BALIKESİR ÜNİVERSİTESİ, MÜHENDİSLİK FAKÜLTESİ
ENDÜSTRİ MÜHENDİSLİĞİ BÖLÜMÜ

2022

ALGORİTMA VE PROGRAMLAMAYA GİRİŞ
Ders Notları

Doç. Dr.
İBRAHİM KÜÇÜKKOÇ

http://ikucukkoc.baun.edu.tr

Balıkesir, 14 Şubat 2022

Balıkesir Üniversitesi, Mühendislik Fakültesi
Endüstri Mühendisliği Bölümü

BIL1202
ALGORİTMA VE PROGRAMLAMAYA GİRİŞ
DERS NOTLARI

Doç. Dr. İbrahim Küçükkoç
Web: ikucukkoc.baun.edu.tr
Email: ikucukkoc@balikesir.edu.tr

Güncelleme Tarihi: 14.02.2022

1

Ders Planı, Değerlendirme Kriterleri, Yararlanılacak Kaynaklar, İçerik

DERSLE İLGİLİ BİLGİLER

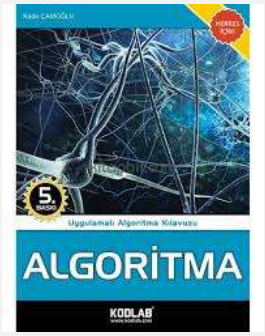
- BIL1202 Algoritma ve Programlamaya Giriş
- Dersin Amacı:
  - Öğrencilerin temel algoritma ve problem çözme yapıları hakkında bilgi sahibi olmasını ve bir problemle karşılaştıklarında bu temel yapıları kullanarak algoritmalar tasarlayabilme ve bu algoritmaları yapısal bir programlama dili ile gerçekleştirebilme yeteneğini kazanmasını sağlamaktır.
- Dersin Web Sayfası: http://ikucukkoc.baun.edu.tr/lectures/BIL1202
  - Duyurular ve ders notlarının paylaşımı bu web sayfasından yapılacaktır. Derse gelmeden önce kontrol edilmeli ve gerekirse çıktı/fotokopi alınarak gelinmelidir.
  - Ayrıca, Öğrenci Bilgi Sistemi üzerinden paylaşılan duyurulardan haberdar olabilmek için OBS'deki email adresinizin doğru ve güncel olduğundan emin olunuz.
  - 21-22 Bahar: Dersler TEAMS üzerinden online olarak işlenecektir.

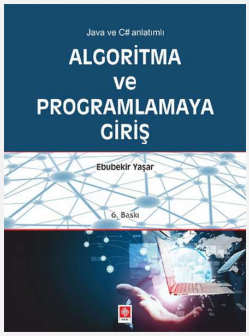
DERSLE İLGİLİ BİLGİLER

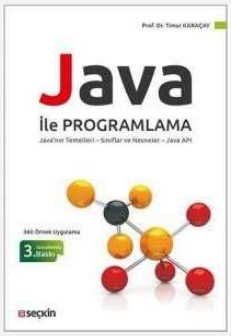
- Değerlendirme:
  - Vize (%40) + Final (%60)
- Derse Katılım:
  - Derslere zamanında gelmeniz gerekmektedir.
  - 5 hafta ve üzeri devamsızlık yapan öğrenciler devamsızlıktan bırakılacak ve final sınavına alınmayacaktır.
  - Derste cep telefonu vb. konuyla alakasız materyallerle ilgilenilmemesi beklenmektedir
- Gerekli Programlar:
  - Java Development Kit - JDK (version 9 veya üzeri)
  - Eclipse IDE for Java Developers (2019 veya üzeri)

DERSLE İLGİLİ BİLGİLER

- Yararlanılacak Kaynaklar:
  - Algoritma: Uygulamalı Algoritma Klavuzu, 5. Baskı, Kadir Çamoğlu, KODLAB, 2011
  - Algoritma Geliştirme ve Programlamaya Giriş, 13. Baskı, Fahri Vatansever, Seçkin Yayıncılık, 2017
  - Algoritma ve Programlamaya Giriş, 6. Baskı, Ebubekir Yaşar, Ekin Basım Yayın, 2016
  - Java ile Programlama, 3. Baskı, Timur Karaçay, Seçkin Yayıncılık, 2016







DERSLE İLGİLİ BİLGİLER

- İÇERİK

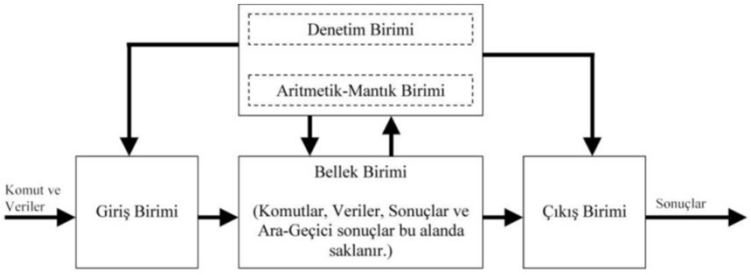
1	Giriş, değerlendirme kriterleri, yararlanılacak kaynaklar, ders planı, temel kavramlar
2	Algoritmaların sınıflandırılması, algoritma geliştirmek
3	Satır kod, sayaç yapıları, koşul/karar durumları
4	Akış diyagramı ve çoklu koşul yapıları
5	Sözde kod, satır algoritmalardan ve akış diyagramlarından sözde kod oluşturma
6	Temel algoritma örnekleri, genel uygulamalar
7	Akış diyagramlarından kodlamaya geçiş, Java programlama dili ve özellikleri
8	JAVA ile Programlamaya giriş, değişkenler
9	Basit koşul yapıları, If-Else, Switch-Case
10	Veri giriş/çıkış işlemleri
11	Döngüler (For, While, Do-While)
11	Tek boyutlu diziler ve uygulamalar
12	İki ve daha çok boyutlu diziler ve uygulamalar
13	Sıralama algoritmaları (seçme, kabarcık, araya eklemeli, hızlı, buble sort, quick sort, insertion sort vs.,)
14	Java program geliştirme uygulamaları

2

Temel Kavramlar, Algoritma Kavramı ve Önemi, Algoritma Türleri

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-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-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, Phython vs.)
  - Uygulama Yazılımları (MS Word, Excel, Autocad, vs.)

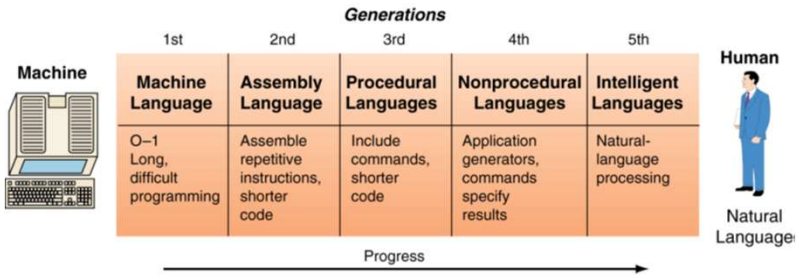
  Yazılım geliştirme sonucu ortaya çıkan ürüne program denir.
  Bir problemin bilgisayar tarafından çözülebilmesi için öncelikle algoritmasının oluşturulması gerekmektedir.

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 tanımlamış olduğu "Analytical Engine" i ile Bernoulli sayılarının hesaplanmasına ilişkin makalesinde olduğu söylenmektedir. Bu nedenle ilk gerçek anlamdaki programlama dillerinden birinin adı Ada Lovelace anısına ADA olarak isimlendirilmiştir.
- 1940'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**dır.
- 16 bit uzunluklu sayılara 1 word luk sayılar 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 Sayilar
- 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** (**A**merican **N**ational **C**ode for **I**nformation **I**nterchange) 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.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	⌂

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	ø
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	░	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	|	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	▉	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	₧	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	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

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

Matematiksel işlemlerin öncelik sırası ?

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

NOT: Bilgisayar diline kodlanmış bir matematiksel ifadede, 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.

İşlem sembolü	Anlamı
=	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;

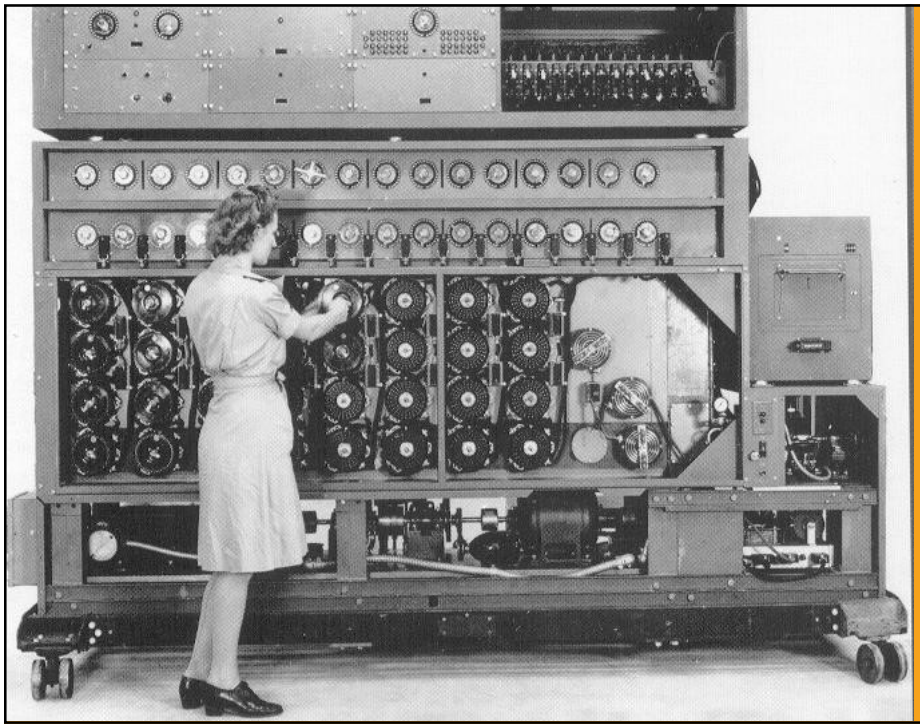
Eğer Yaş>23 VE maaş=asgari ücret ise ismi Yaz

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

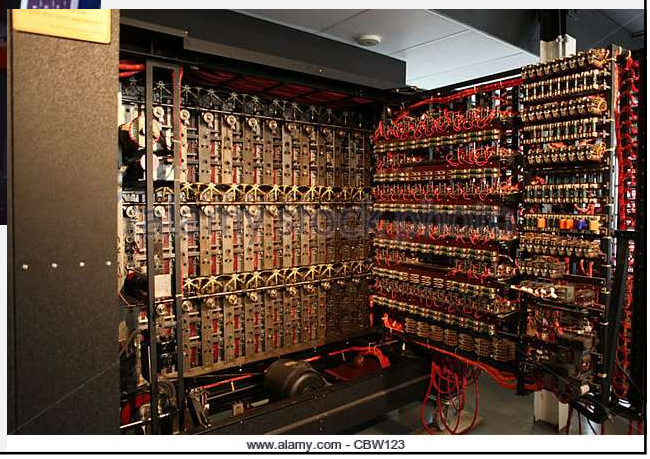
Örnek-2

- 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 (TDili Not>65 veya YDil Not>65) ise ismi Yaz



ALGORİTMA KAVRAMI



Okuyunuz: Alan Turing (https://goo.gl/cqgBzz)

Algoritma Kavramı

- Bu bölümde temel olarak algoritma kavramını işleyeceğiz. Algoritmanin 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 yaptiği 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 başlandı.

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 bütününün 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 admlar herkes tarafından aynı şekilde anlaşılabiliyor olmalı, farklı anlamlara gelebilecek bulanık ifadeler içermemelidir.
  - *2. Sıralı Olma:*
    Her algoritma için bir başlangiç durumu söz konusudur. Çözüm, bu başlangıç durumu gözönünde bulundurularak gereklestirilir. Adimlarin hangi sırada gerçekleştirileceği çok önemlidir ve net bir şekilde belirtilmelidir.
  - *3. Sonluluk:*
    Algoritma sonlu sayıda adımdan oluşmali, sinirli 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 mesaji açıkça alırız. Ancak bilgisayarlar öyle değildir. Onları uzaydan gelmiş yaratıklar gibi düşünebiliriz. Hiçbir şey bilmezler. Ama onlara detaylı olarak verdiğimiz bütün emirleri yerine getirebilirler.
- Iyi tarif edersek, herşeyi hizlica 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 işığı yok. O zaman şöyle yapalım: Önce yürüyelim, sonra sola ve sağa bakalım (tabi eğer ezilmeden karşıya geçebildiysek). Olur mu? Işimiz 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?
- Bulunduğumuz tarafta, araç trafiği sağdan aktığı için, önce sola bakalim. 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 çözmede iki temel yöntem vardir:
  - 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 öğesi problemdir. Problemi ne kadar iyi anlarsak, algoritmayı geliştirmemiz o kadar kolay olur. Eğer problemi iyi anlayamazsak, algoritma geliştirme aşamasında ciddi sikıntılar 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 başlangiç 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 algoritmanin 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 (algoritmayi) bulmaya çalışmalıyız. Bunun için de bulabildiğimiz kadar çok çözüm yolu gelistirip, bunların içinden en uygununu tercih etmeliyiz. Cözüm yolları geliştirirken her bir çözüm yolu için çözümü adımlara ayrıştırıp, 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 sınanır. Tabi bu sefer sınama bilgisayar üzerinden kod çalıştırılarak gerçekleştirilir. Bu sınama 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 yazilım 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 çerçevelendiği aşamadır. Bu aşamada yazılımin 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 yapilır. 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.

Programlamayla İlişkili Kavramlar

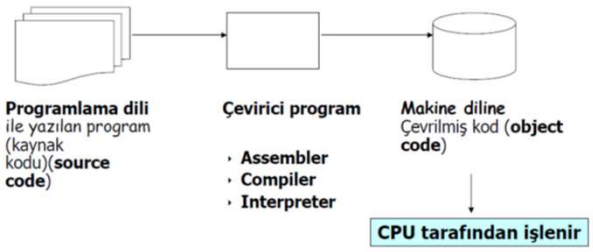
- Kaynak Kod
  - Bir programlama diliyle yazılmış metinlere kaynak kod (source code) denir. Kaynak kod dosyalarinin uzantiları kullanılan programlama diline gene değişir. Örneğin;
    - Java için .java
    - C++ için .cp
    - Visual Basic için .vb
    - C# i in .cs
  - Bir kaynak kod dosyasini Notepad veya Wordpad gibi herhangi bir metin düzenleyici programla açabiliriz.
  - Kaynak kodlar, bilgisayarlar üzerinde direkt olarak çalıştırılamazlar.
- 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.

Programlamayla İlişkili Kavramlar

- Amaç Program
  - Kaynak kodlar, insan tarafından anlaşılabilen ve insan tarafindan oluşturulan program dosyalarıdır. Bu dosyaların bilgisayarlar tarafından anlaşılabilmesi için özel bir işlemden geçmesi ve sonrasında bilgisayarın anlayacağı makine diline çevrilmesi gerekir. Işte 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 amaci; 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 arasinda 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, yazılımci kendi algoritmasını kendi belirler ve programını buna göre yazar.
- Günümüzde artık standart iş yazılımları geliştirilirken yapilan işler için algoritma yazmadan direkt programlama yoluna gidilmektedir. Bu tip projelerde ancak vergi hesaplama, maas hesaplama, prim hesaplama, nöbet çizelgeleme vb. karmaşık durumları çözümlemek için algoritmaya başvurulmaktadır.

Algoritmanın Yazılım Geliştirme Sürecindeki Yeri

- Sonuç olarak yazılım gelistirme 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 yazilim geliştireceksek en hizlı şekilde en iyi kodu yazmamızı sağlayacak bir yazılım geliştirme arabirimi bilgisine sahip olmamız gerekmektedir.
  - Platform: Yukarıda bahsi geçen iki konuda bilgi sahibiysek, yazilım 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 uygulamasi geliştiriyorsanız response, request gibi nesneleri 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 gelitireceksek, XML teknolojisini bilmeliyiz. Eger bir mail alma/yollama programı yazacaksak, SMTP protokolünü bilmeliyiz.
- En Iyi Pratikler (Best Practices): Belirli bir teknolojide bir çözüm üretmek istiyorsak, öncelikle bu iş birileri yapmiş mı diye bakmakta fayda vardir. Eger yaptığımız işi daha önceden yapanlar olduysa, onların bunu nasıl çözdüğünü öğrenerek bu sekilde yapmak en dogrusu 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 oldugunu, tarihçesini, yazilim gelistirme sürecindeki yerini ve programlamayla ilişkisini gördük. Bundan sonraki bölümlerde problem-çözüm-algoritma-programlama döngüsünü adım adım kavrayarak uygulamalarla pekistireceğiz.

Algoritma - Örnek

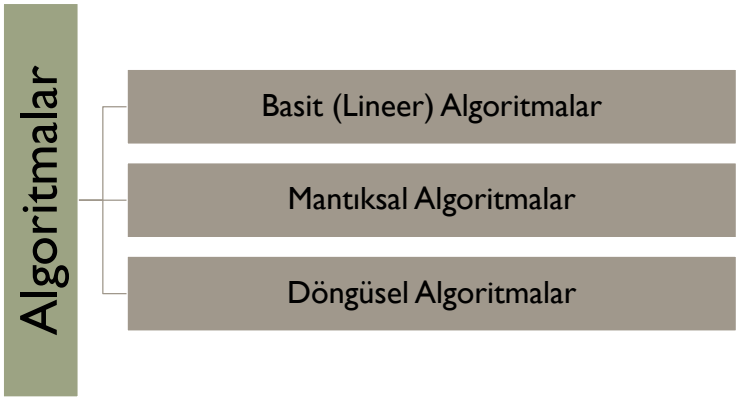
- Algoritma ile oluşturulacak çözümler sözel olarak ifade edilir. Örneğin sabah kalktığımızda kahvaltı yapılacağı zaman kahvaltı hazırlama algoritması oluşturulursa:

- Yataktan kalk
- Mutfağa 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:Mutfağa 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.



Basit (Lineer Algoritmalar)

- İçerisinde mantıksal ifadelerin yer almadığı, program akış dallanmalarının olmadığı algoritmalardır. 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 uzunluğunu al; km
- Adım 2: Girilen değeri 1000 ile çarp; m=km*1000
- Adım 3: Hesaplanan değeri ekrana yazdır; m

Basit (Lineer Algoritmalar)

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

- Adım 1: Başla
- Adım 2: Üç adet sayı al; a, b, c
- Adım 3: Sayıların toplamını hesapla; toplam=a+b+c
- Adım 4: Sayıların çarpımlarını hesapla; çarpım=a*b*c
- Adım 5: Sayıların ortalamasını hesapla; ort=toplam/3
- Adım 6: Sayıların toplamını, çarpımını ve ortalamasını ekrana yazdır; toplam, çarpım, ort.
- Adım 7: Dur

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: Başla
  - Adım 2: Yataktan kalk
  - Adım 3: Mutfağa git
  - Adım 4: Eğer ekmek yoksa ekmek al
  - Adım 5: Çayı hazırla
  - Adım 6: Dolaptan kahvaltılıkları çıkar
  - Adım 7: Bardağın bitince çayını doldur
  - Adım 8: Karnın doyunca sofradan kalk
  - Adım 9: Eğer kahvaltılıklar bitmişse bulaşık makinesine koy
  - Adım 10: Eğer kahvaltılıklar bitmemişse kahvaltılıkları dolaba koy
  - Adım 11: Sofrayı temizle
  - Adım 12: Dur

Mantıksal Algoritmalar

- Girilen üç adet sayı içinden en büyük sayıyı bulan algoritmayı yazalım.
- Adım 1: Başla
- Adım 2: Üç adet sayı al; a,b,c
- Adım 3: En büyük sayı a olsun; eb=a
- Adım 4: Eğer b en büyükten büyük (b>eb) ise en büyük b olsun; eb=b
- Adım 5: Eğer c en büyükten büyük (c>eb) ise en büyük c olsun; eb=c
- Adım 6: En büyük sayıyı ekrana yazdır; eb
- Adım 7: Dur

Mantıksal Algoritmalar

- Girilen üç adet sayı içinden en büyük sayıyı bulan algoritmayı yazalım.
- Adım 1: Başla
- Adım 2: Üç adet sayı al; a,b,c
- Adım 3: Eğer a>b ve a>c ise eb=a
- Adım 4: Eğer b>c ve b>a ise eb=b
- Adım 5: Eğer c>a ve c>b ise eb=c
- Adım 6: En büyük sayıyı ekrana yazdır, eb
- Adım 7: Dur

- Karşılaştırmalar sadece ikili olarak yapılabilir.

Mantıksal Algoritmalar – Örnek 2

- Girilen sayının pozitif, negatif veya sıfır olduğunu bulan algoritma yazalım.
- Adım 1: Başla
- Adım 2: Sayıyı gir; a
- Adım 3: Eğer a sayısı sıfırdan büyük ise ekrana 'pozitif' yaz ve adım 6'ya git
- Adım 4: Eğer a sayısı sıfırdan küçük ise ekrana 'negatif' yaz ve adım 6'ya git
- Adım 5: Eğer a sayısı sıfıra eşit ise ekrana 'sıfır' yaz ve adım 6'ya git
- Adım 6: Dur

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

Örnek Pasta Yapım Süreci

1. Pastanın yapımı için gerekli malzemeleri hazırla
2. Yağı bir kaba koy
3. Şekeri aynı kaba yağın üzerine koy
4. Yağ ve şekeri çırp
5. Karışımın üzerine yumurtayı kır
6. Tekrar çırp
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 çırp
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. Keki fırından çıkart
17. Fırını kapat
18. Kekin ı kapat
19. Kekin soğumasını bekle
20. Keki servis edebilirsin.

