

Temel Kavramlar, Algoritma Kavramı ve  
Önemi, Algoritma Türleri

**Dr. Fatih KALEMKUŞ**

*Kafkas Üniversitesi*

# Temel Kavramlar

## • Bilgisayar Nedir?

- Bilgisayar, aritmetiksel ve mantıksal işlemlerden oluşan bir işi, önceden verilmiş programa göre yapıp sonuçlandıran elektronik bir araçtır.
- Bir bilgisayarın çalışabilmesi için üç temel birime ihtiyaç vardır.
  - Merkezi İşlem Birimi (Central Processing Unit-CPU)
  - Bellek Birimi
  - Giriş-Çıkış Birimi(I/O)



# Temel Kavramlar

- **Merkezi İşlem Birimi:**  
Bilgisayardaki tüm karar verme ve kontrol işlemlerini gerçekleştirir. Matematiksel işlemleri gerçekleştirdiği gibi bilgisayarda hangi birimlerden giriş yapılacak hangi sırada çıkış yapılacak öncelikler nasıl olacak vb. işlemleri de gerçekleştirir.
- **Bellek Birimi:**  
Bilgisayarlar çalıştıkları süre boyunca giriş biriminden aldığı veya hesaplama sonucu elde ettiği verileri bellek üzerinde saklayarak işlemleri gerçekleştirirler.
- **Giriş/Çıkış Birimleri:**  
Kullanıcıdan veya diğer aygıtlardan (fare, klavye, mikrofon, kamera, tarayıcı vb.) bilgisayara veri aktarmak için kullanılan birimlere **Giriş Birimleri**; bilgisayarda bulunan verileri kullanıcıları bilgilendirmek amacıyla veya diğer aygıtlara (ekran, yazıcı, tarayıcı, hoparlör, kulaklık vb.) göndermek amacıyla kullanılan birimlere de **Çıkış Birimleri** denir.

# Donanım ve Yazılım

- Bilgisayar sistemleri yazılım ve donanım olmak üzere iki kısımdan oluşmaktadır.
- Donanım:  
Bilgisayarda gözle görebildiğimiz fiziksel parçalar donanım olarak isimlendirilir. Donanımlar kullanım amaçlarına göre 4 kısımda incelenirler.
  - Merkezi İşlem Birimi
  - Bellek Birimi
  - Depolama Birimleri
  - Çevre Birimleri

# Donanım ve Yazılım

- Yazılım:

Bilgisayarın çalışması için donanım dışında kalan kısma yazılım denilir. Yani, yapılması gereken işleri yapabilmek için donanıma komutlar veren **programlar topluluğudur**.

Genel olarak üç kısımda incelenebilir.

- Sistem Yazılımları (İşletim Sistemi – Windows, Unix, Linux vs.)
- Program Geliştirme Yazılımları (Programlama Dilleri – Java, C, Pascal, Python vs.)
- Uygulama Yazılımları (MS Word, Excel, Autocad, vs.)

**Yazılım geliştirme sonucu ortaya çıkan ürüne program denir.**

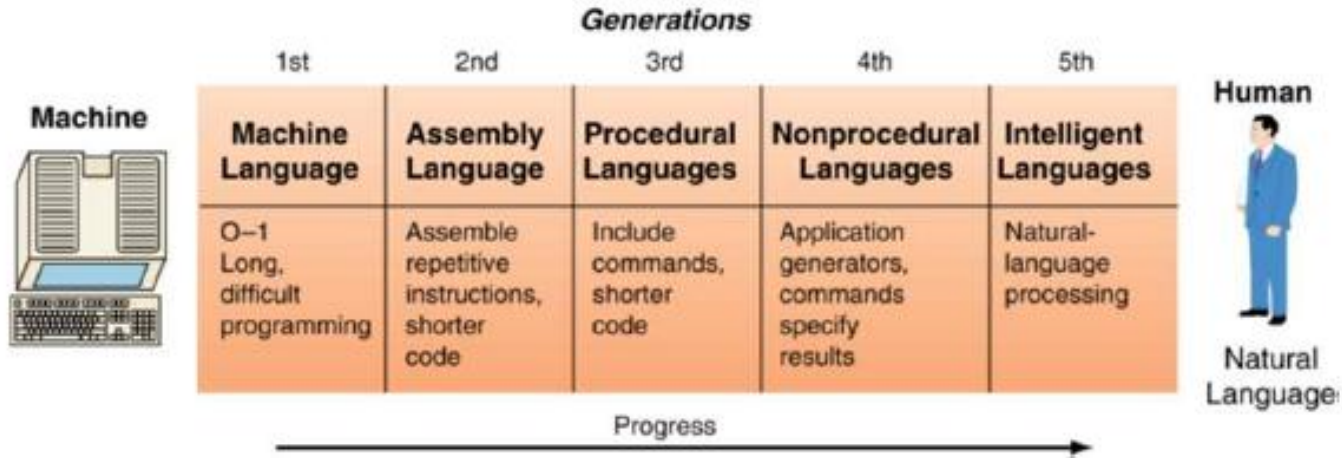
Bir problemin bilgisayar tarafından çözülebilmesi için öncelikle algoritmasının oluşturulması gerekmektedir.

# Programlama Dillerinin Gelişimi

- Program:  
Belirli bir işi gerçekleştirmek için gerekli komutlar dizisi olarak tanımlanabilir.
- Programlama:  
Bir programı oluşturabilmek için gerekli komutların belirlenmesi ve uygun biçimde kullanılmasıdır.
- Programlama Dilleri:  
Bir programın oluşturulmasında kullanılan komutlar, tanımlar ve kuralların belirtildiği programlama araçlarıdır.

Programlama Dili:  
Bilgisayarlara ne yapmaları gerektiğini söylememizi sağlayan özel bir dil

*Tüm yazılımlar programlama dilleri ile yazılır.*



# Programlama Dillerinin Gelişimi

- İlk programın, Ada Lovelace tarafından Charles Babbage'ın tanımladığı "Analytical Engine" ile Bernoulli sayılarının hesaplanmasına ilişkin makalesinde olduğu söylenmektedir. Bu nedenle ilk gerçek anlamdaki programlama dillerinden birinin adı Ada Lovelace anısına ADA olarak isimlendirilmiştir.
- 1940'larda ilk modern bilgisayar ortaya çıktığında, programcılar yalnızca assembly dili kullanarak yazılım yapabiliyorlardı.

