilerleyin  
39 m

↗ DEÜ Kaynaklar Yerleşkesi boyunca ilerlemek için  
sağa dönün  
400 m

📍 Döner kavşaktan 3. çıkışa girin ve DEÜ Kaynaklar  
Yerleşkesi boyunca ilerleyin



# Algoritmaya Giriş

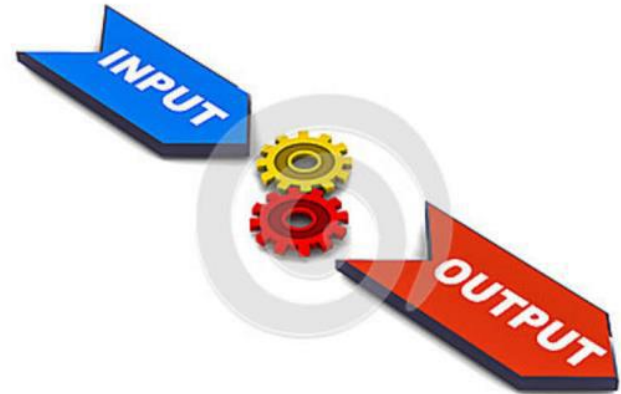
20

- Bir algoritmanın 4 (bazı kaynaklara göre 5, 1. özellik 2 ayrı özellik olarak da gösterilebiliyor) özelliği olmalıdır:
  - ▣ Valid Input / Output (Geçerli Giriş/Çıkış, Sonuç)
  - ▣ Finiteness (Sonluluk)
    - Herhangi bir giriş için, algoritma sınırlı sayıda adımdan sonra sona ermelidir.
  - ▣ Definiteness (Kesinlik, açıklık)
    - Algoritmanın tüm adımları kesin olarak tanımlanmalıdır.
  - ▣ Effectiveness (Etkinlik)
    - Algoritmanın her adımını doğru bir şekilde ve sınırlı bir süre içinde gerçekleştirmek mümkün olmalıdır. Her adımın kesin olması yeterli değildir (veya kesin olarak tanımlanmış), ancak aynı zamanda mümkün olmalıdır.

# Valid Input/Output (Geçerli Giriş/Çıkış)

21

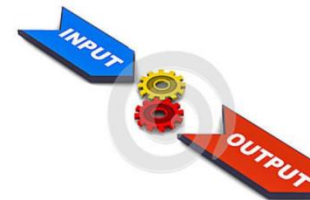
- **Input:** Bir algoritma girdi almayabileceği gibi belirlenmiş bir girdi seti içerisinde belirli sayıda girdi alabilir.
- **Output:** Bir algoritma girdi ile ilişkisi önceden tanımlanmış bir veya daha fazla çıktı üretir.



# Valid Input/Output (Geçerli Giriş/Çıkış)

22

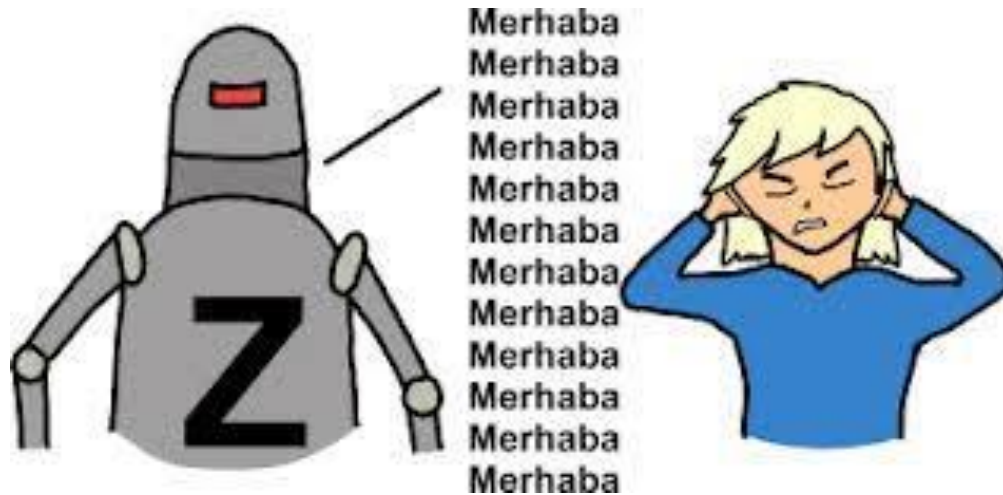
- Girdi ve Çıktıları Belirlemek
  - Problemi iyi tanımlamak için başlangıç ve bitiş noktalarını çok net belirlememiz gerekir.
  - Bizim bulacağımız şey, problemin çözüm yoludur.
  - Problem çözümünde kullanılacak parametreleri iyi tespit etmek gerekir.
  - Aksi halde geliştirilen algoritma problemin çözümü için yetersiz kalabilir.
  - Bunun için algoritmanın girdilerini ve çıktılarını iyice kavramalıyız.