Döngüsel Algoritmalar – Örnek

- Bir sayının faktöriyelini bulan algoritma yazalım.
- Adım 1: Başla
- Adım 2: Faktöriyeli hesaplanacak sayıyı al; n
- Adım 3: Faktöriyel değerini 1 yap; f=1
- Adım 4: İndeks değerini 1 yap; i=1
- Adım 5: Faktöriyel değeri ile indeksi çarp ve hesaplanan değeri faktöriyele yaz; f=f x i
- Adım 6: İndeks değerini 1 artır; i=i+1
- Adım 7: Eğer indeks değeri n'den küçük veya eşitse (i<=n) Git Adım 5
- Adım 8: Dur

Döngüsel Algoritmalar – EBOB-1

- Girilen iki sayının en büyük ortak bölenini (EBOB) bulan algoritma yazalım.
- Adım 1: Başla
- Adım 2: Sayıları al; a, b
- Adım 3: Eğer a>b ise b'yi a'dan çıkar ve tekrar a'ya yaz (a=a-b) ve Adım 3'e git
- Adım 4: Eğer b>a ise a'yı b'den çıkar ve tekrar b'ye yaz (b=b-a) ve Adım 3'e git
- Adım 5: a değerini ekrana yaz
- Adım 6: Dur

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

Döngüsel Algoritmalar – EBOB-2

- Girilen iki sayının en büyük ortak bölenini (EBOB) bulan algoritma yazalım.
- Adım 1: Başla
- Adım 2: Sayıları al; a, b
- Adım 3: Eğer a=b ise ebob=a ve Git Adım 9
- Adım 4: Sayac=1
- Adım 5: Eğer a<b ise kucuksayi=a
- Adım 6: Eğer b<a ise kucuksayi=b
- Adım 7: Eğer (a MOD Sayac = 0) VE (b MOD Sayac = 0) ise ebob=Sayac
- Adım 8: Eğer Sayac<kucuksayi ise Sayac=Sayac+1 ve Git Adım 7
- Adım 9: Yaz ebob
- Adım 10: Dur

Döngüsel Algoritmalar – EBOB-3

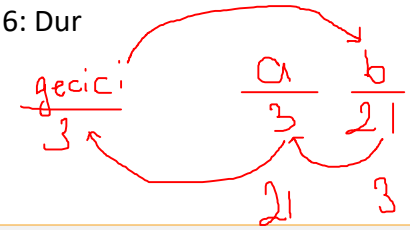
- Girilen iki sayının en büyük ortak bölenini (EBOB) bulan algoritma yazalım.
- Adım 1: Başla
- Adım 2: Sayıları al; a, b
- Adım 3: Eğer a=b ise ebob=a ve Git Adım 6
- Adım 4: Eğer a<b ise sayac=a Değilse sayac=b
- Adım 5: Eğer (a MOD Sayac = 0) VE (b MOD Sayac = 0) ise ebob=sayac; değilse sayac=sayac-1 ve Git Adım 5
- Adım 6: Yaz ebob
- Adım 7: Dur

Döngüsel Algoritmalar – EKOK-1

- Girilen iki sayının en büyük ortak bölenini (EBOB) bulan algoritma yazalım.
- Adım 1: Başla
- Adım 2: Sayıları al; a, b
- Adım 3: carpim=a*b
- Adım 4: Eğer a>b ise b'yi a'dan çıkar ve tekrar a'ya yaz (a=a-b) ve Adım 4'e git
- Adım 5: Eğer b>a ise a'yı b'den çıkar ve tekrar b'ye yaz (b=b-a) ve Adım 4'e git
- Adım 6: En büyük bölen a değerini ekrana yaz; a
- Adım 7: ekok=carpim/a
- Adım 8: Dur

Döngüsel Algoritmalar – EKOK-2

- Adım 1: Başla
- Adım 2: Sayıları al; a,b
- Adım 3: Sayac=0
- Adım 4: Sayac=Sayac+a
- Adım 5: Eğer Sayac MOD b = 0 ise yaz Sayac; değilse Git Adım 4
- Adım 6: Dur



- Adım 1: Başla
- Adım 2: Sayıları al; a,b
- Adım 3: Eğer a<b ise gecici=a, a=b, b=gecici
- Adım 4: Sayac=0
- Adım 5: Sayac=Sayac+a
- Adım 6: Eğer Sayac MOD b = 0 ise yaz Sayac; değilse Git Adım 5
- Adım 7: Dur

Döngüsel Algoritmalar – EKOK-2

- Adım 1: Başla
- Adım 2: Sayıları al; a,b
- Adım 3: Eğer a<b ise gecici=a, a=b, b=gecici
- Adım 4: Eğer a MOD b = 0 ise yaz a ve Git Adım 8
- Adım 5: Sayac=0
- Adım 6: Sayac=Sayac+a
- Adım 7: Eğer Sayac MOD b = 0 ise yaz Sayac; değilse Git Adım 6
- Adım 8: Dur

Döngüsel Algoritmalar – EKOK-3

- Adım 1: Başla
- Adım 2: Sayıları al; a,b
- Adım 3: y=a, z=b
- Adım 4: Eğer a<b ise a=a+y ve Git Adım 4
- Adım 5: Eğer b<a ise b=b+z ve Git Adım 4
- Adım 6: Yaz a
- Adım 7: Dur

3

Algoritma Geliştirmek, Satır Kod

Algoritma Geliştirmek

Bir problem çözmek üzere geliştirilen algoritma üç şekilde yazılabilir:

- Satır algoritma: Problemin çözüm adımları düz metin olarak açık cümlelerle yazılır.
- Akış diyagramı (flow-chart): Problemin çözüm adımları geometrik şekillerle gösterilir.
- Sözde kod (pseudo-code): Problemin çözüm adımları komut benzeri anlaşılır metinlerle veya kısaltmalarla ifade edilir.

Örnek: Klavyeden girilen iki sayıyı toplayıp ekranda gösteren programın algoritmasının üç değişik yöntemle gösterilmesi:

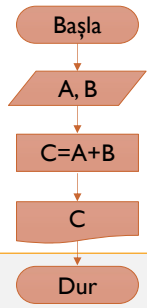
Satır algoritma ile gösterilmesi:

1. Başla
2. Birinci sayıyı (A) klavyeden oku
3. İkinci sayıyı (B) klavyeden oku
4. Girilen sayıları toplayarak sonucu oluştur (C=A+B)
5. Sonucu (C) ekrana yazdır
6. Dur

Sözde kod ile gösterilmesi:

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

Akış diyagramı ile gösterilmesi:



Algoritma Geliştirmek

Algoritmaları geliştirirken değişken, sabit, atama, döngü, karar yapısı, alt yordam gibi bir takım öğeler kullanılır.

Veri: (data): Bilgisayarlarda işlenen tüm bilgiler veri olarak adlandırılırlar. Veriler temel olarak *sayısal ve alfasayısal* olmak üzere ikiye ayrılırlar (Seckin, s.55).

Tanımlayıcı: (identifier): Değişken, sabit, alt yordam, alan gibi programlama birimlerine yazılımcı tarafından verilen isimlerdir.

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

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

Yarıçapı girilen dairenin alanını hesaplayıp yazdıran programın algoritmasını oluşturunuz.

Algoritma Geliştirmek

İsimlendirme Kuralları:

- İngiliz alfabesindeki A-Z veya a-z arası 26 harf kullanılabilir.
- 0-9 arası rakamlar kullanılabilir.
- Simgelerden sadece alt çizgi (_) kullanılabilir.
- Harf veya alt çizgi ile başlayabilir, ancak rakamla başlayamaz veya sadece rakamlardan oluşamaz.
- İlgili programlama dilinin komutu veya saklı/anahtar kelimesi olamaz.

En çok kullanılan standart tanımlayıcı gösterimleri:

- Pascal case: Kelimelerin ilk harfleri büyük. Örn: `AdSoyad, OgrenciNo`
- Camel case: Birinci kelimenin ilk harfi küçük, diğer kelimelerin ilk harfleri büyük.
  Örn: `adSoyad, ogrenciNo`

Algoritma Geliştirmek

Gömülü değer (literal): Kod içine yazılmış olan metinsel, sayısal ya da diğer veri tiplerindeki sabit değerlere gömülü değer denir. `PI=3.14` ifadesindeki `3.14` değeri bir gömülü değerdir. Aynı şekilde `mesaj="merhaba"` ifadesindeki `mesaj` bir değişken veya sabit, `"merhaba"` ise gömülü değerdir.

Aritmetik İşlemler: Programlamadaki en temel işlemlerdir. Aritmetik işlemlerde kullanılan operatörler aşağıdaki gibidir.

İşleç	Adı	Örnek
+	Toplama	C=A+B
-	Çıkartma	D=E-F
*	Çarpma	X=Y*Z
/	Bölme	T=P/S
=	Eşittir	A=B+C

Algoritma Geliştirmek

Mantıksal İşlemler: Bir programın akışı içerisinde belirli bir koşula bağlı olarak akışın hangi yönde ilerleyeceğine karar vermede mantıksal işlemler kullanılır. Bu ifadelerin sonucunda doğru (**`true`**) ya da yanlış (**`false`**) değerleri elde edilir.

Mantıksal işlemlerde kullanılan işleçler aşağıdaki gibidir.

İşleç	Adı	Örnek
>	Büyük	A>B
<	Küçük	A<B
==	Eşit	A==B
<> (veya !=)	Farklı	A<>B veya (A!=B)
>=	Büyük Eşit	A>=B
<=	Küçük Eşit	A<=B

Algoritma Geliştirmek

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

```
1. Başla
2. Öğrencinin numarasını (No) al
3. Öğrencinin adını ve soyadını (AdSoyad) al
4. Öğrencinin vize notunu (VizeNot) al
5. Öğrencinin final notunu (FinalNot) al
6. Ort = 0.4*VizeNot + 0.6*FinalNot
7. Numara (No) ve ortalamayı (Ort) yaz
8. Dur
```

Algoritma Geliştirmek

Sayaç Kullanımı:

Programlarda bazı işlemlerin belirli sayıda yapılması veya işlenen değerlerin sayılması gerekebilir. Örneğin klavyeden girilen bir cümlede ka tane sesli harf olduğunu bulan programda, cümlenin her harfi sırasıyla çağrılır ve sesli harfler kümesine ait olup olmadığı araştırılır. Eğer sıradaki çağrılan harf bu kümeye ait ise, bunları sayacak olan değişkenin değeri artırılır.

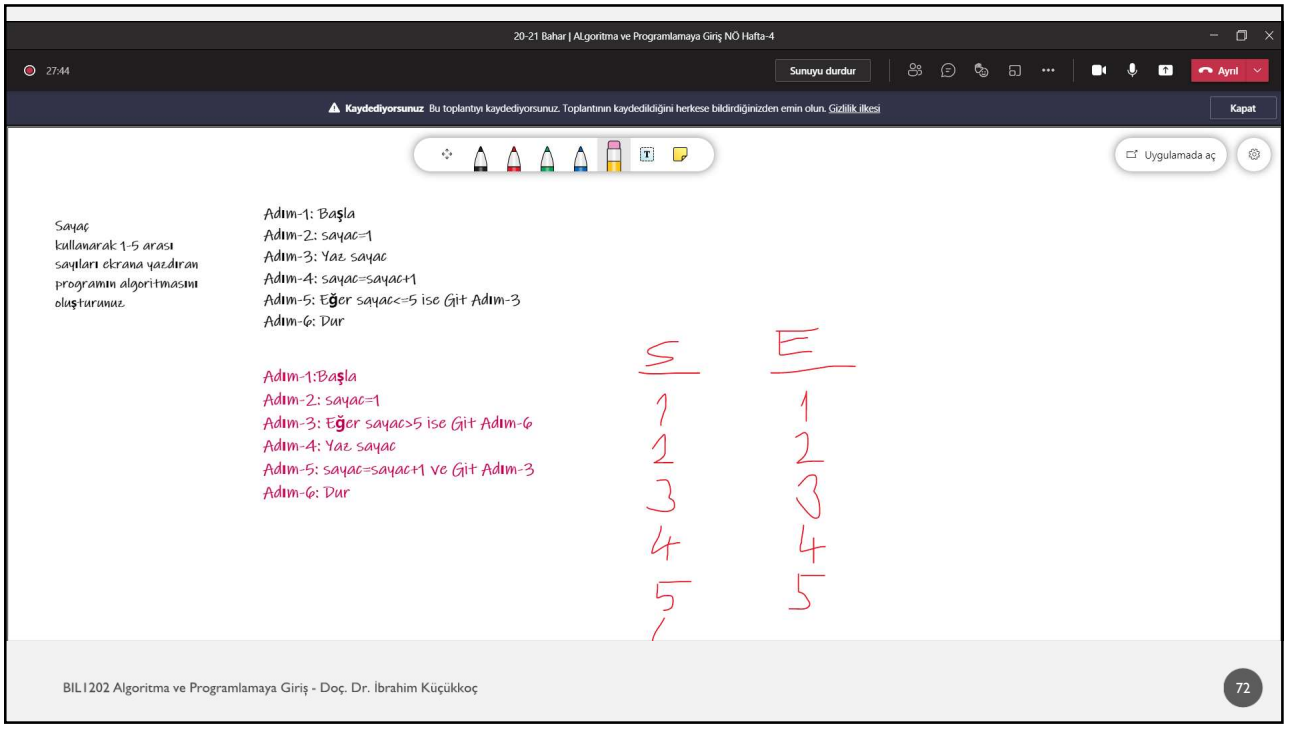
```
sayac=sayac+1
```

Burada sağdaki ifadede değişkenin eski değerinin üzerine `1` eklenerek bulunan sonuç yine değişkenin kendisine yeni değer olarak atanmaktadır.

Sayaç yapısı:

```
sayac=sayac+adım
```

Sayaç kullanarak 1-5 arası sayıları ekrana yazdıran programın algoritmasını oluşturunuz.



Algoritma Geliştirmek

Döngü Kullanımı:

Programlardaki belirli işlem bloklarını (kod parçalarını); aynı veya farklı değerlerle, verilen sayıda gerçekleştiren çevrim yapılarına döngü denir.

Örneğin 1 ile 1000 arasındaki tek sayıları ekrana yazdıracak programda, 1 ile 1000 arasında ikişer ikişer artan bir döngü açılır ve döngü değişkeni ardışık olarak yazdırılır.

Döngü Oluşturma Kuralları:

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

Algoritma Geliştirmek

Aşağıdaki algoritma çalıştırıldığında ekrana ne yazdıracaktır?

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

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

Algoritma Geliştirmek

Ardışık Toplama:

Çalışma prensibi sayacınkine benzer, aynı değerin üzerine yeni değerler eklemek için kullanılır. Genel kullanım şekli:

```
Toplam = Toplam + sayi
```

Toplam değişkenine başlangıç değeri olarak 0 atanır.

Klavyeden girilen 5 adet sayının ortalamasını bulan program:

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


Algoritma Geliştirmek

Adım-1: Başla

Adım-2: Toplam=0

Adım-3: N=25

Adım-3: **DÖNGÜ (i=1 TO N STEP 1)**

Adım-4: Yaz "Bir sayı giriniz:"

Adım-5: Al, sayi

Adım-6: Toplam=Toplam+sayi

Adım-7: **DÖNGÜSONU**

Adım-8: Ort=Toplam/N

Adım-9: Yaz Ort

Adım-10: Dur

Klavyeden girilen 5 adet sayının ortalamasını bulan program:

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

Algoritma Geliştirmek

Adım-1: Başla	Adım-1: Başla
Adım-2: Faktöriyeli hesaplanacak sayıyı al; N	Adım-2: Faktöriyeli hesaplanacak sayıyı al; N
Adım-3: carpim=1	Adım-3: carpim=1
Adım-4: DÖNGÜ (i=2 TO N STEP 1)	Adım-4: DÖNGÜ (i=N TO 2 STEP -1)
Adım-5: carpim=carpim*i	Adım-5: carpim=carpim*i
Adım-6: DÖNGÜSONU	Adım-6: DÖNGÜSONU
Adım-7: Yaz carpim	Adım-7: Yaz carpim
Adım-8: Dur	Adım-8: Dur

Algoritmanın Hazırlanması

- Algoritmadaki adımlar programın sonlu sayıda işlem yapmasını sağlamalıdır.
- Adımlar sıralı ve mantıklı olmalıdır.
- Tekrar eden işlemlerde programın hızı ve etkinliği açısından kaçınılmalıdır.
- Farklı değerlerle gerçekleştirilen aynı işlemleri tekrar tekrar yazmak yerine bunları alt program/fonksiyon olarak oluşturmak daha uygundur.
- Gereksiz işlemlerden kaçınılmalıdır.
- Etkisiz işlemlerden kaçınılmalıdır.
- Bellek verimliliği açısından fazladan veya gereksiz tanımlamalardan kaçınılmalıdır.
- Her algoritmanın bir çıktısı/sonucu olmalıdır.

Algoritma Yazma Kuralları

- Algoritmadaki tüm satırlar 1'den başlayarak numaralandırılmalıdır.
- Bütün algoritmalarda birinci satır "1. Başla" şeklindedir.
- Bütün algoritmalar "Dur" ile biter.
- Algoritmalarda kullanılan "Git" işlem akışı yönlendirme komutu satır numarasıyla birlkte kullanılmalıdır.
- Algoritmalarda kullanılabilecek alt program veya fonksiyonlar, tanımlayıcı isimleri ve parametreleriyle birlikte verilmelidir.
- Algoritmalardaki adımlar, sınırlı sayıda, açık, net ve kesin olmalıdır.
- Algoritmalardaki ifadeler anlaşılır ve mümkün olduğunca sade (az ve öz) olmalıdır.
- Algoritmalardaki ifadeler, herhangi bir programlama diline, donanıma, işletim sistemine vb. bağlı olmamalıdır.

Çalışma Soruları

- Değişken nedir, programlarda değişkenlere neden ihtiyaç duyulmaktadır?
- Girilen üç adet sayıdan en büyüğünü bulan programın algoritmasını hazırlayınız.
- Girilen birbirinden farklı üç sayıyı küçükten büyüğe doğru sıralayan programın algoritmasını hazırlayınız.
- 1-99 arasındaki tek ve çift sayıların toplamları ile çarpımlarını ayrı ayrı hesaplayan programın algoritmasını hazırlayınız.
- Klavyeden girilen pozitif bir tamsayının tek mi çift mi olduğunu tespit eden programın algoritmasını hazırlayınız.
- Klavyeden girilen bir tamsayının pozitif, negatif, veya sıfır olduğunu tespit eden programın algoritmasını hazırlayınız.

4

Akış Diyagramı ve Çoklu Koşul Yapıları

Akış Diyagramı

Başla/Dur: Algoritmayı başlatmak/sonlandırmak için kullanılır.

Başla Dur

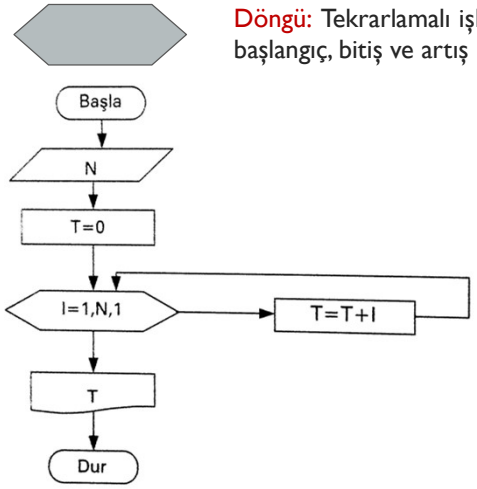
Veri Girişi: Bilgisayara klavyeden veri girişini temsil eden şekildir.

OgrNo A, B, C

İşlem: Programın çalışması sırasında yapılacak işlemleri ifade etmek için kullanılan şekildir.

c=a^2+2ab A=PI*r^2

Akış Diyagramı

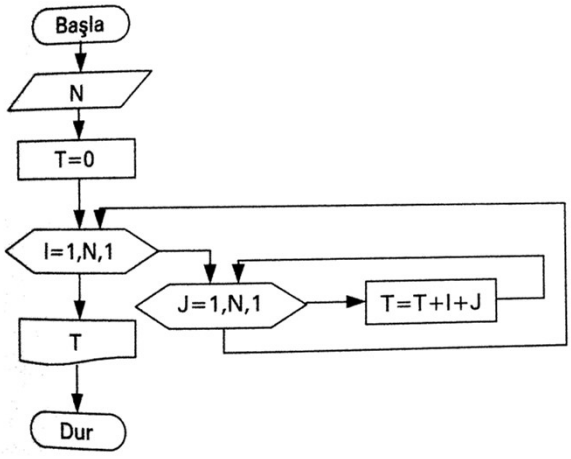


Döngü: Tekrarlamalı işlemler döngü olarak adlandırılır. Şeklin içine döngünün başlangıç, bitiş ve artış (adım) değerleri yazılır.

N	T	I	Ekran
5	0	1	
	1	2	
	3	3	
	6	4	
	10	5	
	15		15

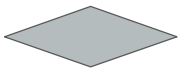
Akış Diyagramı

İç-İçe Döngü Yapısı:

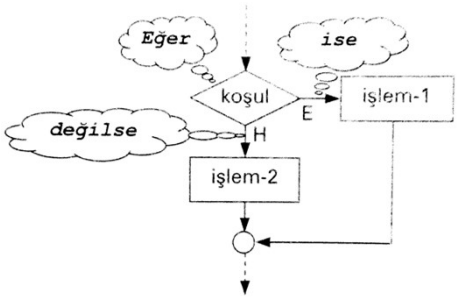


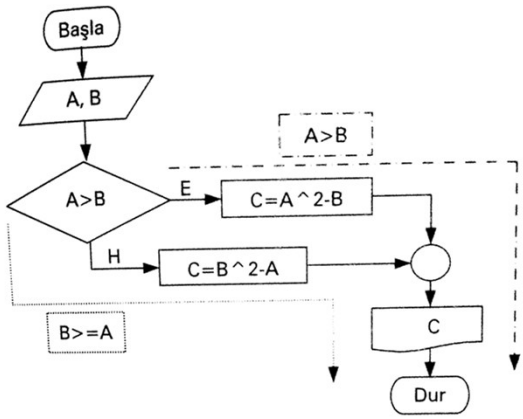
N	T	I	J	Ekran
3	0	1	1	
	2		2	
	5		3	
	9	2	1	
	12		2	
	16		3	
	21	3	1	
	25		2	
	30		3	
	36			36

Akış Diyagramı

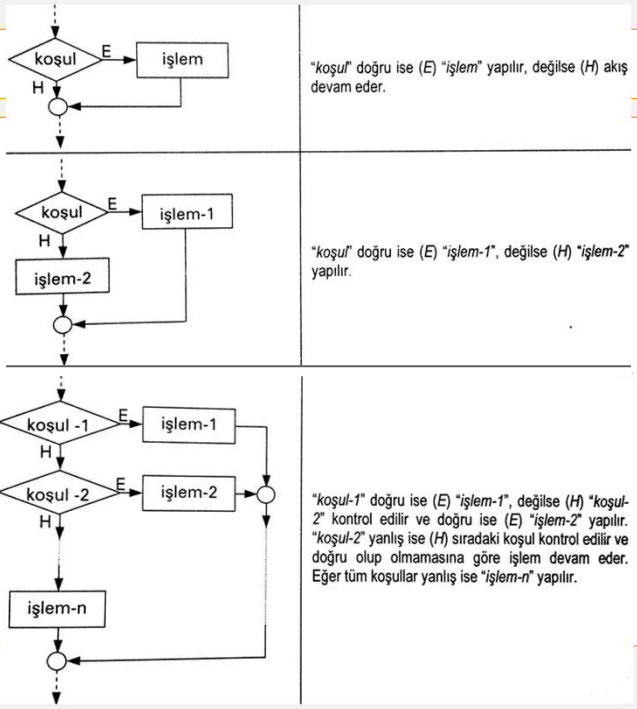


Karar: Karar verme/karşılaştırma işlemlerini temsil eden şekildir. Koşul veya mantıksal operatörlerle bağlı koşullar, şeklin içine yazılır. TRUE, FALSE





Akış Diyagramı



"*koşul*" doğru ise (*E*) "*işlem*" yapılır, değilse (*H*) akış devam eder.

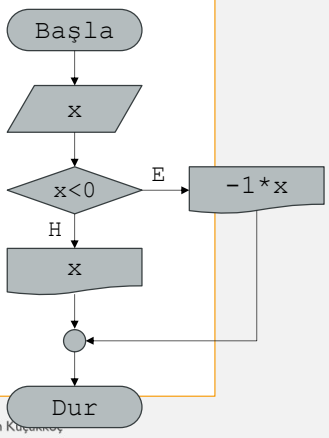
"*koşul*" doğru ise (*E*) "*işlem-1*", değilse (*H*) "*işlem-2*" yapılır.