- 1962 - APL
- 1964 - BASIC
- 1964 - PL/I
- 1970 - Pascal → Yapısal programlama
- 1970 - Forth
- 1972 - C
- 1972 - Prolog
- 1978 - SQL → Nesne yönelimli dillerin ortaya çıkışı
- 1983 - Ada
- 1983 - C++
- 1987 - Perl

## 1990 lar, Internet

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

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

# Sayı Sistemleri

- Günlük yaşantımızda 10 luk sayı sistemi kullanılır. Ancak, bilgisayar sistemleri 2 lik sayı sistemini kullanılırlar. 10 luk sistemde taban 10, ikilik sistemde taban 2 dir.
- Sayı sistemlerinde sayıyı oluşturan her bir rakam **digit** olarak adlandırılır. Onluk sayı sistemlerinde her bir rakam **decimal digit** yada sadece **digit**ken, ikilik sistemde **binary digit** yada kısaca **bit** olarak adlandırılır.
- 123456 6 digitlik onlu sayı  
100101 6 bitlik ikili sayı
- Sayı sembolleri 0 .. (Taban-1) arasındadır.
- Onluk düzende rakamlar 0..9, ikilik düzende rakamlar 0, 1 den oluşur.
- Sayıların oluşturulması
- $123456 = 1 * 10^5 + 2 * 10^4 + 3 * 10^3 + 4 * 10^2 + 5 * 10^1 + 6 * 10^0$
- $100101 = 1 * 2^5 + 0 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0$

İkilik	Onluk
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
...	...

# Sayı Sistemleri

- Sekiz bitlik ikili sayılara bir byte lık sayılar denir.
- 10011101 8 bit yada bir **byte**dir.
- 16 bit uzunluklu sayılara 1 word luk sayılar denmesine rağmen, bu kavram bazen işlemcinin veri yolu uzunluğu kadar bit sayısı ile de eşleştirilmektedir.
- 11001001 11100011 2 byte lık yada 1 **word**luk sayı.
- Ayrıca her 4 bit, bir **Nibble** olarak adlandırılır.

# Sayı Sistemleri

- Bellek Ölçü Birimleri
  - 1 Byte = 8 Bit
  - 1 Kilobyte (KB) =  $10^3$  byte = 1.024 byte.
  - 1 Megabyte (MB) =  $10^6$  byte = 1.048.576 byte.
  - 1 Gigabyte (GB) =  $10^9$  byte = 1.073.741.824 byte.
  - 1 Terabyte (TB) =  $10^{12}$  byte = 1.099.511.627.776 byte.
  - 1 Petabyte (PB) =  $10^{15}$  byte.
  - 1 Exabyte (EB) =  $10^{18}$  byte.
  - 1 Zettabyte (ZB) =  $10^{21}$  byte.
  - 1 Yottabyte (YB) =  $10^{24}$  byte.

# Sayı Sistemleri

- Pozitif ve Negatif Sayılar
- Bir byte'lık en küçük ve en büyük pozitif sayılara bakalım  
00000000 (decimal 0)  
11111111 (decimal 255)
- İkilik sistemde negatif sayılar, çıkarma işleminin toplama aracılığıyla yapılabilmesini sağlamak amacıyla tümleyen sayılarla gösterilir. Tümleyen sayı, verilen sayıyı, o bit sayısı için temsil edilen en büyük sayıya tamamlayan sayıdır. (Pratikte bit evirerek yapılır.)
- Örneğin 00001010'ın tümleyeni 11110101'dir. (255 - 10). Bu türden tümleyene **1'e tümleyen sayı** denir.

# Sayı Sistemleri

- Bilgisayarlar yalnızca sayılarla çalışırlar, oysa bizim harflere ve diğer sembollere de gereksinimimiz vardır. Bu semboller de sayılara karşılık düşürülecek biçimde kodlanırlar. Program örneğin bu sayı ile karşılaşırsa ekrana karşılık düşen sembolü basar, yada klavyeden gelen sayının sembolik karşılığını, yazıcıdan çıkarır.
- Bir çok kodlama türü olmasına karşın dünyada bilgisayar ortamlarında ANSI tarafından 1963 yılında standartlaştırılan **ASCII (American National Code for Information Interchange)** kodlaması yoğun olarak kullanılmaktadır. Ancak günümüzde ,ASCII kodları çok dilliliği sağlayabilmek için yetersiz kaldığından **UNICODE** kodlaması yaygınlaşmaktadır. Ancak pek çok uygulamada ASCII kodlaması hala geçerliliğini korumaktadır.
- ASCII temel olarak 7 bit' tir. 127 karakterden oluşur. Ama Extended kısmıyla birlikte 8 bit kullanılmaktadır. Ancak genişletilmiş kısımdaki semboller yazılım ortamına göre değişebilmektedir.

# Sayı Sistemleri

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`	128	80	Ç	160	A0	á	192	C0	Ľ	224	E0	α
1	01	Start of heading	33	21	!	65	41	A	97	61	a	129	81	ú	161	A1	í	193	C1	ł	225	E1	β
2	02	Start of text	34	22	"	66	42	B	98	62	b	130	82	é	162	A2	ó	194	C2	ŧ	226	E2	Γ
3	03	End of text	35	23	#	67	43	C	99	63	c	131	83	á	163	A3	ú	195	C3	ł	227	E3	π
4	04	End of transmit	36	24	\$	68	44	D	100	64	d	132	84	ä	164	A4	ñ	196	C4	—	228	E4	Σ
5	05	Enquiry	37	25	%	69	45	E	101	65	e	133	85	ä	165	A5	Ñ	197	C5	†	229	E5	σ
6	06	Acknowledge	38	26	&	70	46	F	102	66	f	134	86	Å	166	A6	•	198	C6	‡	230	E6	μ
7	07	Audible bell	39	27	'	71	47	G	103	67	g	135	87	ç	167	A7	°	199	C7	‡	231	E7	ı
8	08	Backspace	40	28	(	72	48	H	104	68	h	136	88	é	168	A8	ˆ	200	C8	ˆ	232	E8	φ
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i	137	89	ë	169	A9	˜	201	C9	ˆ	233	E9	θ
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j	138	8A	è	170	AA	˘	202	CA	ˆ	234	EA	Ω
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k	139	8B	ı	171	AB	˙	203	CB	ˆ	235	EB	ϑ
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l	140	8C	ı	172	AC	˚	204	CC	ˆ	236	EC	∞
13	0D	Carrige return	45	2D	-	77	4D	M	109	6D	m	141	8D	ı	173	AD	ı	205	CD	ˆ	237	ED	∞
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n	142	8E	Ä	174	AE	«	206	CE	ˆ	238	EE	ε
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o	143	8F	Å	175	AF	»	207	CF	ˆ	239	EF	∅
16	10	Data link escape	48	30	0	80	50	P	112	70	p	144	90	É	176	B0	◻	208	DO	ˆ	240	FO	∞
17	11	Device control 1	49	31	1	81	51	Q	113	71	q	145	91	æ	177	B1	◻	209	D1	ˆ	241	F1	±
18	12	Device control 2	50	32	2	82	52	R	114	72	r	146	92	Æ	178	B2	◻	210	D2	ˆ	242	F2	≥
19	13	Device control 3	51	33	3	83	53	S	115	73	s	147	93	ó	179	B3		211	D3	ˆ	243	F3	≤
20	14	Device control 4	52	34	4	84	54	T	116	74	t	148	94	ö	180	B4		212	D4	ˆ	244	F4	[
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u	149	95	ó	181	B5		213	D5	ˆ	245	F5	]
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v	150	96	û	182	B6		214	D6	ˆ	246	F6	ˆ
23	17	End trans. block	55	37	7	87	57	W	119	77	w	151	97	ù	183	B7		215	D7	ˆ	247	F7	∞
24	18	Cancel	56	38	8	88	58	X	120	78	x	152	98	ÿ	184	B8		216	D8	ˆ	248	F8	*
25	19	End of medium	57	39	9	89	59	Y	121	79	y	153	99	ÿ	185	B9		217	D9	ˆ	249	F9	*
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z	154	9A	Û	186	BA		218	DA	ˆ	250	FA	ˆ
27	1B	Escape	59	3B	;	91	5B	[	123	7B	(	155	9B	◊	187	BB		219	DB	ˆ	251	FB	√
28	1C	File separator	60	3C	<	92	5C	\	124	7C		156	9C	◊	188	BC		220	DC	ˆ	252	FC	∞
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	)	157	9D	◊	189	BD		221	DD	ˆ	253	FD	*
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~	158	9E	◊	190	BE		222	DE	ˆ	254	FE	◻
31	1F	Unit separator	63	3F	?	95	5F	~	127	7F	◻	159	9F	f	191	BF		223	DF	ˆ	255	FF	◻

