

Balıkesir Üniversitesi, Mühendislik Fakültesi
Endüstri Mühendisliği Bölümü

BİLİ 202
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



Ders Planı, Değerlendirme Kriterleri, Yararlanılacak Kaynaklar, İçerik

DERSLE İLGİLİ BİLGİLER

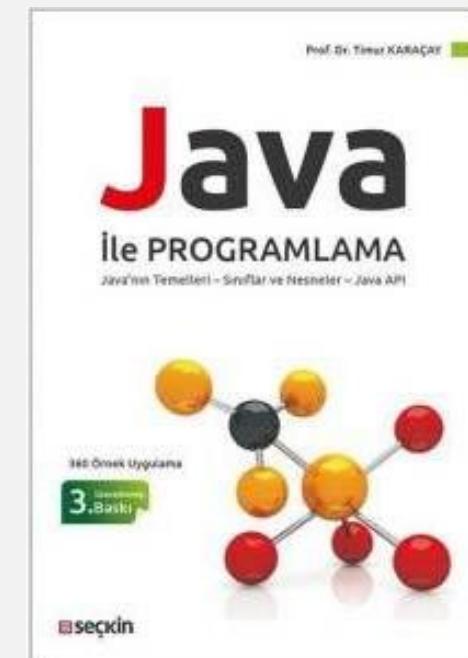
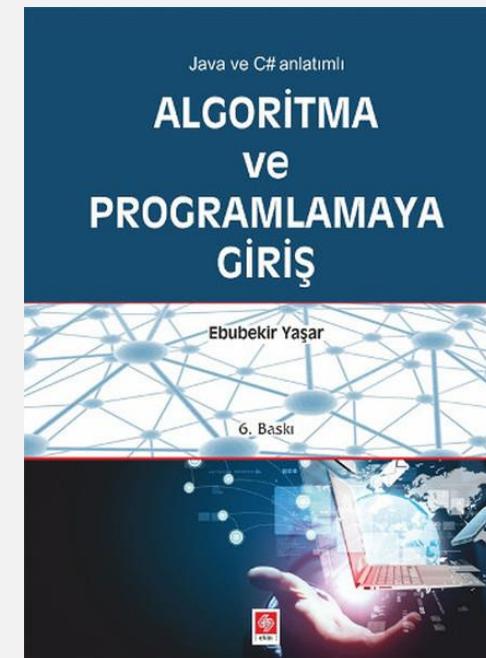
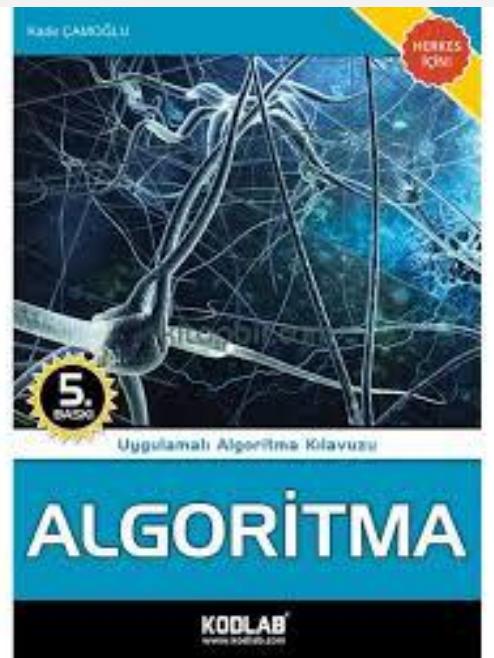
- BIL1202 Algoritma ve Programlamaya Giriş
- Dersin Amacı:
 - Öğrencilerin temel algoritma ve problem çözme yapıları hakkında bilgi sahibi olmasını ve bir probleme karşılaşıklarında bu temel yapıları kullanarak algoritmalar tasarlayabilme ve bu algoritmaları yapısal bir programlama dili ile gerçekleştirebilme 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. konuya 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

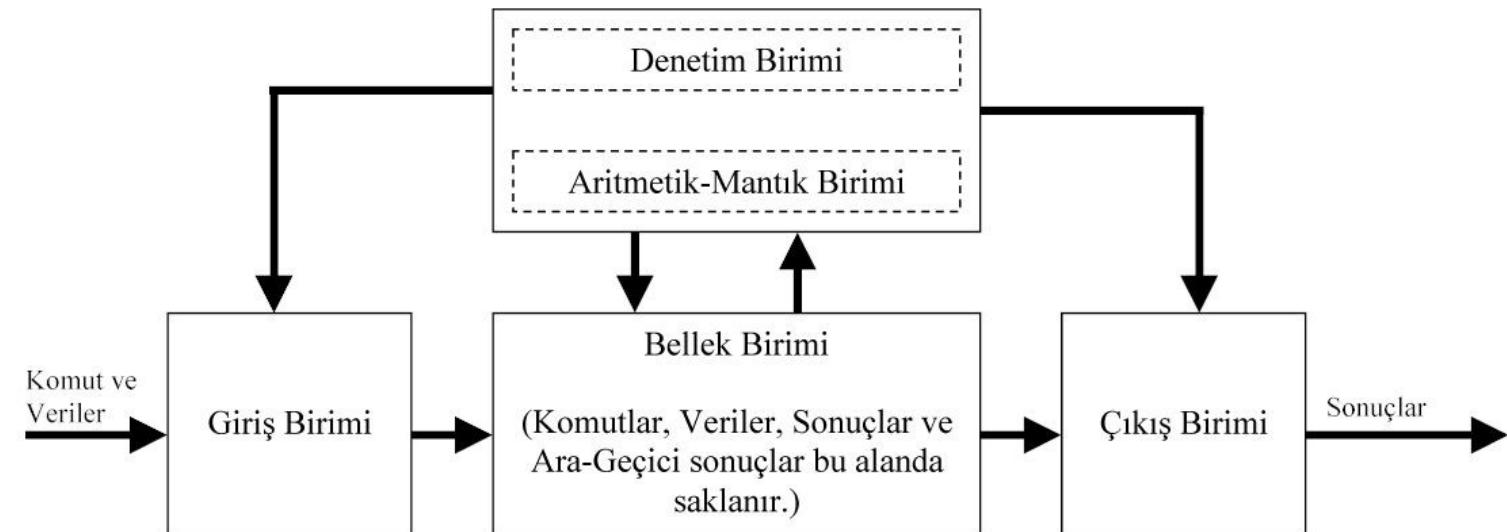
- **İÇERİK**

- 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 algoritmalarından 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

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

Yazılım geliştirme sonucu ortaya çıkan ürüne program denir.

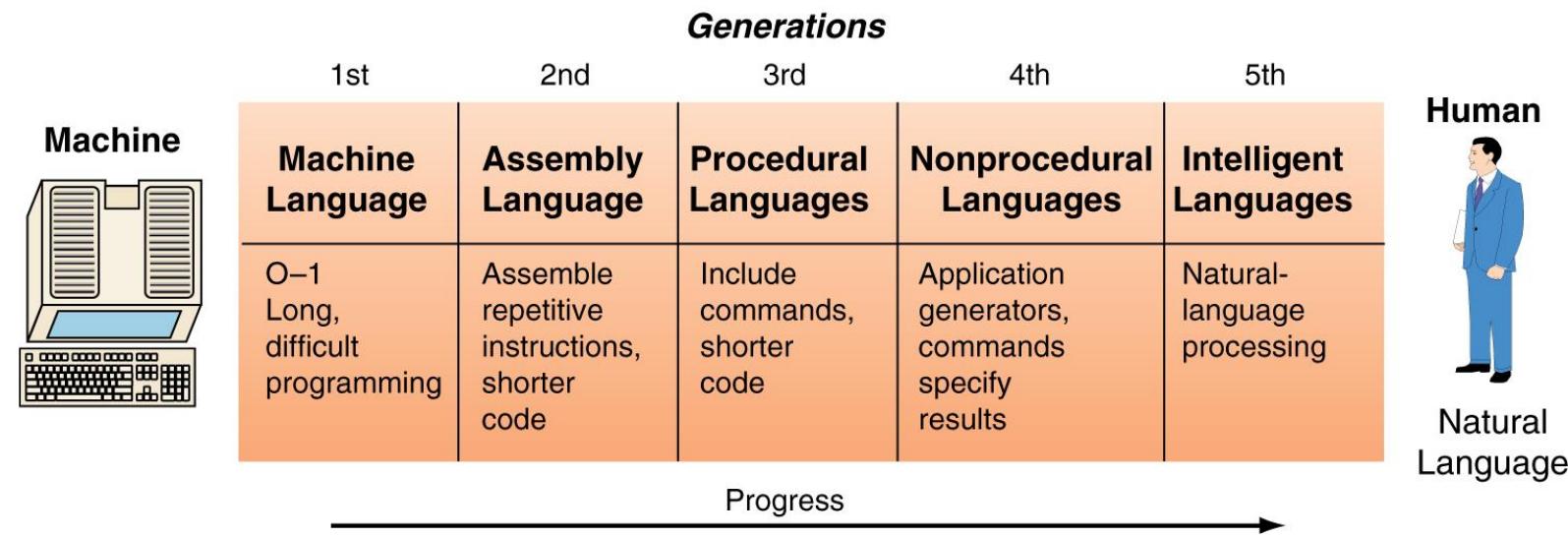
Bir problemin bilgisayar tarafından çözülebilmesi için öncelikle algoritmasının oluşturulması gerekmektedir.

Programlama Dillerinin Gelişimi

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

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

*Tüm yazılımlar
programlama dilleri ile
yazılır.*



Programlama Dillerinin Gelişimi

- İlk programın, Ada Lovelace tarafından Charles Babbage'ın tanımlamış olduğu “Analytical Engine” i ile Bernoulli sayılarının hesaplanması üzerine 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 çıkıştı
- 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 amaçlamaktadır.

Sayı Sistemleri

- Günlük yaşamımızda 10 luk sayı sistemi kullanılır. Ancak, bilgisayar sistemleri 2 lik sayı sistemini kullanırlar. 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 **digitken**, ikilik sistemde binary digit yada kısaca **bit** olarak adlandırılır.
- 123456 6 digitlik onlu sayı
100101 6 bitlik ikili sayı
- Sayı semboller 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 sayılar bir byte lük sayılar denir.
- 10011101 8 bit yada bir **bytedır**.
- 16 bit uzunluklu sayılar I word luk sayılar sayılır denmesine rağmen, bu kavram bazen işlemcinin veri yolu uzunluğu kadar bit sayısı ile de eşleştirilmektedir.
- 11001001 11100011 2 byte lük yada I **wordluk sayı**.
- Ayrıca her 4 bit, bir **Nibble** olarak adlandırılır.

Sayı Sistemleri

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

Sayı Sistemleri

- Pozitif ve Negatif Sayılar
- Bir byte'lık en küçük ve en büyük pozitif sayılarla 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 in 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 sayılarla karşılık düşürülecek biçimde kodlanırlar. Program örneğin bu sayı ile karşılaşrsa ekrana karşılık düşen simbolü basar, yada klavyeden gelen sayının sembolik karşılığını, yazıcıdan çıkarır.
- Bir çok kodlama türü olmasına karşın dünyada bilgisayar ortamlarında ANSI tarafından 1963 yılında standartlaştırılan **ASCII** (American National Code for Information Interchange) kodlaması yoğun olarak kullanılmaktadır. Ancak günümüzde ,ASCII kodları çok dilliliği sağlayabilmek için yetersiz kaldığından UNICODE kodlaması yaygınlaşmaktadır.Ancak pek çok uygulamada ASCII kodlaması hala geçerliliğini korumaktadır.
- ASCII temel olarak 7 bit' tir. 127 karakterden oluşur. Ama Extended kısmıyla birlikte 8 bit kullanılmaktadır. Ancak genişletilmiş kısımdaki semboller yazılım ortamına göre değişimlemeektedir.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	Ø	96	60	`	128	80	Ҫ	160	A0	á	192	C0	Ł	224	E0	ܵ
1	01	Start of heading	33	21	!	65	41	À	97	61	à	129	81	Ӯ	161	A1	í	193	C1	ݰ	225	E1	ܶ
2	02	Start of text	34	22	"	66	42	߱	98	62	߲	130	82	ߴ	162	A2	ó	194	C2	ߵ	226	E2	߳
3	03	End of text	35	23	#	67	43	߳	99	63	ߴ	131	83	߶	163	A3	ú	195	C3	߷	227	E3	߸
4	04	End of transmit	36	24	\$	68	44	߹	100	64	߻	132	84	߸	164	A4	ñ	196	C4	-	228	E4	ߺ
5	05	Enquiry	37	25	%	69	45	߻	101	65	߻	133	85	߻	165	A5	ܺ	197	C5	ܻ	229	E5	ܺ
6	06	Acknowledge	38	26	&	70	46	߱	102	66	߱	134	86	߶	166	A6	ܻ	198	C6	߻	230	E6	ܻ
7	07	Audible bell	39	27	'	71	47	߱	103	67	߱	135	87	ߴ	167	A7	ܻ	199	C7	߻	231	E7	ܻ
8	08	Backspace	40	28	(72	48	߱	104	68	߱	136	88	߶	168	A8	ܻ	200	C8	߻	232	E8	ܻ
9	09	Horizontal tab	41	29)	73	49	߱	105	69	߱	137	89	߶	169	A9	ܻ	201	C9	ܻ	233	E9	ܻ
10	0A	Line feed	42	2A	*	74	4A	߱	106	6A	߱	138	8A	߶	170	AA	ܻ	202	CA	ܻ	234	EA	ܻ
11	0B	Vertical tab	43	2B	+	75	4B	߱	107	6B	߱	139	8B	߶	171	AB	ܻ	203	CB	ܻ	235	EB	ܻ
12	0C	Form feed	44	2C	,	76	4C	߱	108	6C	߱	140	8C	߶	172	AC	ܻ	204	CC	ܻ	236	EC	ܻ
13	0D	Carriage return	45	2D	-	77	4D	߱	109	6D	߱	141	8D	߶	173	AD	ܻ	205	CD	ܻ	237	ED	ܻ
14	0E	Shift out	46	2E	.	78	4E	߱	110	6E	߱	142	8E	߶	174	AE	ܻ	206	CE	ܻ	238	EE	ܻ
15	0F	Shift in	47	2F	/	79	4F	߱	111	6F	߱	143	8F	߶	175	AF	ܻ	207	CF	ܻ	239	EF	ܻ
16	10	Data link escape	48	30	0	80	50	߱	112	70	߱	144	90	ܶ	176	B0	ܻ	208	DO	ܻ	240	FO	ܻ
17	11	Device control 1	49	31	1	81	51	߱	113	71	߱	145	91	ܶ	177	B1	ܻ	209	D1	ܻ	241	F1	ܻ
18	12	Device control 2	50	32	2	82	52	߱	114	72	߱	146	92	ܶ	178	B2	ܻ	210	D2	ܻ	242	F2	ܻ
19	13	Device control 3	51	33	3	83	53	߱	115	73	߱	147	93	ܶ	179	B3	ܻ	211	D3	ܻ	243	F3	ܻ
20	14	Device control 4	52	34	4	84	54	߱	116	74	߱	148	94	ܶ	180	B4	ܻ	212	D4	ܻ	244	F4	ܻ
21	15	Neg. acknowledge	53	35	5	85	55	߱	117	75	߱	149	95	ܶ	181	B5	ܻ	213	D5	ܻ	245	F5	ܻ
22	16	Synchronous idle	54	36	6	86	56	߱	118	76	߱	150	96	ܶ	182	B6	ܻ	214	D6	ܻ	246	F6	ܻ
23	17	End trans. block	55	37	7	87	57	߱	119	77	߱	151	97	ܶ	183	B7	ܻ	215	D7	ܻ	247	F7	ܻ
24	18	Cancel	56	38	8	88	58	߱	120	78	߱	152	98	ܶ	184	B8	ܻ	216	D8	ܻ	248	F8	ܻ
25	19	End of medium	57	39	9	89	59	߱	121	79	߱	153	99	ܶ	185	B9	ܻ	217	D9	ܻ	249	F9	ܻ
26	1A	Substitution	58	3A	:	90	5A	߱	122	7A	߱	154	9A	ܶ	186	BA	ܻ	218	DA	ܻ	250	FA	ܻ
27	1B	Escape	59	3B	:	91	5B	[123	7B	{	155	9B	ܶ	187	BB	ܻ	219	DB	ܻ	251	FB	ܻ
28	1C	File separator	60	3C	<	92	5C	\	124	7C		156	9C	ܶ	188	BC	ܻ	220	DC	ܻ	252	FC	ܻ
29	1D	Group separator	61	3D	=	93	5D]	125	7D	}	157	9D	ܶ	189	BD	ܻ	221	DD	ܻ	253	FD	ܻ
30	1E	Record separator	62	3E	>	94	5E	߻	126	7E	߻	158	9E	ܶ	190	BE	ܻ	222	DE	ܻ	254	FE	ܻ
31	1F	Unit separator	63	3F	?	95	5F	߻	127	7F	߻	159	9F	ܶ	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	$a-b$	$a-b$
Carpma	$a.b$	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	Carpma 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 ; $Y=A^B/C$
Once A^B işlemi yapılacak, ardından bulunan sonuç C ye bölünecektir.]

■ Matematiksel İşlemler

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

Karşılaştırma (Karar) İşlemleri

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

<i>İşlem sembolü</i>	<i>Anlamı</i>
=	<i>Eşittir</i>
<>	<i>Eşit değil</i>
>	<i>Büyük</i> tür
<	<i>Küçük</i> tür
\geq veya \Rightarrow	<i>Büyük eşittir</i>
\leq veya \Leftarrow	<i>Küçük eşittir</i>

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 sağlanması isteniyorsa koşullar arasında VE
- Herhangi birinin sağlanması isteniyorsa koşullar arasında 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 gereklidir. Yani;

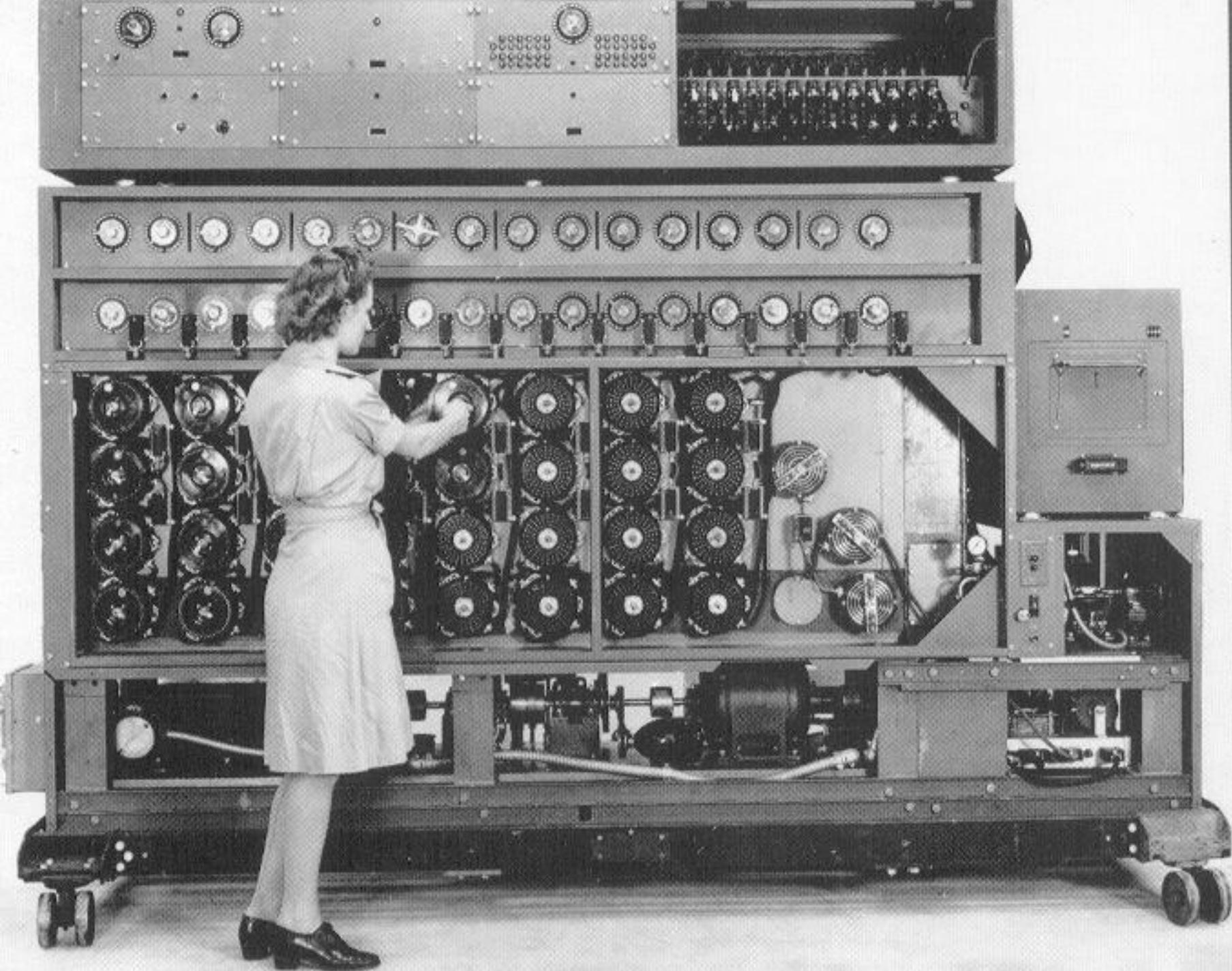
Eğer **Yaş>23 VE maaş=asgari ücret** ise ismi Yaz

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 koşuludur. Diğer iki dersin notlarının herhangi birinin 65 in üzerinde olması gereklidir.