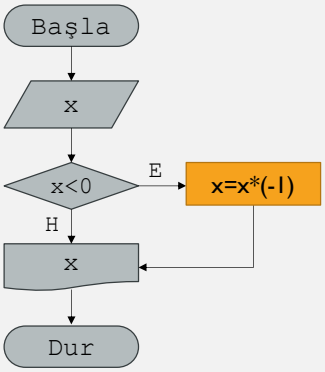
"*koşul-1*" doğru ise (*E*) "*işlem-1*", değilse (*H*) "*koşul-2*" kontrol edilir ve doğru ise (*E*) "*işlem-2*" yapılır. "*koşul-2*" yanlış ise (*H*) sıradaki koşul kontrol edilir ve doğru olup olmamasına göre işlem devam eder. Eğer tüm koşullar yanlış ise "*işlem-n*" yapılır.

Akış Diyagramı

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

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





Akış Diyagramı

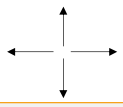
Veri/Bilgi Yazma: Ekrana veya yazıcıya veri/bilgi yazdırmak için kullanılır. Sabit alfasayısal bilgiler yazdırılacak ise çift tırnak içerisinde yazılır.

Önceden Tanımlı İşlem: Özellikle büyük boyutlu programlar bir çok alt programdan/fonksiyondan oluşur. Bu şekilde önceden tanımlanmış programları parametreli veya parametresiz olarak çağırmak için kullanılır.

Bağlantı: Genel anlamda işlem akışlarını birleştiren bir yer olup:
- Farklı yerlere dallanan işlem akışlarını toplamak
- Bir sayfadan diğerine geçen akış diyagramları arasında bağlantı kurmak
- Parça parça çizilen akış diyagramları arasında bağlantı kurmak için kullanılır.

A

1



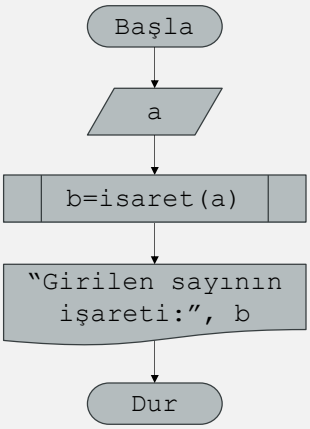
İşlem Akış Yönleri: İşlem akışının hangi yönde olduğunu gösteren oklardır.

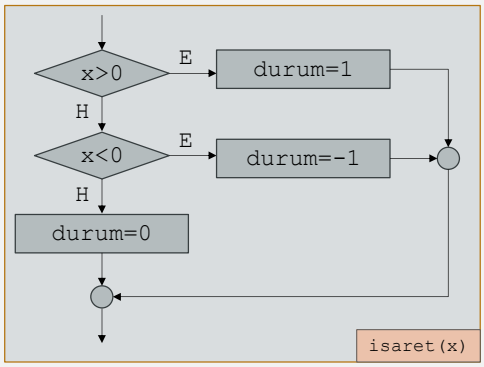
Akış Diyagramı

Örnek şekil	Açıklama
"Uludağ"	Tırnak içindeki 'Uludağ' ismini aynen yazdırır. *Sabit ifadeyi yaz ve alt satıra geç*
x	'x' değişkeninin içeriğini yazdırır ve imleç, bir alt satıra geçer (imleç satırbaşı yapar). *Değişken içeriğini yaz ve alt satıra geç*
y ,	'y' değişkeninin içeriğini yazdırır ve imleç, aynı satırda kalır. Daha sonraki yazdırma işlemleri – herhangi bir konumlandırma ve yönlendirme yoksa – kalınan yerden devam eder. *Değişken içeriğini yaz ve aynı satırda kal*
"Bursa" , a	Tırnak içindeki 'Bursa' ismini ve 'a' değişkeninin içeriğini yazdırır. *Sabit ifade ve değişken içeriğini yazarak alt satıra geç*

A
1

Akış Diyagramı

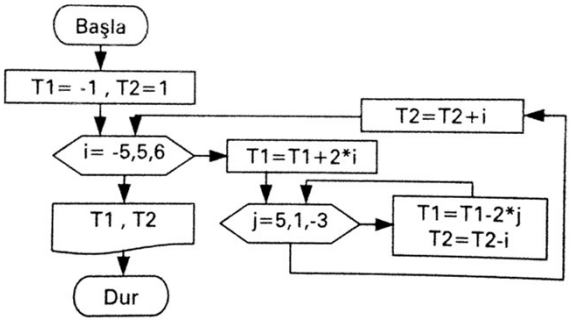




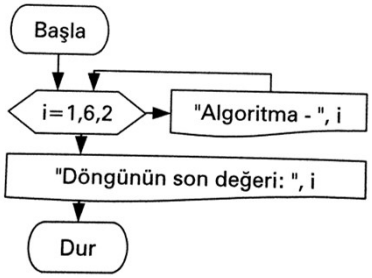
Bir sayının işaretini bulan alt programın yazılıp ana programdan çağrılması

Akış Diyagramı

Aşağıdaki akış diyagramlarının ekran çıktısını elde ediniz.



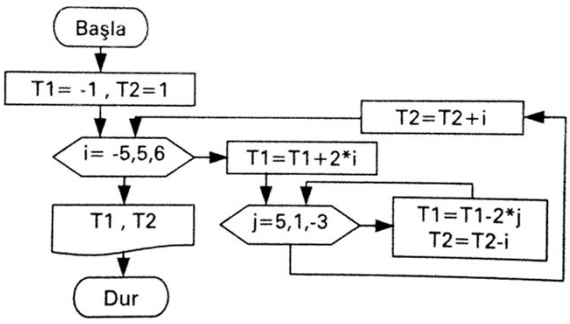
(Seckin, s.80)



(Seckin, s.80)

Akış Diyagramı

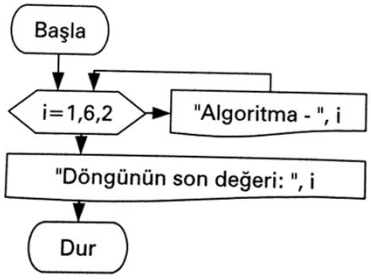
Aşağıdaki akış diyagramlarının ekran çıktısını elde ediniz.



T1	T2	i	j	Ekran
-1	1	-5		
-11			5	
-21	6		2	
-25	11			
-23	6	1	5	
-33	5		2	
-37	4			
	5			**-37 5**

Akış Diyagramı

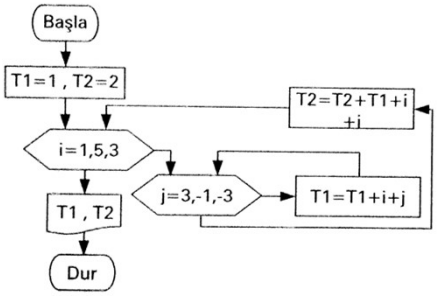
Aşağıdaki akış diyagramlarının ekran çıktısını elde ediniz.



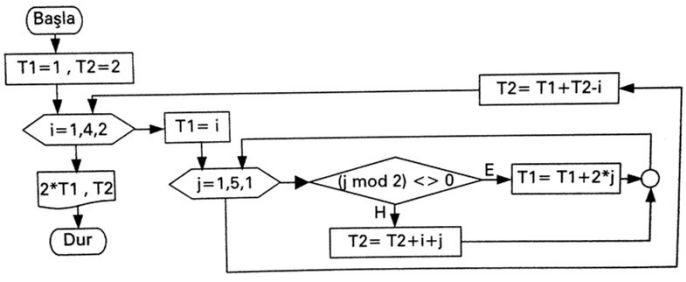
i	Ekran
1	Algoritma - 1
3	Algoritma - 3
5	Algoritma - 5
	Döngünün son değeri: 5

Akış Diyagramı

Aşağıdaki akış diyagramlarının ekran çıktısını elde ediniz.



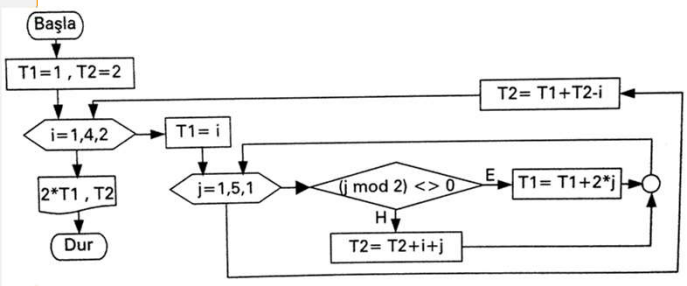
(Seckin, s.81)



(Seckin, s.81)

Akış Diyagramı

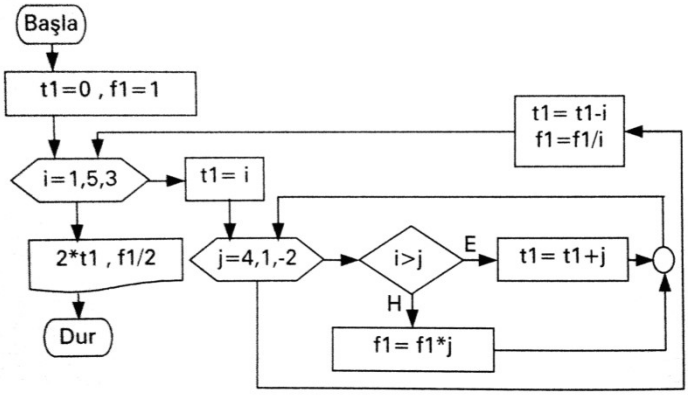
Aşağıdaki akış diyagramlarının ekran çıktısını elde ediniz.



T1	T2	i	j	Ekran
1	2	1		
1			1	
3	5		2	
9			3	
	10		4	
19	28	3	5	
3			1	
5	33		2	
11			3	
	40		4	
21	58		5	**42 58**

Akış Diyagramı

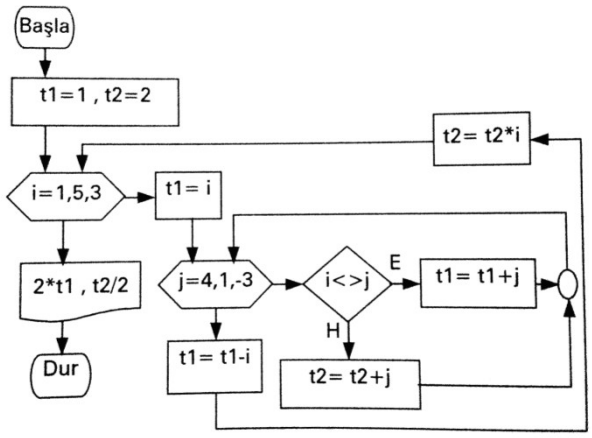
Aşağıdaki akış diyagramının ekran çıktısını elde ediniz.



(Seckin, s.82)

Akış Diyagramı

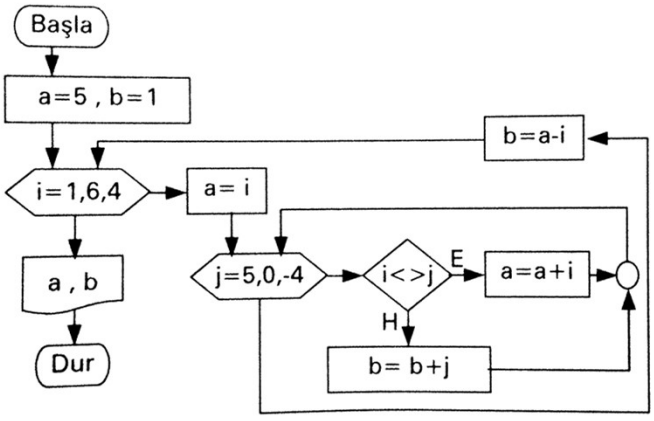
Aşağıdaki akış diyagramının ekran çıktısını elde ediniz.



(Seckin, s.82)

Akış Diyagramı

Aşağıdaki akış diyagramının ekran çıktısını elde ediniz.



(Seckin, s.82)

İki veya Çok Alternatifli Koşul Yapıları

Bazı durumlarda basit koşul yapıları yetse de bazen akışın ikiye (veya daha fazla) ayrılıp birinden devam etmesi gerekebilir. Yani, basit koşul yapılarının yetmediği yerde iki veya daha çok alternatifli koşul yapıları kulanılır.

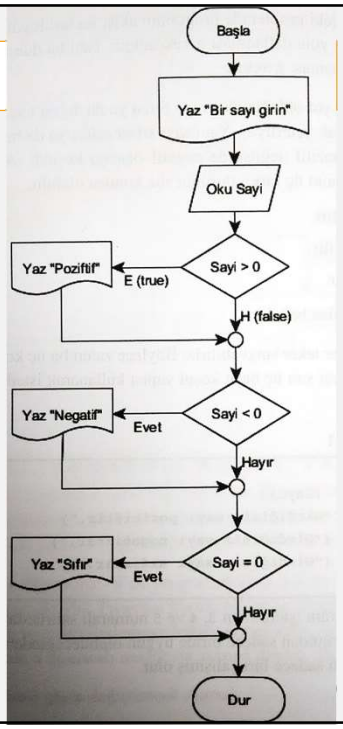
Örnek: Kullanıcıdan bir sayı alarak, girilen sayının pozitif, negative veya sıfır olduğunu ekrana yazdıran programın algoritmasını tasarlayın (*Kodlab s.34*).

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

Bu algoritmada 4, 5, ve 6'ıncı satırlardaki koşullar sırayla sınanır ve komutlardan sadece birisi çalışmış olur.

İki veya Çok Alternatifli Koşul Yapıları

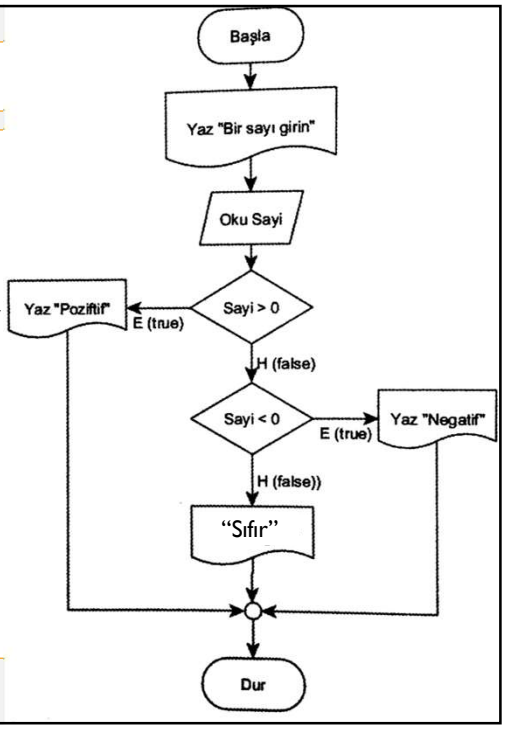
```
1. Başla
2. Yaz "Bir Sayı Girin"
3. Oku Sayı
4. Eğer Sayı>0 ise Yaz "Girdiğiniz Sayı Pozitiftir"
5. Eğer Sayı<0 ise Yaz "Girdiğiniz Sayı Negatiftir"
6. Eğer Sayı=0 ise Yaz "Girdiğiniz Sayı Sıfırdır"
7. Dur
```



İki veya Çok Alternatifli Koşul Yapıları

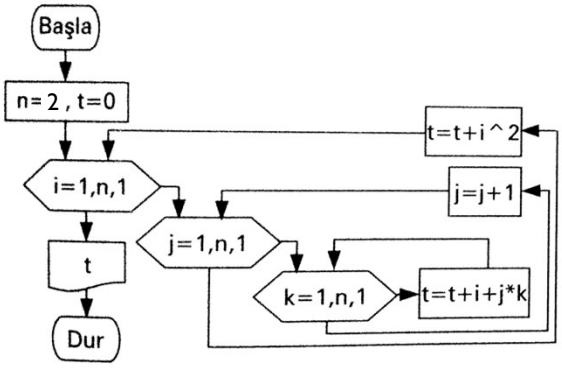
1. Başla
2. Yaz "*Bir Sayı Girin*"
3. Oku Sayı
4. Eğer
   - 4.1. Sayı>0 ise Yaz "*Girdiğiniz Sayı Pozitiftir*"
   - 4.2. Değilse Eğer
     - 4.2.1. Sayı<0 ise Yaz "Girdiğiniz Sayı Negatiftir"
     - 4.2.2. Değilse Yaz "Girdiğiniz Sayı Sıfırdır"
5. Dur

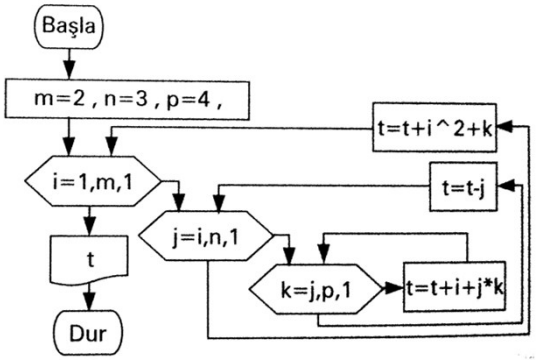
***Bu slayt güncellendi (27.02.2018)!**



Çalışma Soruları

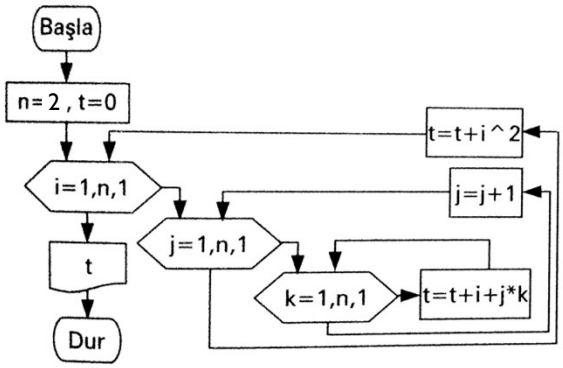
Aşağıdaki akış diyagramlarının ekran çıktılarını elde ediniz.





Çalışma Soruları

Aşağıdaki akış diyagramlarının ekran çıktılarını elde ediniz.

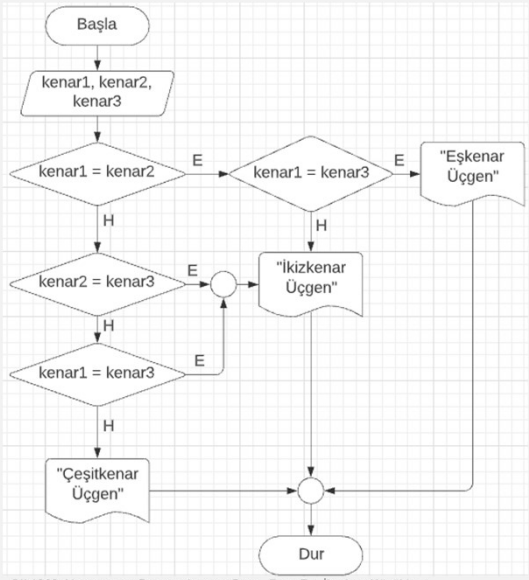


n	t	i	j	k	Ekran
2	0	1	1	1	
	2			2	
	5		2		
	6	2	1	1	
	9			2	
	13		2		
	17				**17**

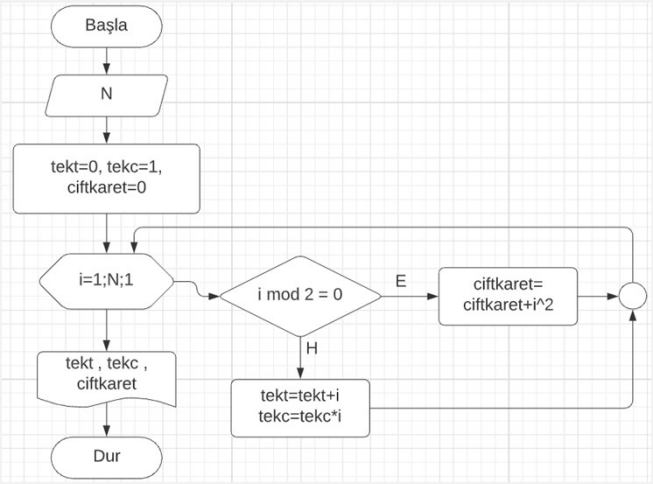
Çalışma Soruları

- **Klavyeden girilen N sayısına göre 1'den N'e kadar olan tek sayıların toplamını ve çarpımını, çift sayıların ise kareleri toplamını bulan programın akış diyagramını çiziniz?**
- Klavyeden girilen ismi, yine klavyeden istenen sayı kadar alt alta yazdıran programın akış diyagramını çiziniz?
- Klavyeden metre (m) cinsinden girilen uzunluğu, kilometre (km) ve santimetre (cm) cinsine dönüştürüp yazdıran programın akış diyagramını çiziniz?
- **Klavyeden üç kenar uzunluğu girilen üçgenin türünü (eşkenar, ikizkenar veya çeşitkenar) tespit edip yazdıran programın akış diyagramını çiziniz?**
- Klavyeden girilen bir sayının yine klavyeden istenen yüzdesini hesaplayıp yazdıran programın akış diyagramını çiziniz?
- Klavyeden bir ürünün fiyatı ve KDV oranı istenmektedir. Buna gore ürünün KDV'li satış fiyatını hesaplayıp yazdıran programın akış diyagramını çiziniz?
- Klavyeden bir ürünün fiyatı ve kar/zarar oranı istenmektedir. Buna göre ürünün satış fiyatını hesaplayıp yazdıran programın akış diyagramını çiziniz?
- Klavyeden 1-7 arasında bir tamsayı istenmektedir. Bu tamsayıya göre haftanın ilgili gününü kelime/isim olarak yazdıran programın akış diyagramını çiziniz?
- Klavyeden girilen 10 sayının ortalamasını hesaplayıp yazdıran programın akış diyagramını çiziniz?

- **Klavyeden üç kenar uzunluğu girilen üçgenin türünü (eşkenar, ikizkenar veya çeşitkenar) tespit edip yazdıran programın akış diyagramını çiziniz?**



Klavyeden girilen N sayısına göre 1'den N'e kadar olan tek sayıların toplamını ve çarpımını, çift sayıların ise kareleri toplamını bulan programın akış diyagramını çiziniz?



N	tekt	tekc	ciftkaret	i	Ekran
5	0	1	0	1	
	1	1	4	2	
	4	3		3	
			20	4	
	9	15		5	**9 15 20**

5

Sözde Kod (Pseudo-code) & *Algoritmalar Arasında Dönüşüm*

Sözde Kod

Problemi çözmek için tasarladığımız algoritmaları kodlamamız gerekir. Bu kodlama herhangi bir programlama dilinde (C, C++, Java, Python vb.) olabileceği gibi, eğer algoritmayı hemen çalıştırmaya ihtiyacımız yoksa ara bir yapı olan sözde koda da dönüştürebiliriz.

Sözde Kod:

Bilgisayarda bir programlama dili olarak çalışmayan, ancak yazı/konuşma dilinden ziyade programlama dillerine daha yakın olan algoritma ifadelerine sözde kod (pseudo-code) denilir.

Sözde kodların İngilizce ifadelerle belirtilmesi dünyada yaygın olarak kabul görmektedir.

Sözde Kod

Bir sözde kod, yapısal olarak dört temel öğeye sahiptir. Bunlar;

1. Okuma/Yazma:

   READ, GET, WRITE, DISPLAY gibi komutlarla temel okuma ve yazma işlemleri gerçekleştirilir. *Hangisi ne zaman kullanılır?*

2. İşlemler:

   Sözde kod içinde gerçekleştirilen toplama, çıkartma, bölme, vb. Aritmetik ve diğer işlemler ve bir değişkene değer atanması gibi olaylardır.