# Temel Kavramlar

- Bilgisayar programları ile gerçekleştirilen işlemler;
  - 1) Matematiksel İşlemler
  - 2) Karşılaştırma (karar) İşlemleri
  - 3) Mantıksal (lojik) İşlemler
- Matematiksel İşlemler
  - Temel aritmetik işlemler
  - Toplama, çıkarma, çarpma, bölme
  - Matematiksel fonksiyonlar
  - Üstel, logaritmik, trigonometrik, hiperbolik ) vb

# Temel Kavramlar

## Matematiksel İşlemler

<i>İşlem</i>	<i>Matematik</i>	<i>Bilgisayar</i>
<i>Toplama</i>	$a+b$	$a+b$
<i>Çıkarma</i>	$a-b$	$a-b$
<i>Çarpma</i>	$a.b$	$a*b$
<i>Bölme</i>	$a:b$	$a/b$
<i>Üs alma</i>	$a^b$	$a^b$

Matematiksel işlemlerin öncelik sırası ?

<i>Sıra</i>	<i>İşlem</i>	<i>Bilgisayar dili</i>
1	<i>Parantezler</i>	$((.....))$
2	<i>Üs alma</i>	$a^b$
3	<i>Çarpma ve bölme</i>	$a*b$ ve $a/b$
4	<i>Toplama ve çıkarma</i>	$a+b$ ve $a-b$

NOT: Bilgisayar diline kodlanmış bir matematiksel ifade, aynı önceliğe sahip işlemler mevcut ise bilgisayarın bu işlemleri gerçekleştirme sırası soldan sağa(baştan sona) doğrudur.

Örneğin ;  $Y=A*B/C$   
Önce  $A*B$  işlemi yapılacak, ardından bulunan sonuç C ye bölünecektir.

# Temel Kavramlar

## ■ Matematiksel İşlemler

Matematiksel Yazılım	Bilgisayara Kodlanması
$a+b-c+2abc-7$	$a+b-c+2*a*b*c-7$
$a+b^2-c^3$	$a+b^2-c^3$
$a - \frac{b}{c} + 2ac - \frac{2}{a+b}$	$a-b/c+2*a*c-2/(a+b)$
$\sqrt{a+b} - \frac{2ab}{b^2-4ac}$	$(a+b)^{(1/2)}-2*a*b/(b^2-4*a*c)$
$\frac{a^2+b^2}{2ab}$	$(a^2+b^2)/(2*a*b)$

# Temel Kavramlar

## Karşılaştırma (Karar) İşlemleri

- İki büyüklükten hangisinin büyük veya küçük olduğu,
- İki değişkenin birbirine eşit olup olmadığı gibi konularda karar verebilir.

<i>İşlem sembolü</i>	<i>Anlamı</i>
=	Eşittir
<>	Eşit değil
>	Büyüktür
<	Küçüktür
>= veya =>	Büyük eşittir
<= veya =<	Küçük eşittir

# Temel Kavramlar

## Mantıksal İşlemler

Mantıksal işlem	Matematiksel sembol	Komut
Ve	.	And
Veya	+	Or
değil	'	Not

"ve, veya, değil" operatörleri hem matematiksel işlemlerde hem de karar ifadelerinde kullanılırlar.

- Bütün şartların sağlatılması isteniyorsa koşullar arasına VE
- Herhangi birinin sağlatılması isteniyorsa koşullar arasına VEYA
- Koşulu sağlamayanlar isteniyorsa DEĞİL mantıksal operatörü kullanılır.

# Mantıksal İşlemler

## Örnek-1

- Bir işyerinde çalışan işçiler arasından yalnızca yaşı 23 üzerinde olup, maaş olarak asgari ücret alanların isimleri istenebilir.
- Burada iki koşul vardır ve bu iki koşulun da doğru olması gerekir. Yani;

Eğer  $Yaş > 23$  VE  $maaş = \text{asgari ücret}$  ise ismi Yaz

Yaz komutu 1. ve 2.koşulun her ikisi de sağlanıyorsa çalışır.