# Finiteness (Sonluluk)

23

- Algoritma her zaman sayılı adımdan sonra sonlanmalı.



# Definiteness (Kesinlik, Açıklık)

24

- Algoritmanın tüm adımları kesin olarak tanımlanmalıdır.



Her bir adım ayrıntılı bir şekilde tanımlanmalıdır: her bir adımda yapılacak işler karmaşıklığa izin vermeyecek şekilde açıklanmalıdır.



# Effectiveness (Etkinlik)

25

- Algoritmanın her adımını doğru bir şekilde ve sınırlı bir süre içinde gerçekleştirmek mümkün olmalıdır. Her adımın kesin olması yeterli değildir (veya kesin olarak tanımlanmış), ancak aynı zamanda mümkün olmalıdır.



- Her bir adım sonlu bir zamanda tamamlanabilecek olabildiğince basit işlemler olarak verilmelidir.

# Algoritmaya Giriş

26

- Problem: Çözülmesi ve bir sonuca ulaştırılması gereken durum.
  - ▣ Bugün ne giyelim?
  - ▣ Evden okula geliş.
  - ▣ Gazlı bir içecek şişesinin kapağını açmak
  - ▣ Yürümek
  - ▣ vs. vs.
- Algoritmaların Gösterimi:
  - ▣ Pseudo code (Kaba Kod)
    - Adım adım yazılmış talimatlar listesi şeklindedir.
  - ▣ Flowcharts (Akış Şemaları)
    - Bir algoritmanın veya bir işlemin (process) çalışma sırasına uygun biçimde ve belirli anlamları olan simgelerin oklar ile bağlanması yoluyla oluşturulan şemalara verilen isimdir.

# Günlük Yaşamdan Bir Örnek

27

**Örnek:** Bir öğrencinin okula giderken izleyeceği yolu ve okula girişinde yapacaklarını tarif edelim.

## Çözüm

1. Evden/Yurttan dışarıya çık,
2. Otobüs durağına git,
3. Otobüsün geldiğinde otobüse bin,
4. Yolcu ücretini öde,
5. İneceğin yere yakınlaştığında arkaya yürü,
6. İneceğini belirten ikaz lambasına bas,
7. Otobüs durunca in,
8. Okula doğru yürü,
9. Okul giriş kapısından içeriye gir,
10. Sınıfa gir,
11. Dersi dinle.

# Algoritma Geliştirme Yöntemleri

28

- Bir problemin çözümü için geliştirilen algoritma üç farklı şekilde ifade edilebilir:
  - **Step form (Adım formu) – Satır Algoritma**
  - **Pseude-code (Kabakod) – Sözde kod**
  - **Flow charts (Akış Şemaları) – Akış diyagramı**

# Algoritma Geliştirme Yöntemleri

29

- Satır Algoritma Problemin çözümü için yapılması gereken işlem adımları öncelik sıraları göz önünde bulundurularak, açık ve net bir şekilde (hiçbir alternatif yoruma izin vermeksizin) sözel olarak ifade edilir.

# Adım Formu (Step Form)

30

- Bu formda algoritma basit adımlar yardımı ile tanımlanır.
- Yapılacaklar listesine benzemesi bu formu en kolay öğrenilen form yapmaktadır. Ancak diğer formları öğrendikten sonra bu yöntemi daha az kullanmaya başlayacaksınız.