3. Karar Yapıları:

   Bir koşulu control edip, bir alternatifin işletilip işletilmeyeceğine veya birden fazla alternatiften hangisinin işletileceğine karar veren mekanizmalardır.

Sözde Kod

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

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

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

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

Sözde Kod

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

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

SORU:
Yukarıdaki yazım ile sağdaki arasında ne fark vardır?

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


Sözde Kod

Örnek:

BiletKart uygulaması için metroda uygulanan ücret tarifesi aşağıdaki gibidir. Buna göre, uygulanan ücret politikasının algoritmasını oluşturunuz.

- Normal tarife: 3 TL
- Öğrenci ve 30 (dahil) yaşından küçük olanlar: 2.5 TL
- Öğrenci ve 30 yaşından büyük olanlar: 2.75 TL
- 60 (dahil) yaşından büyük olanlar: Ücretsiz

NOT: Bir kişi her iki koşulu birden taşıması durumunda, daha düşük olan ücret tarifesi uygulanır.

```
GET ogrenci, yas
bilet=3
IF ogrenci=true THEN
    IF yas<= 30 THEN
        bilet=2.5
    ELSE
        bilet=2.75
    ENDIF
ENDIF
IF yas>=60 THEN
    bilet=0
ENDIF
DISPLAY bilet
```

```
GET ogrenci, yas
IF yas<=30 THEN
    IF ogrenci=true THEN
        bilet=2.5
    ELSE
        bilet=3
    ENDIF
ELSEIF yas < 60
    IF ogrenci=true THEN
        bilet=2.75
    ELSE
        bilet=3
    ENDIF
ELSE
    bilet=0
ENDIF
DISPLAY bilet
```

```
GET ogrenci, yas
IF ogrenci=true AND yas<=30 THEN
    bilet=2.5
ELSEIF ogrenci=true AND yas>30 AND yas <60 THEN
    bilet=2.75
ELSEIF yas>=60 THEN
    bilet=0
ELSE
    bilet=3
ENDIF
DISPLAY bilet
```

Normal tarife: 3 TL
Öğrenci ve 30 (dahil) yaşından küçük olanlar: 2.5 TL
Öğrenci ve 30 yaşından büyük olanlar: 2.75 TL
60 (dahil) yaşından büyük olanlar: Ücretsiz

NOT: Bir kişi her iki koşulu birden taşıması durumunda, daha düşük olan ücret tarifesi uygulanır.

Sözde Kod

4. Tekrarlı Yapılar:

Program içinde bir koşula bağlı olarak ya da belirli bir sayıda tekrar edecek işlemler için kullanılır.

Koşula bağlı tekrarlı yapılarda, koşul bazen tekrarın girişine, bazen de sonuna uygulanır. Aşağıdaki yapıda koşul tekrarlı yapının girişinde uygulanmıştır. Bu durumda eğer koşul *doğru (true)* değeri vermezse, koşul içindeki tekrarlanacak işlemler hiç gerçekleşmez.

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

****** NOT: BREAK komutu, içinde bulunduğu döngüyü kırar.***

Sözde Kod

Aşağıdaki koşula bağlı tekrarlı yapıdaysa, koşulun durumu her ne olursa olsun, tekrarlı yapı içindeki kod en az bir kere çalıştırılacaktır.

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

Sayaç tipi tekrarlı yapılarda, belirli bir sayıdan başlanarak, belirli bir hedefe kadar sayılır. Sayma işleminde artışın ne kadar olacağını STEP deyimi belirtir.

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

****** 17'ye tam bölünebilen en büyük üç basamaklı sayıyı LOOP veya FOR döngüsü kullanarak bulan ve ekrana yazdıran programın algoritmasını sözde kod (pseudo-code) olarak yazınız.***


```
bulundu = false
FOR (i=100 TO 999 STEP 1)
    IF (i mod 17 = 0) THEN
        bulunan = i
        bulundu = true
    ENDIF
ENDFOR
IF (bulundu = false) THEN
    DISPLAY "Böyle bir sayı yoktur"
ELSE
    DISPLAY bulunan
ENDIF
```

```
bulundu = false
FOR (i=999 TO 100 STEP -1)
    IF (i mod 17 = 0) THEN
        DISPLAY i
        bulundu = true
        BREAK
    ENDIF
ENDFOR
IF (bulundu = false) THEN
    DISPLAY "Böyle bir sayı yoktur"
ENDIF
```

```
bulundu = false
i=999
LOOP (i>=100)
    IF (i mod 17 = 0) THEN
        DISPLAY i
        bulundu = true
        BREAK
    ENDIF
    i=i-1
ENDLOOP
IF (bulundu = false) THEN
    DISPLAY "Böyle bir sayı yoktur"
ENDIF
```

```
Sayac=999
LOOP (Sayac>=100)
    ……
    ……
    ……
    ……
    Sayac=Sayac-1
ENDLOOP


FOR (Sayac=999 TO 100 STEP -1)
    ……
    ……
    ……
ENDFOR
```


Algoritmalar Arasında Dönüşüm

Satır Algoritmalardan Sözde Kod Oluşturmak

Bir satır algoritmayı sözde koda dönüştürürken aşağıdaki adımları izleriz:

- Girdi ve çıktılar (değişkenler) belirlenir
- Sıralı adımlar, karar yapıları, tekrarlı yapılar ve işlemler belirlenir
- Yapı, işlem ve adımlar uygun şekilde birleştirilir.

```
1. Başla
2. Yaz 1
3. Yaz 2
4. Yaz 3
5. Dur
```

```
DISPLAY 1
DISPLAY 2
DISPLAY 3
```


Satır Algoritmalardan Sözde Kod Oluşturmak

İki sayıyı alıp, bunları toplayarak toplamı ekrana yazdıran algoritmanın satır kodu ve sözde kodu:

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

```
GET A
GET B
C=A+B
DISPLAY C
```

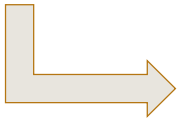
Satır Algoritmalardan Sözde Kod Oluşturmak

```
1. Başla
2. Yaz "Yaşınızı Giriniz"
3. Oku (Yas)
4. Eğer
 4.1. Yas>=18 ise Yaz "Uygulamayı İndirebilirsiniz"
 4.2. Değilse Yaz "Uygulamayı İndiremezsiniz"
5. Dur
```

```
DISPLAY "Yaşınızı Giriniz"
GET Yas
IF Yas>=18 THEN
   DISPLAY "Uygulamayı İndirebilirsiniz"
ELSE
   DISPLAY "Uygulamayı İndiremezsiniz"
ENDIF
```


Satır Algoritmalardan Sözde Kod Oluşturmak

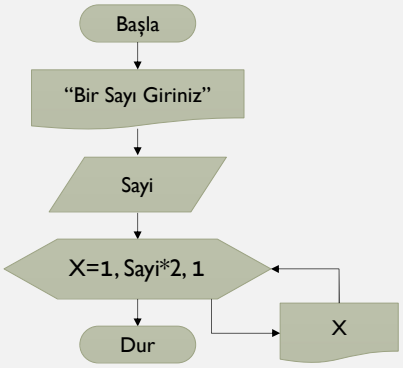
```
1. Başla
2. Toplam=0
3. DÖNGÜ (X=1 TO 100 STEP 1)
4. Yaz "Bir Sayı Girin"
5. Oku Sayi
6. Toplam=Toplam+Sayi
7. DÖNGÜSONU
8. Yaz Toplam
5. Dur
```



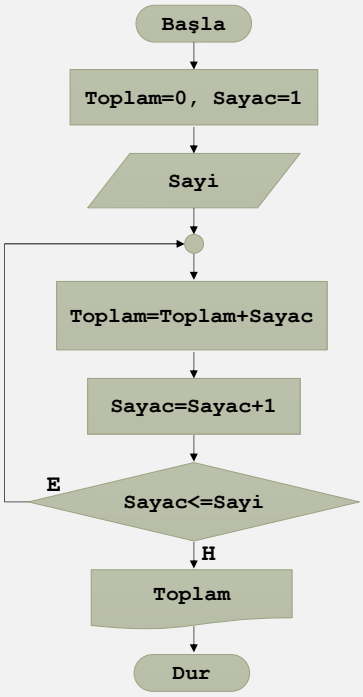
```
Toplam=0
FOR(X=1 TO 100 STEP 1)
   DISPLAY "Bir Sayı Girin"
   GET Sayi
   Toplam = Toplam + Sayi
ENDFOR
DISPLAY Toplam
```

Satır Algoritmalardan Akış Diyagramı Oluşturmak

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



123



Akış Diyagramlarından Sözde Kod Oluşturmak

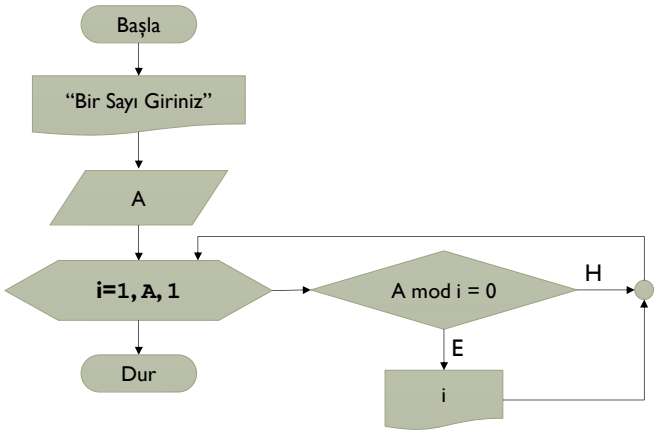
```
Toplam=0, Sayac=1
GET Sayi
LOOP
   Toplam=Toplam+Sayac
   Sayac=Sayac+1
ENDLOOP (Sayac<=Sayi)
DISPLAY Toplam
```

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

Örnek

- Klavyeden girilen pozitif bir A tamsayısının tam bölenlerini bulup listeleyen programı tasarlayarak
  - Satır algoritma
  - Akış diyagramı ve
  - Sözde kod

olarak ifade ediniz.



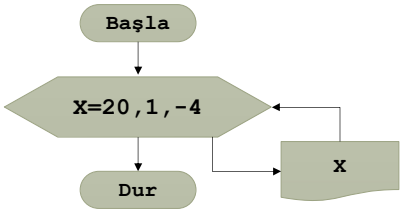

```
DISPLAY "Bir sayı giriniz"
GET A
FOR(i=1 TO A STEP 1)
   IF (A MOD i = 0) THEN
      DISPLAY i
   ENDIF
ENDFOR
```

125

Örnek

- 20'den başlayıp 1'e kadar, dörder dörder geriye doğru sayıp ekrana yazdıran algoritmayı tasarlayınız.

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




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

Çalışma Sorusu – 1

- Klavyeden girilen n sayısına göre;
  1'den n'e kadar tamsayıların toplamını (t1)
  1'den n'e kadar tek tamsayıların toplamını (t2)
  2'den n'e kadar çift sayıların toplamını (t3)

hesaplayan ve ekrana yazdıran programı tasarlayarak;

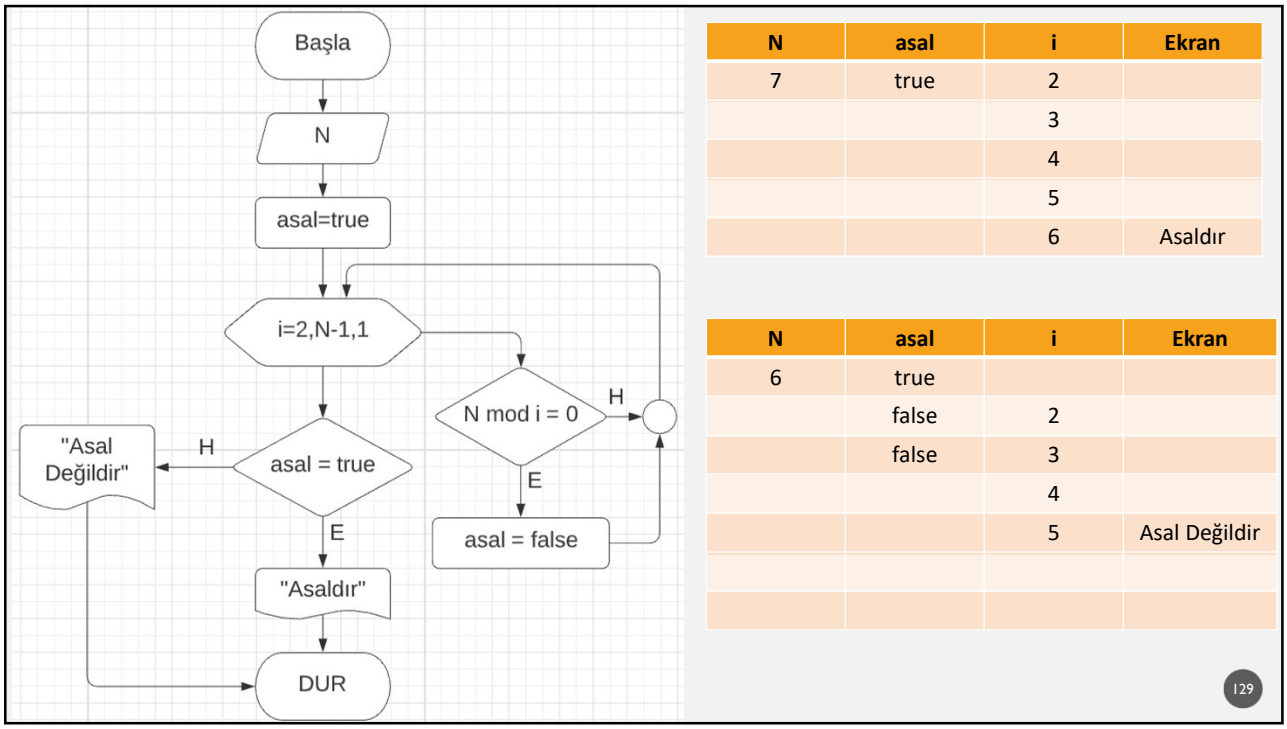
- Satır algoritma
- Akış diyagramı ve
- Sözde kod

olarak ifade ediniz (*Seçkin, s.185*).

Çalışma Sorusu – 2

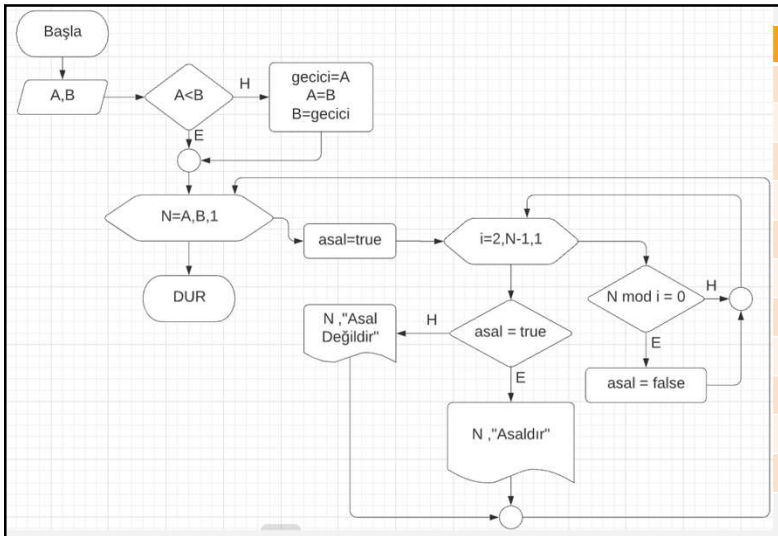
- Klavyeden girilen A ve B sayıları arasındaki (sınırlar dahil) asal sayıları bulup ekrana yazdıran programı tasarlayıp;
  - Satır algoritma
  - Akış diyagramı ve
  - Sözde kod

olarak ifade ediniz (*Kodlab, s. 166*).



N	asal	i	Ekran
7	true	2	
		3	
		4	
		5	
		6	Asaldır

N	asal	i	Ekran
6	true		
	false	2	
	false	3	
		4	
		5	Asal Değildir



A	B	gec	N	asal	i	Ekran
6	3	6				
3	6		3	true	2	3 Asaldır
			4	true	2	
				false	3	4 Asal Değildir
			5	true	2	
					3	
					4	5 Asaldır
			6	true	2	
				false	3	
				false	4	
					5	6 Asal Değildir

Çalışma Sorusu – 3

- Fibonacci sayı dizisi 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89... dizilişindeki sayılardan oluşan bir dizidir. Dizideki ilk iki sayı "0, 1" dir ve sonra gelen sayılar, kendisinden önceki iki sayının toplamıdır.

0	1	1
İki önceki sayı	Bir önceki sayı	Yeni sayı

- Fibonacci dizisinin ilk 10 elemanını hesaplayarak ekrana yazdıran programı tasarlayarak
  - Satır algoritma
  - Akış diyagramı ve
  - Sözde kod

olarak ifade ediniz (*Kodlab, s.160*).


```
sayi1=0, sayi2=1
DISPLAY sayi1
DISPLAY sayi2
FOR (sayac=1 TO 8 STEP 1)
    sayi3=sayi1+sayi2
    DISPLAY sayi3
    sayi1=sayi2
    sayi2=sayi3
ENDFOR
```

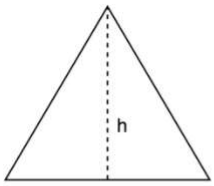
sayi1	sayi2	sayi3	sayac	Ekran
0	1			0
		1	1	1
1	1	2	2	1
1	2	3	3	2
2	3	5	4	3
3	5	8	5	5
5	8	13	6	8
8	13	21	7	13
13	21	34	8	21
				34

6

Temel Algoritma Örnekleri & Genel Uygulamalar

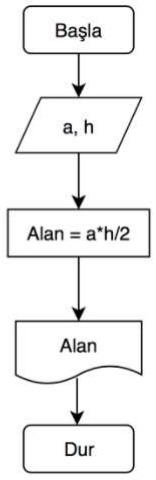
Üçgenin Alanı - 1

Klavyeden bir kenar uzunluğu ve o kenara ait yüksekliği girilen üçgenin alanını hesaplayan programın satır kodunu ve akış diyagramını geliştiriniz.



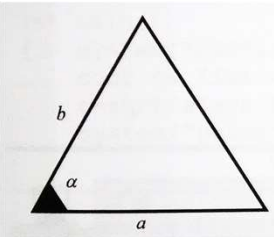
Alan = (a.h) / 2

1. Başla
2. Kenar uzunluğunu (a) gir
3. Yüksekliği (h) gir
4. Alan=a*h/2
5. Yaz Alan
6. Dur



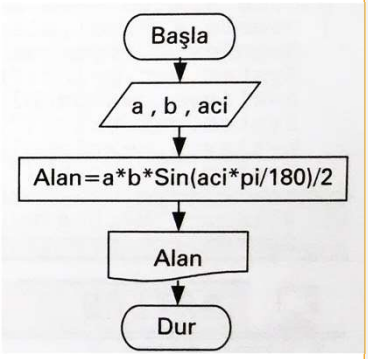
Üçgenin Alanı - 2

Klavyeden iki kenarı ve derece cinsinden aradaki açısı girilen üçgenin alanını hesaplayan programın satır kodunu ve akış diyagramını geliştiriniz.



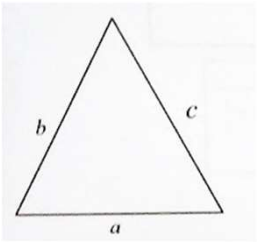
$$Alan = a.b.Sin(\alpha)/2$$

1. Başla
2. Birinci kenarı (a) gir
3. İkinci kenarı (b) gir
4. Aradaki açıyı (aci) gir
5. Alan=a*b*Sin(aci*pi/180)/2
6. Yaz Alan
7. Dur



Üçgenin Alanı - Heron Formülü

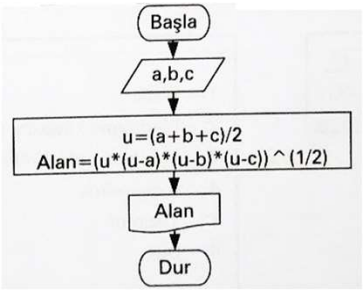
Klavyeden tüm kenar uzunlukları girilen üçgenin alanını hesaplayan programın satır kodunu ve akış diyagramını geliştiriniz.



$$u = \frac{a+b+c}{2}$$

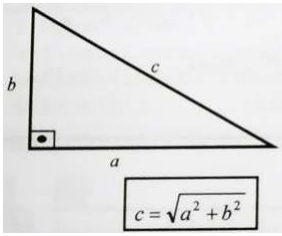
$$Alan = \sqrt{u(u-a)(u-b)(u-c)}$$

1. Başla
2. Birinci kenarı (a) gir
3. İkinci kenarı (b) gir
4. Üçüncü kenarı (c) gir
5. u=(a+b+c)/2
6. Alan=(u*(u-a)*(u-b)*(u-c)) ^ (1/2)
7. Yaz Alan
8. Dur

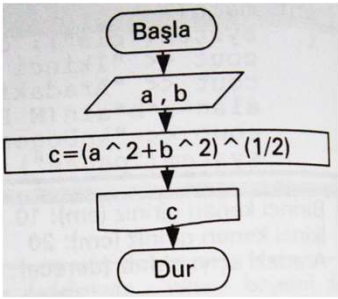


Pisagor Teoremi

Klavyeden dik kenarlarının uzunluğu verilen bir üçgende, hipotenüsün uzunluğunu bulan programı satır kod ve akış diyagramı olarak ifade ediniz.

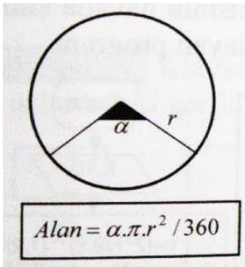


1. Başla
2. Birinci dik kenarı (a) gir
3. İkinci dik kenarı (b) gir
4. c=(a^2+b^2)^(1/2)
5. Yaz c
6. Dur

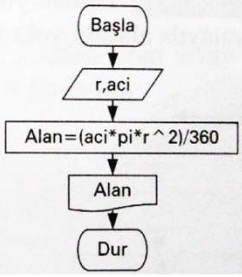


Daire Diliminin Alanı

Klavyeden yarıçapı ve derece cinsinden açısı girilen daire diliminin alanını hesaplayan programı satır kod ve akış diyagramı olarak ifade ediniz.



1. Başla
2. Yarıçapı (r) gir
3. Açıyı (aci) gir
4. Alan=(aci*pi*r^2)/360
5. Yaz Alan
6. Dur

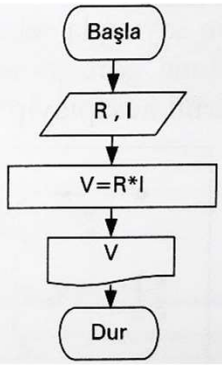


Ohm Kanunu

Klavyeden bir iletkenin direnciyle içinden geçen akım girildiğinde, uçlarındaki gerilimi hesaplayan programı satır kod ve akış diyagramı olarak ifade ediniz.