## Örnek-2

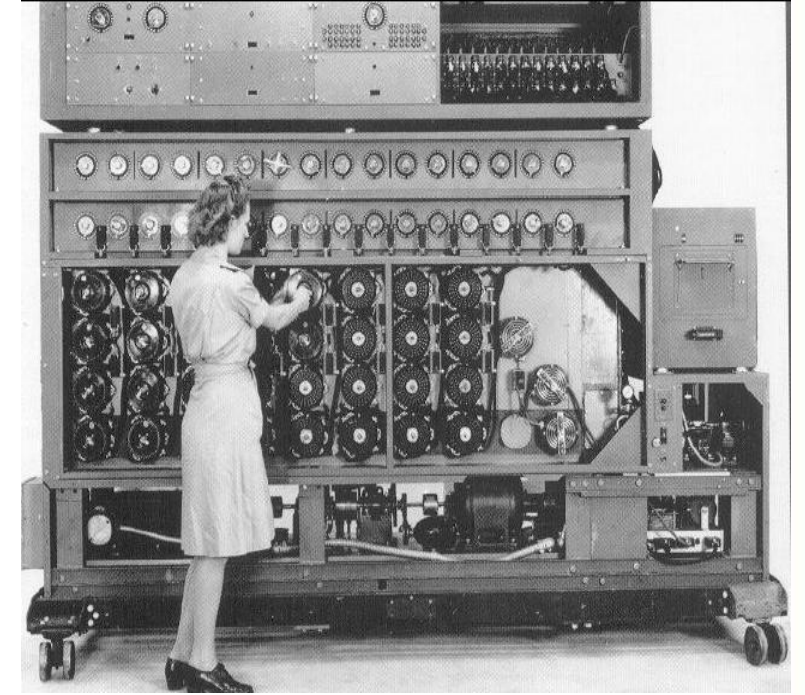
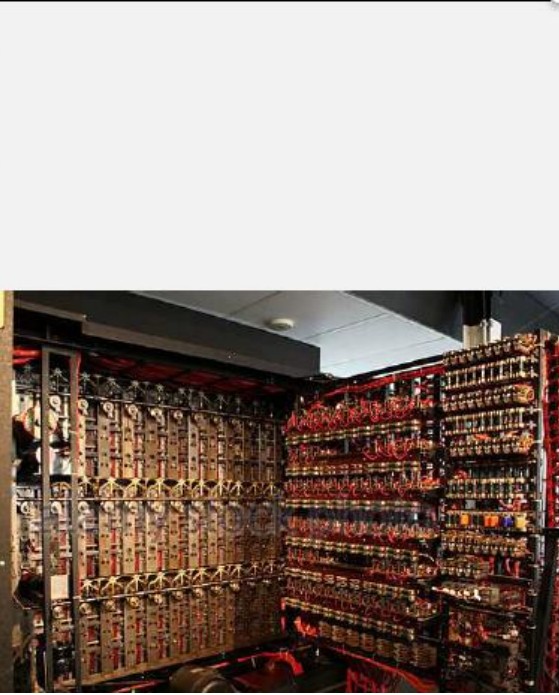
- Bir sınıfta Bilgisayar dersinden 65 in üzerinde not alıp, Türk Dili veya Yabancı Dil derslerinin herhangi birinden 65'in üzerinde not alanların isimleri istenmektedir.
- Burada 3 koşul vardır ve Bilgisayar dersinden 65 in üzerinde not almış olmak temel koşuldur. Diğer iki dersin notlarının herhangi birinin 65 in üzerinde olması gerekir.

Eğer  $Bilg.Not > 65$  VE  $(TDil Not > 65$  veya  $YDil Not > 65)$  ise ismi Yaz

# Algoritma Kavramı



Okuyunuz: [Alan Turing](https://goo.gl/cqgBzz) (<https://goo.gl/cqgBzz>)



# Algoritma Kavramı

- Bu bölümde temel olarak algoritma kavramını işleyeceğiz. Algoritmanın ne olduğunu, tarihçesini, algoritmayla ilgili kavramları işleyeceğiz. Böylece algoritmanın ne olduğu ve ne olmadığı kafamızda iyice belirginleşecek ve algoritmaları nerede kullanacağımızı kavrayacağız.
- Algoritma, 800'lü yıllarda yaşamış olan Acem matematikçi Muhammad ibn Musa al-Khwärizmi'nin yaptığı çalışmalarda ortaya konmuştur. 12. yüzyılda bu çalışmalar Latince'ye çevrilirken, çalışmaların sahibi olan al-Kharizmi'nin adından ötürü yaptığı bu çalışma “*algorithm*” olarak çevrilmiştir. Bu kelime Türkçe'ye ise algoritma olarak girmiştir.
- Tarihçesinden de görüleceği üzere algoritma, bilgisayar dünyasına girmeden önce, matematik alanındaki problemlerin çözümü için kullanılmaktaydı. Daha sonra bilgisayarların geliştirilmesiyle bu alandaki problemlerin çözümünde de kullanılmaya başladı.

# Algoritma Kavramı

- *Algoritma, en basit ifadeyle, bir problemi çözmek için takip edilecek sonlu sayıda adımdan oluşan bir çözüm yoludur.*  
Diğer bir ifadeyle algoritma, bir problemin mantıksal çözümünün adım adım nasıl gerçekleştirileceğinin sözlü ifadesidir.
- Algoritma ile oluşturulan çözümler sözel olarak ifade edildiğinden daha standart herkesin gördüğünde ortak olarak aynı sonucu çıkarabileceği hale getirmek için akış diyagramları kullanılır. Akış diyagramları sembollerden oluşmaktadır. Her sembolün belli bir işlevi vardır.
- Algoritması oluşturulmuş bir problemin bilgisayar ortamına aktarılmış haline **program** denir.
- **Program, problemin çözümünde yapılması gereken işlemler bütünüdür kod karşılığıdır.**
- Algoritmaların program haline getirilmesi için programlama dilleri kullanılır.
- Programlama dilleri kullanılarak yazılımlar geliştirilir.

# Algoritma Kavramı

- Algoritmanın temel özellikleri şunlardır:

- *1. Kesinlik:*

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

- *2. Sıralı Olma:*

Her algoritma için bir başlangıç durumu söz konusudur. Çözüm, bu başlangıç durumu gözönünde bulundurularak gerçekleştirilir. Adımların hangi sırada gerçekleştirileceği çok önemlidir ve net bir şekilde belirtilmelidir.

- *3. Sonluluk:*

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

# Algoritma Kavramı

- Şimdi algoritmanın tanımını ve özelliklerini günlük yaşamdan basit bir örnekle pekiştirelim. Diyelim ki araç trafiği olan bir yolda karşıya geçmek istiyoruz. Bu durumda çözmemiz gereken problem (buna yapılması gereken iş de diyebiliriz) karşıya geçmektir. O zaman bu problemin çözümü için bir yol bulmamız gerekiyor.
- Şöyle bir ifadeye ne dersiniz?
- Önce araba var mı kontrol et, ardından yürü!
- Bu ifade özünde doğrudur. Ancak yeterince açık değildir. Bunu hayatında ilk defa karşıdan karşıya geçecek birine söylersek, kim bilir nasıl anlar?
- Yolun kenarına park etmiş araba var mı? Evet var. O zaman kaldırımdan yürüyeyim.
- Yolun kenarına park etmiş araba var mı? Hayır yok. O zaman yolun ortasından yola paralel yürüyeyim.
- Sol taraftan gelen araba var mı? Hayır yok. O zaman sola bakarak karşıya yürüyeyim
- Bu böylece sürüp gider.