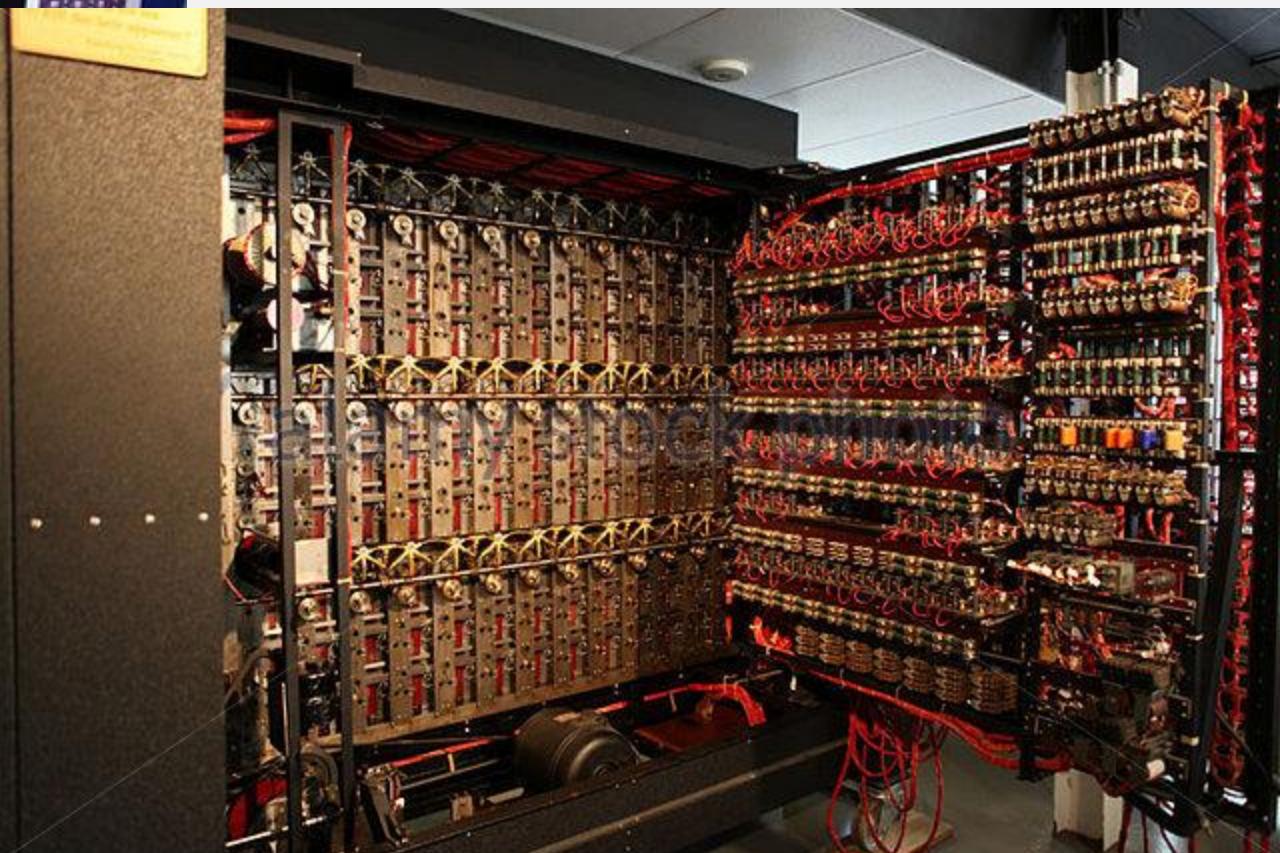
Eğer **Bilg.Not>65 VE (TDili Not>65 veya YDil Not>65)** ise ismi Yaz



ALGORİTMA KAVRAMI



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



Algoritma Kavramı

- Bu bölümde temel olarak algoritma kavramını işleyeceğiz. Algoritmanın ne olduğunu, tarihçesini, algoritmayla ilgili kavramları işleyeceğiz. Böylece algoritmanın ne olduğu ve ne olmadığı kafamızda iyice belirginleşecek ve algoritmaları nerede kullanacağımızı kavrayacağız.
- Algoritma, 800'lü yıllarda yaşamış olan Acem matematikçi Muhammad ibn Musa al-Khwārizmi'nin yaptığı çalışmalarında 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 çıkarabileceğini hale getirmek için akış diyagramları kullanılır. Akış diyagramları sembollerden oluşmaktadır. Her simbolün belli bir işlevi vardır.
- Algoritması oluşturulmuş bir problemin bilgisayar ortamına aktarılmış haline **program** denir.
- **Program, problemin çözümünde yapılması gereken işlemler bütününe 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 anınlara gelebilecek bulanık ifadeler içermemelidir.

- *2. Sıralı Olma:*

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

- *3. Sonluluk:*

Algoritma sonlu sayıda adımdan oluşmalı, sınırlı bir zaman diliminde tamamlanmalıdır. Her algoritmanın bir son noktası, bitisi 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 çözümümüz gereken problem (buna yapılması gereken iş de diyebiliriz) karşıya geçmektir. O zaman bu problemin çözümü için bir yol bulmamız gerekiyor.
- Şöyledir ifadeye ne dersiniz?
- Önce araba var mı kontrol et, ardından yürü!
- Bu ifade özünde doğrudur. Ancak yeterince açık değildir. Bunu hayatında ilk defa karşısından karşıya geçecek birine söylesek, kim bilir nasıl anlar?
- Yolun kenarına park etmiş araba var mı? Evet var. O zaman kaldırımdan yürüyeyim.
- Yolun kenarına park etmiş araba var mı? Hayır yok. O zaman yolun ortasından yola paralel yürüyeyim.
- Sol taraftan gelen araba var mı? Hayır yok. O zaman sola bakarak karşıya yürüyeyim
- Bu böylece sürüp gider.

Algoritma Kavramı

- Evet, biz yetişkin ve eğitimli insanlar "Önce araba var mı kontrol et, ardından yürü! ifadesinden verilmek istenen mesajı açıkça alırız. Ancak bilgisayarlar öyle değildir. Onları uzaydan gelmiş yaratıklar gibi düşünebiliriz. Hiçbir şey bilmezler. Ama onlara detaylı olarak verdigimiz bütün emirleri yerine getirebilirler.
- İyi tarif edersek, herşeyi hızla anlayıp kolayca uygulayabilirler.
- Şimdi problemi daha net ve kesin ifadelerle çözmeye çalışalım. Ama nereden başlayacağız? Evde miyiz? Trafik ışıklarının yanında miyiz? Yaya geçidinin önünde miyiz? Bu gibi durumlar önemlidir.
- Bu örnekte yolun kenarındayız ve trafik ışıği yok. O zaman şöyle yapalım: Önce yürüyelim, sonra sola ve sağa bakalım (tabi eğer ezilmeden karşıya geçebildiysek). Olur mu? İşimiz şansa kalır. Eğer araba yoksa olur. Araba varsa ezilme ihtimalimiz var.
- O zaman adımları gerçekleştirme sırası da önemli. Yolun kenarından başlayıp, önce sola bilmeli, gelen araba yoksa ya da karşıya geçebileceğimiz kadar uzak bir mesafede yürümeli. Sağ tarafta da araç yoksa ya da gelen araç yeterince uzak mesafede 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 bakalım. Sonra da sağ taraftan araba geliyor mu diye sağa bakalım. Acaba biz sağa bakarken soldan gelen bir araba olmuş mudur diye şimdi tekrar sola bakalım. Bu arada ya sağdan bir araba geliyorsa! Tekrar sağa mı baksak. Ya soldan araba geliyorsa..
- Gördüğünüz gibi yukarıdaki gibi düşünürsek sonsuz bir döngüye gireriz. Ya da araba yoksa, yürü!" ifadesi gereği sonsuza kadar yürüyecek miyiz? Bu yüzden algoritmalar sonlu sayıda adımdan ve bir bitiş durumundan oluşmalıdır. Yoksa zaten problem çözülmüş olmaz. Yolun bu tarafında kala kalırız. Veya sonsuza kadar yürümeye devam ederiz.

Problem Çözmek

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

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

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

Problem Çözmek

3. **Çözüm Yolları (Algoritmalar) Geliştirmek:** Bir problemin çözümü için çoğunlukla birden fazla seçenekimiz olur. İçinde bulunduğuuz duruma göre bazen zaman sıkışıklığından ilk bulduğumuz çözüm yolunu uygulamak durumunda kalırız. Ama eğer yeterince vakit varsa, en iyi çözüm yolunu (algoritmayı) bulmaya çalışmalıyız. Bunun için de bulabildiğimiz kadar çok çözüm yolu geliştirip, bunların içinden en uygununu tercih etmeliyiz. Çözüm yolları geliştirirken her bir çözüm yolu için çözümü adımlara ayırtırıp, 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 yazılım bir gerçek yaşam problemini çözdüğünü söyleyebiliriz.
- Peki, bir yazılımı geliştirmek için neler yapmak gereklidir?
- Bir yazılımı geliştirmek temel olarak şu adımları gerektirir:
 - Analiz:** Analiz aşaması, gereksinimlerin belirlendiği, bu gereksinimlerin çözümünlendiği ve çerçevelendiği aşamadır. Bu aşamada yazılımin ne yapacağı, hangi ihtiyacı karşılayacağı, hangi problemi çözeceği belirlenir.
 - Tasarım:** Analizle belirlenen yazılımin en uygun şekilde nasıl gerçekleştirilebileceğinin belirlenmesidir. Belirlenen gereksinimlere ve koşullara bakılarak hangi programlama dili, teknoloji, mimari, araç vb. kullanılacağı, çözümün planı, modeli, mimarisi tasarlınır.



Yazılım Geliştirme Süreci

3. **Geliştirme:** Bir önceki aşamada belirlenmiş olan tasarım, artık hayatı geçirilmeye, yazılım geliştirilmeye başlanır. Kodlama bu aşamada yapılır. Bu aşamada, kodlamayla birlikte veritabanı geliştirme, arayüz tasarımı, çeşitli konfigürasyonlar ve dokümantasyonlar da yapılmaktadır.
4. **Hatalardan Arındırma:** Geliştirme aşamasında ortaya çıkan arayüz, kod, veritabanı, doküman gibi ürünlerin istenilen seye uygun olup olmadığı test edilir. Eğer yazılımin 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 ilişkili Kavramlar

- **Kaynak Kod**
 - Bir programlama diliyle yazılmış metinlere kaynak kod (source code) denir. Kaynak kod dosyalarının uzantıları kullanılan programlama diline gene değişir. Örneğin;
 - Java için .java
 - C++ için .cp
 - Visual Basic için .vb
 - C# i in .cs
 - Bir kaynak kod dosyasını 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.

Programlamaya İlişkili Kavramlar

• Amaç Program

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

• Derleyici

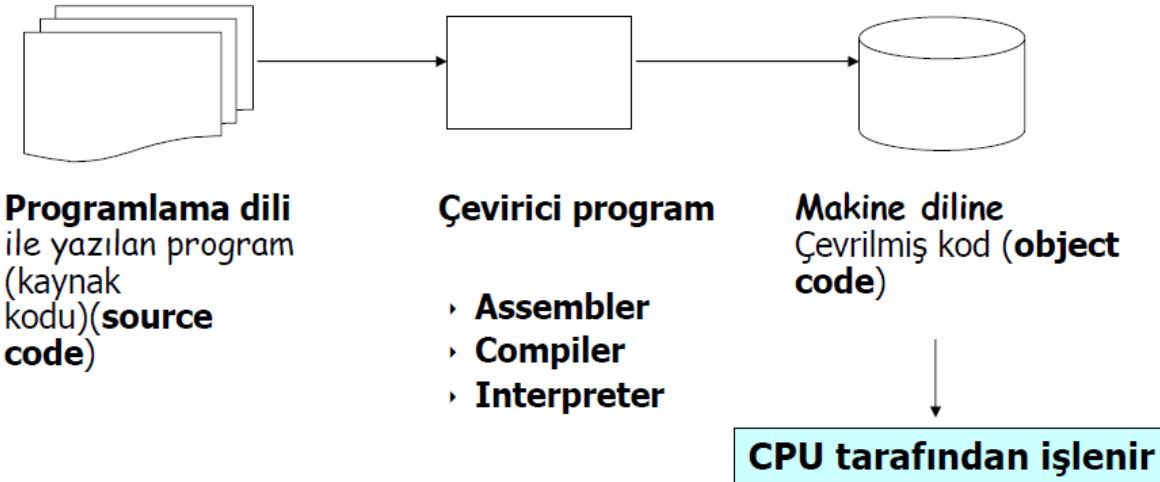
- Herhangi bir programlama diliyle yazılmış olan kaynak kodu, makine diline yani amaç programa dönüştüren özel programlara derleyici (compiler) adı verilir. Derleyicilerin kaynak kodu amaç programa dönüştürmeleri işlemeye de derleme (compile) denilmektedir.

• Yorumlayıcı

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

Programlama Dillerinin Gelişimi

- **Dilden dile çevrim**



Algoritmanın Yazılım Geliştirme Sürecindeki Yeri

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

Algoritmanın Yazılım Geliştirme Sürecindeki Yeri

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

Algoritmanın Yazılım Geliştirme Sürecindeki Yeri

- **Teknoloji:** Geliştirdiğimiz yazılımın üzerinde çalıştığı teknolojilere hakim olmamız gereklidir. Örneğin; XML dosyaları işleyerek sipariş alacak bir yazılım geliştireceksek, XML teknolojisini bilmeliyiz. Eğer bir mail alma/yollama programı yazacaksak, SMTP protokolünü bilmeliyiz.
- **En İyi Pratikler (Best Practices):** Belirli bir teknolojide bir çözüm üretmek istiyorsak, öncelikle bu iş birileri yapmış mı diye bakmakta fayda vardır. Eğer yaptığımız işi daha önceden yapanlar olduysa, onların bunu nasıl çözdüğünü öğrenerek bu şekilde yapmak en doğrusu olacaktır. İşte bu daha önceki pratiklerin en iyilerini bulup bunlardan faydalananmalıyız.
- **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 algoritmalarдан faydalanzıız.
- Örneğin; bir dağıtım aracının uğrayacağı 10 nokta için en uygun rotanın bulunması için algoritma geliştiririz.

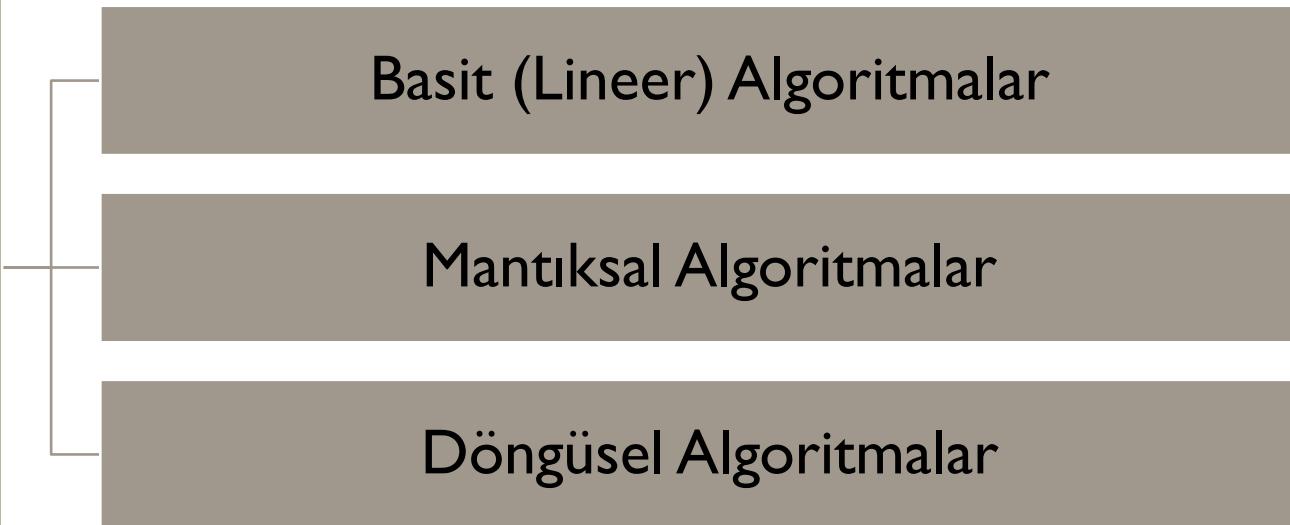
Bu bölümde algoritmanın ne olduğunu, tarihçesini, yazılım geliştirme sürecindeki yerini ve programlamaya ilişkisini gördük. Bundan sonraki bölümlerde problem-çözüm-algoritma-programlama döngüsünü adım adım kavrayarak uygulamalarla pekiştireceğiz.

Algoritma - Örnek

- Algoritma ile oluşturulacak çözümler sözel olarak ifade edilir. Örneğin sabah kalktığımızda kahvaltı yapacağı zaman kahvaltı hazırlama algoritması oluşturulursa:
 - Yataktan kalk
 - Mutfağa git
 - Ekmek al
 - Çay hazırla
 - Dolaptan kahvaltlılıkları çıkar
 - Bardağın bitince çayını doldur
 - Karnın doyunca sofradan kalk
 - Kahvaltlılıkları dolaba koy
 - Sofrayı temizle
- Adım 1:Yataktan kalk
 - Adım 2:Mutfağa git
 - Adım 3:Ekmek al
 - Adım 4:Çay hazırla
 - Adım 5:Dolaptan kahvaltlılıkları çıkar
 - Adım 6:Bardağın bitince çayını doldur
 - Adım 7:Karnın doyunca sofradan kalk
 - Adım 8:Kahvaltlılıkları 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 algoritmalarla akış düz bir halde baştan sona doğru olacaktır. Çoğunlukla küçük hesaplamaları gerçekleştirmek için kullanılırlar. Bir önceki algoritmeye 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 yazdırın algoritma;

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

Mantıksal Algoritmalar

- Algoritma içerisinde mantıksal karşılaştırmaların bulunduğu yapılardır. Mantıksal karşılaştırmalara göre algoritmanın akışı farklı adımlara geçecektir. Bu şekilde oluşturulan algoritmaya Mantıksal Algoritmalar denir.
- İlk oluşturulan algoritma biraz daha ayrıntılarırsa karar yapılarının ortaya çıktığı görülür. Örnek:
 - Adım 1: Başla
 - Adım 2: Yataktan kalk
 - Adım 3: Mutfaga git
 - Adım 4: Eğer ekmek yoksa ekmek al
 - Adım 5: Çayı hazırla
 - Adım 6: Dolaptan kahvaltlıkları çıkar
 - Adım 7: Bardağın bitince çayını doldur
 - Adım 8: Karnın doyunca sofradan kalk
 - Adım 9: Eğer kahvaltlıklar bitmişse bulaşık makinesine koy
 - Adım 10: Eğer kahvaltlıklar bitmemişse kahvaltlıkları 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 algoritmaların 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 algoritmalar döngüsel algoritma denir.
- Yani döngüsel algoritmaların mantıksal karşılaştırma sonucunda program daha önceki adımlara gider.

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 faktöriyeye yaz; $f=f \times i$
- Adım 6: İndeks değerini 1 artır; $i=i+1$
- Adım 7: Eğer indeks değeri n 'den küçük veya eşitse ($i \leq n$) Git Adım 5
- Adım 8: Dur

Döngüsel Algoritmalar – EBOB- I

- 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 $kucuksayı=a$
- Adım 6: Eğer $b < a$ ise $kucuksayı=b$
- Adım 7: Eğer ($a \text{ MOD } Sayac = 0$) VE ($b \text{ MOD } Sayac = 0$) ise $ebob=Sayac$
- Adım 8: Eğer $Sayac < kucuksayı$ ise $Sayac=Sayac+1$ ve Git Adım 7
- Adım 9: Yaz ebob
- Adım 10: Dur

Döngüsel Algoritmalar – EBOB-3

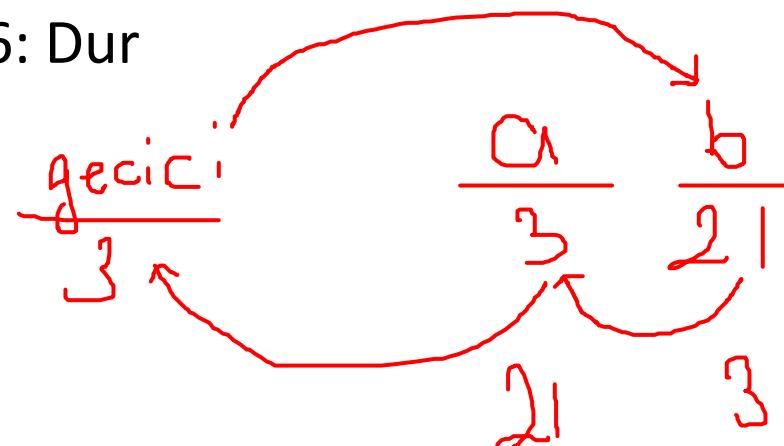
- Girilen iki sayının en büyük ortak bölenini (EBOB) bulan algoritma yazalım.
- Adım 1: Başla
- Adım 2: Sayıları al; a, b
- Adım 3: Eğer $a=b$ ise $ebob=a$ ve Git Adım 6
- Adım 4: Eğer $a < b$ ise $sayac=a$ Değilse $sayac=b$
- Adım 5: Eğer ($a \text{ MOD } sayac = 0$) VE ($b \text{ MOD } sayac = 0$)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

Döngüsel Algoritmalar – EKOK- I

- Girilen iki sayının en büyük ortak bölenini (EBOB) bulan algoritma yazalım.
- Adım 1: Başla
- Adım 2: Sayıları al; a, b
- Adım 3: $\text{carpim}=a*b$
- 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: $\text{ekok}=\text{carpim}/a$
- 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 $Sayac \text{ MOD } b = 0$ 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 $Sayac \text{ MOD } b = 0$ 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 $a \text{ MOD } b = 0$ ise yaz a ve Git Adım 8
- Adım 5: $Sayac=0$
- Adım 6: $Sayac=Sayac+a$
- Adım 7: Eğer $Sayac \text{ MOD } b = 0$ 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 yazılabılır:

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

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

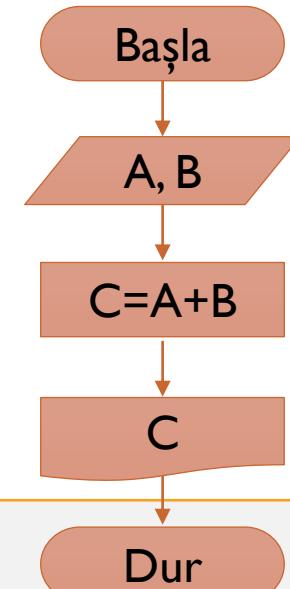
Satır algoritma ile gösterilmesi:

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

AKİŞ DİYAGRAMI 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 bölümündür. Örneğin $C=A+B$ gibi 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. $\text{PI}=3.14$ gibi bir ifadeden sonra PI sabiti programın her yerinde 3.14 değerini ifade eder.