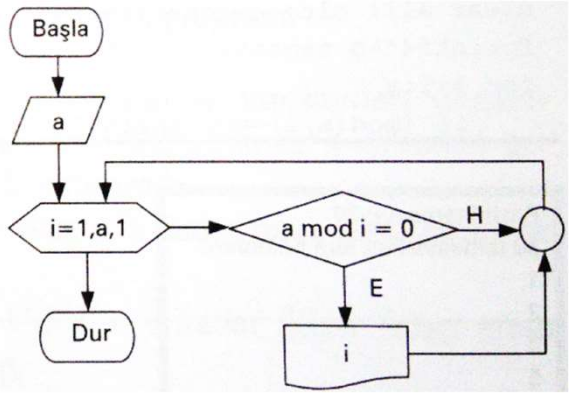
$V = R.I$

1. Başla
2. İletkenin direncini (R) gir
3. İletkenin içinden akan akımı (I) gir
4. V=R*I
5. Yaz V
6. Dur



Bir Tamsayının Tam Bölenlerinin Bulunması

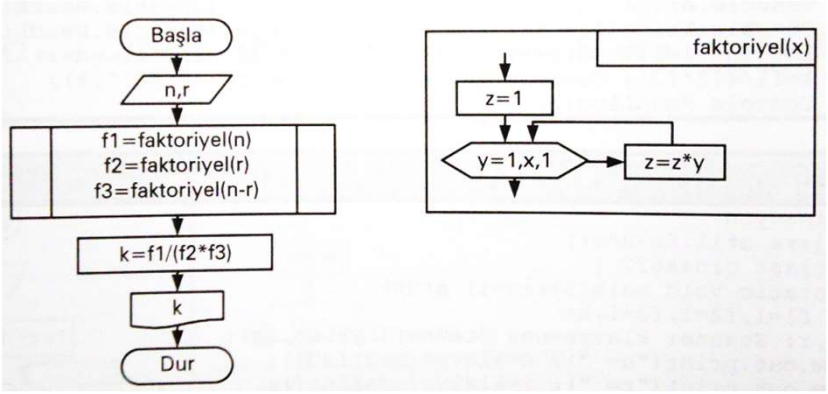
Klavyeden girilen pozitif bir *a* tamsayısının tam bölenlerini hesaplayıp listeleyen programı satır kod ve akış diyagramı olarak ifade ediniz.



İyileştirme?

Kombinasyon Hesaplama

Klavyeden eleman sayısı girilen bir kümenin belirtilen kombinasyonlarının sayısını hesaplayan programı satır kod ve akış diyagramı olarak ifade ediniz.



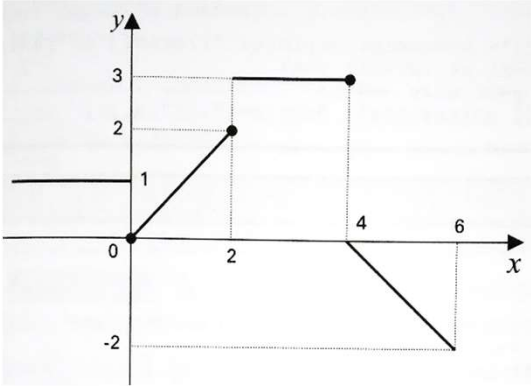
$$C(n,r) = \frac{n!}{r!\,(n-r)!}$$

Birinci Dereceden Denklemin Kökü

Klavyeden katsayıları girilen birinci dereceden denklemin kökünü hesaplayan programın satır kodunu ve akış diyagramını geliştiriniz.

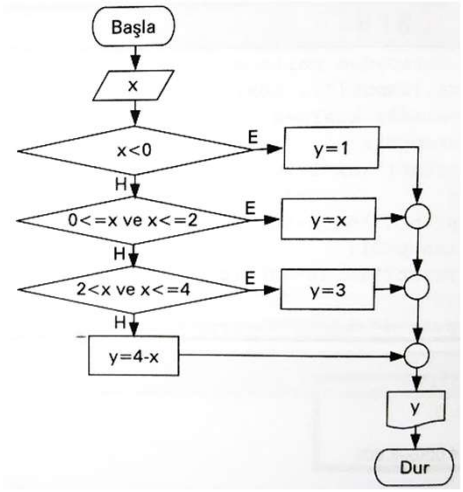
Grafiği Verilen Fonksiyon

Bir y=f(x) fonksiyonu, grafiksel olarak aşağıdaki gibi verilmektedir. Buna göre klavyeden girilen x değeri için y'yi hesaplayıp ekrana yazdıran programın satır kodunu ve akış diyagramını geliştiriniz.



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

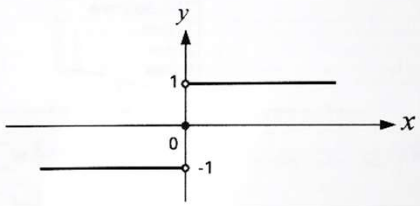
1. Başla
2. x değerini gir
3. Eğer x<0 ise y=1
4. Eğer x>=0 ve x<=2 ise y=x
5. Eğer x>2 ve x<=4 ise y=3
6. Eğer x>4 ise y=4-x
7. Yaz y
8. Dur



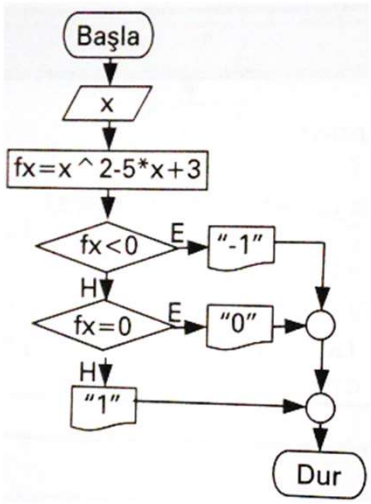
İşaret (Signum) Fonksiyonu

İşaret (signum) fonksiyonu'nun tanımı ve grafiksel gösterimi aşağıda verilmektedir. Tanıma göre klavyeden girilen x değeri için $f(x) = x^2 - 5x + 3$ fonksiyonunun işaretini hesaplayan programın satır kodunu ve akış diyagramını geliştiriniz.

$$Sgn(f(x)) = \begin{cases} -1 & , \ f(x) < 0 \\ 0 & , \ f(x) = 0 \\ 1 & , \ f(x) > 0 \end{cases}$$



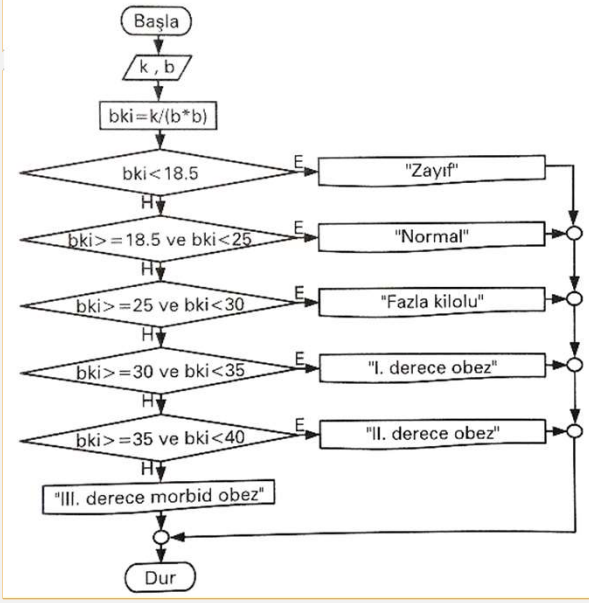

```
1. Başla
2. x değerini gir
3. fx=x^2-5*x+3
4. Eğer fx<0 ise Yaz "-1"
5. Eğer fx=0 ise Yaz "0"
6. Eğer fx>0 ise Yaz "1"
7. Dur
```



İdeal Vücut Ağırlığı

Klavyeden kilo (kg) ve boy (m) bilgisini alıp, aşağıdaki tabloya göre ideal kilo durumunu değerlendirip ekrana yazdıran programın akış diyagramını geliştiriniz.

	BKI (kg/m²)
Zayıf	18,5 altında
Normal	18,5 - 24,9
Fazla kilolu	25 - 29,9
I. derece obez	30 - 34,9
II. derece obez	35 - 39,9
III. derece morbid obez	40 ve üzerinde



Narsist Sayı (Armstrong Sayısı)

n haneli bir sayının basamaklarının n'inci üstlerinin toplamı, sayının kendisine eşitse, böyle sayılara narsist sayılar (veya Armstrong sayıları) denir.

Örneğin, 153 sayısı 3 haneli bir narsist sayıdır.

Çünkü 1^3 + 5^3 + 3^3 = 153 olmaktadır.

Soru:

3 haneli en büyük narsist sayıyı hesaplayıp ekrana yazdıran programı geliştiriniz.

Soru:

Klavyeden girilen bir sayının, narsist sayı olup olmadığını belirleyip ekrana yazdıran programın sözde kodu:

a,b,c tamsayı olmak üzere a:yüzler, b: onlar, c: birler basamağı

```
GET sayi
a=sayi/100
b=(sayi mod 100)/10
c=sayi mod 10
IF (a^3+b^3+c^3 = sayi) THEN
   DISPLAY "Girilen sayı Narsist bir sayıdır"
ELSE
   DISPLAY "Girilen sayı Narsist sayı değildir"
ENDIF
```

Soru:
3 haneli en büyük narsist sayıyı hesaplayıp ekrana yazdıran programın sözde kodu:
a,b,c tamsayı olmak üzere a:yüzler, b: onlar, c: birler basamağı

```
bulunan=-1
FOR (sayi=999 TO 100 STEP -1)
  a=sayi/100
  b=(sayi mod 100)/10
  c=sayi mod 10
  IF (a^3+b^3+c^3 = sayi) THEN
     bulunan=sayi
     BREAK
  ENDIF
ENDFOR
IF bulunan=-1 THEN
  DISPLAY "Böyle bir sayı yoktur"
ELSE
  DISPLAY bulunan
ENDIF
```

```
bulunan=-1
FOR (sayi=100 TO 999 STEP 1)
  a=sayi/100
  b=(sayi mod 100)/10
  c=sayi mod 10
  IF (a^3+b^3+c^3 = sayi) THEN
     bulunan=sayi
  ENDIF
ENDFOR
IF bulunan=-1 THEN
  DISPLAY "Böyle bir sayı yoktur"
ELSE
  DISPLAY bulunan
ENDIF
```


Soru 1:

Klavyeden girilen bir sayının tam sayı olup olmadığını nasıl bulursunuz(Seçkin s.213)?

Soru 2:

Klavyeden girilen b tamsayısına göre $a^3 - a^2 = b$ şartını sağlayan $0 < a < 100$ tamsayısını nasıl bulursunuz (Seçkin s.213)?

Soru 3:

Girilen pozitif bir tamsayının, iki sayının kareleri toplamı şeklinde yazılıp yazılamayacağını nasıl hesaplarsınız (Seçkin s.215)?

- Klavyeden girilen b tamsayısına göre $a^3 - a^2 = b$ şartını sağlayan $0 < a < 100$ tamsayısını nasıl bulursunuz?

```
GET b
FOR (a=0 TO 100 STEP 1)
   IF (a^3-a^2=b) THEN
       DISPLAY a
   ENDIF
ENDFOR
```

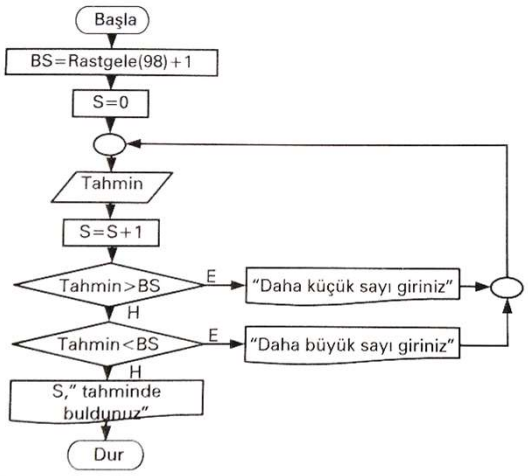
```
GET b
var=false
FOR (a=0 TO 100 STEP 1)
   IF (a^3-a^2=b) THEN
       DISPLAY a
       var=true
   ENDIF
ENDFOR
IF (var=false) THEN
    DISPLAY "Koşulu sağlayan a sayısı yoktur"
ENDIF
```


- Girilen pozitif bir tamsayının, iki sayının kareleri toplamı şeklinde yazılıp yazılamayacağını nasıl hesaplarsınız?

```
GET sayi
bulundu=false
FOR (a=1 TO sayi STEP 1)
   FOR (b=1 TO sayi STEP 1)
       IF (sayi=a^2+b^2) THEN
           DISPLAY a , b
           bulundu=true
       ENDIF
   ENDFOR
ENDFOR
IF (bulundu=false) THEN
   DISPLAY "Uygun sayılar yoktur"
ENDIF
```

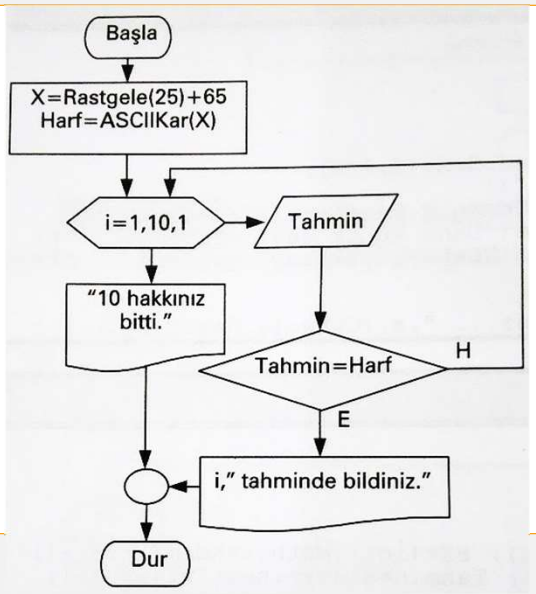
Sayı Tahmin Oyunu

Bilgisayarın ürettiği 1-99 arası bir tamsayının, kullanıcı tarafından tahmin edilmesi oyunu için geliştirilen akış diyagramı



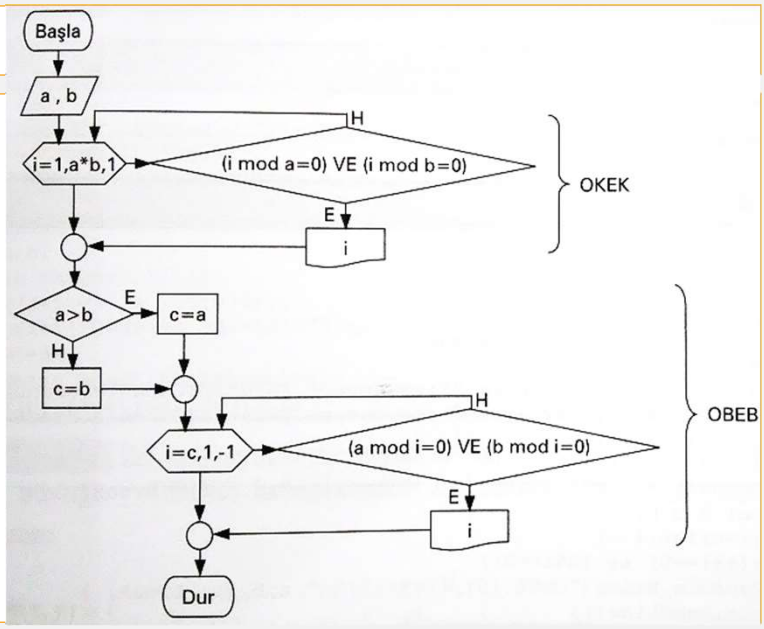
Harf Tahmin Oyunu

Bilgisayarın ürettiği rastgele büyük harfin, en fazla 10 denmede tahmin edilmesi oyunu için geliştirilen akış diyagramı



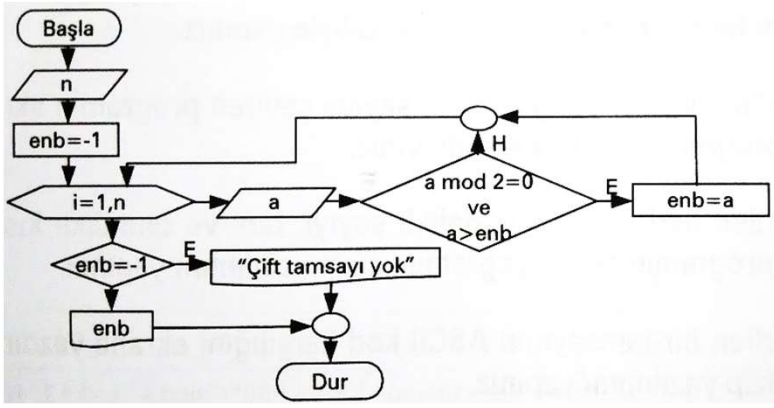
OKEK-OBEB

Klavyeden girilen iki pozitif tamsayının ortak katlarının en küçüğü (OKEK) ile ortak katlarının en büyüğünü (OBEB) hesaplayıp ekrana yazdıran programın akış diyagramı



Girilen Tamsayılardan En Büyük Çift Olanını Bulma

Klavyeden girilen n tane pozitif tamsayıdan, en büyük çift tamsayıyı bulan programın akış diyagramı



Çalışma Soruları

- Klavyeden girilen negatif sayıyı pozitif sayıya çeviren programı tasarlayıp akış diyagramını çiziniz.
- Klavyeden girilen bir sayının karesini, küpünü ve karekökünü hesaplayıp yazdıran programı tasarlayıp sözde kodunu yazınız ve akış diyagramını çiziniz.
- Klavyeden girilen üç sayıyı büyükten küçüğe doğru sıralayan programın sözde kodunu yazınız ve akış diyagramını çiziniz.
- 1-99 arasındaki, haneleri toplamı tek olan tamsayıların listesini veren programın akış diyagramını çiziniz.
- Klavyeden girilen N değerine göre (N>20), 10-N arası tüm sayıların toplamını, 5-N arası tek sayıların çarpımını, 14-N arası çift sayıların toplamını hesaplayıp yazdıran programın sözde kodunu yazınız ve akış diyagramını çiziniz.

7

Java Programlama Dili ve Algoritmadan Kodlamaya Geçiş

Java'da Veri Tipleri

Java'da kullanılacak olan değişkenler ve veri tipleri önceden bildirilmelidir.

1. Tamsayı Veri Tipleri

Veri Tipi	Aktarılabilecek En Küçük Değer	Aktarılabilecek En Büyük Değer	Bellekte Kapladığı Alan (byte)
byte	-2^7	2^7-1	1
short	-2^{15}	$2^{15}-1$	2
int	-2^{31}	$2^{31}-1$	4
long	-2^{63}	$2^{63}-1$	8

2. Ondalıklı Sayı Veri Tipleri

Veri Tipi	Aktarılabilecek En Küçük Değer	Aktarılabilecek En Büyük Değer	Bellekte Kapladığı Alan (byte)
float	$3,4.10^{-38}$	$3,4.10^{38}$	4
double	$1,7.10^{-308}$	$1,7.10^{308}$	8

161

Java'da Veri Tipleri

3. Alfasayısal Veri Tipleri

Tek karakter ve karakter grubu (kelime, cümle vb.) için kullanılabilecek iki veri tipi olup aşağıda gösterilmiştir:

Veri Tipi	Anlamı
char	Tek tırnak içinde bir karakter aktarılabilir.
String	Çift tırnak içinde birden fazla karakter aktarılabilir.

Java'da Kullanılan Operatörler

Aritmetik Operatörler

Aritmetik operatörler için bilmemiz gereken en önemli kural operatörlerin öncelik sırasıdır. İşlemlerimiz bu öncelik sırasına göre yapılmaktadır.

Operatör	Anlamı
*	Çarpma
/	Bölme
%	Kalan (Mod Alma)
+	Toplama
-	Çıkarma

*, / ve % operatörleri, + ve –'ye göre önceliklidir.

*, /, + ve – operatörlerinin int veya float (double) türde operand kabul etmelerine karşılık kalan operandları sadece int türde operand olarak kabul eder. % operandı bölmede kalanı hesaplar.

Java'da Kullanılan Operatörler

Aritmetiksel Atama Operatörleri

Aritmetik operatörler için bilmemiz gereken en önemli kural operatörlerin öncelik sırasıdır. İşlemlerimiz bu öncelik sırasına göre yapılmaktadır.

Operatör	Sembolü	Kullanılışı	Anlamı
Atama	=	x=y	y'nin değerini x'e ata
Topla ata	+=	x+=y	x + y'nin değerini x'e ata
Çıkar ata	-=	x-=y	x - y'nin değerini x'e ata
Çarp ata	*=	x*=y	x * y'nin değerini x'e ata
Böl ata	/=	x/=y	x / y'nin değerini x'e ata
Kalanını ata	%=	x%=y	x % y'nin değerini x'e ata

Tablodan kolayca anlayacağımız üzere, x + = y ifadesi x = x + y ifadesine, x % = y ifadesi de x = x % y ifadesine denktir.

Java'da Kullanılan Operatörler

İlişkisel Operatörler

True veya false değeri döndürür. Koşullu yapılarda kullanılmaktadır.

Sembolü	Anlamı
<	Küçük
>	Büyük
<=	Küçük Eşit
>=	Büyük Eşit
==	Eşit
!=	Eşit Değil

Java'da Kullanılan Operatörler

Mantıksal Operatörler

Bilgisayar dillerinin hemen hepsinde, program akışını kontrol edebilmek ve yönlendirebilmek için mantıksal operatörler kullanılır. Java dilinde kullanılan mantıksal operatörler aşağıda açıklanmıştır.

Sembolü	Anlamı
&&	VE
||	VEYA
!	DEĞİL

```
boolean b;
b = !(3 > 2); // b is false
b = !(2 > 3); // b is true

boolean b;
b = 3 > 2 && 5 < 7; // b is true
b = 2 > 3 && 5 < 7; // b is now false

boolean b;
b = 3 > 2 || 5 < 7; // b is true
b = 2 > 3 || 5 < 7; // b is still true
b = 2 > 3 || 5 > 7; // now b is false
```

Bazı Matematiksel İşlem Komutları

π: `Math.PI`	Mutlak Değer: `Math.abs(x)`
e: `Math.E`	Mod: `%`
x^y: `Math.pow(x,y)`	En Büyük: `Math.max(x)`
$\sqrt{x}$: `Math.sqrt(x)`	En Küçük: `Math.min(x)`
Rastgele(x): `Math.random()*(x+1)`	Sırala: `Arrays.sort()`
Radyan → Derece: `Math.toDegrees()`	$Ln(x)$: `Math.log(x)`
Derece → Radyan: `Math.toRadians()`	$Log(x)$: `Math.log10(x)`
e^x: `Math.exp(x)`	$Sin(x)$: `Math.sin(x)`
Üste Yuvarla: `Math.ceil(x)`	$Cos(x)$: `Math.cos(x)`
Aşağı Yuvarla: `Math.floor(x)`	$Tan(x)$: `Math.tan(x)`
En Yakın Tamsayıya Yuvarla: `Math.round(x)`	

Bazı Alfasayısal İşlem Komutları – String İfadeler İçin

Uzunluk: `.length()`

Büyüt: `.toUpperCase()`

Küçült: `.toLowerCase()`

Ters: `.reverse()`

Bul: `.indexOf()` **veya** `.contains()`

Değiştir: `.replace`

Dönüştür Sayısal Tam: `.parseInt()`

Dönüştür Sayısal Ondalıklı: `.parseFloat()`

Dönüştür Alfasayısal: `.toString()`

Belirli Bir İndeksten Sonra Stringi Bölme: `.substring()`

Belirli Bir İndeksteki Karakteri Alma: `.charAt()`

Java Program Yapısı

Program Başlığı:

Program hakkındaki açıklamaları ya da ismini içeren ifade veya ifadelerdir.

```
//  açıklamalar veya program başlığı
```

Sınıf Çağırma Bölümü:

Java dilinde "*sınıf*"lar (*class*), "*paket*" (*package*) olarak adlandırılan dosyalarda toplanmışlardır. Diğer sınıfların, yazılacak programda kullanılabilmesi için önceden çağrılması gerekir. Herhangi bir Java programı yazıldığında, Java standart kütüphanesi (*java.lang* paketi) otomatik olarak çağrılır. Fakat kullanılacak diğer paketlere ait sınıflar, nesneler, fonksiyonlar kullanılacaksa bunların "*import*" ile çağrılması gerekir.

```
import paket.sınıf;
```


Java Program Yapısı

Örneğin `"import java.util.Scanner"` komutu ile Scanner sınıfı ilgili programda artık kullanılabilir veya `"import java.util.*"` ile de "java.util" paketindeki tüm sınıflar çağrılıp kullanılabilir.