# Algoritma Kavramı

- Evet, biz yetişkin ve eğitimli insanlar "Önce araba var mı kontrol et, ardından yürü!" ifadesinden verilmek istenen mesajı açıkça alırız. Ancak bilgisayarlar öyle değildir. Onları uzaydan gelmiş yaratıklar gibi düşünebiliriz. Hiçbir şey bilmezler. Ama onlara detaylı olarak verdiğimiz bütün emirleri yerine getirebilirler.
- İyi tarif edersek, herşeyi hızlıca anlayıp kolayca uygulayabilirler.
- Şimdi problemi daha net ve kesin ifadelerle çözmeye çalışalım. Ama nereden başlayacağız? Evde miyiz? Trafik ışıklarının yanında mıyız? Yaya geçidinin önünde miyiz? Bu gibi durumlar önemlidir.
- Bu örnekte yolun kenarındayız ve trafik ışığı yok. O zaman şöyle yapalım: Önce yürüyalım, sonra sola ve sağa bakalım (tabii eğer ezilmeden karşıya geçebildiysek). Olur mu? İşimiz şansa kalır. Eğer araba yoksa olur. Araba varsa ezilme ihtimalimiz var.
- O zaman adımları gerçekleştirme sırası da önemli. Yolun kenarından başlayıp, önce sola bakmalı, gelen araba yoksa ya da karşıya geçebileceğimiz kadar uzak bir mesafedeyse sağa bakılmalı. Sağ tarafta da araç yoksa ya da gelen araç yeterince uzak mesafedeyse karşıya yürünmeli.

# Algoritma Kavramı

- Şimdi biraz daha iyi bir çözüm yolu bulmuş olduk. Peki, şuna ne dersiniz?
- Bulduğumuz tarafta, araç trafiği sağdan aktığı için, önce sola bakalım. Sonra da sağ taraftan araba geliyor mu diye sağa bakalım. Acaba biz sağa bakarken soldan gelen bir araba olmuş mudur diye şimdi tekrar sola bakalım. Bu arada ya sağdan bir araba geliyorsa! Tekrar sağa mı baksak. Ya soldan araba geliyorsa..
- Gördüğünüz gibi yukarıdaki gibi düşünürsek sonsuz bir döngüye gireriz. Ya da araba yoksa, yürü!" ifadesi gereği sonsuza kadar yürüyecek miyiz? Bu yüzden algoritmalar sonlu sayıda adımdan ve bir bitiş durumundan oluşmalıdır. Yoksa zaten problem çözülmüş olmaz. Yolun bu tarafında kala kalırız. Veya sonsuza kadar yürümeye devam ederiz.

# Problem Çözmek

- Problem çözmeye iki temel yöntem vardır:
  - Deneysel, deneyimsel ya da deneme yanılma yöntemi
  - Algoritma geliştirmek

Problemi çözmek için çözüm yolu (algoritma) geliştirmenin temel adımları şöyledir

1. **Problemi Tanımlamak:** Algoritmanın amacı belirli bir problemi çözmektir. Bu nedenle algoritma geliştirmenin esas ögesi problemdir. Problemi ne kadar iyi anlarsak, algoritmayı geliştirmemiz o kadar kolay olur. Eğer problemi iyi anlayamazsak, algoritma geliştirme aşamasında ciddi sıkıntılar yaşar, tekrar tekrar problemi tanımlama aşamasına geri döneriz. Daha da kötüsü problemi yanlış anlarsak, bizi beklenmeyen bir sonuca götüren bir algoritma yazma ihtimalimiz söz konusu olacaktır.
2. **Girdi ve Çıktıları Belirlemek:** Problemi iyi tanımlamak için başlangıç ve bitiş noktalarını çok net belirlememiz gerekir. Bizim bulacağımız şey, problemin çözüm yoludur. Ama problem çözüldüğünde ortaya çıkacak şeyi, problem içerisindeki parametreleri bilmeliyiz ki algoritmamızı geliştirelim. Bunun için algoritmanın girdilerini ve çıktılarını iyice kavramalıyız.

# Problem Çözmek

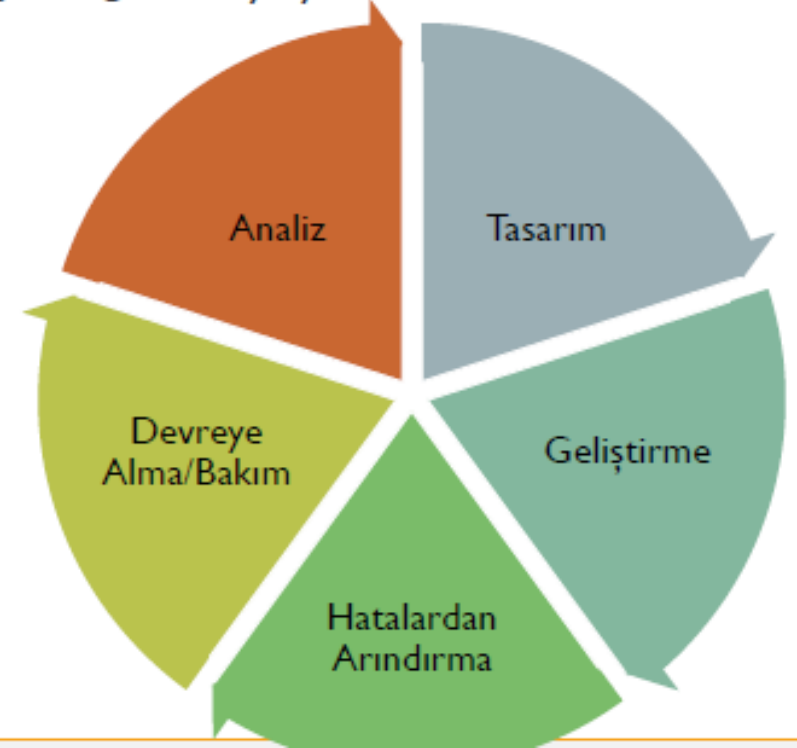
- 3. Çözüm Yolları (Algoritmalar) Geliştirmek:** Bir problemin çözümü için çoğunlukla birden fazla seçeneğimiz olur. İçinde bulunduğumuz duruma göre bazen zaman sıkışıklığından ilk bulduğumuz çözüm yolunu uygulamak durumunda kalırız. Ama eğer yeterince vakit varsa, en iyi çözüm yolunu (algoritmayı) bulmaya çalışmalıyız. Bunun için de bulabildiğimiz kadar çok çözüm yolu geliştirip, bunların içinden en uygununu tercih etmeliyiz. Çözüm yolları geliştirirken her bir çözüm yolu için çözümü adımlara ayırıştırıp, daha sonra da bu adımları uygun şekilde birbirleriyle ilişkilendirmeliyiz.
  - 4. Çözümün Sınanması ve İyileştirilmesi:** Algoritmayı geliştirdikten sonra, henüz kodlamadan kağıt üzerinde nasıl çalışacağını sınamalıyız. Bunu yaptığımızda eğer algoritmada bir eksiklik ya da hata çıkarsa, bunu düzeltmeli ve tekrar sınamalıyız. Sınama aşamasında eğer bellek ya da işlemci kullanımıyla ilgili bir iyileştirme fırsatı yakaladıysak, gerekli iyileştirmeleri de yaparak algoritmamızı olgunlaştırmalıyız.
- Aslında algoritma geliştirme için gerekli adımlar 4 numaralı maddede biter. Ancak bilgisayar programları için algoritmalar geliştirdiğimiz zaman iki maddeye daha ihtiyacımız vardır.