Yarıçapı girilen dairenin alanını hesaplayıp yazdırın programın algoritmasını oluşturunuz.

Algoritma Geliştirmek

İsimlendirme Kuralları:

- İngiliz alfabetesindeki 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. $\text{PI}=3.14$ 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 işaretler 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 yazdırın 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 çağrılar 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 eklerek bulunan sonuç yine değişkenin kendisine yeni değer olarak atanmaktadır.

Sayaç yapısı:

sayac=sayac+adım

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

27:44

Sunuyu durdur



...



Aynıl

Kapat

⚠ Kaydediyorsunuz Bu toplantıyı kaydediyorsunuz. Toplantının kaydedildiğini herkese bildirdiğinizden emin olun. [Gizlilik ilkesi](#)



Uygulamada aç



Sayıc
kullanarak 1-5 arası
sayıları ekrana yazdırın
programın algoritmasını
oluşturunuz

Adım-1: Başla

Adım-2: sayıac=1

Adım-3: Yaz sayıac

Adım-4: sayıac=sayıac+1

Adım-5: Eğer sayıac<=5 ise Git Adım-3

Adım-6: Dur

Adım-1: Başla

Adım-2: sayıac=1

Adım-3: Eğer sayıac>5 ise Git Adım-6

Adım-4: Yaz sayıac

Adım-5: sayıac=sayıac+1 ve Git Adım-3

Adım-6: Dur

S E

1
1
2
3
4
5
/

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. DÖNGÜ SONU
6. Yaz T
7. Dur

Algoritma Geliştirmek

Ardışık Toplama:

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

$$\text{Toplam} = \text{Toplam} + \text{sayı}$$

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, sayı

Adım-6: Toplam=Toplam+sayı

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-2: Faktöriyeli hesaplanacak sayıyı al; N

Adım-3: carpim=1

Adım-4: DÖNGÜ (i=2 TO N STEP 1)

Adım-5: carpim=carpim*i

Adım-6: DÖNGÜSONU

Adım-7: Yaz carpim

Adım-8: Dur

Adım-1: Başla

Adım-2: Faktöriyeli hesaplanacak sayıyı al; N

Adım-3: carpim=1

Adım-4: DÖNGÜ (i=N TO 2 STEP -1)

Adım-5: carpim=carpim*i

Adım-6: DÖNGÜSONU

Adım-7: Yaz carpim

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 algoritmaların birinci satırı “1. Başla” şeklinde olmalıdır.
- Bütün algoritmalar “Dur” ile biter.
- Algoritmaların “Git” işlem akışı yönlendirme komutu satır numarasıyla birlikte kullanılmalıdır.
- Algoritmaların kullanabileceğiniz alt program veya fonksiyonlar, tanımlayıcı isimleri ve parametreleriyle birlikte verilmelidir.
- Algoritmaların adımları sınırlı sayıda, açık, net ve kesin olmalıdır.
- Algoritmaların ifadeleri anlaşılır ve mümkün olduğunda sade (az ve öz) olmalıdır.
- Algoritmaların ifadeleri, herhangi bir programlama diline, donanıma, işletim sisteme 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.

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.

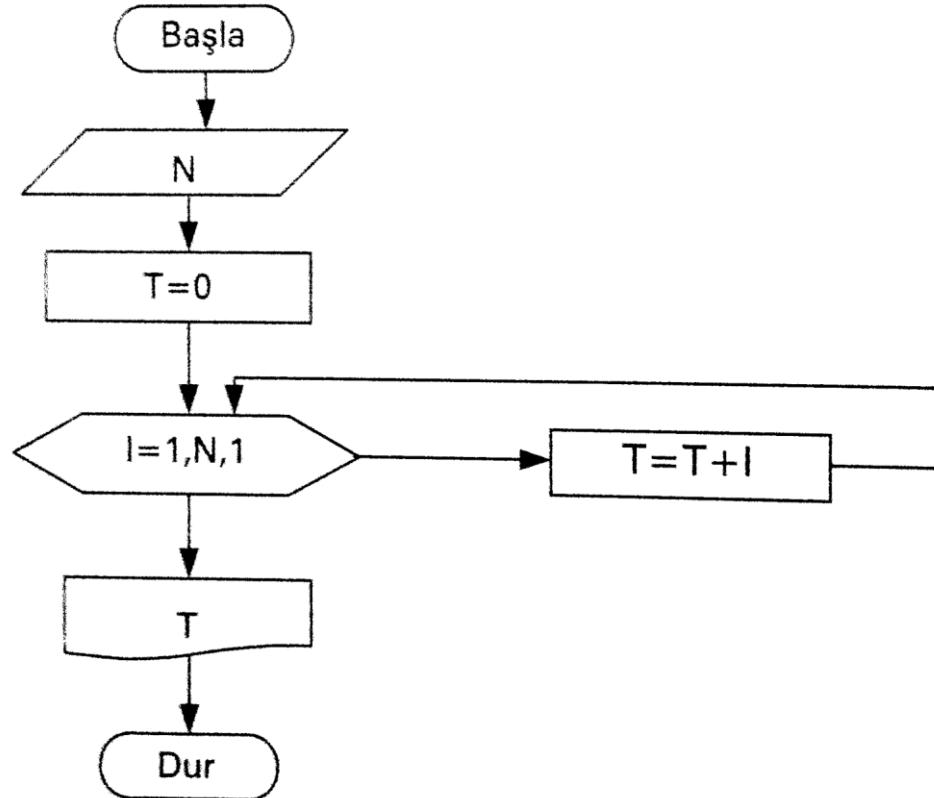
$$c=a^2+2ab$$

$$A=\pi r^2$$

Akış Diyagramı



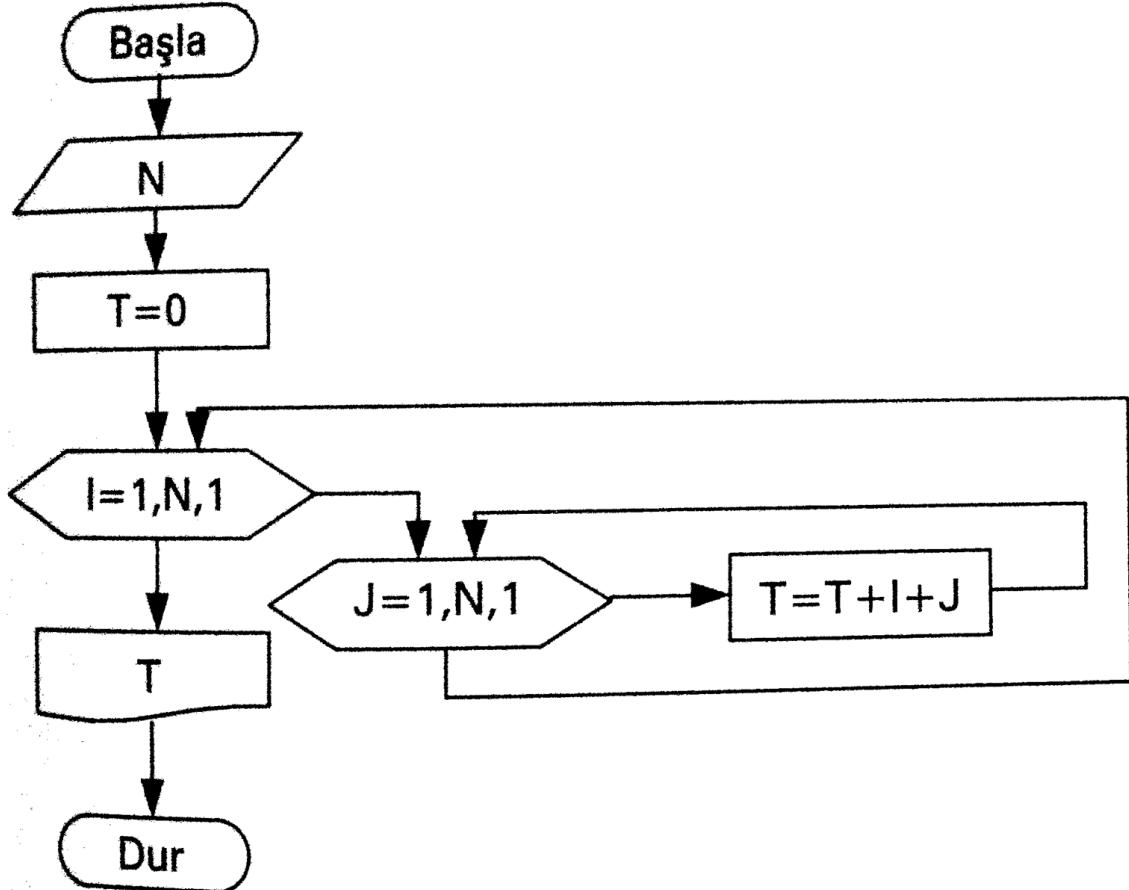
Döngü: Tekrarlamalı işlemler döngü olarak adlandırılır. Şeklin içine döngünün başlangıç, bitiş ve artış (adım) değerleri yazılır.



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

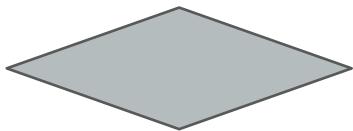
Akış Diyagramı

İç-İçe Döngü Yapısı:

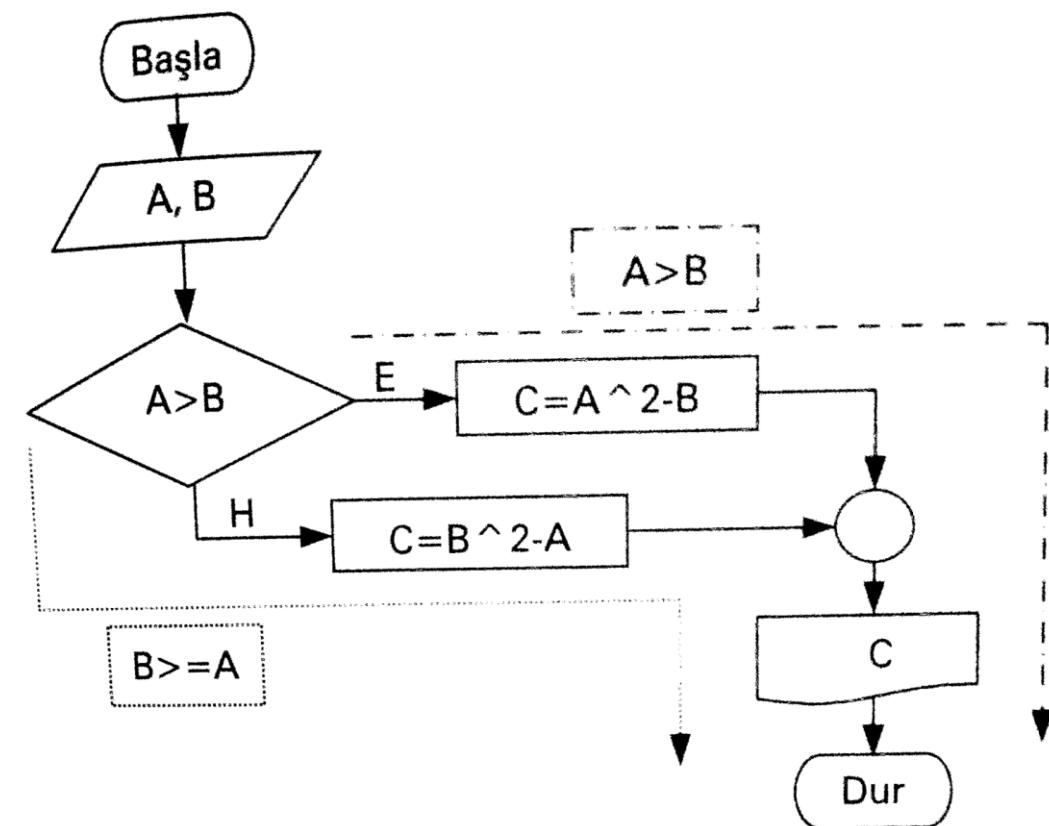
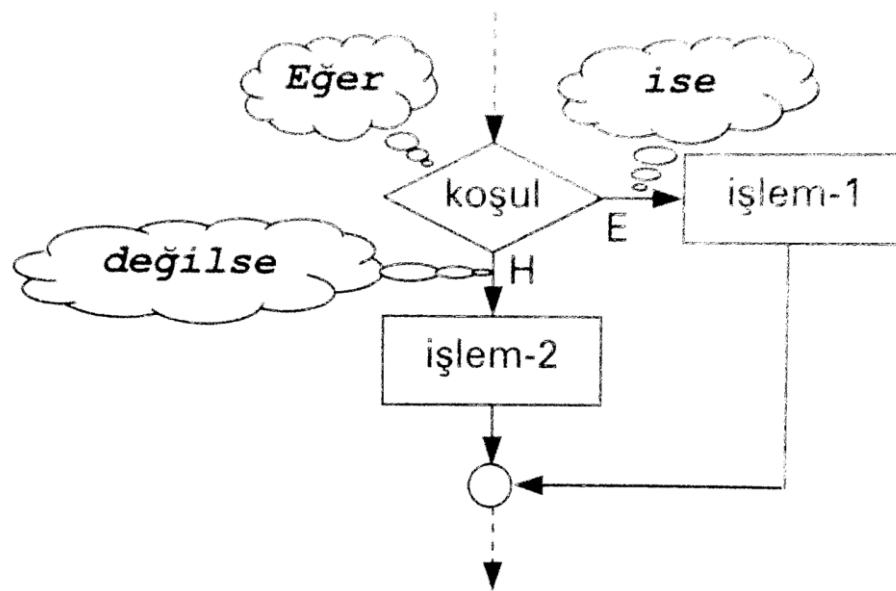


N	T	I	J	Ekran
3	0	1	1	
	2		2	
	5		3	
	9	2	1	
	12		2	
	16		3	
	21	3	1	
	25		2	
	30		3	
	36			36

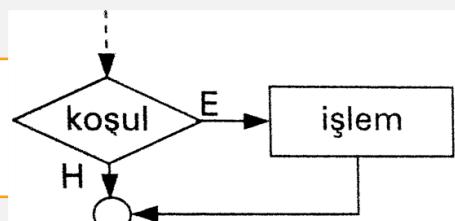
Akış Diyagramı



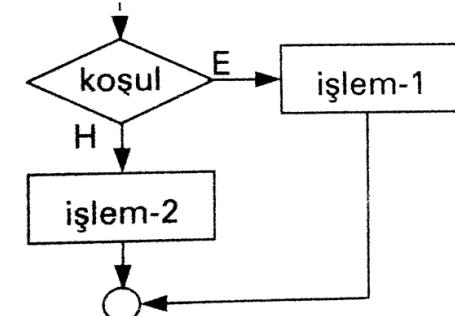
Karar: Karar verme/karşılaştırma işlemlerini temsil eden şekildir. Koşul veya mantıksal operatörlerle bağlı koşullar, şeklin içine yazılır. TRUE, FALSE



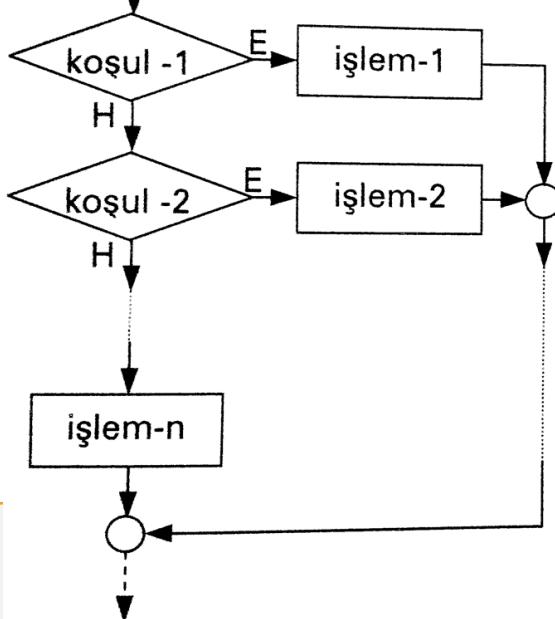
Akış Diyagramı



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



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

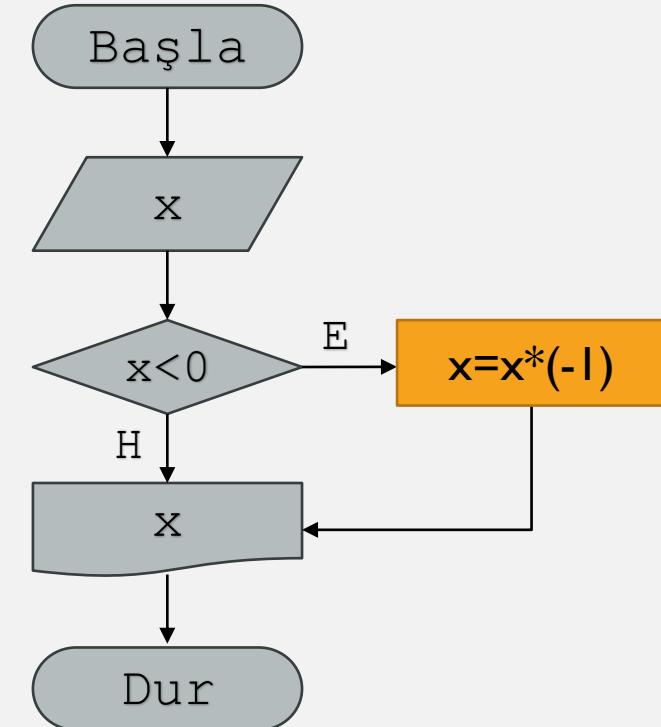
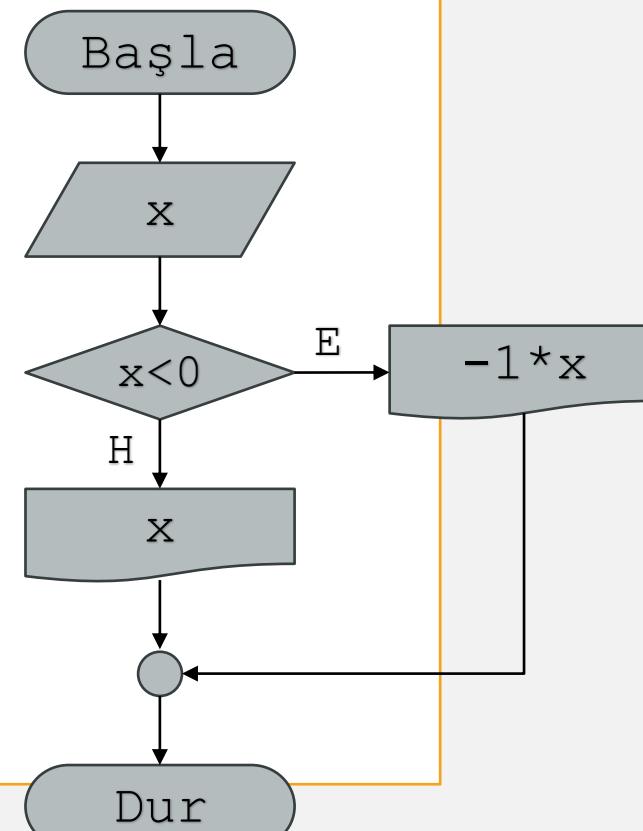


"koşul-1" doğru ise (E) "işlem-1", değilse (H) "koşul-2" kontrol edilir ve doğru ise (E) "işlem-2" yapılır. "koşul-2" yanlış ise (H) sıradaki koşul kontrol edilir ve doğru olup olmamasına göre işlem devam eder. Eğer tüm koşullar yanlış ise "işlem-n" yapılır.

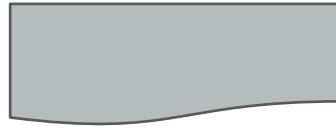
Akış Diyagramı

Örnek: Klavyeden girilen bir sayının mutlak değerini ekranaya yazdırın programın akış diyagramını çizelim.

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



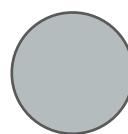
Akış Diyagramı



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

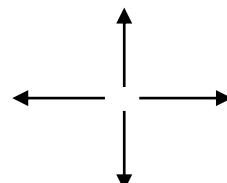


Önceden Tanımlı İşlem: Özellikle büyük boyutlu programlar bir çok alt programdan/fonksiyondan oluşur. Bu şekilde önceden tanımlanmış programları parametreli veya parametresiz olarak çağrırmak için kullanılır.



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.