Paket	Sınıfları
java.lang	Java programlama dilinin temel sınıfları
java.applet	Applet uygulamaları sınıfları
java.awt	Grafiksel arayüz uygulamaları sınıfları
java.io	Sistem giriş/çıkış sınıfları
java.sql	Veritabanı programlama sınıfları
javax.net	Ağ uygulamaları sınıfları

Java Program Yapısı

Sınıflar:

Java ile geliştirilen uygulamaların bileşenleri ".class" uzantılı dosyalarda saklanırlar. Java'da sınıf tanımlama en genel haliyle aşağıdaki gibi yapılır.

```
denetleyiciler class sınıfAdı{

        program kodları

}
```


Java Program Yapısı

Değişken Tanımlama:

Java'da kullanılacak değişkenler önceden bildirilmelidir.

```
veri tipi  değişken adı;
```

Sabit Tanımlama:

Java'da sabit tanımlamak için "final" kullanılmaktadır.

```
final veri tipi  sabit adı = sabit değeri;
```

Java Program Yapısı

Ekrana "merhaba" yazan programı inceleyelim.

```
//Merhaba
public class Merhaba {
    public static void main (String[ ] args) {
        System.out.println ("Merhaba");
    }
}
```

Dosya adı ile sınıf adının aynı olması gerekir. "main"deki "public" deyimi, sınıfın veya yöntemin herkese açık (dışarıdan erişilebilir) olduğunu belirtir. "static" deyimi sınıf tarafından paylaşıldığını, "void" de bir değer geri göndermediğini (dönmediğini) belirtir.

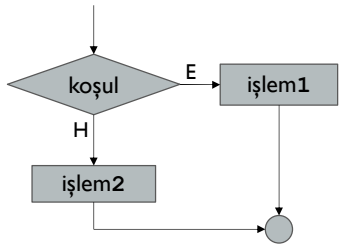
Akış Diyagramından Kodlamaya Geçiş

Sembollerin Karşılıkları

Şekil	Java'daki Karşılığı
Başla	Açıklamalar, bildirimler
Değişken	değişken=nextInt(); değişken=nextLine(); …
Değişken	System.out.println(değişken); System.out.print(değişken); …
İşlem	İşlem
değişken= başla,dur,adım	`for (başla; şart; adım) { }`

Sembollerin Karşılıkları

Şekil



Java'daki Karşılığı

```
if (koşul) {
      işlem1;
} else {
      işlem2;
}
```

Veri Giriş Komutları

Java'da klavyeden veri girişi için "java.util" paketindeki "Scanner" sınıfının yöntemleri (System.in) kullanılır. Bu nedenle programın başında "`import java.util.Scanner`" ile sınıf çağrılır. "Scanner" sınıfının bazı yöntemleri aşağıdaki gibi özetlenebilir.

Paket	Sınıfları
next()	Klavyeden girilen ifadeyi ilk özel karakterine (boşluk) kadar alır.
nextBoolean()	Klavyeden girilen ifadeyi boolean tipinde alır.
nextByte()	Klavyeden girilen ifadeyi byte tipinde alır.
nextDouble()	Klavyeden girilen ifadeyi double tipinde alır.
nextInt()	Klavyeden girilen ifadeyi int tipinde alır.
nextLine()	Klavyeden girilen tüm satırı alır.
nextLong()	Klavyeden girilen ifadeyi long tipinde alır.
nextShort()	Klavyeden girilen ifadeyi short tipinde alır.

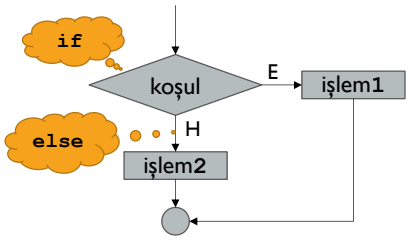
Veri Giriş/Çıkış Komutları

```
//Veri Girişleri
import java.util.Scanner;
public class VeriGiris {
     public static void main (String[ ] args) {
           String a;
           int b;
           Scanner klavye = new Scanner(System.in);
           System.out.print ("Bir cümle giriniz: ");
           a=klavye.nextLine();
           System.out.println ("Girdiğiniz cümle: "+a);
           System.out.print ("Bir sayı giriniz: ");
           b=klavye.nextInt();
           System.out.println ("Girdiğiniz sayı: "+b);
     }
}
```

Karar (Karşılaştırma) Komutları – IF ELSE YAPISI

Koşulların kontrolünde kullanılan komutlardır. Karar komutları farklı yapıda olabilirler:

- Yarım Form: Sadece koşul doğru ise yapılacak işlemler vardır.
- Tam Form: Koşul doğru olduğunda ve koşul yanlış olduğunda yapılacak işlemler vardır.
- Çok Koşullu Form: Birçok koşulun durumuna göre yapılacak işlemler vardır.



Karar (Karşılaştırma) Komutları – IF ELSE YAPISI

Yarım forma ilişkin Java'daki kodlama biçimi aşağıdaki gibidir. Görüldüğü üzere sadece koşul sağlanırsa çalıştırılacak komutlar vardır. Koşul sağlanmazsa program akışına kaldığı yerden devam eder.

```
if ( koşul ) {
    .....
    .....
}
```

Koşul Sağlanırsa Yapılacak İşlemler

Karar (Karşılaştırma) Komutları – IF ELSE YAPISI

Tam forma ilişkin Java'daki kodlama biçimi aşağıdaki gibidir. Görüldüğü üzere koşulun sağlanması ve sağlanmaması durumunda çalıştırılacak olan komutlar verilmiştir.

```
if ( koşul ) {
     .....
     .....
} else {
     .....
     .....
}
```

Koşul Sağlanırsa Yapılacak İşlemler

Koşul Sağlanmazsa Yapılacak İşlemler

Karar (Karşılaştırma) Komutları – IF ELSE YAPISI

Çok koşullu forma ilişkin Java'daki kodlama biçimi ise aşağıdaki gibidir.

```
if ( koşul1 ) {
     .....
     .....
} else if ( koşul2 ) {
     .....
     .....
} else {
     .....
     .....
}
```

Koşul1 Doğru ise Yapılacak İşlemler

Koşul1 Yanlış, Koşul2 Doğru ise Yapılacak İşlemler

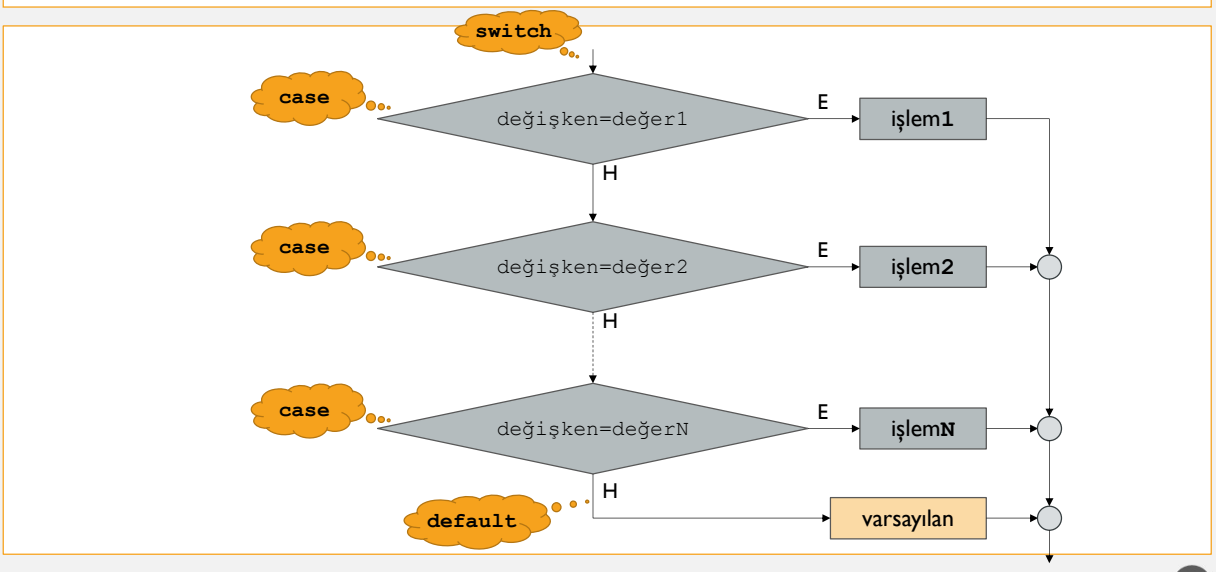
Koşul1 ve Koşul2 Yanlış ise Yapılacak İşlemler

```
if ( koşul1 ) {
    ……
    ……
} else if ( koşul2 ) {
    ……
    ……
} else if ( koşul3 ) {
    ……
    ……
}
```


Karar (Karşılaştırma) Komutları – IF ELSE YAPISI

```
//Karar 1
import java.util.Scanner;
public class ornek {
     public static void main (String[] args) {
            Scanner klavye = new Scanner(System.in);
            System.out.print ("Bir tamsayı giriniz: ");
            int a=klavye.nextInt();
            if (a>0) {
               System.out.println ("Pozitif");
            } else if (a<0){
               System.out.println("Negatif");
            } else{
               System.out.println("Sıfır");
            }
     }
}
```

Karar (Karşılaştırma) Komutları – SWITCH-CASE



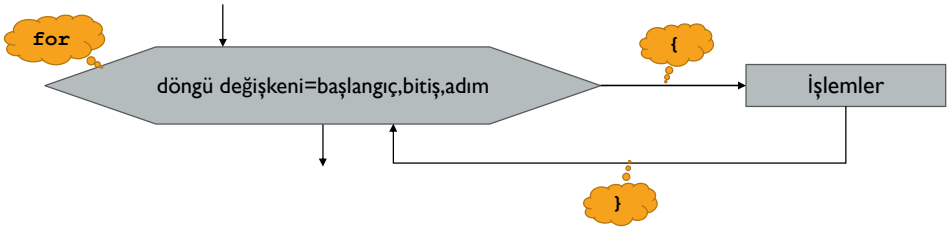

```
//Karar 2
import java.util.Scanner;
public class ornek {
      public static void main (String[] args) {
               Scanner klavye = new Scanner(System.in);
               System.out.print ("Notunuzu (1-5) giriniz: ");
               int a=klavye.nextInt();
               switch (a) {
                  case 1: {
                     System.out.println ("Çok zayıf");
                     break;
                  } case 2: {
                     System.out.println ("Zayıf");
                     break;
                  } case 3: {
                     System.out.println ("Orta");
                     break;
                  } case 4: {
                     System.out.println ("İyi");
                     break;
                  } case 5: {
                     System.out.println ("Çok iyi");
                     break;
                  } default: {
                     System.out.println("Geçersiz Not");
                     break;
                  }
               }
      }
}
```

Döngü Komutları

Tekrarlı işlemlerin yapılmasını sağlarlar. Döngüler üçe ayrılırlar:

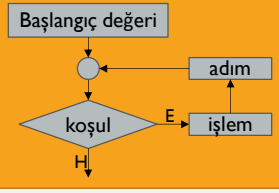
- Sayıcılı döngüler: Döngü işlemleri bir sayaca bağlı olarak gerçekleştirilir.
- Ön koşullu döngüler: Döngü işlemleri, döngü öncesinde kontrol edilen koşula bağlı olarak gerçekleştirilir.
- Son koşullu döngüler: Döngü işlemleri, döngü sonunda kontrol edilen koşula bağlı olarak gerçekleştirilir. Bu durumda döngü en az bir kez çalıştırılır.

Döngü Komutları - for




```
for (tip başlangıç değeri; koşul; adım) {
      .....
      .....                    } İşlemler
}
```

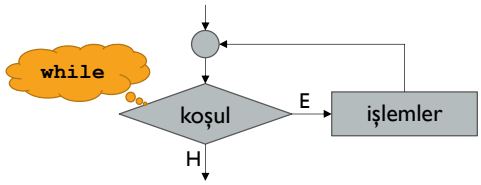
Alternatif gösterim



Döngü Komutları - for

```
//Döngü 1
public class ornek {
    public static void main (String[ ] args) {
          int t=0;
          int N=0;
          for (int i=1; i<=N; i++){
               t+=i;
          }
          System.out.println ("Birden N'e kadar sayıların
          toplamı: "+t);
    }
}
```


Döngü Komutları - while



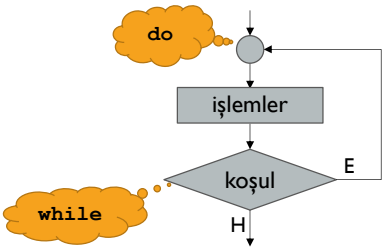

```
while ( koşul ) {
      ..........
      ..........
}
```

işlemler

Döngü Komutları - while

```
//Döngü 2
import java.util.Scanner;
public class ornek {
     public static void main (String[ ] args) {
            float t=0;
            Scanner klavye = new Scanner(System.in);
            System.out.print ("Tek sayıların üst sınırı: ");
            int N=klavye.nextInt();
            int i=1;
            while (i<=N) {
                  t+=i;
                  i+=2;
            }
            System.out.println("Toplam: "+t);
     }
}
```


Döngü Komutları - do while




```
do {
     .....
     .....
} while ( koşul );
```

} işlemler

Döngü Komutları - do while

```
//Döngü 2
import java.util.Scanner;
public class ornek {
     public static void main (String[ ] args) {
            float t=0;
            Scanner klavye = new Scanner(System.in);
            System.out.print ("Çift sayıların üst sınırı: ");
            int N=klavye.nextInt();
            int i=2;
            do {
               t+=i;
               i+=2;
            } while (i<=N);
            System.out.println("Toplam: "+t);
     }
}
```


ÖDEV

Bir satış elemanının sattığı ürün miktarına göre alacağı günlük ücret aşağıdaki gibi belirleniyor:

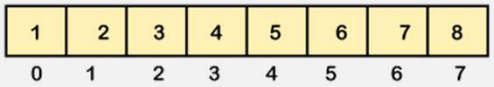
- Günlük satış miktarı 50 adetten az ise 15 PB tutarındaki sabit ücrete, satılan ürün başına 1 PB değerinde prim eklenerek günlük ücret belirlenir.

- Günlük satış miktarı 50 adet ya da daha fazla ise, bu durumda günlük sabit ücret 15 PB alınarak, satılan ürün başına da ilk 50 adet ürün için 2 PB, 50 adedi aşan kısım için de 3 PB prim verilerek günlük ücret belirlenir.

Bir satıcının günlük satış miktarı bilgisayara girildiğinde satıcının alacağı günlük ücreti hesaplayan bir Java programı yazınız.

Array Length in Java

Array Length = Array's Last Index + 1



Array's last index = 7
Arraylength = 7 + 1 = 8

Dizinin uzunluğu: 8

long form

```
double[] a;
a = new double[N];
for (int i = 0; i < N; i++)
   a[i] = 0.0;
```

declaration
creation
initialization

short form

```
double[] a = new double[N];
```

initializing declaration

```
int[] a = { 1, 1, 2, 3, 5, 8 };
```


Çalışma Sorusu:

Önceki slaytlarda (örneğin 5 ve 6 numaralı slaytlar) algoritmaları verilen programları JAVA dilinde kodlayınız.

Kaynaklar

- Algoritma Geliştirme ve Programlamaya Giriş, 13. Baskı, Fahri Vatansever, Seçkin Yayıncılık, 2017
- Algoritma: Uygulamalı Algoritma Klavuzu, 5. Baskı, Kadir Çamoğlu, KODLAB, 2011
- Algoritma ve Programlamaya Giriş, 6. Baskı, Ebubekir Yaşar, Ekin Basım Yayın, 2016
- Java ile Programlama, 3. Baskı, Timur Karaçay, Seçkin Yayıncılık, 2016
- Algoritma ve Programlamaya Giriş Ders Notları, Kadriye Ergün, Balıkesir Üniversitesi, Erişim Tarihi 5 Ocak 2018.
- Algoritma ve Programlama Ders Notu, Umut Engin Ayten, Yıldız Teknik Üniversitesi, Erişim Tarihi 5 Ocak 2018.

JAVA KODLAMA ÖRNEKLERİ

```
import java.util.*;

public class IlkProgram {

	public static void main(String[] args) {

		//şlsajdlksdjfşldskflşsdkfşlds ışlsıfposdıfşosdıf pos8df ı

		/*
		int a = 2;
		int b;

		int c;


		b = 16;

		c = a + b;
		c = a * b;
		System.out.print(c);
		System.out.println("c");
		System.out.println(a);
		System.out.println();
		System.out.println("Merhaba Dünya");
		System.out.println(c);
		 */
		/*
		c = b / a;

		System.out.println(2 * c);
		System.out.println(c);

		double sayi = Math.pow(a, 5.3);
		System.out.println(sayi);

		*/

		/*
		double kenar1 = 3;
		double kenar2 = 4;
		double kenar3;

		//kenar3 = Math.sqrt(kenar1*kenar1 + kenar2*kenar2);

		kenar3 = Math.sqrt(Math.pow(kenar1, 2) + Math.pow(kenar2, 2));

		System.out.println(kenar3);
		*/

		/*
		int a1 = 5;
		double a2 = 2;
		double a3 = 5/2.0;

		System.out.println("a3: " + a3);
		*/

		/*
		int sayi1 = 99;

		if (sayi1 > 100) {
			System.out.println("Sayı 100'den büyüktür");
		} else {
			System.out.println("Sayı 100'den büyük DEĞİLDİR");
		}
		*/

		/*
		int sayi1 = 131;
```

```
                if(sayi1 % 2 == 0) {
                        System.out.println("Sayı çifttir");
                } else {
                        System.out.println("Sayı tektir");
                }
                */

                Scanner klavye = new Scanner (System.in);




                int alinansayi;
                System.out.println("Lütfen bir sayı giriniz: ");
                alinansayi=klavye.nextInt();
                System.out.println("alinansayi: " + alinansayi);
                if(alinansayi%7==0) {
                        System.out.println("Alınan sayı 7ye tam bölünmektedir");
                } else {
                        System.out.println("Alınan sayı 7ye tam BÖLÜNMEMEKTEDİR");
                }

        }

}
```

```
import java.util.Scanner;

public class W12 {

	public static void main(String[] args) {
		//System.out.println("Algoritma ve Programlamaya Giriş");

		/*
		Scanner tara = new Scanner(System.in);
		System.out.println("Bir sayı giriniz");
		int sayi1 = tara.nextInt();
		System.out.println("Başka bir sayı giriniz");
		int sayi2 = tara.nextInt();
		double toplam = sayi1 + sayi2;
		System.out.print("Girdiğiniz sayıların toplamı: ");
		System.out.println(toplam);
		double kar1 = Math.sqrt(toplam);
		System.out.println("Girilen sayıların karekökü: "+ kar1);
		*/

		/*
		Scanner klavye = new Scanner(System.in);
		System.out.println("Lütfen ad-soyad giriniz");
		String adsoyad = klavye.nextLine();
		System.out.println("Girilen ad-soyad: " + adsoyad);
		*/

		/*
		Scanner klavye2 = new Scanner(System.in);

		System.out.println("Bir sayı giriniz: ");
		int sayi1 = klavye2.nextInt();
		System.out.println("Girilen sayı: " + sayi1);

		if(sayi1>0) {
			System.out.println("Girilen sayı sıfırdan büyüktür");
		} else if(sayi1<0) {
			System.out.println("Girilen sayı sıfırdan küçüktür");
		} else {
			System.out.println("Girilen sayi SIFIRDIR");
		}
		*/

		/*
		Scanner klavye = new Scanner(System.in);

		System.out.println("Vize Notunu Giriniz: ");
		int vizeNot = klavye.nextInt();

		System.out.println("Final Notunu Giriniz: ");
		int finalNot = klavye.nextInt();

		double ortalama = vizeNot*0.4 + finalNot*0.6;

		System.out.println("Not ortalaması: " + ortalama);

		if(ortalama>=50 && finalNot>=50) {
			System.out.println("Geçer Not");
		} else {
			System.out.println("Kalır Not");
		}
		*/

		/*
		Scanner klavye = new Scanner(System.in);

		System.out.println("İlk Sayıyı Giriniz: ");
		int s1 = klavye.nextInt();

		System.out.println("İkinci Sayıyı Giriniz: ");
		int s2 = klavye.nextInt();

		System.out.println("Üçüncü Sayıyı Giriniz: ");
```

```
                int s3 = klavye.nextInt();

                double kartop = Math.pow(s1, 2) + Math.pow(s2, 2) + Math.pow(s3, 2);
                double carpim = s1*s2*s3;
                if (kartop<carpim) {
                        System.out.println("Kareler Toplamı Çarpımdan Küçüktür");
                } else {
                        System.out.println("Kareler Toplamı Çarpımdan Küçük DEĞİLDİR");
                }

                */

                Scanner klavye = new Scanner(System.in);

                System.out.println("Lütfen yaş giriniz: ");
                int yas = klavye.nextInt();

                System.out.println("Öğrencilik durumu giriniz (Evet ise true, hayır ise false");
                boolean ogrenci = klavye.nextBoolean();

                if(ogrenci==true && yas <=30) {
                        System.out.println("Bilet fiyatı: 2.5 TL");
                } else if(ogrenci==true && yas >30 && yas<=60) {
                        System.out.println("Bilet fiyatı: 2.75 TL");
                } else if (yas>60){
                        System.out.println("Bilet fiyatı: 0 TL");
                } else {
                        System.out.println("Bilet fiyatı: 3 TL");
                }

        }
}
```

```
import java.util.Scanner;

public class W13 {

	public static void main(String[] args) {

		Scanner tara = new Scanner(System.in);
		/*
		//klavyeden iki sayı alıp karşılaştırmak
		System.out.println("İlk sayıyı giriniz: ");
		int a = tara.nextInt();

		System.out.println("İkinci sayıyı giriniz: ");
		int b = tara.nextInt();

		if (a > b) {
			System.out.println("a sayısı b'den büyüktür");
		} else if (b > a) {
			System.out.println("b sayısı a'dan büyüktür");
		} else {
			System.out.println("a ve birbirine eşittir");
		}
		*/

		/*
		//klavyeden alınan 10 adet sayının toplamının bulunması
		int toplam=0;
		for(int i=1; i<=10; i++) {
			System.out.println(i + ". sayıyı giriniz: ");
			int sayi1 = tara.nextInt();
			toplam=toplam+sayi1;
		}
		System.out.println("Girilen sayıların toplamı: " + toplam);
		*/

		/*
		//klavyeden alınan n adet sayının toplamının bulunması
		System.out.println("Kaç adet sayıyı toplamak istiyorsunuz?");
		int adet=tara.nextInt();

		int toplam=0;
		for(int i=1; i<=adet; i++) {
			System.out.println(i + ". sayıyı giriniz: ");
			int sayi1 = tara.nextInt();
			toplam=toplam+sayi1;
		}
		System.out.println("Girilen " + adet + " sayının toplamı: " + toplam);
		*/

		/* Üç basamaklı, 7ye tam bölünebilen en büyük sayı
		for(int i=999; i>=100; i--) {
			if(i%7==0) {
				System.out.print(i + " ");
				break;
			}
		}
		*/


		/* Alınan üç basamaklı bir sayının basamak değerleri toplamının ortalaması
		System.out.println("Üç basamaklı bir sayı giriniz: ");
		int alinan = tara.nextInt();

		while(alinan<100 || alinan>999) {
			System.out.println("Lütfen sayının üç basamaklı olduğundan emin olunuz!");
			System.out.println("Üç basamaklı bir sayı giriniz: ");
			alinan = tara.nextInt();
		}
		System.out.println("Alınan Sayı: " + alinan);
```

```
                int a=alinan/100; //yüzler basamağı
                int b=(alinan%100)/10; //onlar basamağı
                int c=alinan%10; //birler basamağı
                double ortalama= (a+b+c)/3.0;
                System.out.println(ortalama);
                */

                /*
                //klavyeden alınan pozitif bir sayının faktöriyelinin hesaplanması
                System.out.println("Faktöriyeli hesaplanacak sayıyı giriniz: ");
                int sayi = tara.nextInt();

                while(sayi<0) {
                        System.out.println("Pozitif bir sayı giriniz: ");
                        sayi = tara.nextInt();
                }
                System.out.println("Alınan sayı: " + sayi);

                double carpim = 1;
                int i=1;
                do{
                        carpim=carpim*i;
                        i++;
                }while(i<=sayi);
                System.out.println("Alınan sayının faktöriyeli: " + carpim);
                                */

                //Sayı tahmin oyunu
                int BS = (int) (Math.random()*100)+1; //1 ile 100 arasında rastgele tamsayı
                System.out.println("Lütfen tahmininizi giriniz: ");
                int tahmin=tara.nextInt();
                int sayac=1;
                while(tahmin!=BS) {
                        if(tahmin>BS) {
                                System.out.println("Daha küçük bir sayı giriniz: ");
                                tahmin=tara.nextInt();
                        } else if(tahmin<BS){
                                System.out.println("Daha büyük bir sayı giriniz: ");
                                tahmin=tara.nextInt();
                        }
                        sayac++;
                }
                System.out.println(sayac + " adımda doğru tahmin ettiniz");

        }
}
```