# Pseudo Kod

31

- Pseudo kod algoritma geliřtirmeye yardımcı olmak için kullanılan yapay ve resmi olmayan bir dildir.
- Pseudo code günlük konuşma dili gibidir;
  - ▣ Kolay
  - ▣ Kullanıcı dostu
  - ▣ Gerçek olmayan bir programlama dili.
- Bilgisayarlar tarafından çalıştırılmaz.
- Bir programı yazmaya başlamadan önce onun hakkında düşünmenize yardımcı olur.
- Sadece eylem ifadelerinden oluşur. Tanımlamalar çalıştırılabilir ifadeler değildir ve herhangi bir eyleme sebep olmaz. Bu sebeple pseudo kod içinde yer almazlar.

# Pseudo Kod

32

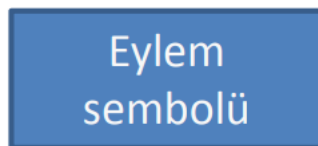
- Bazı temel pseudo kod komutları şunlardır:
  - ▣ Başla: Programın başladığını ifade eder.
  - ▣ Bitir: Programın bittiğini ifade eder.
  - ▣ Oku: Kullanıcı girişi için yazılır.
  - ▣ Yaz: Kullanıcıya bilgi veya sonuç göstermek için yazılır.
  - ▣ Eğer ... İse ...: Şartlara göre akışın değişmesinde kullanılır.
  - ▣ Eğer .... Değilse...: Şartlara göre akışın değişmesinde kullanılır.



# Akış Diyagramı

33

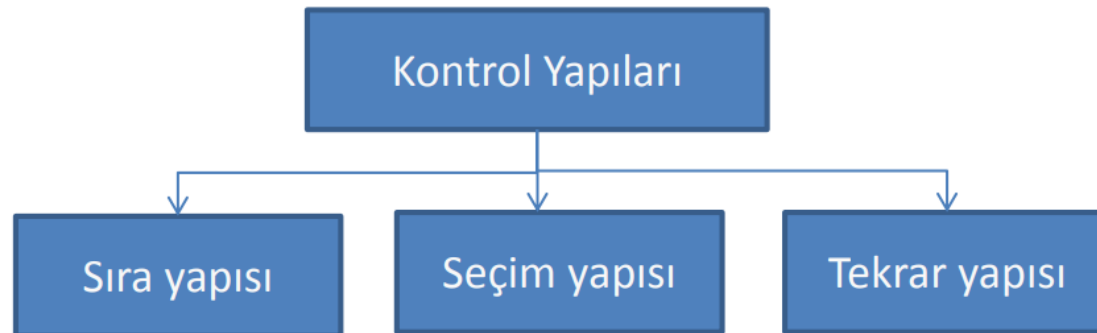
- ❑ Çeşitli anlamlar ifade eden ve birbirine oklarla bağlanan şekillerle görsel olarak algoritmanın adımlarını ifade etmektir.
- ❑ Akış şemaları Dikdörtgen, Baklava, Elips, Daire gibi özel amaçlı bazı sembollerin çizilmesi ile oluşturulurlar.



# Kontrol Yapıları

34

- Araştırmacılar tüm programların sadece üç kontrol yapısı ile yazılabileceğini gösterdi.
- Bu yapılar, sıra yapısı, seçim yapısı ve tekrarlama yapısı.



# Kontrol Yapıları (Sıra Yapısı)

35

- Bir programdaki ifadeler bir biri ardına yazıldıkları sırada çalıştırılırlar.
- Aksi belirtilmediği sürece C ifadeleri bir biri ardına yazıldıkları sıra ile çalıştırılırlar.

# Algoritma

36

- Temelde algoritmamızı üç ana bölüme ayırabiliriz:
  - ▣ Giriş: Bilgisayarın üzerinde çalışacağı veri kullanıcı tarafından girilir.
  - ▣ İşlem: Bilgisayar girilen bilgiyi işler.
  - ▣ Çıkış: Bilgisayar insanların anlayacağı şekilde ekrana sonucu gösterir.

# Algoritma

37

## □ Giriş:

- ▣ Ürünün adını al.
- ▣ Ürünün miktarını al.
- ▣ Bir dosyadan ürünün fiyatını oku.