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



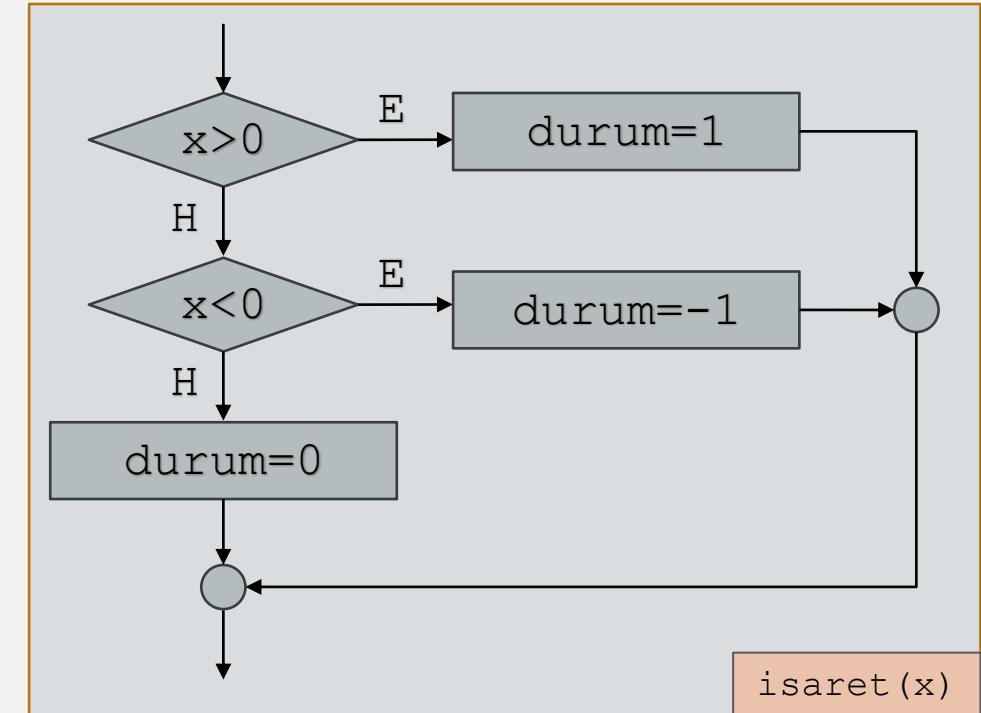
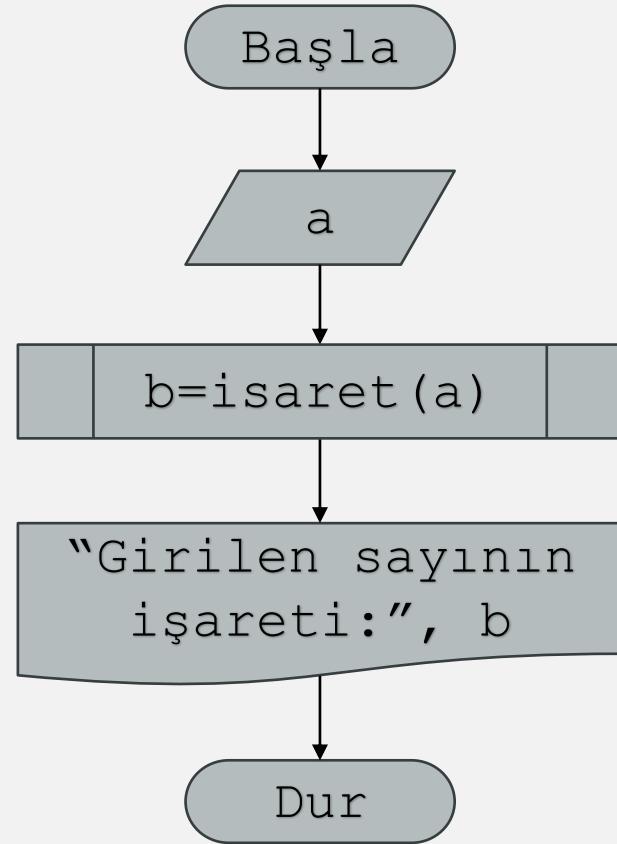
Akış Diyagramı

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

A

1

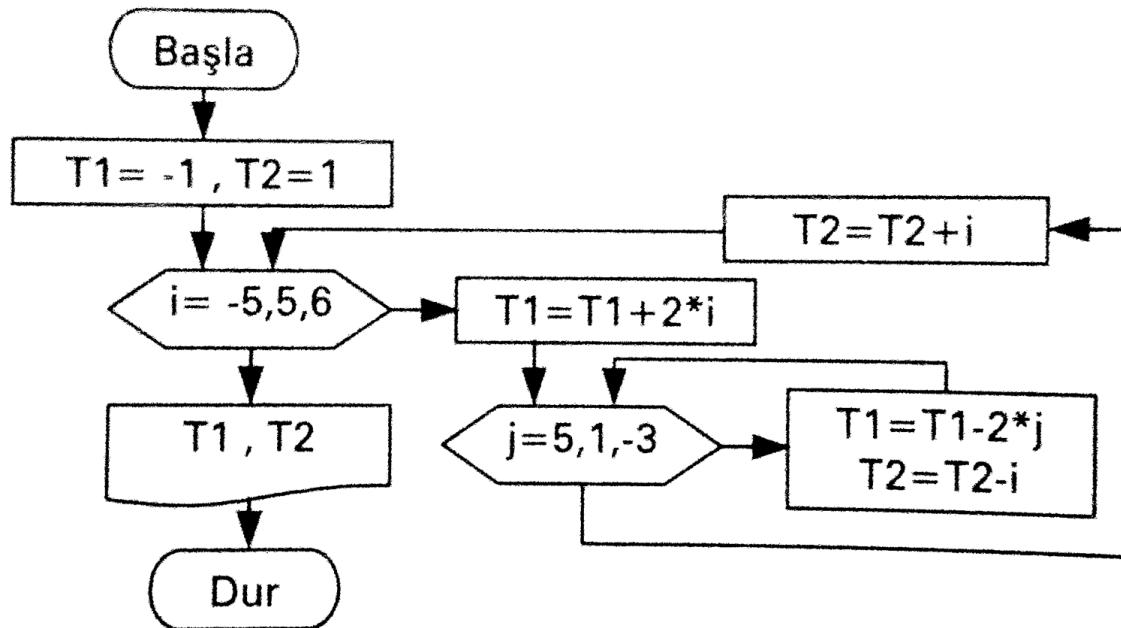
Akış Diyagramı



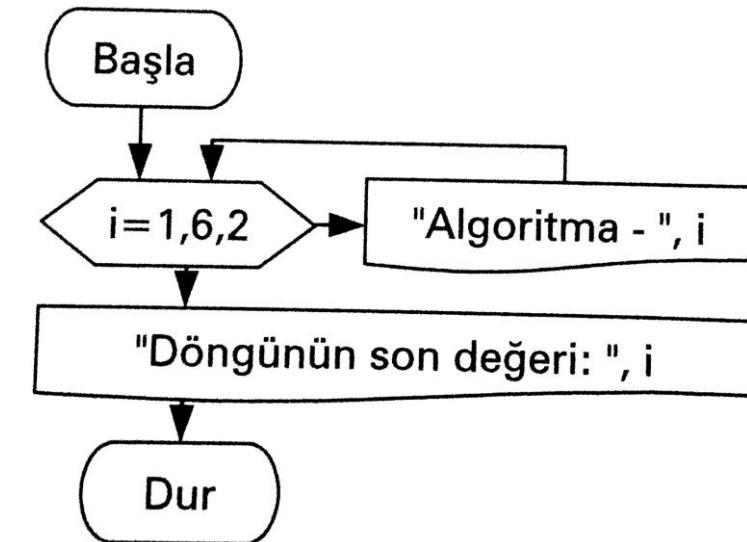
Bir sayının işaretini bulan alt programın yazılıp ana programdan çağrılması

Akış Diyagramı

Aşağıdaki akış diyagramlarının ekran çıktısını elde ediniz.



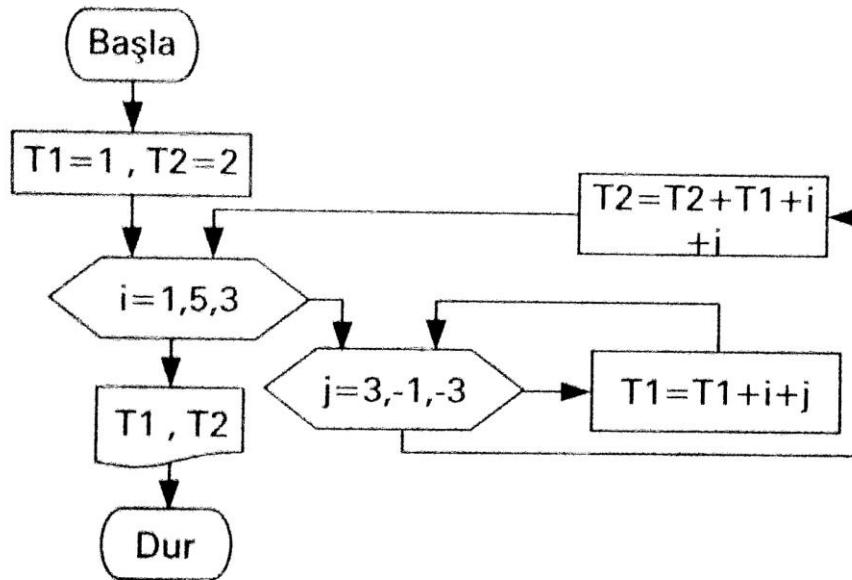
(Seckin, s.80)



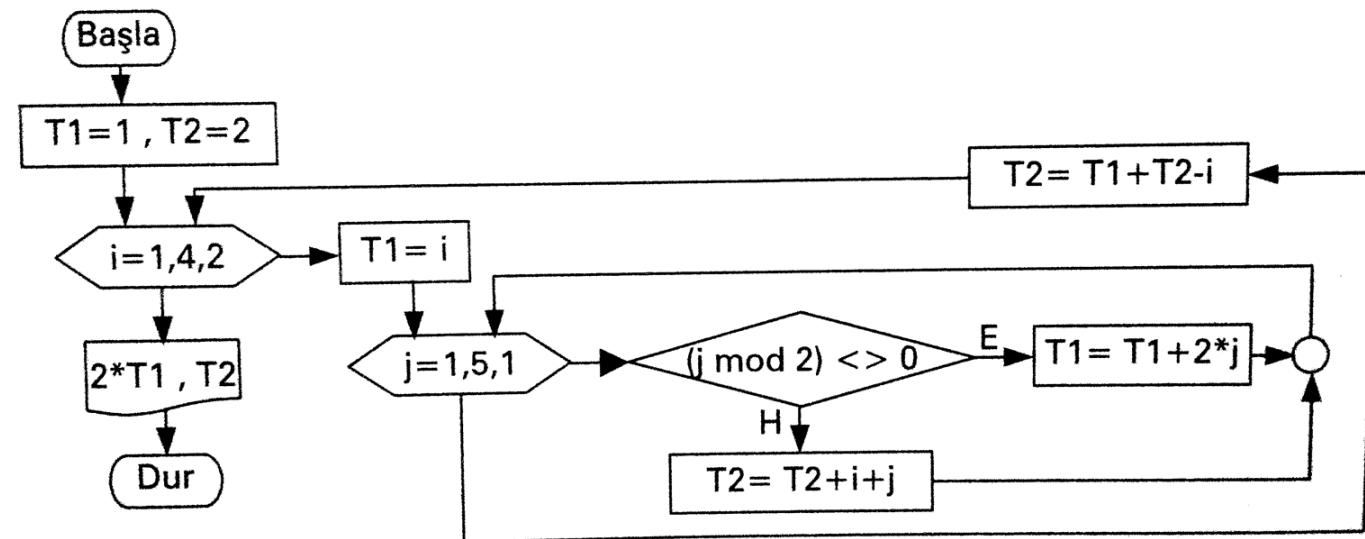
(Seckin, s.80)

Akış Diyagramı

Aşağıdaki akış diyagramlarının ekran çıktısını elde ediniz.



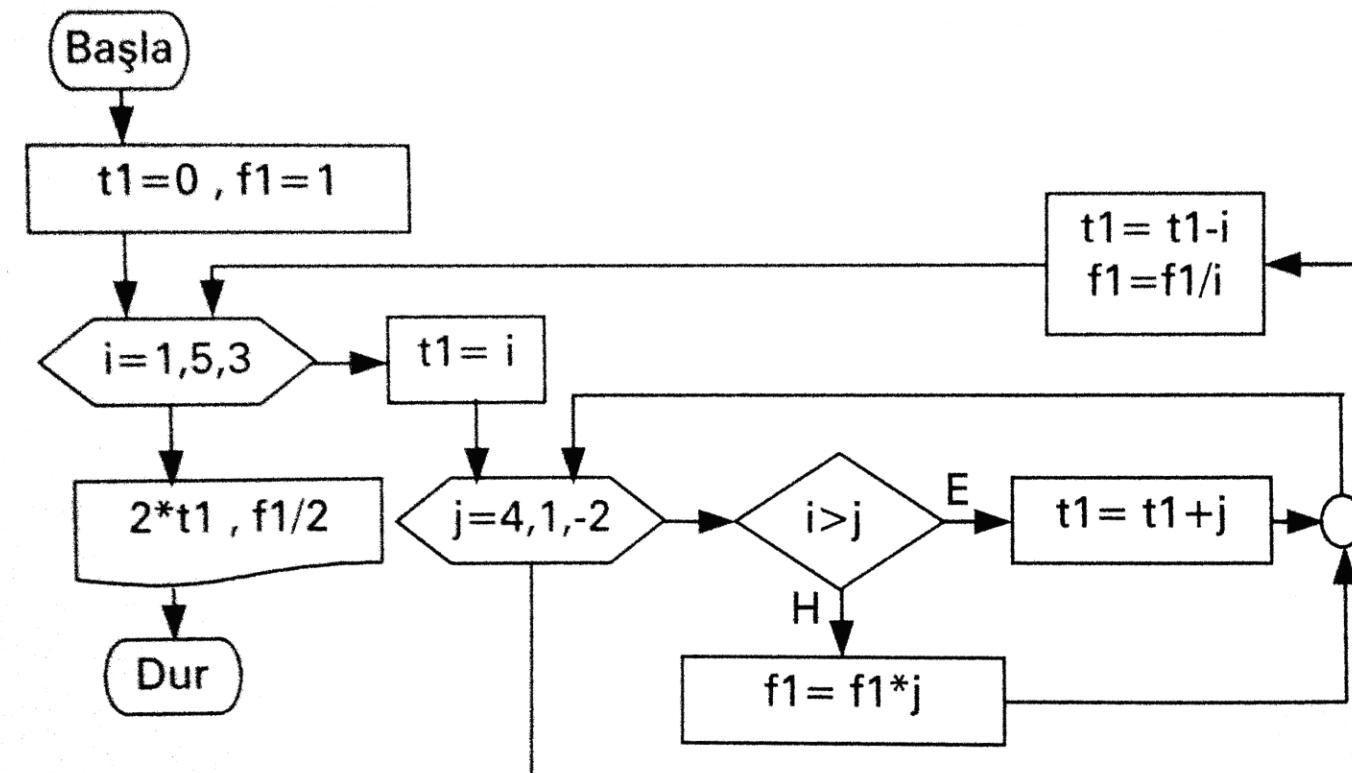
(Seckin, s.81)



(Seckin, s.81)

Akış Diyagramı

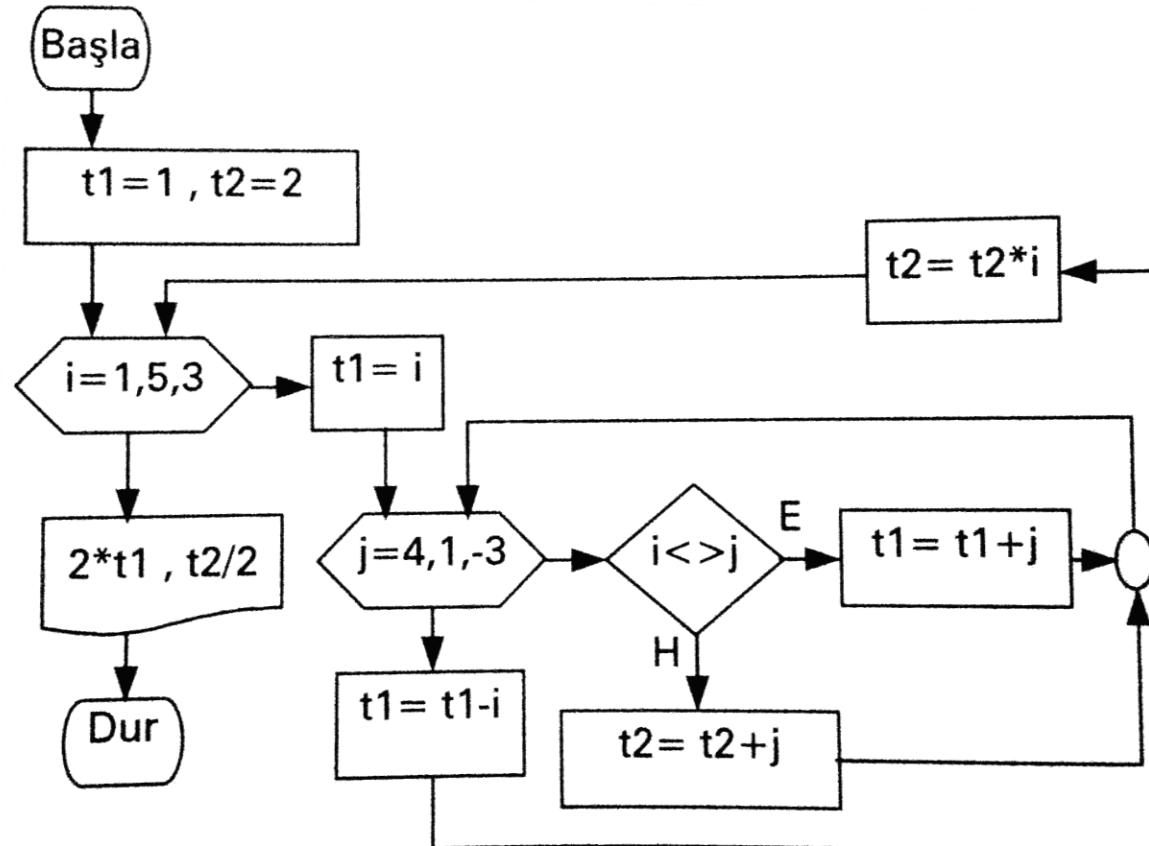
Aşağıdaki akış diyagramının ekran çıktısını elde ediniz.



(Seckin, s.82)

Akış Diyagramı

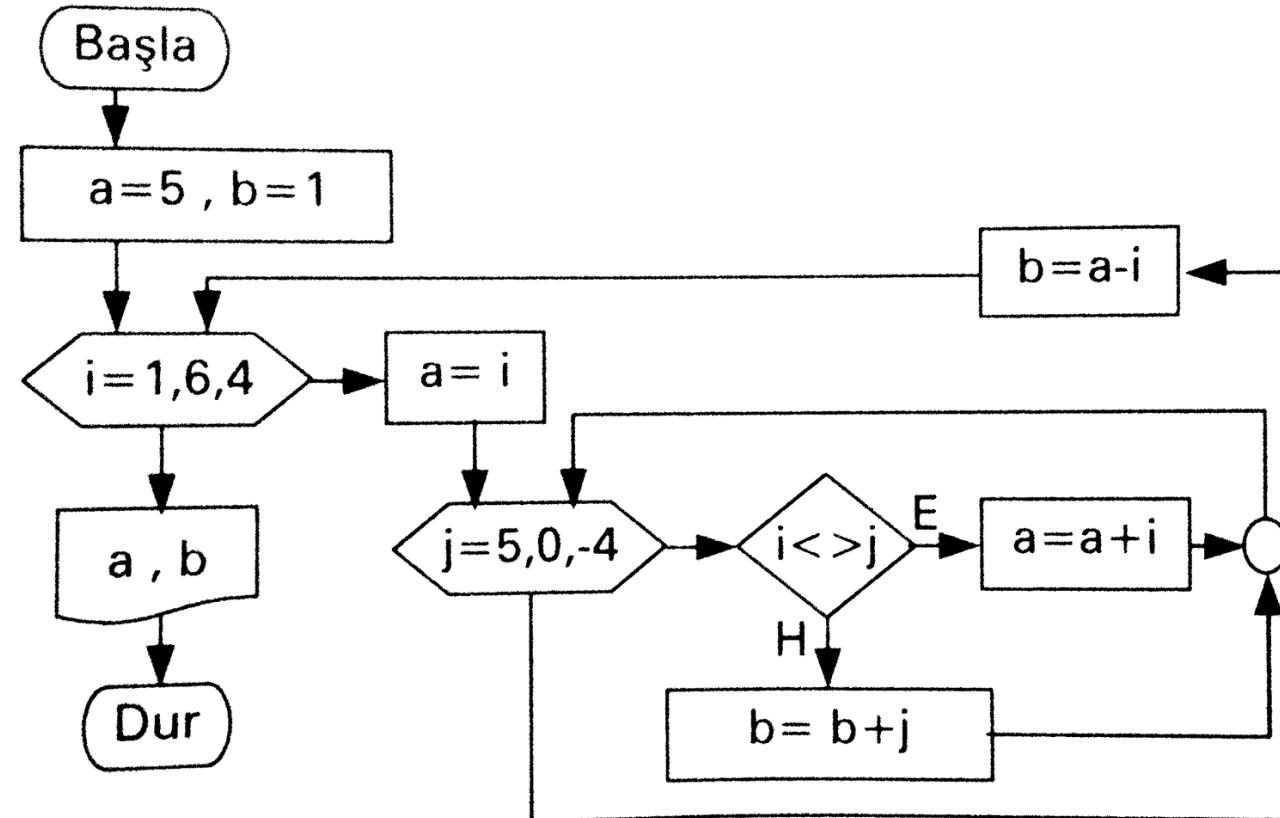
Aşağıdaki akış diyagramının ekran çıktısını elde ediniz.



(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) ayrılp birinden devam etmesi gerekebilir. Yani, basit koşul yapılarının yetmediği yerde iki veya daha çok alternatifli koşul yapıları kullanılır.