# Problem Çözmek

5. **Algoritmanın Kodlanması:** Geliştirilen algoritma belirli bir programlama dilinde kodlanır. Böylece kağıt üzerindeki çözümümüz bilgisayar üzerinde çalışabilecek hale gelmiş olur. Algoritmayı kodlarken kullanılan programlama dili ve platformunun özellikleri de göz önünde bulundurularak kodun doğruluğu ve performanslı oluşu sağlanır.
6. **Kodun Sınanması ve İyileştirilmesi:** Yazılan kod da algoritmada olduğu gibi sınılanır. Tabi bu sefer sınama bilgisayar üzerinden kod çalıştırılarak gerçekleştirilir. Bu sinama sırasında ortaya çıkan hatalar ve performans sorunları giderilerek program iyileştirilir.

# Yazılım Geliştirme Süreci

- Algoritmalar, matematik biliminden bilgisayar bilimine miras yoluyla geçmiş problem çözme yöntemleridir. Geliştirilen tüm yazılımlar, ya müşterinin bir problemini çözmektedir ya da mevcut bir ihtiyacını karşılar. Her iki durum için de geliştirilen yazılım bir gerçek yaşam problemini çözdüğünü söyleyebiliriz.
- Peki, bir yazılımı geliştirmek için neler yapmak gerekir?
- Bir yazılımı geliştirmek temel olarak şu adımları gerektirir:
  1. **Analiz:** Analiz aşaması, gereksinimlerin belirlendiği, bu gereksinimlerin çözümlendiği ve çerçeveselendiği aşamadır. Bu aşamada yazılımın ne yapacağı, hangi ihtiyacı karşılayacağı, hangi problemi çözeceği belirlenir.
  2. **Tasarım:** Analizle belirlenen yazılımın en uygun şekilde nasıl gerçekleştirilebileceğinin belirlenmesidir. Belirlenen gereksinimlere ve koşullara bakılarak hangi programlama dili, teknoloji, mimari, araç vb. kullanılacağı, çözümün planı, modeli, mimarisi tasarlanır.



# Yazılım Geliştirme Süreci

3. **Geliştirme:** Bir önceki aşamada belirlenmiş olan tasarım, artık hayata geçirilmeye, yazılım geliştirilmeye başlanır. Kodlama bu aşamada yapılır. Bu aşamada, kodlamayla birlikte veritabanı geliştirme, arayüz tasarımı, çeşitli konfigürasyonlar ve dokümantasyonlar da yapılmaktadır.
4. **Hatalardan Arındırma:** Geliştirme aşamasında ortaya çıkan arayüz, kod, veritabanı, doküman gibi ürünlerin istenilen seye uygun olup olmadığı test edilir. Eğer yazılımın çeşitli noktalarında hatalar bulunursa, bu hatalar düzeltilerek yeniden test edilir. Böylece yazılım hatalardan mümkün olduğunca arındırılana kadar bu işlem devam eder.
5. **Devreye Alma ve Bakım:** Bu aşamada, hatalardan arındırılmış yazılım kullanılacağı alana kurulur. Kullanıcıya gerekli eğitim verilir ve bir süre yazılımı kullanmasında destek olunur.

# Programlamaya İlişkin Kavramlar

- Kaynak Kod

- Bir programlama diliyle yazılmış metinlere kaynak kod (source code) denir. Kaynak kod dosyalarının uzantıları kullanılan programlama diline göre değişir. Örneğin;

- Java için .java

- C++ için .cpp

- Visual Basic için .vb

- C# için .cs

- Bir kaynak kod dosyasını Notepad veya Wordpad gibi herhangi bir metin düzenleyici programla açabiliriz.

- Kaynak kodlar, bilgisayarlar üzerinde direkt olarak çalıştırılmazlar.

- Kod Düzenleyici

- Herhangi bir programlama dilinde program yazmak için, Notepad bile kullanılabilir. Ancak geliştirdiğimiz kodla ilgili ipuçları vermesi, hatalarımızı bularak bize göstermesi, hatta bazı hatalarımızı otomatik olarak düzeltmesi nedeniyle, ilgili programlama diline özel yazılmış kod düzenleyicileri kullanırız.

# Programlamaya İlişkin Kavramlar

- Amaç Program

- Kaynak kodlar, insan tarafından anlaşılabilen ve insan tarafından oluşturulan program dosyalarıdır. Bu dosyaların bilgisayarlar tarafından anlaşılabilmesi için özel bir işlemden geçmesi ve sonrasında bilgisayarın anlayacağı makine diline çevrilmesi gerekir. İşte makine diline çevrilmiş ve bilgisayar üzerinde çalıştırılabilir olan bu programa amaç program denir.

- Derleyici

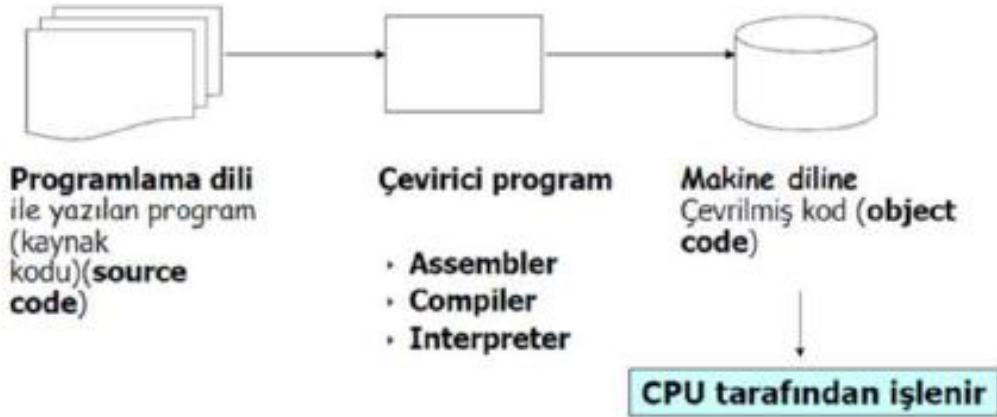
- Herhangi bir programlama diliyle yazılmış olan kaynak kodu, makine diline yani amaç programa dönüştüren özel programlara derleyici (compiler) adı verilir. Derleyicilerin kaynak kodu amaç programa dönüştürmeleri işlemine de derleme (compile) denilmektedir.