## □ İşlem:

- ▣ Ürünün toplam fiyatını hesapla.
- ▣ Gerekli indirimi yap.

## □ Çıkış:

- ▣ Toplam satış değerini yaz.

## □ Örnekteki adımları algoritma olarak alt alta toplarsak:

1. Başla.
2. Ürünün adını al.
3. Ürünün miktarını al.
4. Bir dosyadan ürünün fiyatını oku.
5. Ürünün toplam fiyatını hesapla.
6. Gerekli indirimi yap.
7. Toplam satış değerini yaz.
8. Bitir.

# Örnek: Bir porsiyon menemen yapma algoritması

38

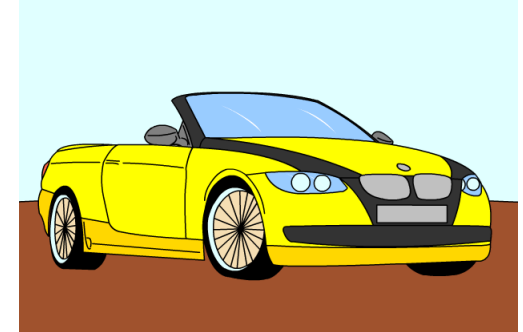
1. Başla
2. 2 yumurta al
3. Yarım yemek kaşığı yağ al
4. Bir domatesi yıka ve küçük küpler şeklinde doğra
5. Bir biberi yıka ve küçük küpler şeklinde doğra
6. Hepsini bir tavaya koy
7. Tavanın altındaki ocağı yak
8. Bir çimdik tuz ve karabiber ekle
9. İstenilen kıvama gelinceye kadar
  - \* Tahta bir kaşık ile karıştır
10. Ocağı kapat
11. Tavadakileri bir yemek tabağına koy
12. Bitir



# Örnek: Park halindeki bir aracı çalıştırıp hareket ettiriniz.

39

1. Başla
2. Şoför kapısını aç
3. Sürücü koltuğuna otur
4. Kemerini tak
5. Anahtarı kontak girişine yerleştir
6. Koltuğu kendine göre ayarla
7. Dikiz aynasını ve yan aynaları kendine göre ayarla
8. Sol ayak ile en soldaki debriyaj pedalına sonuna kadar bas
9. Vitesi boş konumuna getir
10. Anahtarı, motor çalışıncaya kadar çevir, çalışınca bırak
11. Vitesi 1 konumuna getir (Sol üstte)
12. Sol sinyali çalıştır (Direksiyonun solundaki kolu aşağıya indir)
13. Aynalardan trafiğin uygun olup olmadığını kontrol et
14. Değilse 12. adıma geri dön uygunsa 15.adıma geç
15. Eğer çekili ise el frenini indir
16. Sağ ayak ile en sağdaki gaz pedalına hafifçe bas
17. Sol ayağını debriyajdan hafifçe kaldır
18. Hareket edince sol ayağını tamamen kaldır
19. Bitir



# Örnek: Brownie yapmak

40

1. Başla
2. Eğer yağ yumuşak değilse erit
3. Krema kıvamına gelinceye kadar yağ ile şekeri karıştır.
4. Yumurta ve vanilyayı ekle, karıştır
5. Kakao ve unu ekle ve iyice karıştır
6. Karışımı yağlanmış kek kalıbına dök
7. Mikrodalga fırında 8-9 dakika pişir
8. Bitir





# Alıştırma: Çay demleme

41



# Alıştırma: Çay demleme

42



# Algoritmaları nerede görebilirsiniz?

43

- IKEA mobilyası birleřtirmek
- Sözlükten kelime arama
- Kağıt uçak yapmak
- Okuldan eve gitmek
- Yapboz çözme
- Sudoku çözme
- Rubik küp çözme



# Örnek: iki sayının toplamını bulan algoritmayı yazınız.

44

1. BAŞLA
2. Birinci sayıyı al
3. İkinci sayıyı al
4. Bu iki sayının toplamını bul
5. Toplam sonucunu yazdır. (Sonucu söyle)
6. BİTİR.