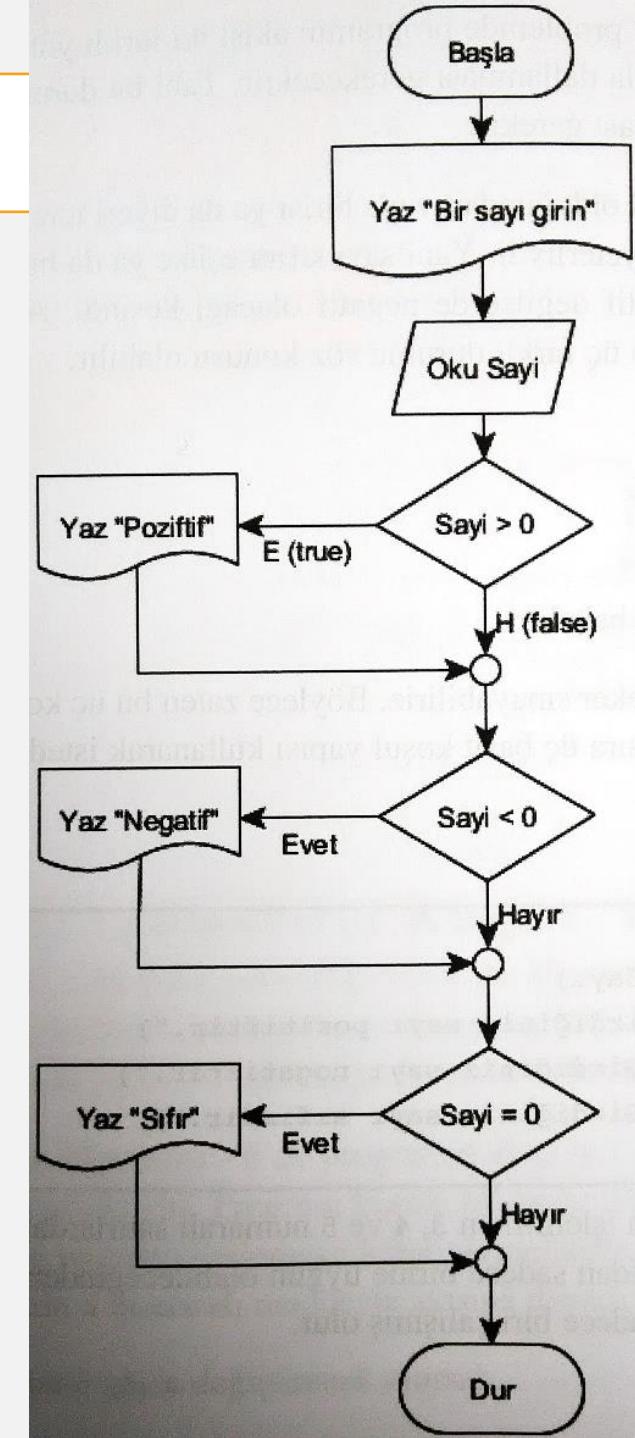
Örnek: Kullanıcıdan bir sayı alarak, girilen sayının pozitif, negative veya sıfır olduğunu ekrana yazdırın programın algoritmasını tasarllayın (Kodlab s.34).

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

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

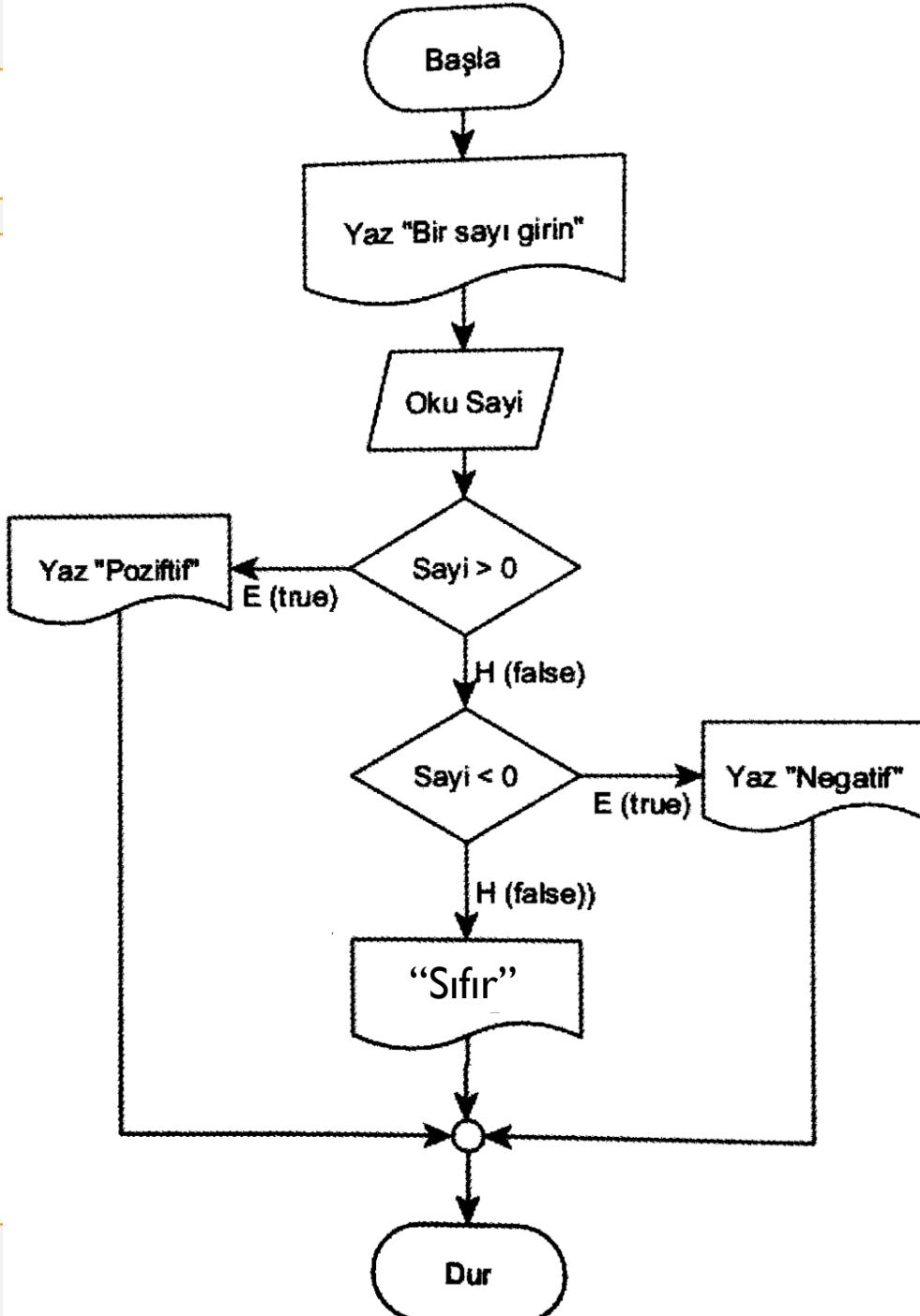
İki veya Çok Alternatifli Koşul Yapıları

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



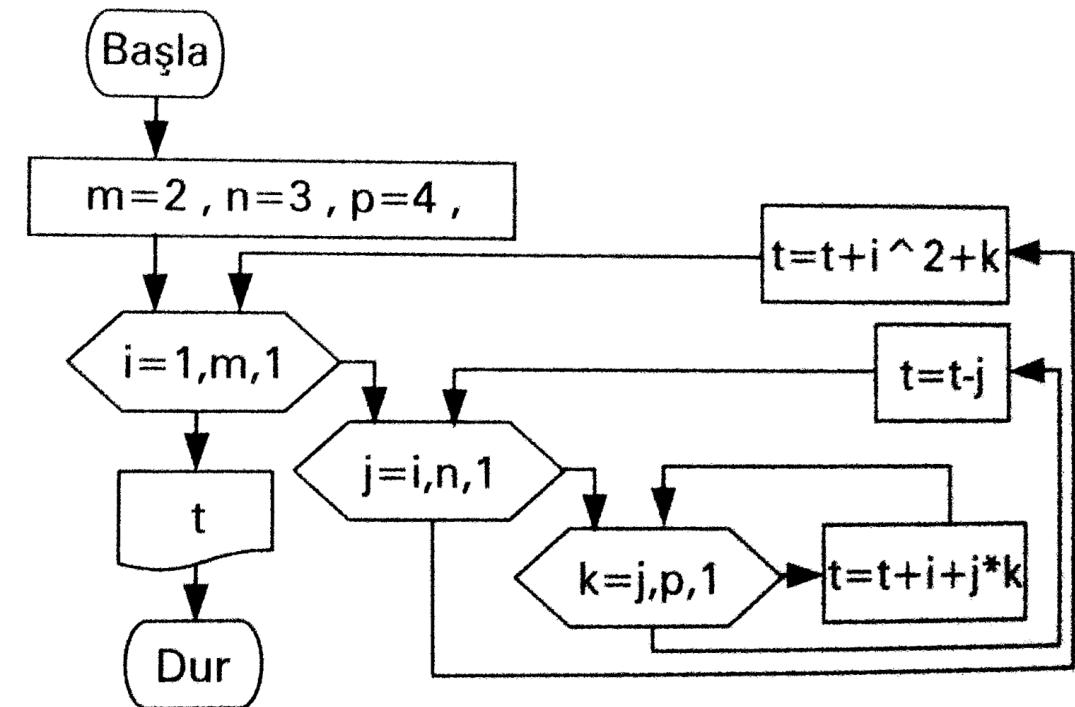
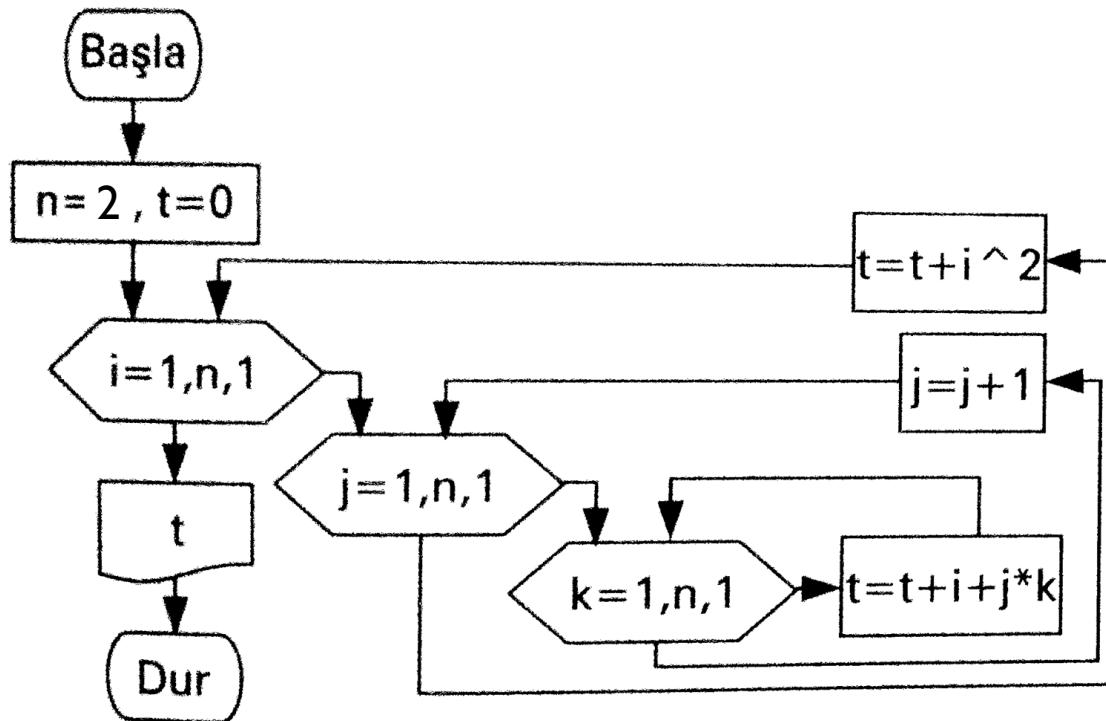
İki veya Çok Alternatifli Koşul Yapıları

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



Çalışma Soruları

Aşağıdaki akış diyagramlarının ekran çıktılarını elde ediniz.



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 yazdırın programın akış diyagramını çiziniz?
- Klavyeden metre (m) cinsinden girilen uzunluğu, kilometre (km) ve santimetre (cm) cinsine dönüştürüp yazdırın programın akış diyagramını çiziniz?
- **Klavyeden üç kenar uzunluğu girilen üçgenin türünü (eşkenar, ikizkenar veya çeşitkenar) tespit edip yazdırın programın akış diyagramını çiziniz?**
- Klavyeden girilen bir sayının yine klavyeden istenen yüzdesini hesaplayıp yazdırın programın akış diyagramını çiziniz?
- Klavyeden bir ürünün fiyatı ve KDV oranı istenmektedir. Buna göre ürünün KDV'li satış fiyatını hesaplayıp yazdırın programın akış diyagramını çiziniz?
- Klavyeden bir ürünün fiyatı ve kar/zarar oranı istenmektedir. Buna göre ürünün satış fiyatını hesaplayıp yazdırın programın akış diyagramını çiziniz?
- Klavyeden 1-7 arasında bir tamsayı istenmektedir. Bu tamsayıya göre haftanın ilgili gününü kelime/isim olarak yazdırın programın akış diyagramını çiziniz?
- Klavyeden girilen 10 sayının ortalamasını hesaplayıp yazdırın programın akış diyagramını çiziniz?

Sözde Kod (Pseudo-code) & Algoritmalar Arasında Dönüşüm

Sözde Kod

Problemi çözmek için tasarladığımız algoritmaları kodlamamız gereklidir. Bu kodlama herhangi bir programlama dilinde (C, C++, Java, Python vb.) olabileceği gibi, eğer algoritmayı hemen çalıştırılmaya ihtiyacımız yoksa ara 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 yapılp yapılmayacağına karar verir.

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

Sözde Kod

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

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

SORU:

Yukarıdaki yazım ile sağdaki arasında ne fark vardır?

```
IF [koşul1] THEN  
    Koşul1 doğru ise gerçekleştirilecek işlemler  
    IF [koşul2] THEN  
        Koşul 1 ve Koşul2 doğru ise gerçekleştirilecek işlemler  
    ELSE  
        Koşul1 doğru ve Koşul2 yanlış ise gerçekleştirilecek 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 yazdırın programın algoritmasını sözde kod (pseudo-code) olarak yazınız.**

Algoritmalar Arasında Dönüşüm

Satır Algoritmalarından 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 Algoritmalarından Sözde Kod Oluşturmak

İki sayıyı alıp, bunları toplayarak toplamı ekrana yazdırın 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 Algoritmalarından Sözde Kod Oluşturmak

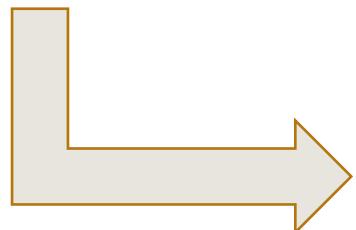
1. Başla
2. Yaz "Yaşınızı Giriniz"
3. Oku (Yas)
4. Eğer
 - 4.1. Yas>=18 ise Yaz "Uygulamayı İndirebilirsiniz"
 - 4.2. Değilse Yaz "Uygulamayı İndiremezsiniz"
5. Dur



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

Satır Algoritmalarından Sözde Kod Oluşturmak

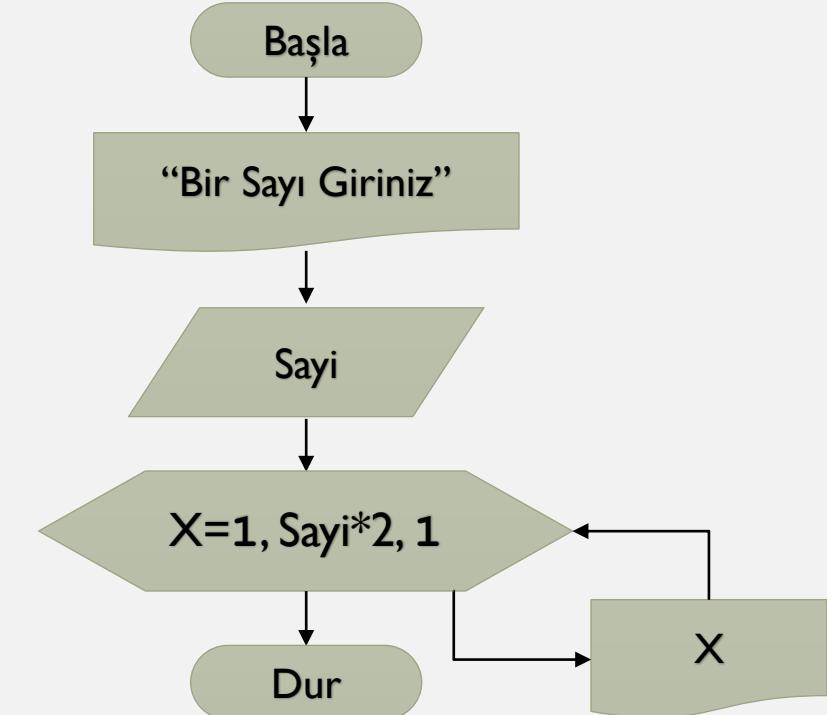
1. Başla
2. Toplam=0
3. DÖNGÜ (X=1 TO 100 STEP 1)
4. Yaz "Bir Sayı Girin"
5. Oku Sayı
6. Toplam=Toplam+Sayı
7. DÖNGÜSONU
8. Yaz Toplam
9. Dur

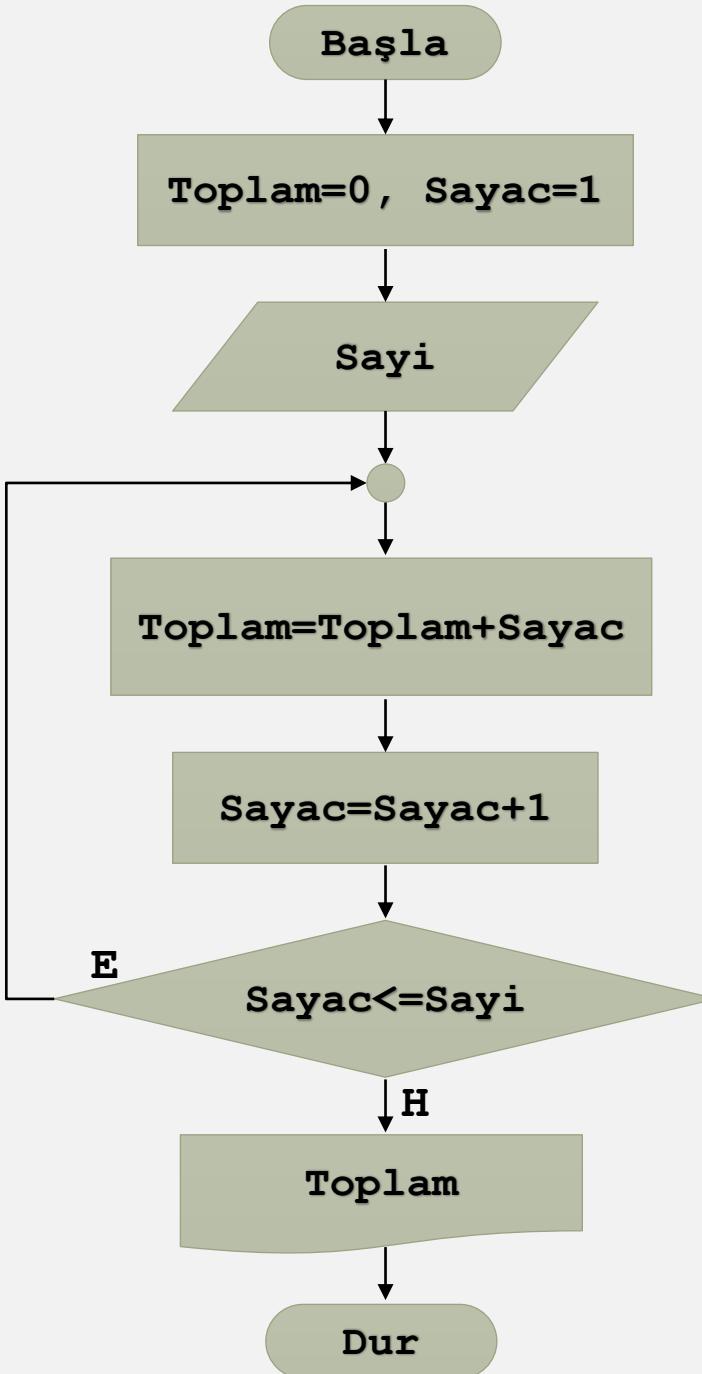


```
Toplam=0
FOR (X=1 TO 100 STEP 1)
    DISPLAY "Bir Sayı Girin"
    GET Sayı
    Toplam = Toplam + Sayı
ENDFOR
DISPLAY Toplam
```

Satır Algoritmalarından Akış Diyagramı Oluşturmak

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





Akış Diyagramlarından Sözde Kod Oluşturmak

Toplam=0 , Sayac=1

GET Sayı

LOOP

Toplam=Toplam+Sayac

Sayac=Sayac+1

ENDLOOP (Sayac<=Sayı)

DISPLAY Toplam

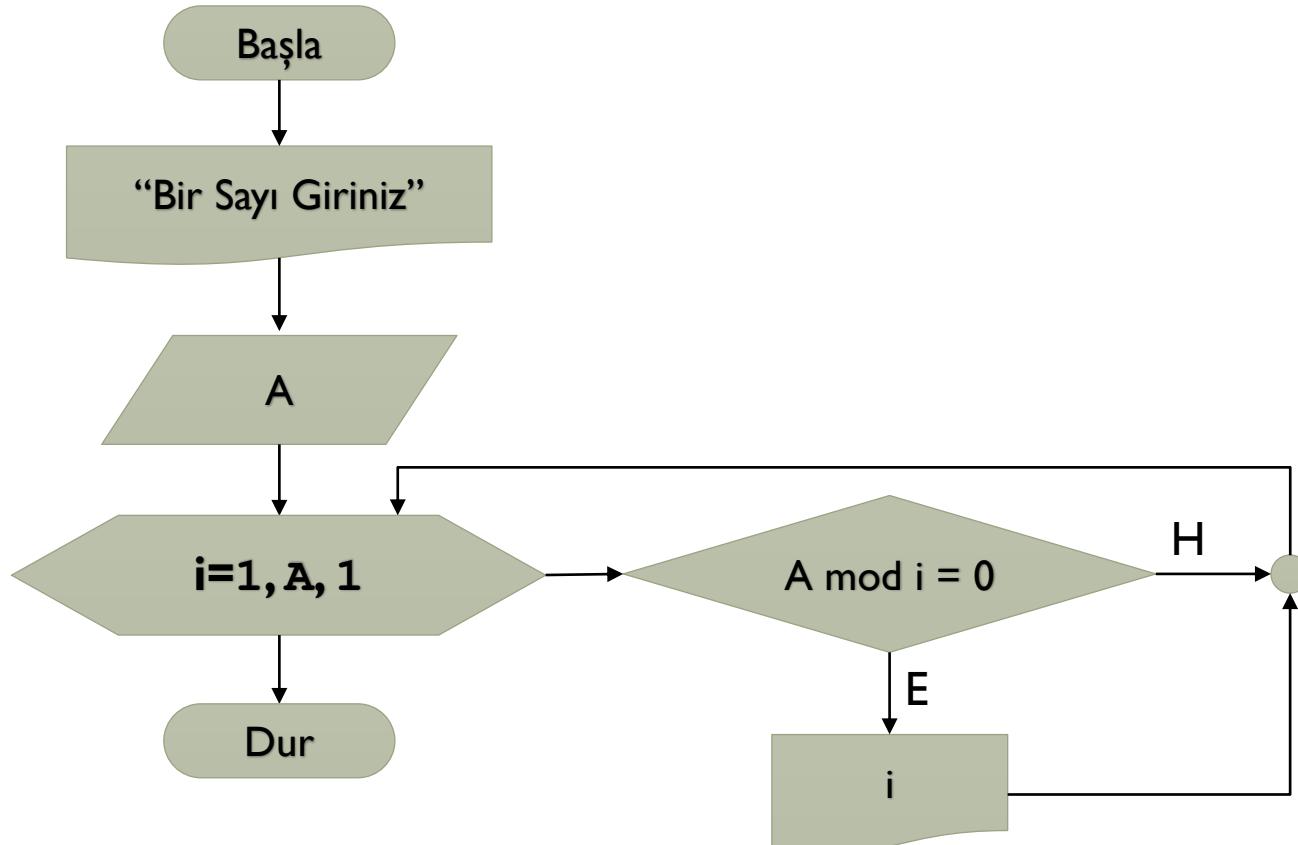
```

Toplam=0
GET Sayı
FOR Sayac=1 TO Sayı 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 kodolarak ifade ediniz.