Selection Sort

Seçme Sıralaması

Selection Sort algoritması $O(n^2)$ grubuna giren bir sıralama yöntemidir. Dolayısıyla büyük sayıda verileri sıralamak için elverişsizdir. Bunun yanında, *bubble sort* algoritmasındaki takas işlemlerinin çoğunu yapmadığı için, bubble sort algoritmasının iyileştirilmiş biçimi sayılır. Çünkü takas işlemlerinin sayısını $O(n^2)$ den O(n) ye düşürür. Dolayısıyla daha etkilidir. Ancak, yaptığı mukayese işlemleri gene $O(n^2)$ düzeyindedir.

Algoritmanın çalışma ilkesi basittir.

1. Başlarken dizinin ilk öğesini en küçük öğe kabul eder. Tabii, bu kabul geçicidir. Sonra kalan terimler arasında daha küçükler varsa, onların en küçüğü olan terimle takas eder. O terimi en sola koyar; bu terim sıralama sonunda ilk terim olacaktır.
2. Diziden seçilen bu en küçük terimi çıkarınca, geri kalan alt dizine aynı yöntemi uygular. Altdiziden seçtiği en küçük öğeyi, ilk seçtiğinin sağına koyar. Dizinin sol ucunda iki terimli alt dizi küçükten büyüğe doğru sıralı bir altdizi oluşturur.
3. Bu işleme kalan altdizinler bitene kadar devam edilir. Her adım başlarken, sol yanda sıralı bir altdizi, sağ yanda sırasız bir alt dizi vardır. Her adımda sıralı dizi bir terim artar, sırasız dizi bir terim azalır. Sağ yandaki sırasız altdizi bitince, sıralama işlemi biter.

Örnek:

int [] *a = {3,17,86,-9,7,-11,38}* arrayi verilsin. Görselliği sağlamak için, bu arrayi

3 17 86 -9 7 -11 38

dizisi biçiminde yazalım. Bu dizinin öğelerini *selection sort* algoritması ile sıralayacağız. Dizinin (array) indislerinin 0 dan başladığını anımsayınız. *a[0] = 3,..., a[6] = 38* dir.

Selection sort algoritması, verilen diziyi sıralı ve sırasız olmak üzere iki alt diziye ayırır. Sırasız alt dizinin en küçük öğesini bulup seçer ve onu sıralı diziye en büyük öğe olarak katar.

Başlangıçta bütün dizi sırasızdır. Dizinin ilk öğesini seçip, tek öğeli (sıralı) bir alt dizi oluşturabiliriz. Geçici olarak, verilen dizinin ilk öğesini enküçük öğe *(minimal)* imiş gibi kabul edelim. Sonra mukayese ile daha küçük terim olup olmadığını araştıracağız. Ondan daha küçükler varsa, onların en küçüğünü *a[0]* ile takas edeceğiz. Böylece verilen dizinin en küçük terimi en sola yerleşir.

1.Aşama

Başlangıçta *a[0] = 3* olduğu için *minimal = 3* olur. Bu eşitliği sağlayan indise *minIndex* diyelim. İlk değeri *minIndex = 0* dır.

a[0]	*a[1]*	*a[2]*	*a[3]*	*a[4]*	*a[5]*	*a[6]*
3	*17*	*86*	*-9*	*7*	*-11*	*38*

Sonra *minimal* 3 sayısını, sırayla dizinin öteki terimleriyle karşılaştırarak, 3 den daha küçük öğe olup olmadığını, varsa onların en küçüğünün hangisi olduğunu arayalım. 17 ve 86 terimleri koşulu

sağlamaz; onları atlıyoruz. Üçüncü a[3] terimi ile mukayese yapınca -9 < 3 olduğunu görüyoruz. Bu durumda daha küçük olan -9 teriminin indisini *minIndex* yapıyoruz.

minIndex = 3

Bu andan sonra minimal öğemiz 3 değil -9 olmuştur. Ondan sonraki terimleri -9 ile karşılaştıracağız. 7 koşulu sağlamaz, onu atlıyoruz. Beşinci *a[5]* teriminde -11 < -9 olduğu için, *minIndex = 5* olur. Bu aşamada minimal öğe indisi 5 olan -11 öğesidir. Kalan 38 terimini -11 ile mukayese ediyor ve koşulu sağlamadığını görüyoruz. O halde dizinin en küçük öğesi indisi *minIndex = 5* olan -11 öğesidir. Dolayısıyla, 3 ile -11 öğelerinin yerlerini değiştiriyoruz (takas işlemi).

-11	17	86	-9	7	3	38	*minIndex = 5*

2.Aşama

Bu aşamada dizi *sıralı {-11}* ve *sırasız {17, 86, -9, 7, 3, 38}* olmak üzere iki altdiziye ayrılmıştır. Şimdi sırasız altdiziye yukarıdaki seçme algoritmasını uygulayarak, enküçük öğesini seçebiliriz. Bunun -9 olacağını görüyoruz. Alt dizinin ilk öğesi olan 17 terimi ile en küçük öğesi olan -9 terimlerinin yerlerini değiştiriyoruz (takas). Sonunda, -9 terimini sıralı alt diziye ekliyoruz:

-11	-9	86	17	7	3	38	*minIndex = 3*

3.Aşama

Bu aşamada dizi *sıralı {-11, -9}* ve *sırasız {86, 17, 7, 3, 38}* olmak üzere iki altdiziye ayrılmıştır. Şimdi sırasız altdiziye seçme algoritmasını uygulayarak, enküçük öğesini seçebiliriz. Bunun 3 olacağını görüyoruz. Alt dizinin ilk öğesi olan 86 terimi ile en küçük öğesi olan 3 terimlerinin yerlerini değiştiriyoruz (takas). Sonunda, 3 terimini sıralı alt diziye ekliyoruz:

-11	-9	3	17	7	86	38	*minIndex = 0*

4.Aşama

Bu aşamada dizi *sıralı {-11, -9, 3}* ve *sırasız {17, 7, 86, 38}* olmak üzere iki altdiziye ayrılmıştır. Şimdi sırasız altdiziye seçme algoritmasını uygulayarak, enküçük öğesini seçebiliriz. Bunun 7 olacağını görüyoruz. Alt dizinin ilk öğesi olan 17 terimi ile en küçük öğesi olan 7 terimlerinin yerlerini değiştiriyoruz (takas). Sonunda, 7 terimini sıralı alt diziye ekliyoruz:

-11	-9	3	7	17	86	38	*minIndex = 4*

5.Aşama

Bu aşamada dizi *sıralı {-11, -9, 3, 7}* ve *sırasız {17, 86, 38}* olmak üzere iki altdiziye ayrılmıştır. Şimdi sırasız altdiziye seçme algoritmasını uygulayarak, enküçük öğesini seçebiliriz. Bunun 17 olacağını görüyoruz. Alt dizinin ilk öğesi zaten 17 terimidir. Dolayısıyla bir takas işlemi gerekmiyor. Sonunda, 17 terimini sıralı alt diziye ekliyoruz:

-11	-9	3	7	17	86	38	*minIndex = 2*

6.Aşama

Bu aşamada dizi *sıralı {-11, -9, 3, 7, 17}* ve *sırasız {86, 38}* olmak üzere iki altdiziye ayrılmıştır. Şimdi sırasız altdiziye seçme algoritmasını uygulayarak, enküçük öğesini seçebiliriz. Bunun 38 olacağını görüyoruz. Alt dizinin ilk öğesi olan 86 terimi ile en küçük öğesi olan 38 terimlerinin yerlerini değiştiriyoruz (takas). Sonunda, 38 terimini sıralı alt diziye ekliyoruz:

-11	-9	3	7	17	38	86	*minIndex = 6*

7.Aşama

Bu aşamada dizi *sıralı {-11, -9, 3, 7, 17, 38}* ve *sırasız {86}* olmak üzere iki altdiziye ayrılmıştır. Sırasız dizi tek öğeli olduğu için, en küçük öğesi kendisidir. 86 terimini sıralı alt diziye ekliyoruz:

-11	-9	3	7	17	38	86	*minIndex = 2*

Böylece verilen dizi sıralanmış olur.

Şimdi selection sort algoritmasını yapan bir java metodu yazalım:

```
void selectionSort(int [] dizi,int n)
{
    int yedek;
    int minIndex;
    for(int i=0; i<n-1; i++)
    {
        minIndex=i;
        for(int j=i; j<n; j++)
        {
            if (dizi[j] < dizi[minIndex]) minIndex=j;
        }
        temp=dizi[i];
        dizi[i]=dizi[minIndex];
        dizi[minIndex]=yedek;
    }
}
```

Yukarıdaki metodu bir java uygulamasında çalıştıralım:

```
package sıralama;

public class SelectionSort {
   int[] a = {3,17,86,-9,7,-11,38}  ;

   void selectionSort(int [] dizi)
   {
      int yedek;
      int minIndex;
      for(int i=0; i< dizi.length; i++)
      {
         minIndex=i;
         for(int j=i; j<dizi.length; j++)
         {
            if (dizi[j] < dizi[minIndex]) minIndex=j;
         }
         yedek=dizi[i];
         dizi[i]=dizi[minIndex];
         dizi[minIndex]=yedek;
      }
   }

 void diziYaz(int[] arr){
      for(int i=0; i < arr.length;i++){
         System.out.print(arr[i] + " ");
      }
   }

   public static void main(String[] args) {
      SelectionSort ss = new SelectionSort();
      System.out.println("\nSıralamadan önce:");
      ss.diziYaz(ss.a);
      ss.selectionSort(ss.a);
      System.out.println("\n\nSıralamadan sonra:");
      ss.diziYaz(ss.a);

   }
}
```

```
public class W14 {

	public static void main(String[] args) {

		int [] numbers = {3, 6, -1, 5, 9, 5, 8, 56, 42, -4, 8};
		/*
		System.out.println(numbers[0]+numbers[10]);
		System.out.println(numbers.length);
		System.out.println(numbers[numbers.length-1]);
		*/
		//dizinin tüm elamanlarını yazdırma
		for(int i=0; i<numbers.length; i++) {
			System.out.print(numbers[i] + " ");
		}
		System.out.println();

		//dizinin sıfırdan büyük elemanlarını yazdırma
		for(int i=0; i<numbers.length; i++) {
			if(numbers[i]>0) {
				System.out.print(numbers[i] + " ");
			}
		}

		System.out.println();
		//dizinin çift indekslerindeki elemanlarını yazdırma
		for(int i=0; i<numbers.length; i=i+2) {
				System.out.print(numbers[i] + " ");
		}

		System.out.println();
		//dizinin çift elemanlarını yazdırma
		for(int i=0; i<numbers.length; i++) {
			if(numbers[i]%2==0) {
				System.out.print(numbers[i] + " ");
			}
		}
	}
}
```

```
import java.util.Scanner;

public class W14_2 {

	public static void main(String[] args) {

		int [] dizi1 = new int[5];
		dizi1[0]=3;

		dizi1 [dizi1.length-1]= dizi1.length ;
		for(int i=0; i<dizi1.length; i++) {
			System.out.print(dizi1[i]+" ");
		}

		System.out.println();
		int sayac=0;
		for(int i=0; i<dizi1.length; i++) {
			if(dizi1[i]!=0) {
				sayac=sayac+1;
			}
		}
		System.out.println("Dizinin " + sayac + " elemanı sıfırdan farklıdır");

		System.out.println();
		int toplam=0;
		for(int i=0; i<dizi1.length; i++) {
			toplam=toplam+dizi1[i];
		}
		System.out.println("Dizi elemanlarının toplamı: " + toplam);









	}

}
```

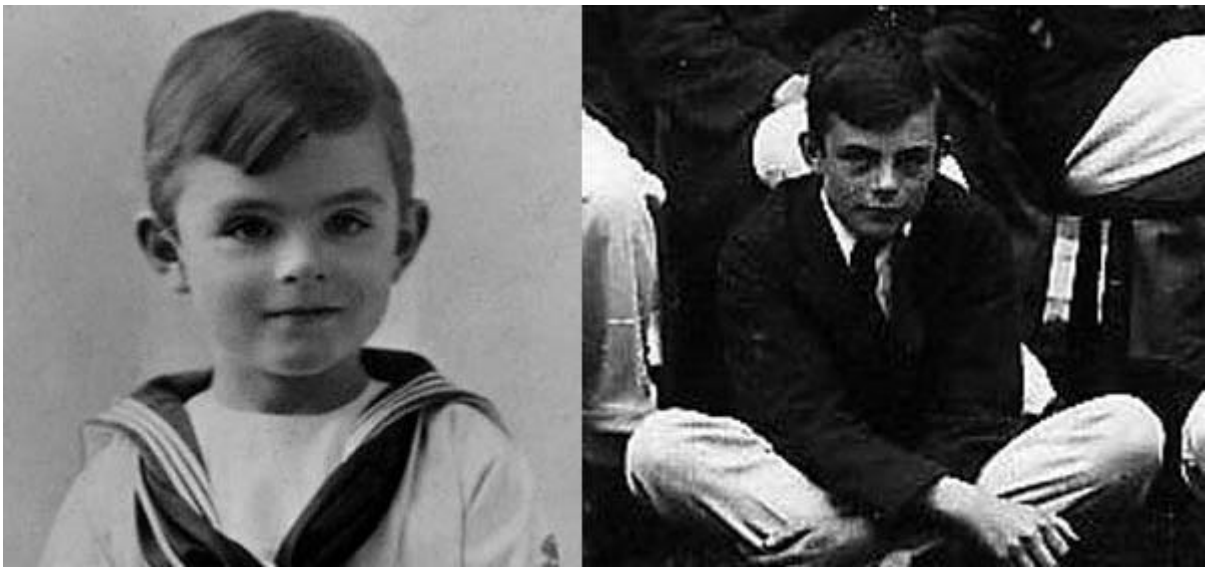
```
public class W14_3 {

	public static void main(String[] args) {
		int[] a = { 3, 17, 86, -9, 7, -11, 38 };

		int yedek;
		for (int i = 0; i < a.length; i++) {
			int minIndex = i;
			for (int j = i+1; j < a.length; j++) {
				if (a[j] < a[minIndex]) {
					minIndex = j;
				}
			}
			yedek = a[i];
			a[i] = a[minIndex];
			a[minIndex] = yedek;
		}
		for (int i = 0; i < a.length; i++) {
			System.out.print(a[i] + " ");
		}



	}
}
```

O elmayı ısıran adam Alan Turing

20/02/2015 10:06 Haber: Oktay Volkan Alkaya -oktay.alkaya@radikal.com.tr / Arşivi

Muhtemelen bu yazıyı okumanıza vesile olan cihazın temelini atan kişi olarak, işletim sisteminin sahibi olan firmanın kurucusu olduğunu düşünüyorsunuzdur. Ancak aslında tüm bunların ardında İngiliz bir bilim insanı var: Alan Turing. Peki kim bu alan Turing? İşte size büyük bir dehanın hikayesi...

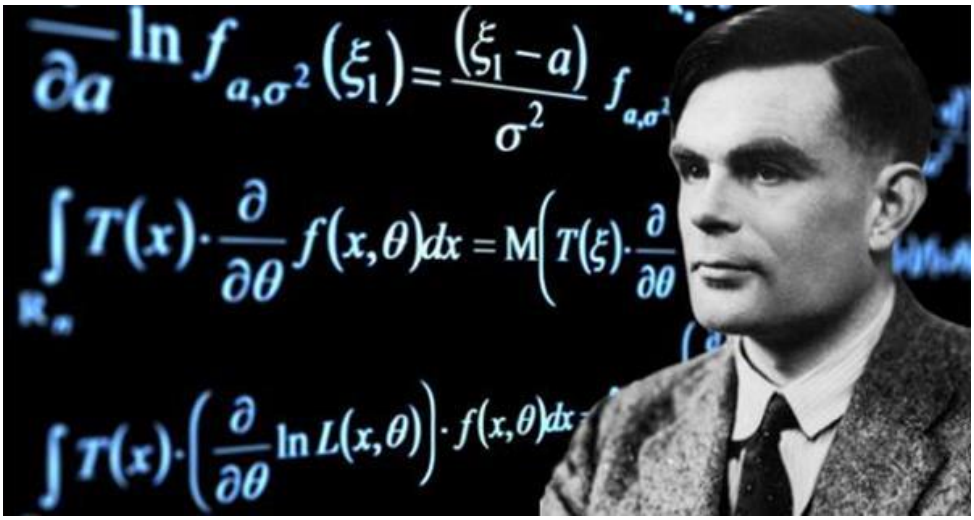
İkinci **Dünya** Savaşı'nın ve geleceğin teknolojisinin kaderini değiştiren adam olarak bilinen Alan Turing, tüm çağların en dahi bilim insanlarından biriydi. Adı Einstein kadar bilinmese de, Steve Jobs gibi modern **zaman** efsanelerinden olmasa da aslında Turing'in başardıkları, bilim yolunda pek çok buluşun yolunu açan bir ışık gibidir.



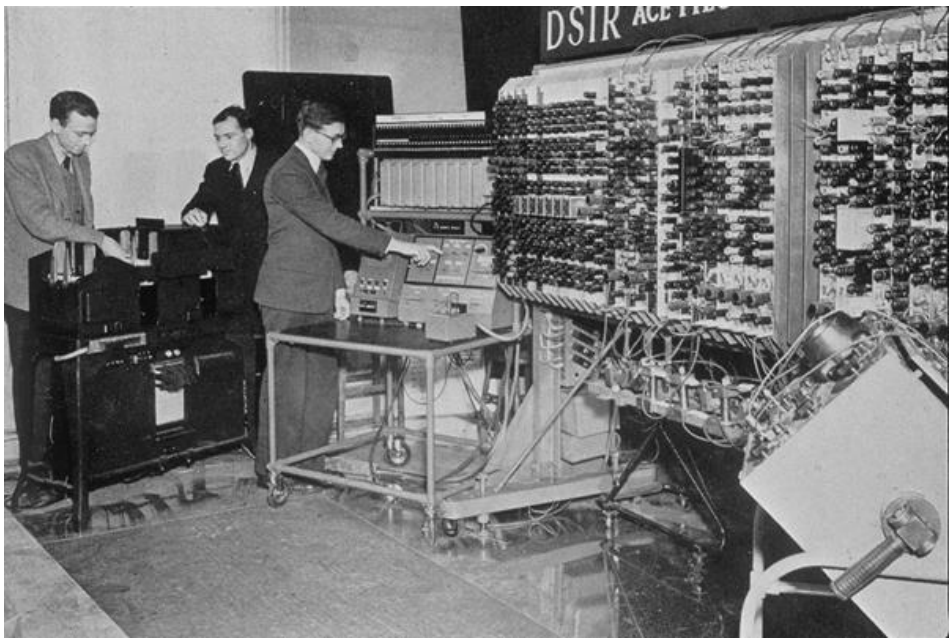
Turing'in hayatı o dönemler İngiliz sömürgesi olan Hindistan'da çalışan anne ve babasının, 1900'lerin başında onun doğumu için İngiltere'ye gelmesiyle başlar. Küçük

yaşlarda çok zeki bir çocuk olduğunu belli eden Turing, 6 yaşında gündüz okuluna gitmeye başlar ve eğitim hayatı boyunca matematik bilimine odaklanır. Turing 1928'de henüz 16 yaşındayken Albert Einstein'ın çalışmasıyla karşılaştı; onu kavramakla kalmadı; bunu Einstein'ın Newton hareket savlarını kendi kendine çalışak ortaya çıkardı.