# Örnek: Doğum yılı 2000 olan bir kişinin yaşını bulan algoritmayı yazınız.

45

## Algoritma :

1. Başla
2. Yaşı hesapla ( $\text{yaş} \leftarrow 2022 - 2000$ )
3. Yaşı ekrana yazdır.
4. Bitir

# Örnek: Doğum yılı verilen bir kişinin yaşını bulan algoritmayı yazınız.

46

## Algoritma:

1. Başla
2. Doğum yılını sor (dyili)
3. Yaşı hesapla ( $\text{yaş} \leftarrow 2022 - \text{dyili}$ )
4. Yaşı ekrana yazdır.
5. Bitir

# Algoritmaya Giriş

47

- Alıştırma:
- İki sayıdan büyük olanı bulan algoritmayı yazınız.

Algoritma :

1. Başla
2. Kullanıcı iki sayı girsin. (x, y)
3. Eğer x, y'den büyükse x'i yazdır, değilse y'yi yazdır.
4. Bitir

# Algoritmaya Giriş

48

- Alıştırma:
- İki sayıdan **küçük** olanı bulan algoritmayı yazınız.



# Algoritmaya Giriş

49

- Alıştırma:
- İki sayıdan **küçük** olanı bulan algoritmayı yazınız.

## Algoritma :

1. Başla
2. Kullanıcı iki sayı girsin. (x, y)
3. Eğer x, y'den **küçükse** x'i yazdır, değilse y'yi yazdır.
4. Bitir

# Algoritmaya Giriş

50

- Alıştırma:
- Gerekli parametreleri kullanıcı tarafından girilen bir silindirin hacmini hesaplayıp ekrana yazdırın. (Yarıçapı ( $r$ ) ve yüksekliği ( $h$ ), ( $\pi$ ) girilsin.)

# Algoritmaya Giriş

51

- Alıştırma:
- Gerekli parametreleri kullanıcı tarafından girilen bir silindirin hacmini hesaplayıp ekrana yazdırın.

## Algoritma :

1. Başla
2. Kullanıcı yarıçapı (r) ve yüksekliği (h), ( $\pi$ ) girsin.
3. Silindirin hacmini bulun.  
$$\text{Hacim} = \pi \times r \times r \times h$$
4. Silindirin hacmini ekrana yazdırın.
5. Bitir

# Pseudocode (Kabakod)

52

Bir algoritmanın, herhangi bir programlama dili olmadan, ama bir programa benzer ifadesidir.

# Örnek

53

## Step Form

1. BAŞLA
2. İlk sayıyı al
3. İkinci sayıyı al
4. Bu iki sayının toplamını bul
5. Sonucu söyle
6. BİTİR.

## Pseudocode

1. Başla
2. X değerini oku
3. Y değerini oku
4.  $\text{Toplam} \leftarrow X + Y$
5. Toplam'ı Yaz
6. BİTİR.

# Örnek: Üçgenin alanını hesaplayan algoritma

54

1. Başla
2. Taban değerini oku
3. Yükseklik değerini oku
4. Taban ile yüksekliği çarp, ve sonucu 2 ile böl
5. Sonucu yazdır
6. Bitir.

## Step Form

1. START
2. Read the base
3. Read the height
4. Multiply the base by the height, and then divide by 2
5. Print the result
6. STOP.

## Pseudocode

b-base, h-height, area-A

1. START
2. Read b
3. Read h
4.  $A = (b * h) / 2$
5. Print A
6. STOP.

# Uygulamalar

56

- ❑ İki sayıdan büyük olanı bulan algoritmayı yazınız.
- ❑ Üç sayıdan büyük olanı bulan algoritmayı yazınız.
- ❑ Üç sayıdan küçük olanı bulan algoritmayı yazınız.
- ❑ Üç sayının ortancasını bulan algoritmayı yazınız.
- ❑ 5918 sayısından büyük olanı bulan algoritmayı yazınız.
- ❑ Verilen 5 adet sayının ortalamasını bulan algoritmayı yazınız.
- ❑ Vize, final ve ödev notları verilen bir öğrencinin dönem sonu not ortalamasını bulan algoritmayı yazınız.  
(Vize[%20], Ödev [%40], Final [%40])
- ❑ Bir A4 kağıdı kullanarak uçak yapma algoritmasını yazınız.



# Sorular???

57