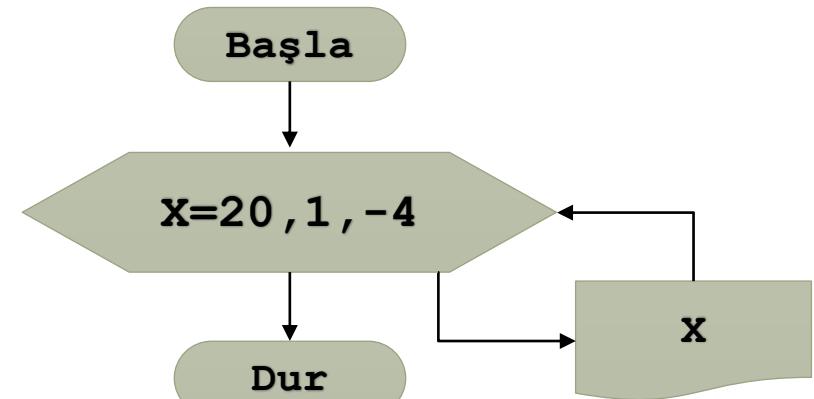
```
DISPLAY "Bir sayı giriniz"  
GET A  
FOR(i=1 TO A STEP 1)  
  IF (A MOD i = 0) THEN  
    DISPLAY i  
  ENDIF  
ENDFOR
```



Örnek

- 20'den başlayıp 1'e kadar, dörder dörder geriye doğru sayıp ekrana yazdırın 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 – I

- 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 yazdırın 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 yazdırın programı tasarlayıp;
 - Satır algoritma
 - Akış diyagramı ve
 - Sözde kodolarak ifade ediniz (*Kodlab*, s. 166).

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.

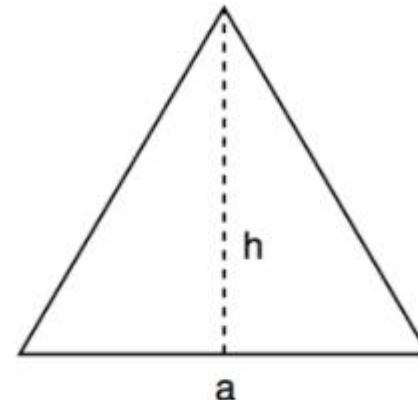
0	1	1
İki önceki sayı	Bir önceki sayı	Yeni sayı

- Fibonacci dizisinin ilk 10 elemanını hesaplayarak ekrana yazdırın programı tasarlayarak
 - Satır algoritma
 - Akış diyagramı ve
 - Sözde kodolarak ifade ediniz (*Kodlab*, s. 160).

Temel Algoritma Örnekleri & Genel Uygulamalar

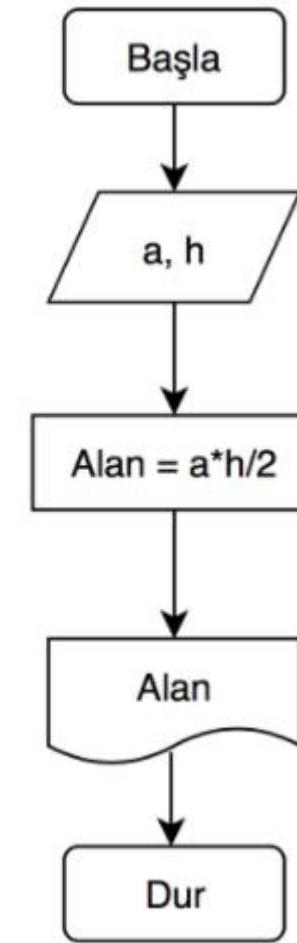
Üçgenin Alanı - I

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



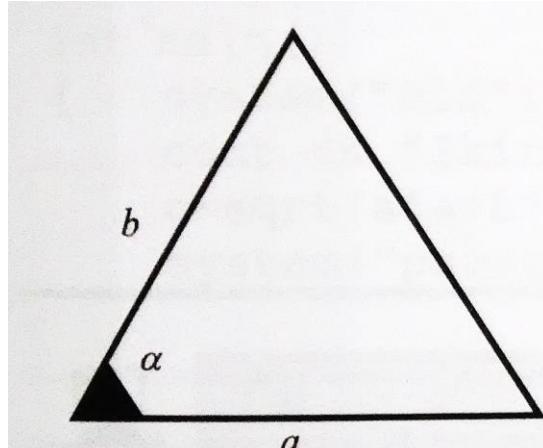
$$\text{Alan} = (a \cdot h) / 2$$

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



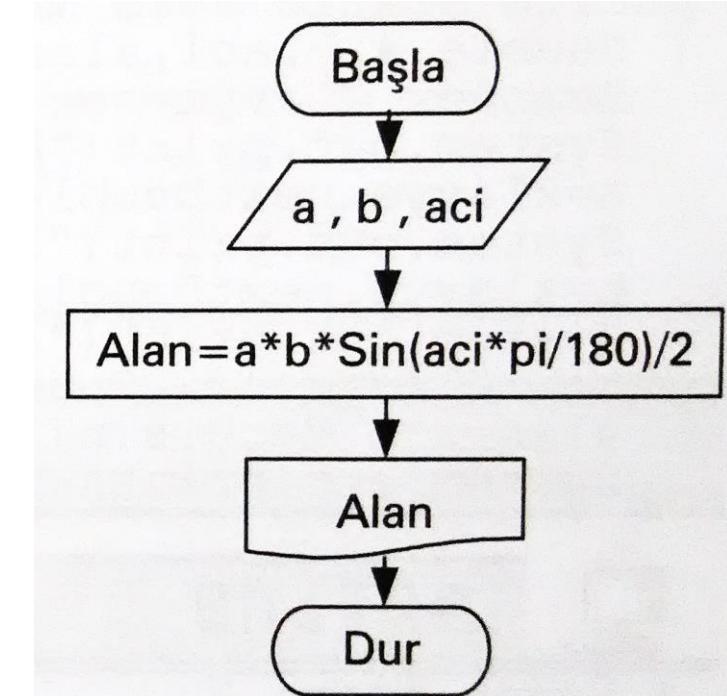
Üçgenin Alanı - 2

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



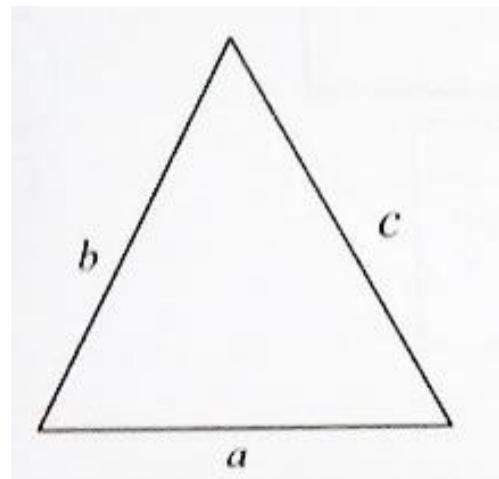
$$\text{Alan} = a.b.\text{Sin}(\alpha)/2$$

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



Üçgenin Alanı - Heron Formülü

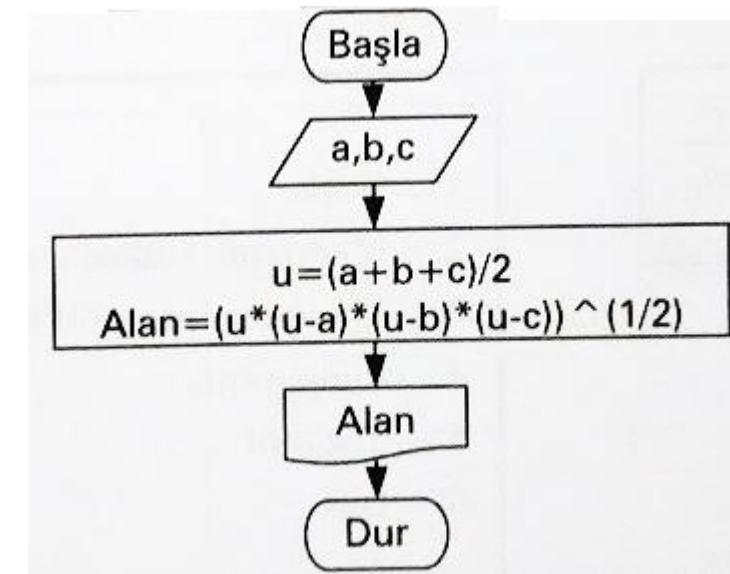
Klavyeden tüm kenar uzunlukları girilen üçgenin alanını hesaplayan programın satır kodunu ve akış diyagramını geliştiriniz.



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

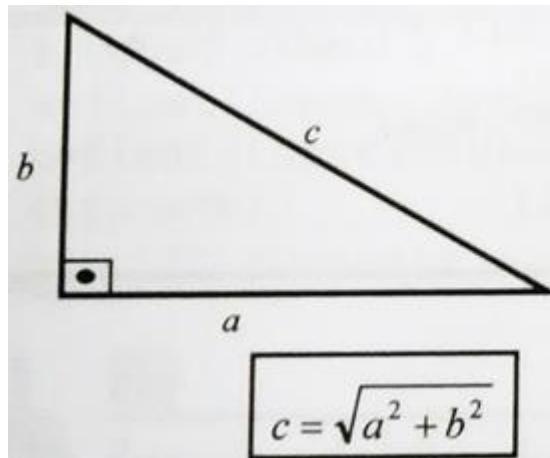
$$\text{Alan} = \sqrt{u(u-a)(u-b)(u-c)}$$

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

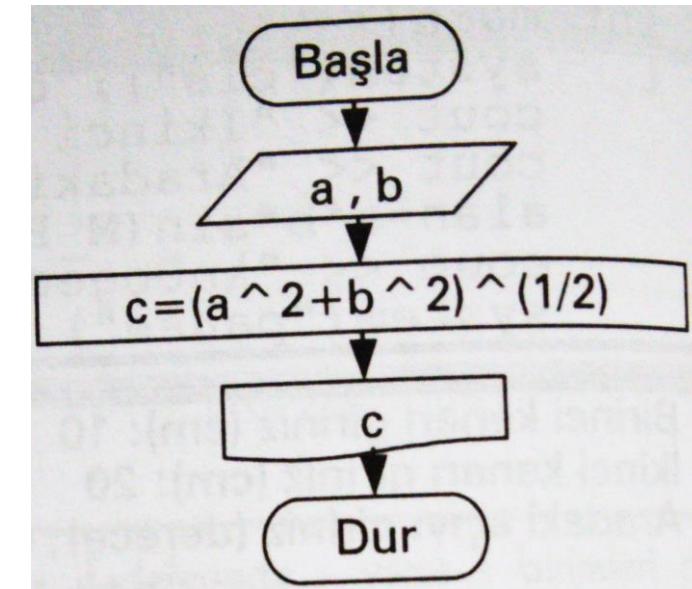


Pisagor Teoremi

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

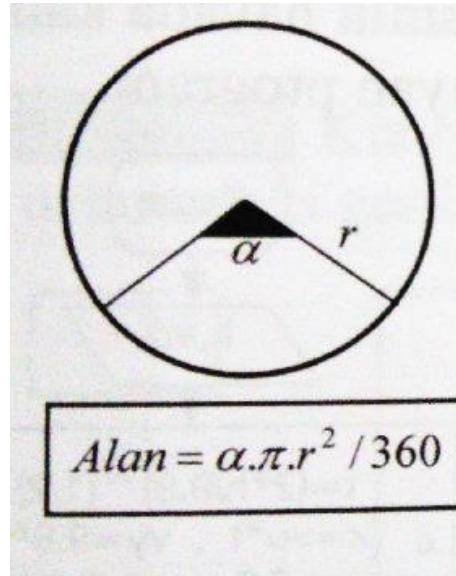


1. Başla
2. Birinci dik kenarı (a) gir
3. İkinci dik kenarı (b) gir
4. $c=(a^2+b^2)^{1/2}$
5. Yaz c
6. Dur

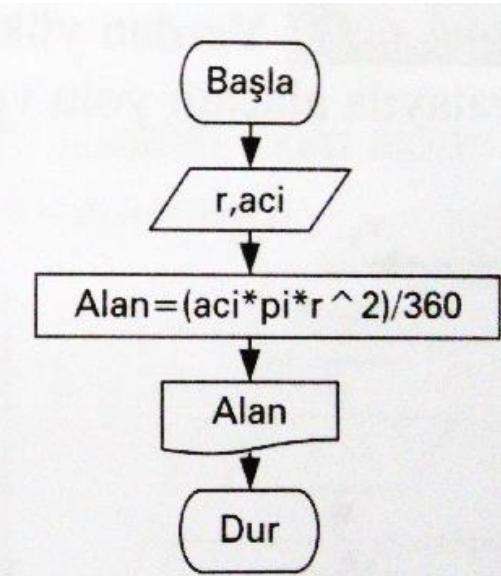


Daire Diliminin Alanı

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

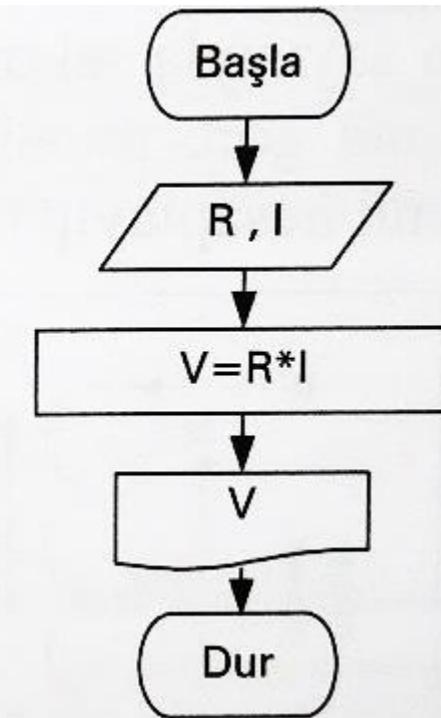


Ohm Kanunu

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

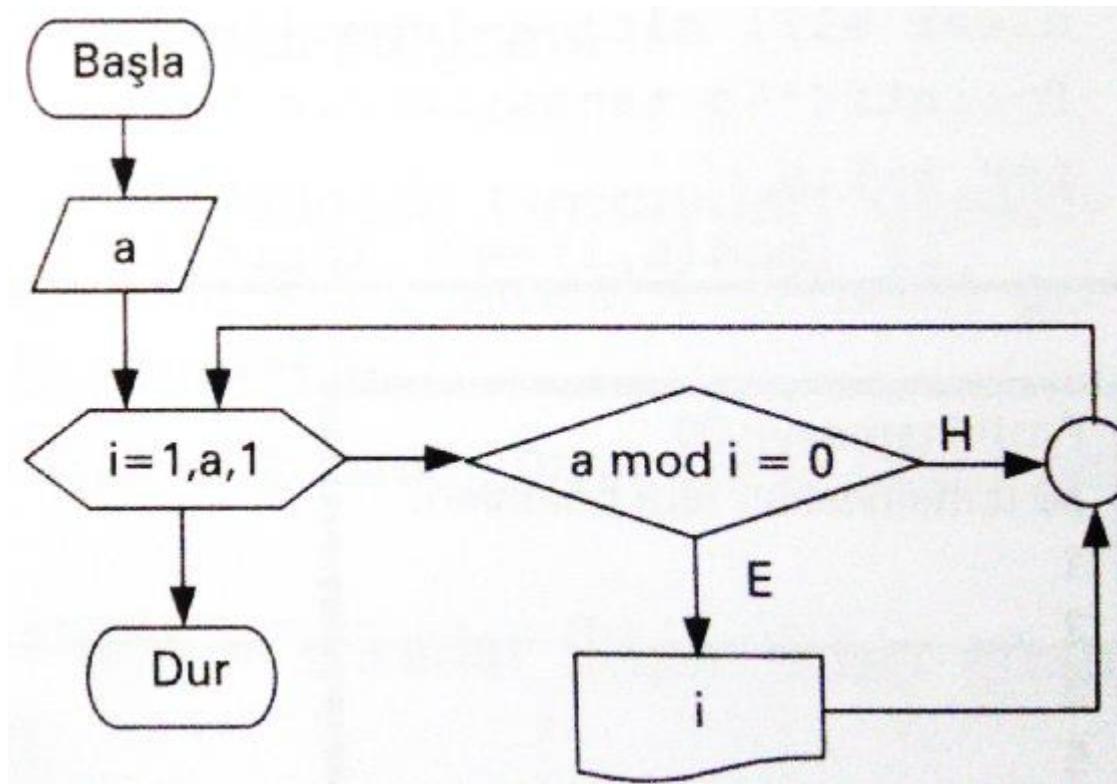
$$V = R \cdot I$$

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



Bir Tamsayıının Tam Bölenlerinin Bulunması

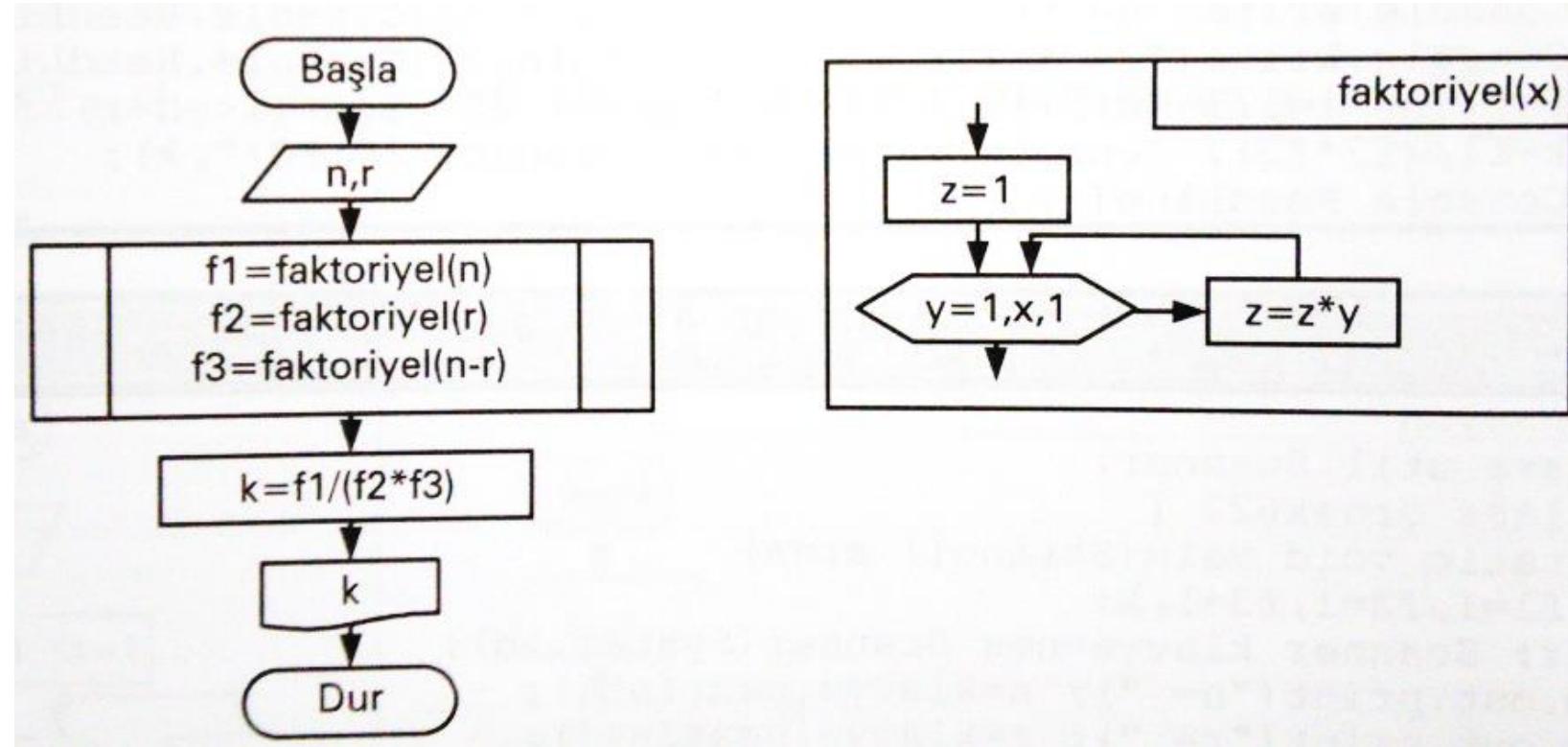
Klavyeden girilen pozitif bir a tamsayısının tam bölenlerini hesaplayıp listeleyen programı satır kod ve akış diyagramı olarak ifade ediniz.