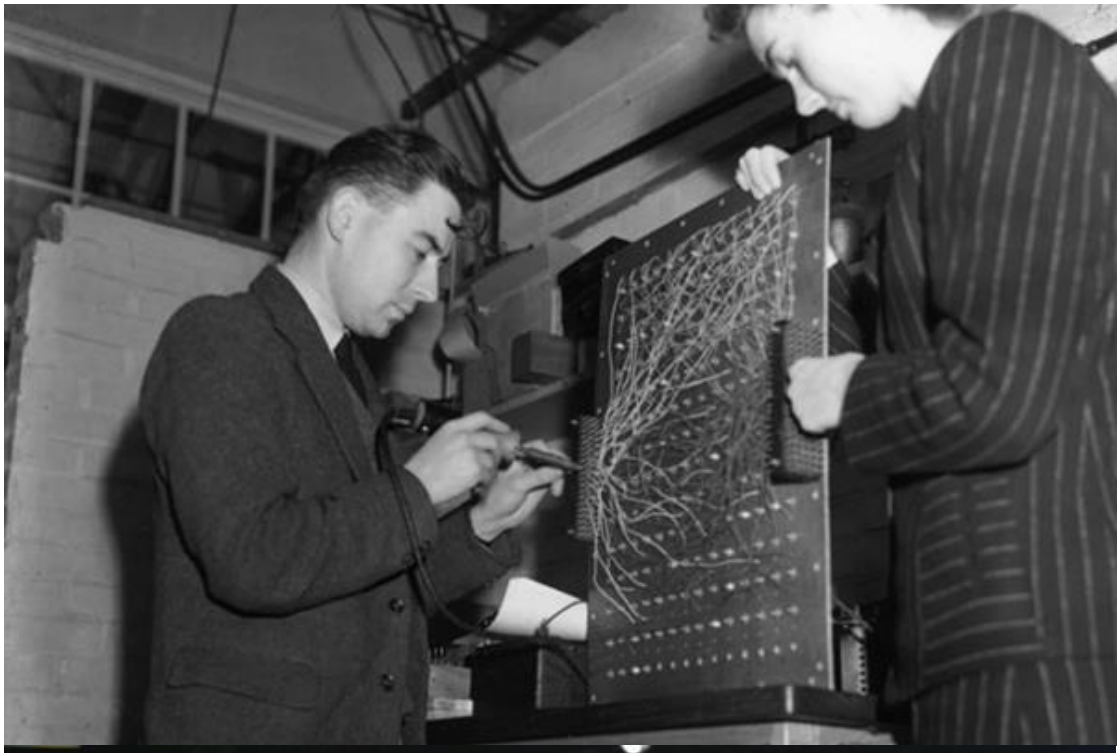
Alan Turing bu dönemde okulda kendinden yaşça biraz daha büyük akademik öğrenci Christopher Morcom'la yakın arkadaşlık ve aşk ilişkisi kurdu. Morcom, tüberküloz hastalığı nedeniyle, Sherborne'daki son sömestirinin bitmesinden sadece birkaç hafta kala öldü. Bu olaydan çok etkilenen Turing'in dini inancı yıkıldı ve ateist oldu. İnsan beyninin çalışması da dâhil, tüm dünya fenomenlerinin materiyalistik olduğu inancını benimsedi.

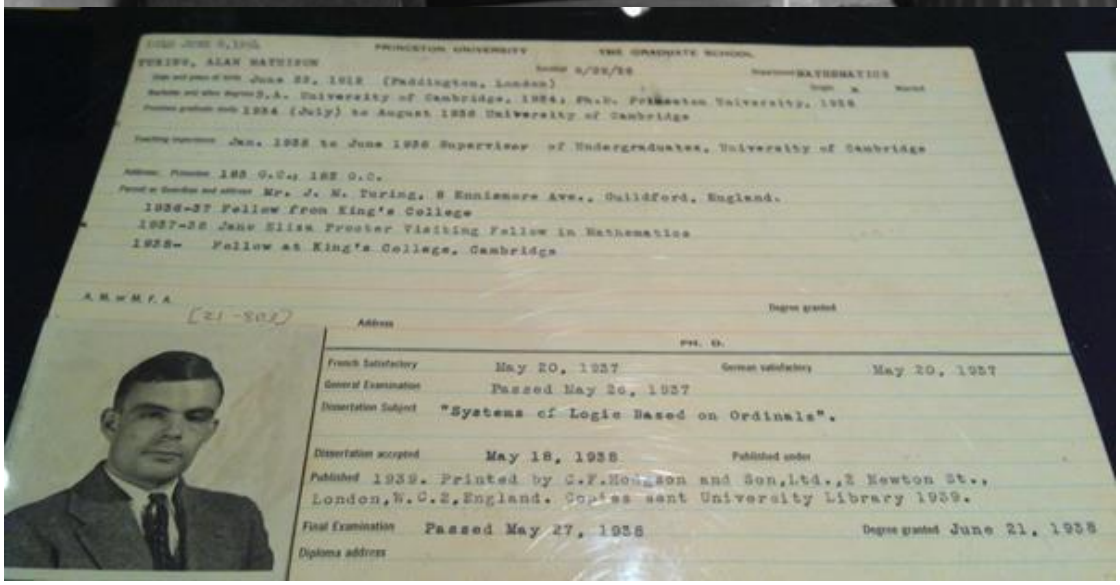


Turing'in klasik eski Yunanca ve Latince çalışmalara istekli olmaması ve matematik ve bilimi daima tercih etmesi onun Cambridge Trinity Koleji'ne bir burs kazanmasına engel oldu. İkinci tercihi olan Cambridge Kings Kolej'e gitti. 1931'den 1934'e kadar orada öğrenciydi, seçkin bir dereceyle diploma aldı ve merkezi limit teoremi üzerinde hazırladığı bir tez yazısı dolayısıyla 1935'te Kings Kolej'e akademik üye seçildi. 28 Mayıs 1936'da sunduğu Hesaplanabilir Sayılar: Karar Verme Probleminin bir Uygulaması adlı çok önemli bir makalesinde, Kurt Gödel'in 1931'de evrensel aritmetik-tabanlı biçimsel diliyle hazırladığı hesaplama ve kanıtın sınırları ispat sonuçlarını yeniden formüle ederek, onun yerine şimdi Turing makineleri diye andığımız, daha basit ve formel usullere dayanan ispatı ortaya attı. Eğer bir algoritma ile temsil edilmesi mümkün ise düşünülmesi mümkün olan her türlü matematiksel problemin böyle bir çesit makine kullanılarak çözülebileceğini ispat etmiş oldu.



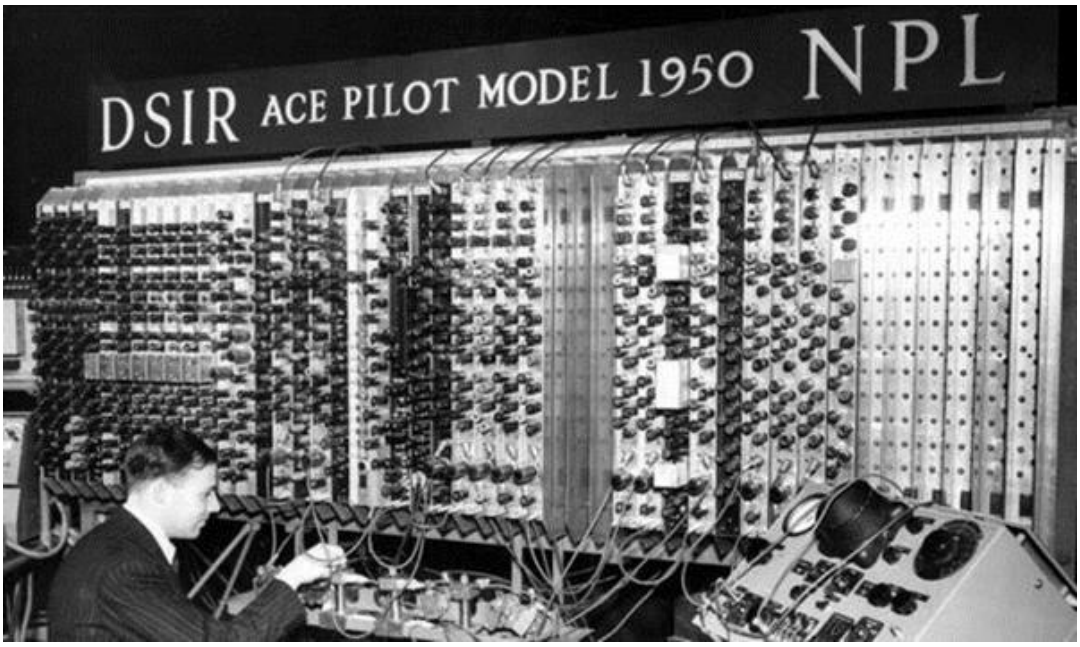
İkinci Dünya Savaşı sırasında, Turing, Bletchley Park'ta Alman şifrelerini kırma girişimlerinde baş katılımcılardan biriydi. Savaştan önce Marian Rejeski, Jerzy Rozycki ve Henryk Zygalski tarafından Polonya Şifre Bürosunda geliştirilen kriptanaliz üzerine eklemeler yaptı. Hem Enigma makinası hem de bu makinaya eklenen Lorenz SZ 40/42 makinasının şifrelerinin kırılmasına birçok anlayışla katkıda bulundu. Turing, Eylül 1938 itibariyle Hükümet Kod ve Şifre Okulu adındaki, İngiliz şifre kod kırma organizasyonunda yarı-zamanlı çalışmaya başladı. Alman Enigma makinası problemi üzerinde çalıştı ve GCCS'de kıdemli kod kırıcı Dilly Knox'la işbirliği yaptı. 4 Eylül 1939'da, Birleşmiş Krallık'ın Almanya'ya karşı savaş ilan etmesinin ertesi günü, Turing askeri hizmet görmek için GCCS'nin savaş zamanı üssü Bletchley Park'a katıldı.



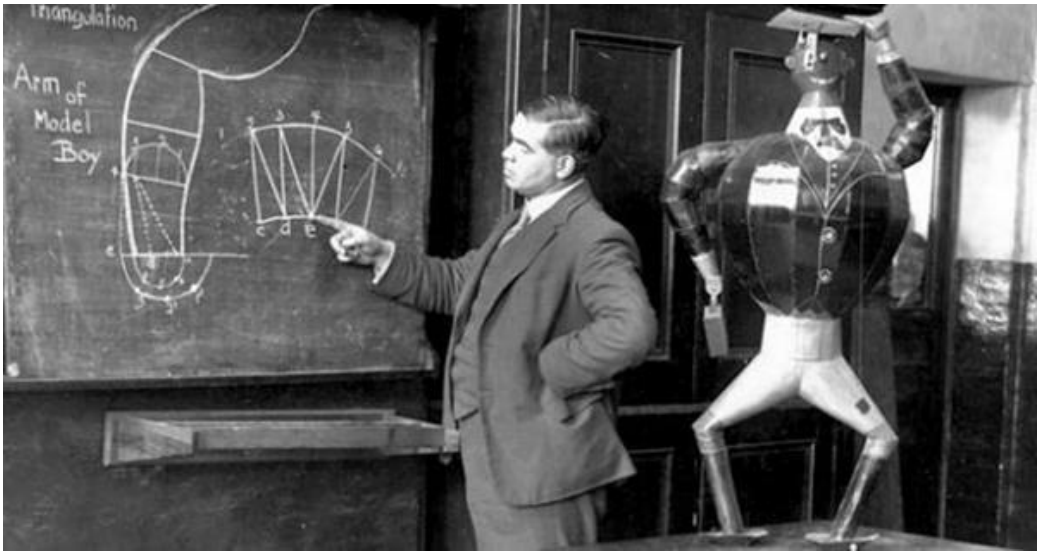


Bletchley Park’a katılışından birkaç hafta sonra, Turing Enigma’yı hızlı kırmaya yardımcı olacak elektromekanik bir makine tasarladı; bu makinaya Bombe adı daha önce 1932'de Polonya tasarımlı makinelerinden geliştirilmiş olan cihaza verilen Bomba adına atıfla verildi. Matematikçi Gordon Welchman’ın önerileriyle eklemelerle, Bombe Enigma, korumalı mesaj trafiğine saldırmada en onemli ve tek tam otomatikleştirilmiş kod kırma makinası olarak kullanıldı. Turing ile aynı dönemde Bletchley Park’ta kriptanaliz üzerine çalışan Profesör Jack Good daha sonra Turing'i şu sözlerle onurlanmdırmıştır: "Turing'in en önemli katkısı, bence, kriptanalitik makine Bombe’nin tasarımıdır. Bunun esası eğitilmemiş bir kulak için çok saçma gelen bir mantık teoremine, hatta herşeyi anlayabileceğimizin muhtemel olduğuna dair çelişkili bir fikre dayanmaktaydı."

Aralık 1940’ta Turing Hut-8 adında bir ekiple çalışmaya başladı. Diğer servislerin kullandığı göster geç sistemlerinden daha karmaşık olan, deniz kuvvetleri Enigma göster geç sistemini çözdü. Turing ayrıca Deniz Kuvvetleri Enigmasını kırmaya yardımcı olması için ‘Banburismus’ adı verilen Bayes tipi istatistik tekniği keşfetti. Çok da iyi bir koşucu olan Turing 1941 baharında, Hut-8’deki iş arkadaşı Joan Clarke’a evlilik teklifinde bulundu, ancak yazın her iki tarafın anlaşmasıyla bu nişan bozuldu.

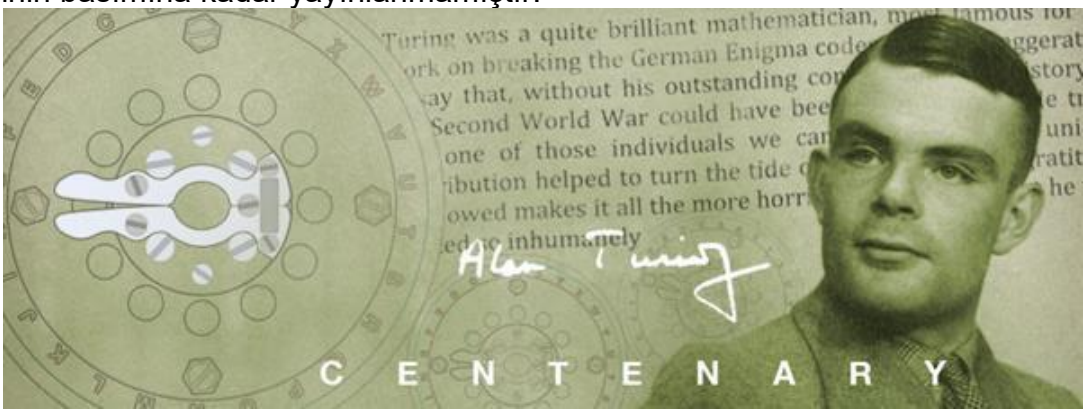


Turing 1945'ten 1947'ye kadar ACE (Otomatik Bilgisayar Motoru) tasarımında çalıştığı Ulusal Fizik Laboratuvarı'ndaydı. 19 Şubat 1946'da ilk program-hafızalı bilgisayarın detaylı dizaynının makalesini sundu. ACE uygulanabilir bir dizayn olmasına rağmen, Bletchley Park'taki savaş zamanı çalışmalarını saran esrarengizlik proje başlangıcının ertelenmelerine öncülük etti ve onu hayal aleminden çıkardı. 1947'nin sonlarında altı yıllık devamlı çalışmadan sonra kendi istedigi bir alanda istediği gibi çalışmak üzere Cambridge'e döndü. O Cambridge'teyken yokluğunda Pilot ACE yapıldı. İlk programı 10 Mayıs 1950'de gerçekleştirildi. Bu ödnemde Turing yapay zekaya yöneldi ve şu anda Turing testi olarak bilinen, bir makine için 'zeki' denilebilme standardını saptama girişimi olan bir deney ileri sürdü. İddiası eğer soru soran kişiyi, diyalog içerisinde olduğunun bir insan olduğu konusunda kandırabilirse, bir bilgisayar için düşünmenin söz konusu olabileceğiydi.



1948’te Turing aynı sınıftan mezun olduğu meslektaşı D.G. Champernowne ile çalışırken henüz var olmayan bir bilgisayar için satranç programı yazmaya başladı. 1952’de programı gerçekleştirmeye yetecek kadar bir bilgisayarı güçlendirerek, Turing bilgisayarını taklit ettiği, her bir hamlesi yaklaşık yarım saat alan bir oyun oynadı. Oyun kaydedildi, Champernowne’nın karısına karşı oyunu kazandığı söylense bile, program Turing’in meslektaşı Alick Glennie’ye karşı kaybetmiştir.

Turing 1952’den 1954’teki ölümüne kadar matematiksel biyoloji, özellikle morfogenez üzerine çalışmıştır. 1952’de Turing örnek biçimlendirme hipotezini öne sürerek, ‘ Morfogenezin Kimyasal Temeli ‘ adlı bir makale yazmıştır. Bu alandaki ilgi odağı canlıların yapısındaki Fibonacci numaralarının varlığını, Fibonacci filotaksisini anlamaktır. Örnek biçimlendirme alanının şu an merkezi olan reaksiyon-difüzyon denklemini kullanmıştır. Son makaleleri 1992’de A.M. Turing’in Derleme Çalışmaları eserinin basımına kadar yayınlanmamıştır.

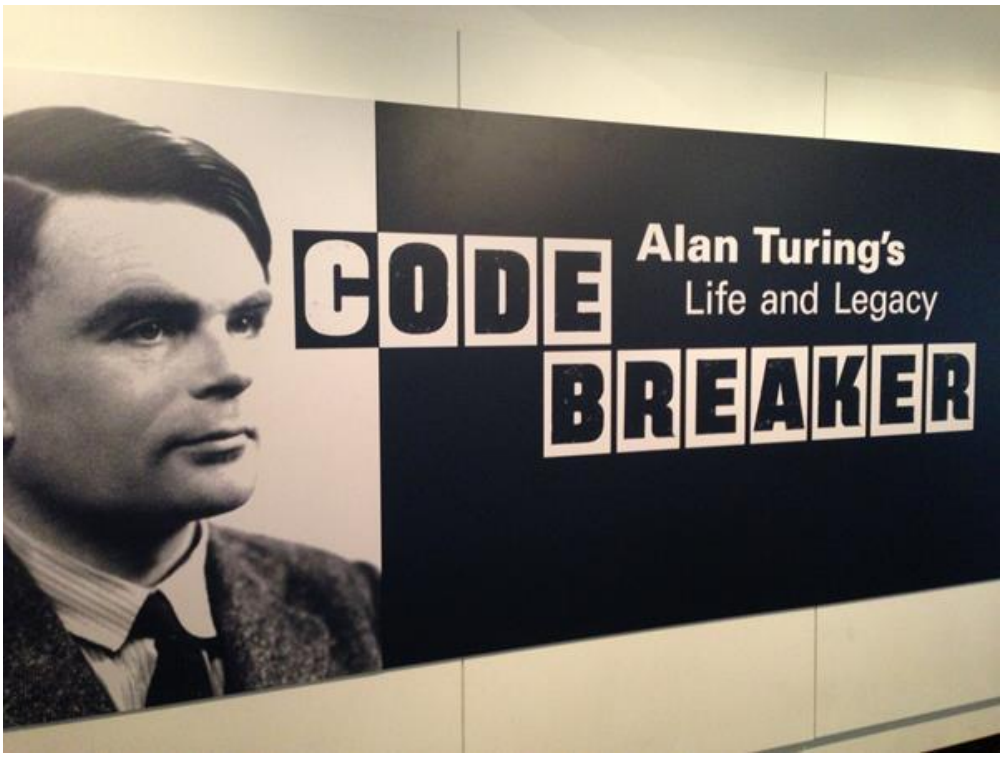


Özel hayatında ise Turing sıkıntılı bir ömür sürmüştü. Homoseksüellik İngiltere'de yasadışıydı ve bir akıl hastalığı olarak dikkate alınmakla birlikte ceza-i yaptırımı olan suç sınıfına girmekteydi. Ocak 1952'de Turing'in 19 yaşında bir genç olan Alan Murray ile bir sinemada tanıştı ve Alan Murray birkaç defa Turing'in evine giderek onunla birlikte kaldı. Birkaç hafta sonra Alan Murray bir tanıdığı ile birlikte Turing'in evini soymaya gitti. Turing bu hırsızlığı polise bildirdi. Polis hırsızları yakaladı ve soruşturma sırasında Alan Murray'in Turing ile homoseksüel ilişkisi olduğu gerçeği ortaya çıktı. Turing de bunun gerçek olduğunu itiraf etti. Turing ve Murray 1885 Ceza Kanunu'na Ek Yasa'nın 11. Kısmı gereğince müstehcen uygunsuzluktan suçlanıp mahkemeye verildiler. Turing pişman değildi ve 50 yıl önce Oscar Wilde'ın başına geldiği gibi aynı suçtan mahkûm edildi.

8 Haziran 1954’te temizlikçisi onu Manchester'deki evinde ölü buldu. Bir gün evvel, yatağının kenarında bıraktığı yarı-yenmiş siyanür-zehirli elmayı yemek suretiyle siyanür zehirlenmesinden öldüğü açıklandı. Elmanın kendisi nedense hiçbir siyanür zehiri testine tabi tutulmadı. Ölüm sebebinin siyanür zehirlenmesi olması iddiasına rağmen naaşına post-mortem yapılmadı. Peri masalı Pamuk Prenses hikayesindeki gibi bu ölümün ardında Turing'e bir suikast düzenlenmiş olması kuvvetli bir ihtimal olarak görülmektedir. Turing ölümünden sonra aldığı ödüller ve adına yapılan anıtlar

ve heykellerle onurlandırılmış bir isim olarak bilim tarihinin en önemli isimlerinden biridir.

Turing'in Elmayı ısırarak hayatına son vermesi ise ölümünün ardından çok büyük bir dünya markasına ilham kaynağı olduğu rivayet edilir. Bilgisayar teknolojisinin babası olarak görülen Turing'in ardında ısırılmış bir elma bırakması, modern zamanların en popüler ısırılmış elmasıyla bağdaştırılır. Bu elma elbette ki "Apple" şirketinin logosudur. İlginç bir şekilde Apple'ın ilk logosu da LGBT renklerini taşımasıyla sanki Turing'in eşcinsel hayatına bir gönderme niteliğindedir. Ve ilginç bir şekilde bugün Steve Jobs'un ardından Apple'ın başında dünyanın en meşhur ve güçlü eşcinseli bulunmaktadır. Apple yöneticileri her ne kadar Turing'le ilgili bir logo ya da renk tercihinde bulunmadıklarını iddia etseler de, söz konusu 'şehir efsanesi' gerçek olarak kabul görmektedir.



Turing'in hayatı şimdiye kadar 3 farklı filmde konu edilmiştir. 2013'te Alan Turing, Le Code De La Vie ve 1996 yapımı olan Breaking The Code adlı filmlerde Turing'in hayatı anlatılmıştır. 3. ve son film olan The Imitation Game ise 2014 yapımı olup ülkemizde Şubat 2015'in üçüncü haftasında vizyona girmiştir. Şu ana kadar yapılan en iddialı Alan Turing filmi olan Imitation Game'e yakından bakacak olursak;

2014 Sonbahar sezonunda gelmesi beklenen ancak ertelenen gösterim tarihi sebebiyle Şubat'a kadar beklememiz gereken "Yapay Oyunlar" , Norveçli yönetmen Morten Tyldum'un elinden çıktı. Tyldum'un ilk büyük çalışması olan yapım özellikle oyunculuk performanslarıyla şimdiden Oscar'a aday bir proje. Benedict Cumberbatch 'ın 2. Dünya Savaşı sırasında Enigma kodunu kıran Alan Turing'i canlandırdığı filmde, Matthew Goode, Charles Dance, Keira Knightley ve Mark Strong'dan oluşan geniş ve güçlü bir kadro karşımıza çıkıyor. Kasım ayında; kod çözme, casusluk, ve vatana ihanet konularını irdeleyecek olan Imitation Game, kuru gürültü 2. Dünya Savaşı filmlerinin uzağında elit bir yapım görünümünde.