- Yorumlayıcı

- Kaynak kodunu satır satır, komut komut derleyerek makine diline çeviren ve çalıştıran programlara yorumlayıcı (intepreter) adı verilmektedir. Yorumlayıcının amacı; programcının yazdığı programı satır satır işleterek, çalışmasını izlemesini ve varsa hatalarını bularak düzeltmesini sağlamaktır.

# Programlama Dillerinin Gelişimi

## • Dilden dile çevrim



# Algoritmanın Yazılım Geliştirme Sürecindeki Yeri

- Algoritmalar, yazılım geliştirme sürecinde, programlamayla tasarım arasında bir yerde kalırlar. Ancak programlama yani geliştirme sürecine daha yakındırlar. Büyük projelerde bazen yazılım mimarları karmaşık bir işin çözümü için önceden çalışarak çözümü bulur ve bunun algoritmasını oluştururlar. Daha sonra da yazılımcı bu algoritmayı alarak programlar. Orta ve küçük çaplı projelerdeyse, yazılımcı kendi algoritmasını kendi belirler ve programını buna göre yazar.
- Günümüzde artık standart iş yazılımları geliştirilirken yapılan işler için algoritma yazmadan direkt programlama yoluna gidilmektedir. Bu tip projelerde ancak vergi hesaplama, maaş hesaplama, prim hesaplama, nöbet çizelgeleme vb. karmaşık durumları çözümlmek için algoritmaya başvurulmaktadır.

# Algoritmanın Yazılım Geliştirme Sürecindeki Yeri

- Sonuç olarak yazılım geliştirme sürecinde ihtiyacımız olan bilgi alanlarını şöyle tanımlayabiliriz:
  - **Programlama dili:** Yazılım geliştireceksek, en azından bir programlama diline hakim olmamız gerekir.
  - **Yazılım geliştirme arabirimi:** IDE (Integrated Development Environment) olarak da bilinen yazılım geliştirme arabirimi, kod düzenleyici, arayüz tasarlayıcı, kod derleyici ve yorumlayıcıyı bir arada barındıran ve yazılım geliştiricilere hayatı kolaylaştıran bir araçtır. Eğer yazılım geliştireceksek en hızlı şekilde en iyi kodu yazmamızı sağlayacak bir yazılım geliştirme arabirimi bilgisine sahip olmamız gerekmektedir.
  - **Platform:** Yukarıda bahsi geçen iki konuda bilgi sahibiysek, yazılım geliştireceğimiz platformu iyice kavramalı, bu platforma özel programlama bilgilerini edinmeliyiz. Yazılım geliştirme platformları temel olarak masaüstü işletim sistemleri (Windows), internet ve mobil olarak düşünülebilir. Yani VB.NET ile kodlamayı biliyor olabiliriz ancak Web uygulaması geliştiriyorsanız response, request gibi nesnelere de biliyor olmalıyız.

# Algoritmanın Yazılım Geliştirme Sürecindeki Yeri

- **Teknoloji:** Geliştirdiğimiz yazılımın üzerinde çalıştığı teknolojilere hakim olmamız gerekir. Örneğin; XML dosyaları işleyerek sipariş alacak bir yazılım geliştireceksek, XML teknolojisini bilmeliyiz. Eğer bir mail alma/yollama programı yazacaksak, SMTP protokolünü bilmeliyiz.
- **En İyi Pratikler (Best Practices):** Belirli bir teknolojide bir çözüm üretmek istiyorsak, öncelikle bu iş birileri yapmış mı diye bakmakta fayda vardır. Eğer yaptığımız işi daha önceden yapanlar olduysa, onların bunu nasıl çözdüğünü öğrenerek bu şekilde yapmak en doğrusu olacaktır. İşte bu daha önceki pratiklerin en iyilerini bulup bunlardan faydalanmalıyız.
- **İş Bilmek (Know-How):** Yazılım geliştirileceği alana özel bilgilere know-how denir. Yazılımı geliştireceğimiz işi bilmeden o iş için doğru ve kaliteli program yazmamız mümkün değildir.

# Algoritmanın Yazılım Geliştirme Sürecindeki Yeri

- **Algoritma:** Algoritma, geliştirilen yazılım içerisindeki karmaşık bir problemin çözümünü bulmamızı sağlar. Yapılacak işin en iyi nasıl çözüleceğini bulmak için algoritmalardan faydalanırız.
- Örneğin; bir dağıtım aracının uğrayacağı 10 nokta için en uygun rotanın bulunması için algoritma geliştiririz.

*Bu bölümde algoritmanın ne olduğunu, tarihçesini, yazılım geliştirme sürecindeki yerini ve programlamayla ilişkisini gördük. Bundan sonraki bölümlerde problem-çözüm-algoritma-programlama döngüsünü adım adım kavrayarak uygulamalarla pekiştireceğiz.*

# Algoritma-Örnek

- Algoritma ile oluşturulacak çözümler sözel olarak ifade edilir. Örneğin sabah kalktığımızda kahvaltı yapılacağı zaman kahvaltı hazırlama algoritması oluşturulursa:
  - Yataktan kalk
  - Mutfığa git
  - Ekmek al
  - Çayı hazırla
  - Dolaptan kahvaltılıkları çıkar
  - Bardağın bitince çayını doldur
  - Karnın doyunca sofradan kalk
  - Kahvaltılıkları dolaba koy
  - Sofrayı temizle
- Adım 1:Yataktan kalk  
Adım 2:Mutfığa git  
Adım 3:Ekmek al  
Adım 4:Çayı hazırla  
Adım 5:Dolaptan kahvaltılıkları çıkar  
Adım 6:Bardağın bitince çayını doldur  
Adım 7:Karnın doyunca sofradan kalk  
Adım 8:Kahvaltılıkları dolaba koy  
Adım 9:Sofrayı temizle

# Algoritmaların Sınıflandırılması

- Algoritmalar karmaşıklık yapılarına göre 3 grupta incelenirler.