İyileştirme?

Kombinasyon Hesaplama

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



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

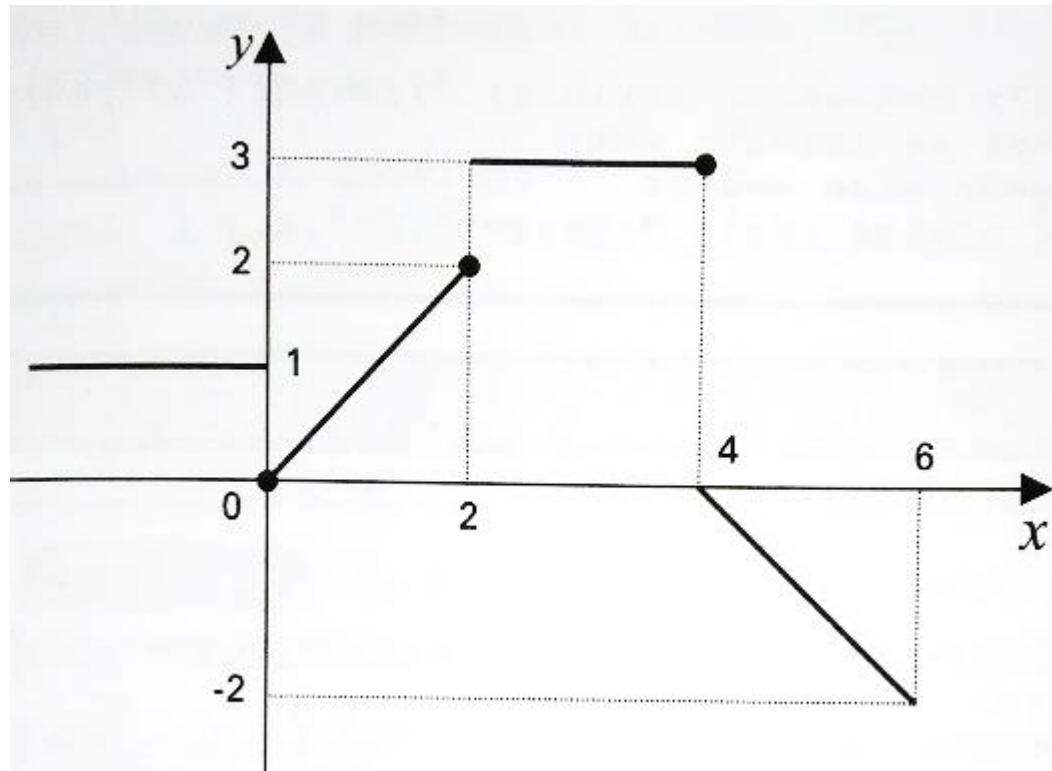
Birinci Dereceden Denklemin Kökü

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



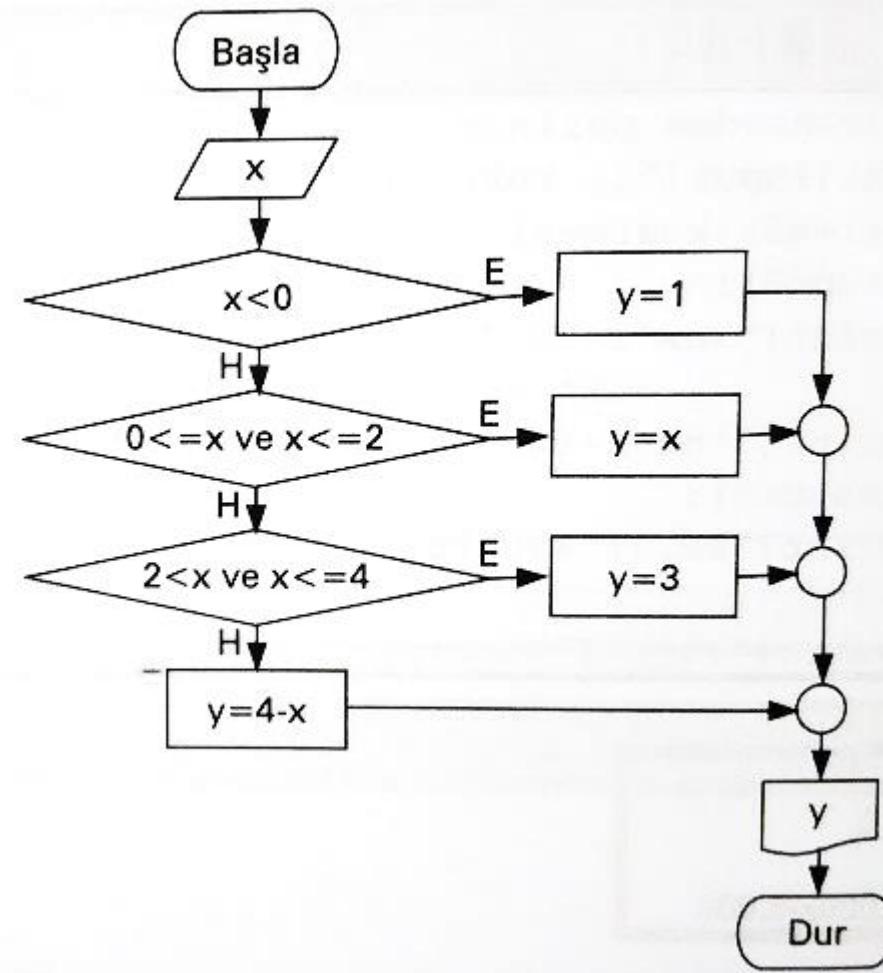
Grafiği Verilen Fonksiyon

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



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

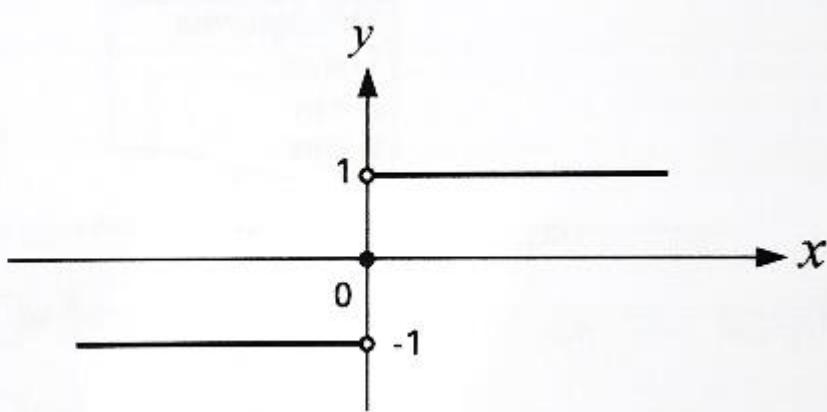
1. Başla
2. x değerini gir
3. Eğer $x < 0$ ise $y = 1$
4. Eğer $x \geq 0$ ve $x <= 2$ ise $y = x$
5. Eğer $x > 2$ ve $x \leq 4$ ise $y = 3$
6. Eğer $x > 4$ ise $y = 4 - x$
7. Yaz y
8. Dur



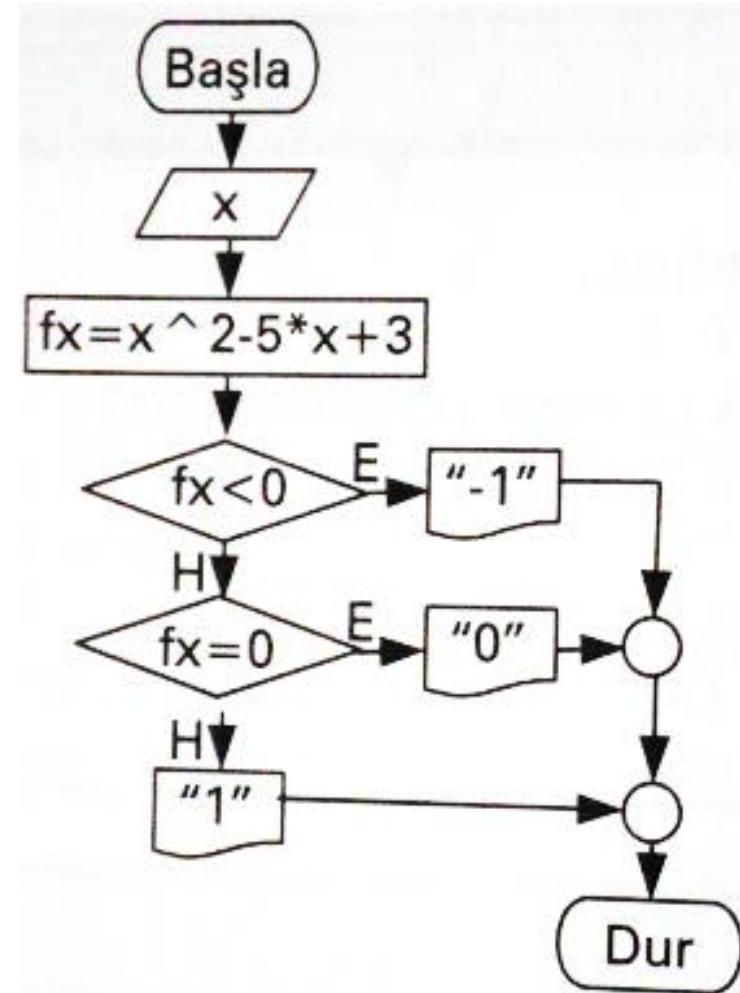
İşaret (Signum) Fonksiyonu

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.

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



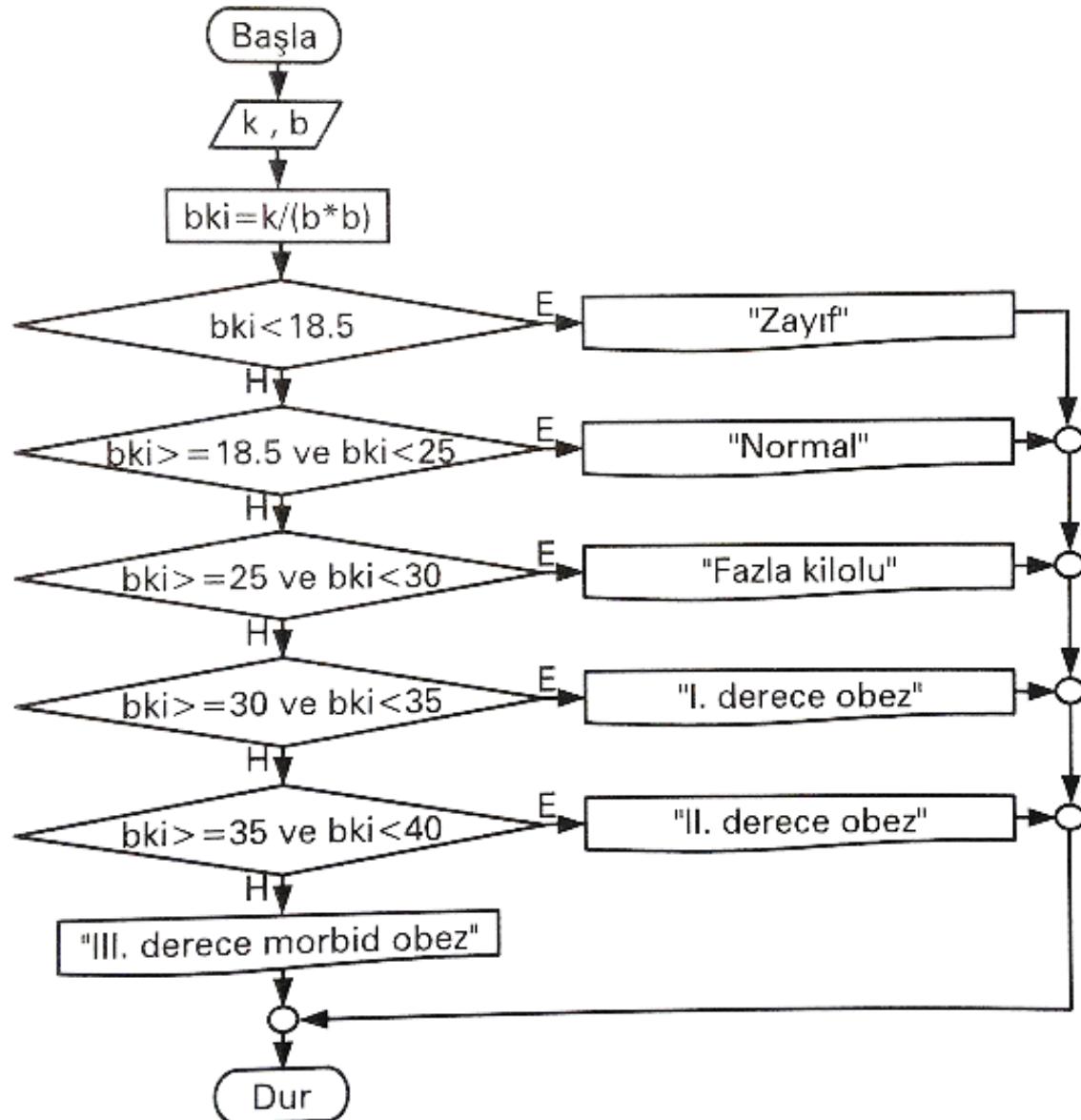
1. Başla
2. x değerini gir
3. $fx = x^2 - 5*x + 3$
4. Eğer $fx < 0$ ise Yaz "-1"
5. Eğer $fx = 0$ ise Yaz "0"
6. Eğer $fx > 0$ ise Yaz "1"
7. Dur



İdeal Vücut Ağırlığı

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

	BKI (kg/m ²)
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 sayılaraya 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 yazdırın programı geliştiriniz.

Soru 1:

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

Soru 2:

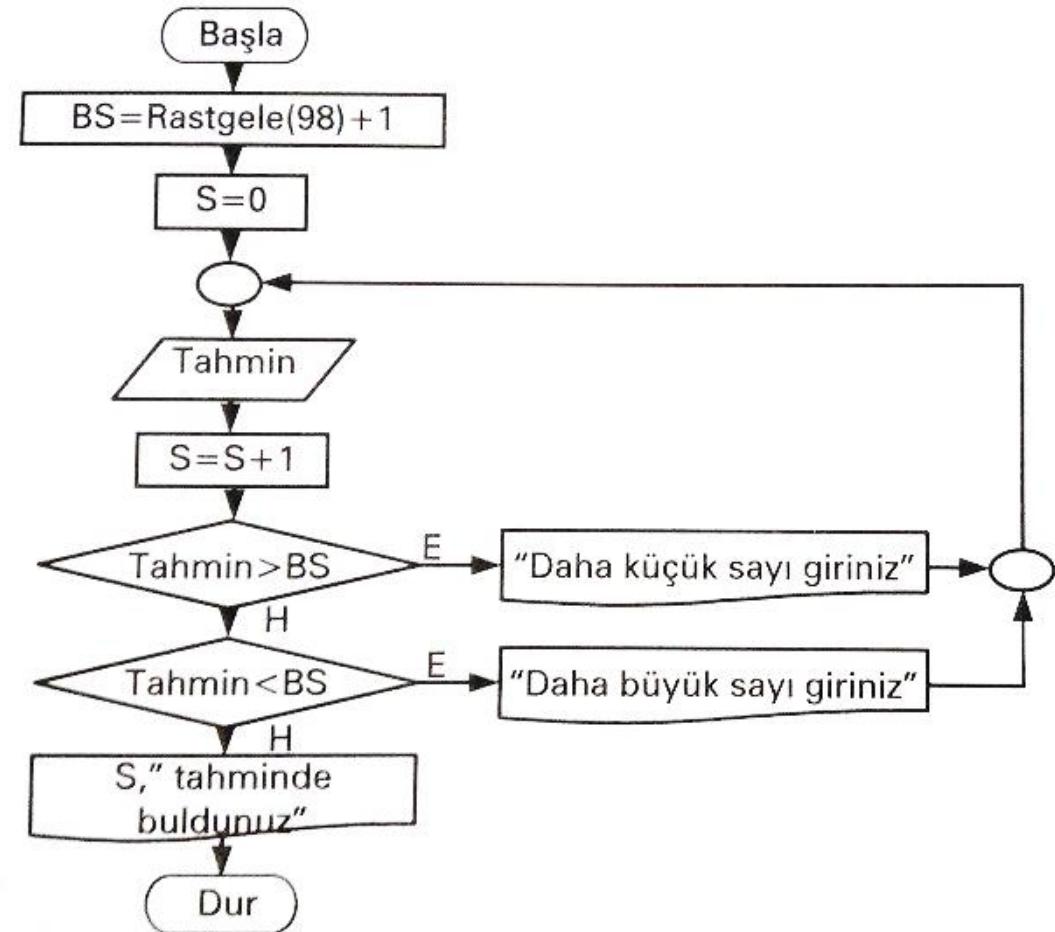
Klavyeden girilen b tamsayısına göre $a^3 - a^2 = b$ şartını sağlayan $0 < a < 100$ tamsayısını nasıl bulursunuz (Seçkin s.213) ?

Soru 3:

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

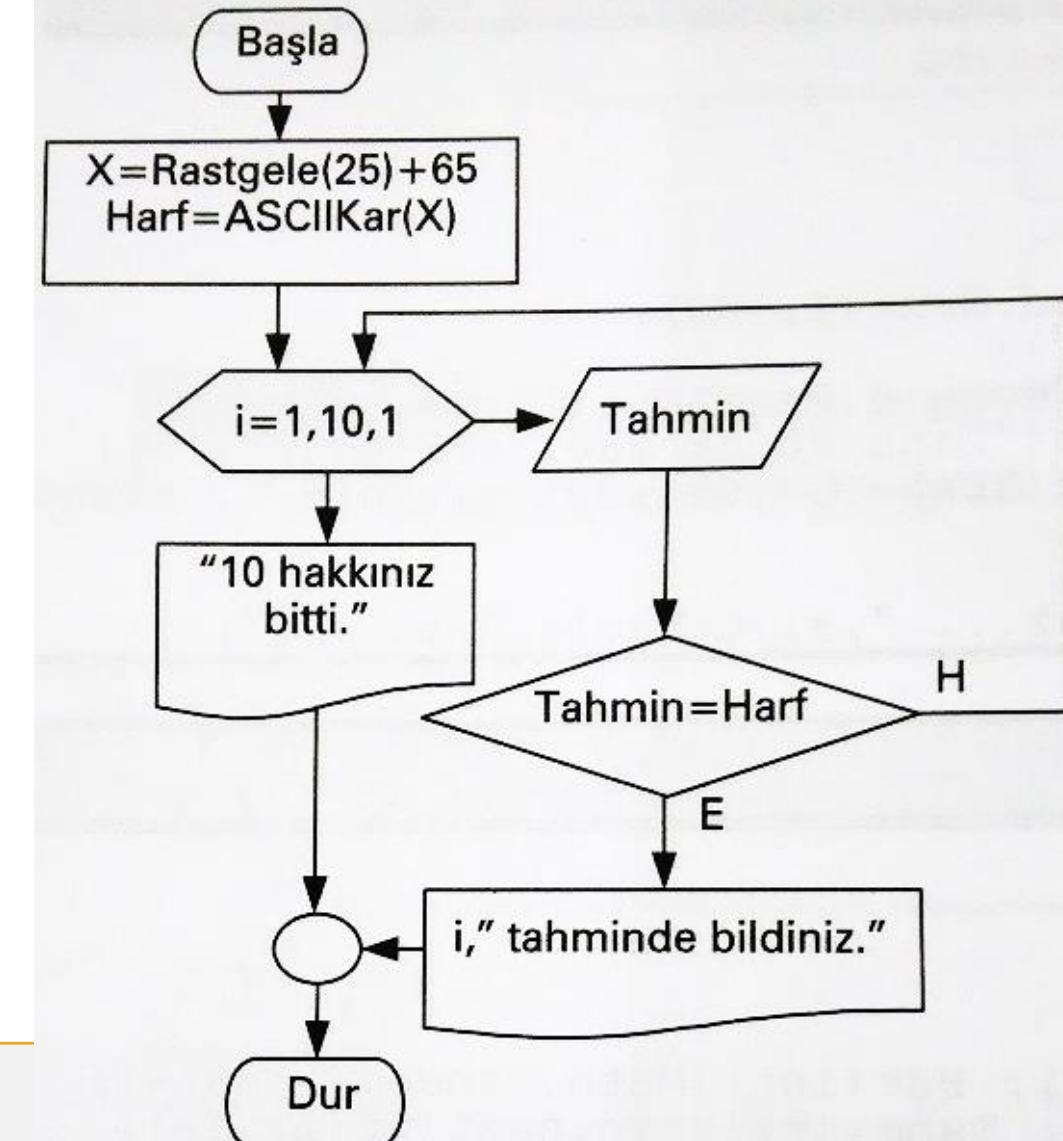
Sayı Tahmin Oyunu

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



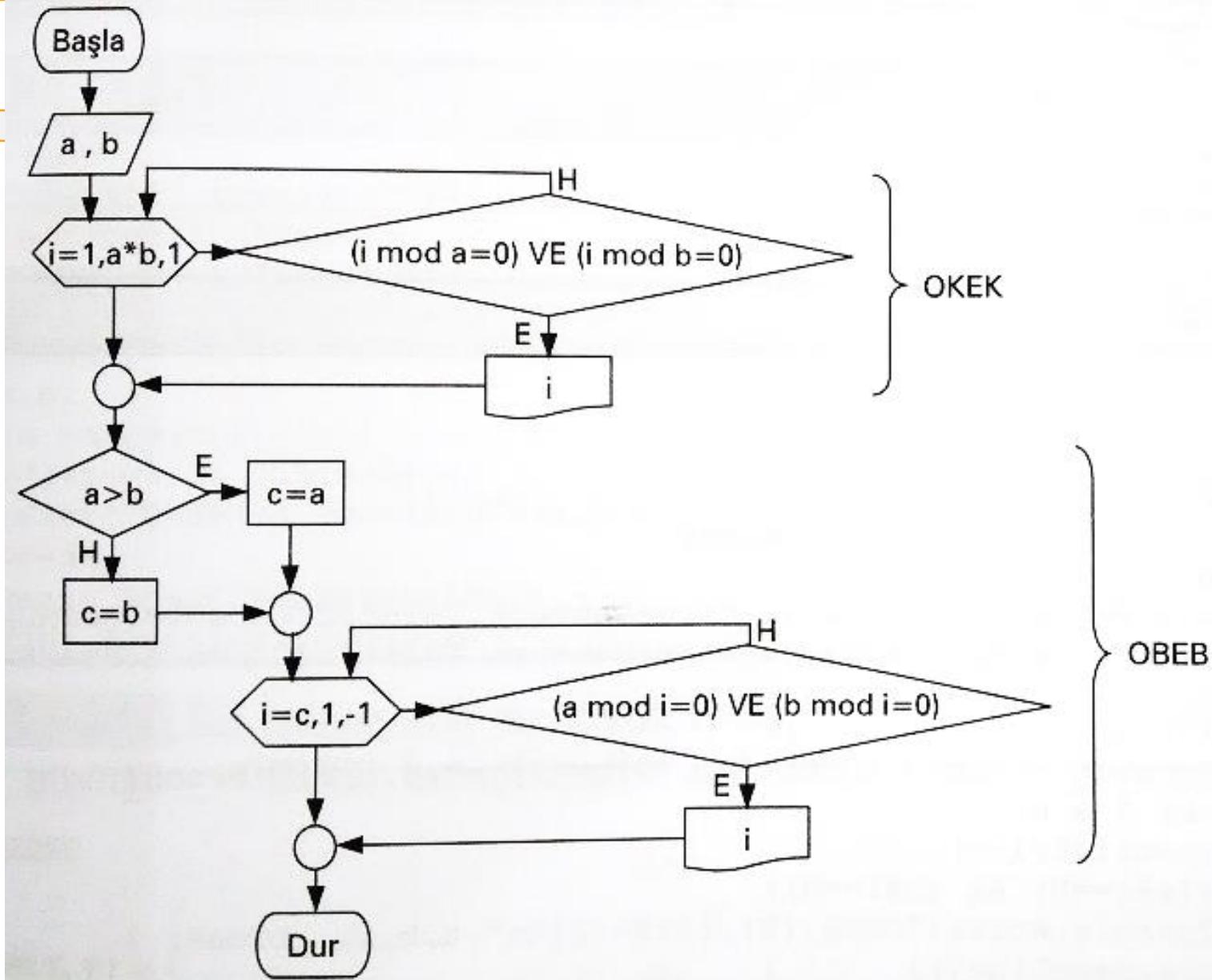
Harf Tahmin Oyunu

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



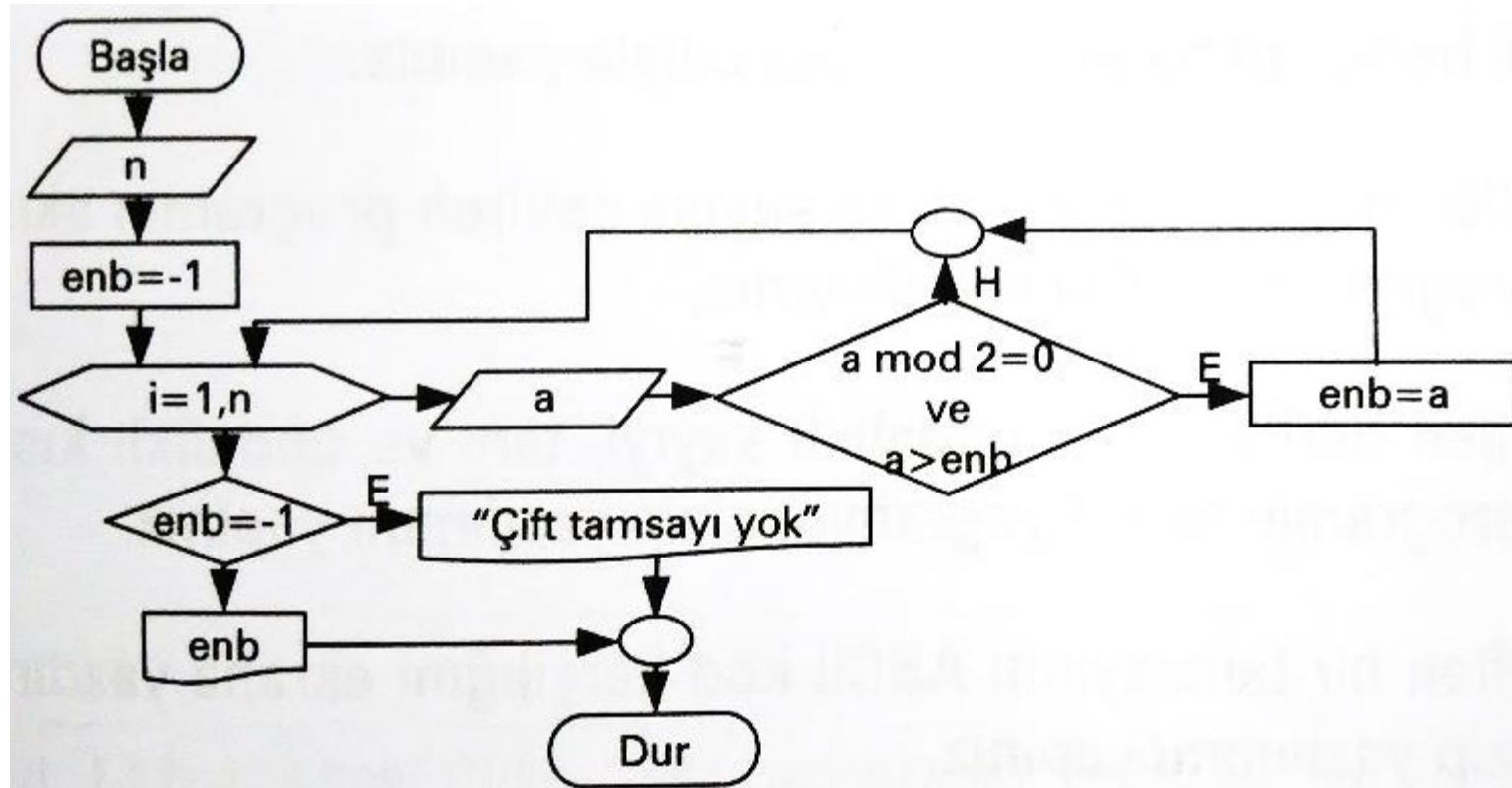
OKEK-OBEB

Klavyeden girilen iki pozitif tamsayının ortak katlarının en küçüğü (OKEK) ile ortak katlarının en büyüğünü (OBEB) hesaplayıp ekrana yazdırın 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 yazdırın 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 yazdırın programın sözde kodunu yazınız ve akış diyagramını çiziniz.

Java Programlama Dili ve Algoritmadan Kodlamaya Geçiş

Java'da Veri Tipleri

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

I. 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	$3,4 \cdot 10^{-38}$	$3,4 \cdot 10^{38}$	4
double	$1,7 \cdot 10^{-308}$	$1,7 \cdot 10^{308}$	8

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
<code>&&</code>	VE
<code> </code>	VEYA
<code>!</code>	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**

e : **Math.E**

x^y : **Math.pow(x, y)**

\sqrt{x} : **Math.sqrt(x)**

Rastgele(x): **Math.random() * (x+1)**

Radyan \rightarrow Derece: **Math.toDegrees()**

Derece \rightarrow Radyan: **Math.toRadians()**

e^x : **Math.exp(x)**

Üste Yuvarla: **Math.ceil(x)**

Aşağı Yuvarla: **Math.floor(x)**

En Yakın Tamsayıya Yuvarla: **Math.round(x)**

Mutlak Değer: **Math.abs(x)**

Mod: $\%$

En Büyük: **Math.max(x)**

En Küçük: **Math.min(x)**

Sırala: **Arrays.sort()**

$Ln(x)$: **Math.log(x)**

$Log(x)$: **Math.log10(x)**

$Sin(x)$: **Math.sin(x)**

$Cos(x)$: **Math.cos(x)**

$Tan(x)$: **Math.tan(x)**

Bazı Alfasayısal İşlem Komutları – String ifadeler için

Uzunluk: `.length()`

Büyük: `.toUpperCase()`

Küçük: `.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:

Programındaki 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 çağrılmazı gereklidir. Herhangi bir Java programı yazıldığında, Java standart kütüphanesi (*java.lang* paketi) otomatik olarak çağrıılır. Fakat kullanılacak diğer paketlere ait sınıflar, nesneler, fonksiyonlar kullanılabilecekse bunların “*import*” ile çağrılmazı gereklidir.

```
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 çağrılp kullanılır.

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 gereklidir. “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



Başla

Java'daki Karşılığı

Açıklamalar, bildirimler

Değişken

değişken=nextInt(); değişken.nextLine(); ...

Değişken

System.out.println(değişken); System.out.print(değişken); ...

İşlem

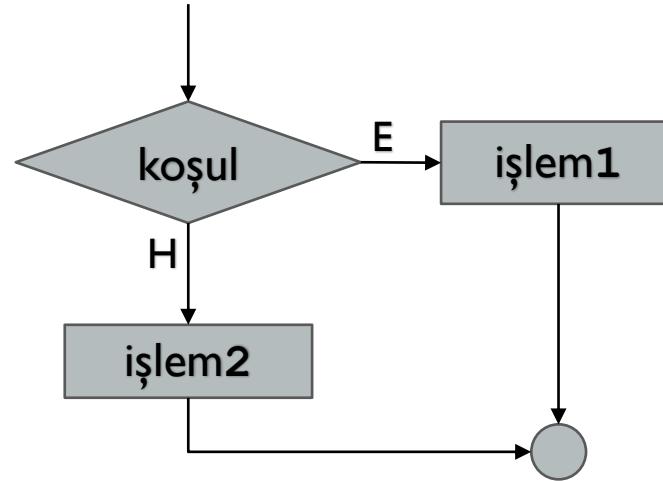
İşlem

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 çağrıılır. “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.

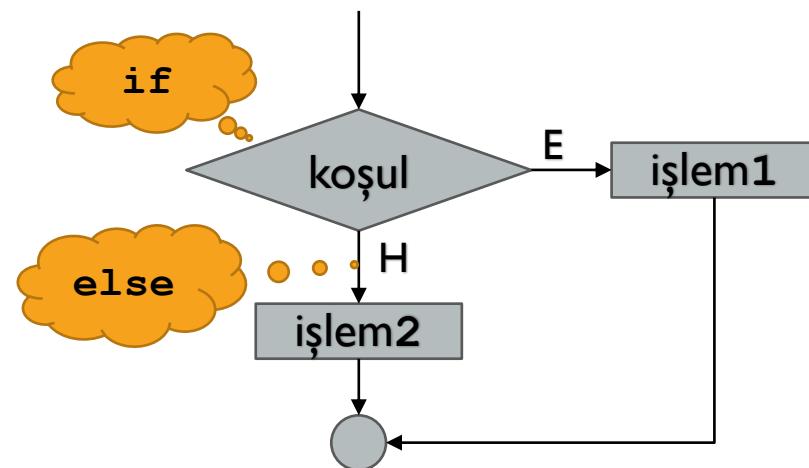
Veri Giriş/Çıkış Komutları

```
//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 yanlışlığında 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 biçimini 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 biçimini 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 biçimini ise aşağıdaki gibidir.

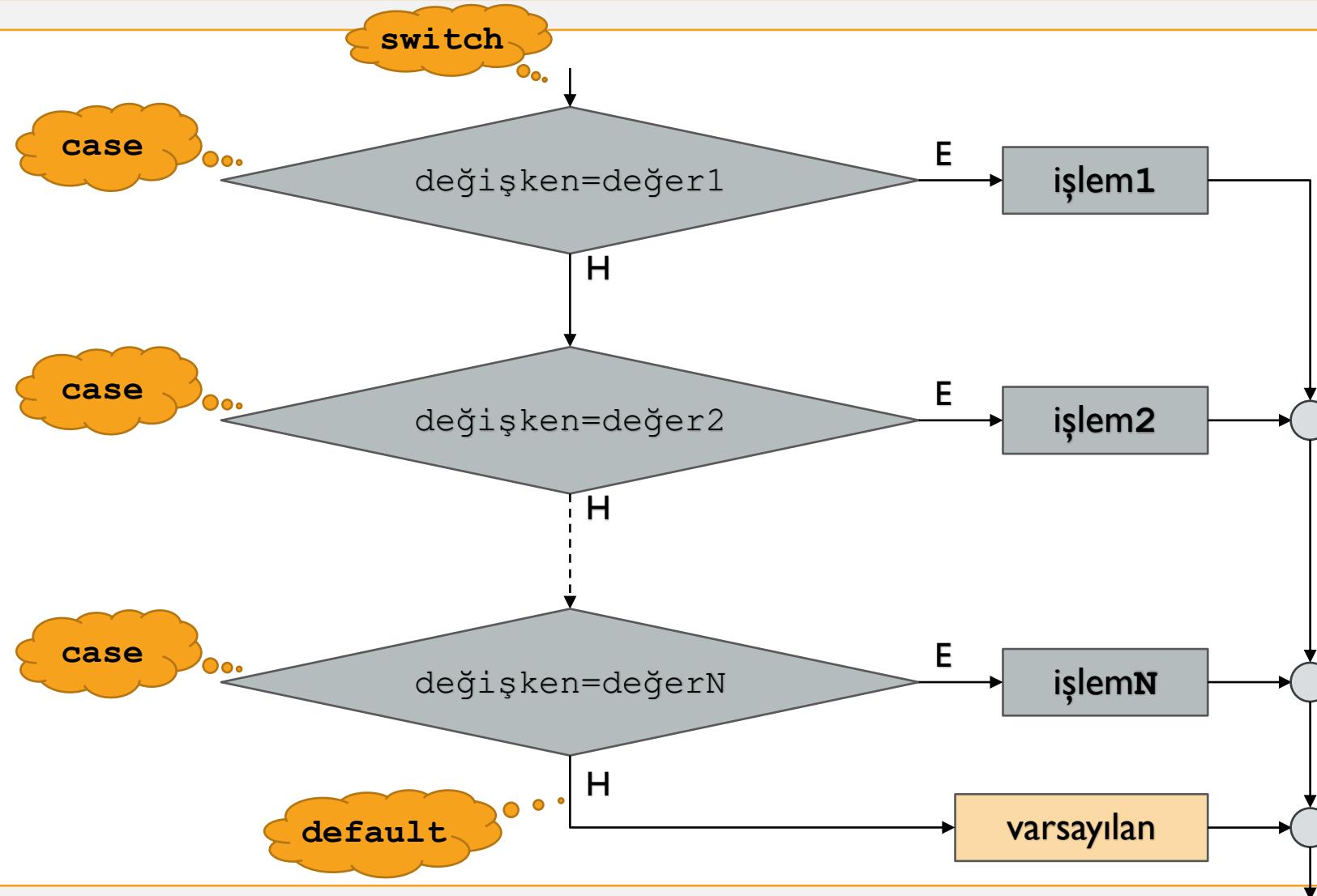
```
if ( koşul1 ) {  
    .....  
    ..... } } Koşul1 Doğru ise Yapılacak İşlemler  
} else if ( koşul2 ) {  
    .....  
    ..... } } Koşul1 Yanlış, Koşul2 Doğru ise  
} else {  
    .....  
    ..... } } Koşul1 ve Koşul2 Yanlış ise  
}  
    .....  
    .....
```

```
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



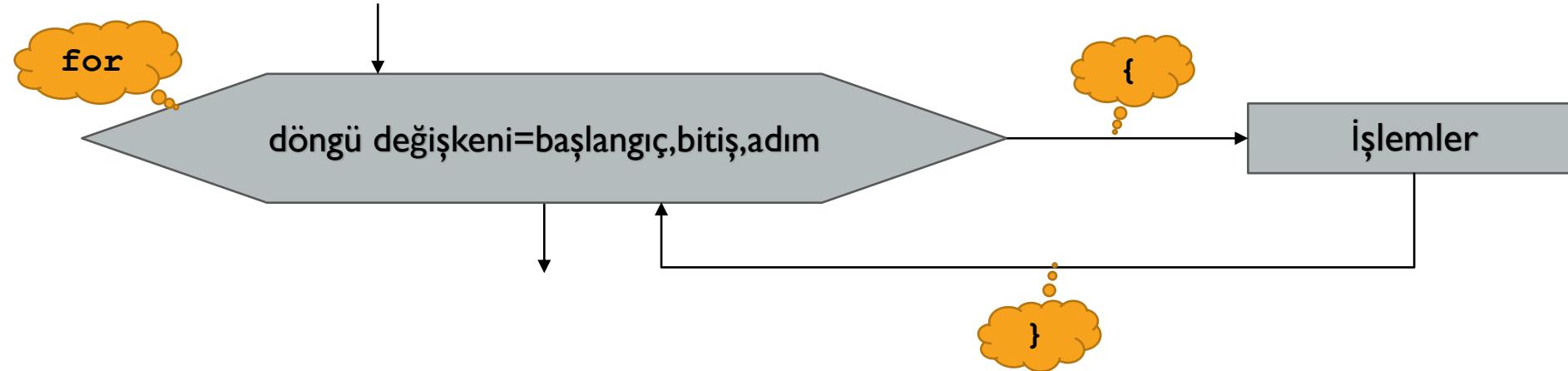
```
//Karar 2
import java.util.Scanner;
public class ornek {
    public static void main (String[] args) {
        Scanner klavye = new Scanner(System.in);
        System.out.print ("Notunuzu (1-5) giriniz: ");
        int a=klavye.nextInt();
        switch (a) {
            case 1: {
                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:

- Sayıçılı 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.

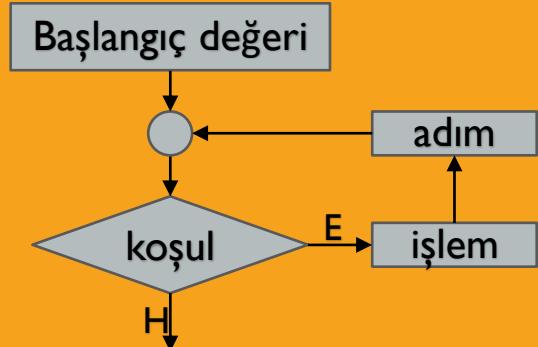
Döngü Komutları - for



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

} } İşlemler

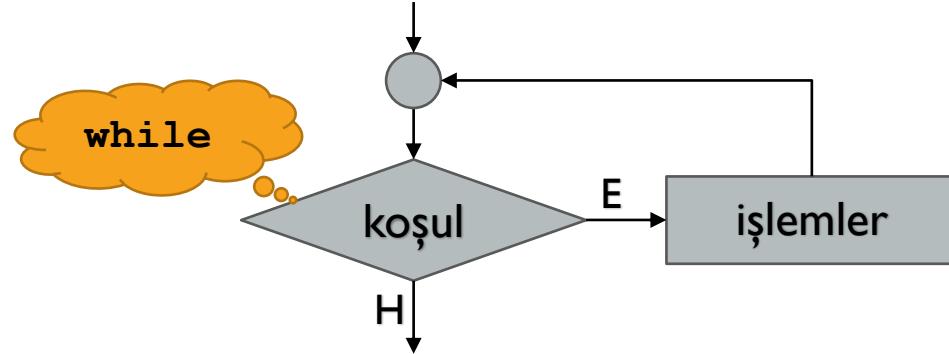
Alternatif gösterim



Döngü Komutları - for

```
//Döngü 1
public class örnek {
    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

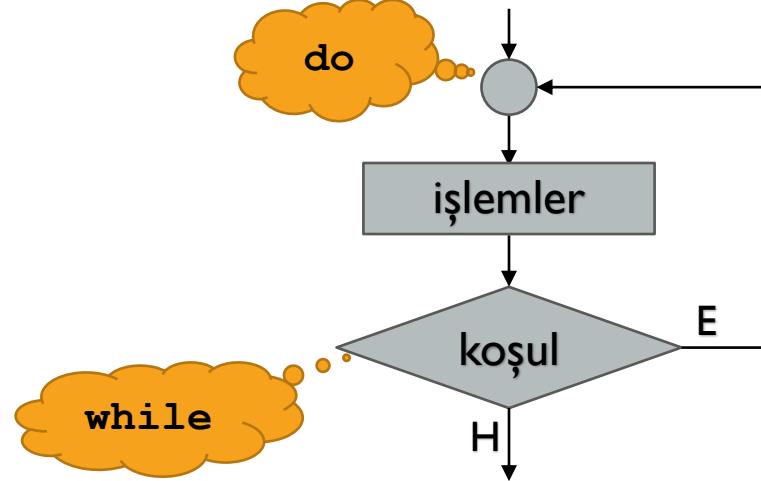


```
while ( koşul ) {  
    .....  
    ..... }  
}
```

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

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 elemanın 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

1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7

Array's last index = 7

Arraylength = 7 + 1 = 8

Dizinin uzunluğu: 8

long form

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