Algoritmalar

Basit (Linear) Algoritmalar

Mantıksal Algoritmalar

Döngüsel Algoritmalar

# Basit (Linear) Algoritmalar

- İçerisinde mantıksal ifadelerin yer almadığı, program akış dallanmalarının olmadığı algoritmalar. Bu algoritmalarda akış düz bir halde baştan sona doğru olacaktır. Çoğunlukla küçük hesaplamaları gerçekleştirmek için kullanılırlar. Bir önceki algoritmaya bakılırsa bir karar yapısının olmadığı görülmektedir.

Örnek:

- Adım 1: Hesaplanacak kilometre uzunluğunuz giriniz;  $km$
- Adım 2: Girilen değeri 1000 ile çarpınız;  $m=km*1000$
- Adım 3: Hesaplanan değeri ekrana yazdırınız;  $m$

# Basit (Linear) Algoritmalar

Örnek: Dışarıdan girilen üç adet sayısının toplamını, çarpımını ve ortalamasını hesaplayan algoritma;

- Adım 1: Üç adet sayı giriniz;  $a, b, c$
- Adım 2: Sayıların toplamını hesaplayınız;  $toplam = a + b + c$
- Adım 3: Sayıların çarpımlarını hesaplayınız;  $çarpım = a * b * c$
- Adım 4: Sayıların ortalamasını hesaplayınız;  $ort = toplam / 3$
- Adım 5: Sayıların toplamını, çarpımını ve ortalamasını ekrana yazdırınız;  $toplam, çarpım, ort$ .

# Mantıksal Algoritmalar

- Algoritma içerisinde mantıksal karşılaştırmaların bulunduğu yapılardır. Mantıksal karşılaştırmalara göre algoritmanın akışı farklı adımlara geçecektir. Bu şekilde oluşturulan algoritmalara Mantıksal Algoritmalar denir.
- İlk oluşturulan algoritma biraz daha ayrıntılanırsa karar yapılarının ortaya çıktığı görülür.

Örnek:

- Adım 1: Yataktan kalk
- Adım 2: Mutfığa git
- Adım 3: Eğer ekmek yoksa ekmek al
- Adım 4: Çayı hazırla
- Adım 5: Dolaptan kahvaltılıkları çıkar
- Adım 6: Bardağın bitince çayını doldur
- Adım 7: Karnın doyunca sofradan kalk
- Adım 8: Eğer kahvaltılıklar bitmişse bulaşık makinesine koy
- Adım 9: Eğer kahvaltılıklar bitmemişse kahvaltılıkları dolaba koy
- Adım 10: Sofrayı temizle

# Mantıksal Algoritmalar

- Girilen üç adet sayı içinden en büyük sayıyı bulan algoritma yazalım.
- Adım 1: Üç adet sayı giriniz;  $a, b, c$
- Adım 2: En büyük sayı  $a$  olsun;  $eb=a$
- Adım 3: Eğer  $b$  en büyükten büyük ( $b > eb$ ) ise en büyük  $b$  olsun;  $eb=b$
- Adım 4: Eğer  $c$  en büyükten büyük ( $c > eb$ ) ise en büyük  $c$  olsun;  $eb=c$
- Adım 5: En büyük sayıyı ekrana yazdır;  $eb$

# Mantıksal Algoritmalar

- Sayının pozitif, negatif veya sıfır olduğunu bulan algoritma yazalım.
- Adım 1: Sayıyı giriniz; **a**
- Adım 2: Eğer a sayısı sıfırdan büyük ise ekrana 'pozitif' yaz ve adım 5'e git
- Adım 3: Eğer a sayısı sıfırdan küçük ise ekrana 'negatif' yaz ve adım 5'e git
- Adım 4: Eğer a sayısı sıfıra eşit ise ekrana 'sıfır' yaz ve adım 5'e git
- Adım 5: Program bitti.

# Döngüsel Algoritmalar

- Program için geliştirilen algoritmada bir işlem birden fazla tekrar ediyorsa döngülü algoritma yapısı kullanılır.
- Döngüsel algoritmalarda mantıksal karşılaştırma yapısı özel olarak kullanılır.
- Eğer algoritma içerisinde kullanılan mantıksal karşılaştırma işlemi sonucunda programın akışı karşılaştırma yapılan yerden daha ileriki bir adıma değil de daha önceki adıma gidiyorsa bu şekilde oluşturulmuş algoritmalara döngüsel algoritma denir.
- Yani döngüsel algoritmalarda mantıksal karşılaştırma sonucunda program daha önceki adımlara gider.

# Döngüsel Algoritmalar

## Örnek Pasta Yapım Süreci

1. Pastanın yapımı için gerekli malzemeleri hazırla
2. Yağı bir kaba koy
3. Şekerini aynı kaba yağın üzerine koy
4. Yağ ve şekerini çirp
5. Karışımın üzerine yumurtayı kır
6. Tekrar çirp
7. Kıvama geldi mi diye kontrol et
8. a. Kıvamlı ise 9. adıma devam et  
b. Değilse 6. adıma dön.
9. Karışıma un koy
10. Karışıma vanilya, kabartma tozu vb. koy
11. Karışımı Kıvama gelinceye kadar çirp
12. Pastayı Kek kalıbına koy
13. Yeteri kadar ısınan fırına pastayı koy
14. Pişimi diye kontrol et
15. a. Pişmiş ise 16. adıma devam et  
b. Değilse 14. adıma dön
16. Kek'i fırından çıkart
17. Fırını kapat
18. Kek'in ı kapat
19. Kek'in soğumasını bekle
20. Kek'i servis edebilirsin.

# Döngüsel Algoritmalar

- Bir sayının faktöriyelini bulan algoritma yazalım.

Adım 1: faktöriyeli hesaplanacak sayıyı giriniz;  $n$

Adım 2: faktöriyel değerini 1 yap;  $f=1$

Adım 3: indeks değerini 1 yap;  $i=1$

Adım 4: faktöriyel değeri ile indeksi çarp ve hesaplanan değeri faktöriyele yaz;  $f=f \times i$

Adım 5: indeks değerini bir artır;  $i=i+1$

Adım 6: eğer indeks değeri hesaplanacak sayıdan küçük veya eşit ise Adım 4' git

Adım 7: faktöriyel değerini ekrana yaz;  $f$

# 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;  $a$

ÖDEV:

Girilen iki sayının en küçük ortak katını (EKOK) bulan algoritmayı geliştiriniz.

# Sonu



*Dr. Fatih KALEMKUŐ*

# Sorular



*Dr. Fatih KALEMKUŞ*

# TEŞEKKÜRLER

*Dr. Fatih KALEMKUŞ*