Algoritma Ve Programlamaya Giriş

Programlama Temelleri

Bilişim Dünyası

Bilgisayarın Çalışma Mantığı

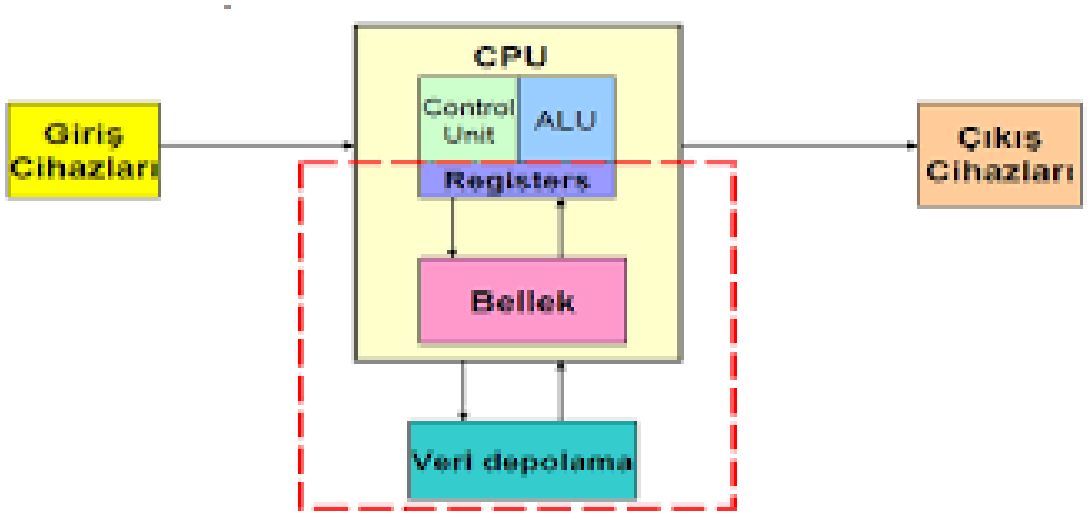
Bilgisayar, kullanıcıdan aldığı verilerle mantıksal ve aritmetiksel işlemleri yapan yaptığı işlemlerin sonucunu saklayabilen, sakladığı bilgilere istenildiğinde ulaşılabilen elektronik bir makinedir. Bu işlemleri yaparken veriler girilir ve işlenir. Ayrıca, istenildiğinde yapılan işlemler depolanabilir ve çıktısı alınabilir.

Giriş: Kişi tarafından veya bilgisayar tarafından sağlanan verilerdir. Bu veriler, sayılar, harfler, sözcükler, ses sinyalleri ve komutlardır. Veriler giriş birimleri tarafından toplanır.

İşlem: Veriler insanların amaçları doğrultusunda, programın yetenekleri ölçüsünde işlem basamaklarından geçer.

Bellek: Verilerin depolandığı yerdir. Giriş yapılan ve işlenen veriler bellekte depolanır.

Çıkış: Bilgisayar tarafından işlem basamaklarından geçirilerek üretilen yazı, resim, tablo, müzik, grafik, görüntü, vb.nin ekrandan ya da yazıcı, hoparlör gibi değişik çıkış birimlerinden alınmasıdır.



Bilgisayarın Anladığı Dil

Bilgisayarlar biz insanların konuştuğu dili anlamazlar. Yani günlük hayatta kullandığımız sözcükleri, renkleri, sesleri, sayıları bilgisayarlar anlamazlar. Peki bilgisayarların anladığı dil nedir? Bilgisayarlar sadece 0 ve 1 rakamlarını anlar ve üzerinde işlem yapar. Bizler günlük hayatta 10luk sayı sistemini kullanırken bilgisayarlar ikilik sayı sistemini kullanmaktadır. Bizim girdiğimiz sayılar, sözcükler, renkler ve sesler bilgisayarlar tarafından 0 ve 1 rakamlarına dönüştürülerek işlem yapılır ve yine bizim anlayacağımız dile çevrilir.

- Sayısal dönüşümler olur fakat metinsel ifadeler de mi ikilik sayı sistemine dönüştürülür?

Bilgisayarda renklerin ve karakterlerin sayısal karşılıkları bulunur.

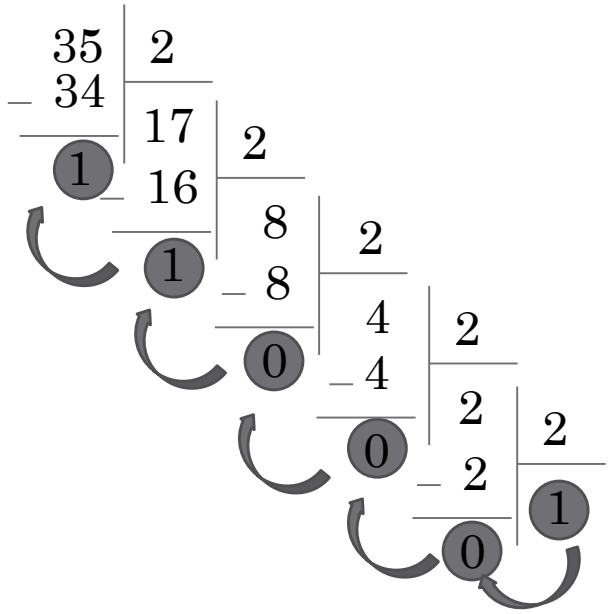
Bilgisayar elektronik aygıt olduğu için elektrik sinyalleri ile işlem yapmaktadır ve 0 elektrik sinyalinin yokluğunu ve 1 elektrik sinyalinin varlığını gösterir. İşlemci içinde bulunan milyarlarca transistörler girilen verileri 0 ve 1 rakamlarına dönüştürür.

0 ve 1 rakamlarına bilgisayar alanında **bit** denilmektedir. Ve 8 tane bit bir araya gelerek bir karakteri oluşturur. 8 bitlik bir bilgi aynı zamanda 1 byte olarak adlandırılır. Bilgisayarda bir karakter 1 byte alan kaplamaktadır.

10 Tabanındaki Sayıyı 2 Tabanına Dönüştürmek

Bu tip dönüşümlerini lise matematik derslerinde aslında görmüşüzdür. Verilen 10 tabanındaki sayıyı 2 tabanına dönüştürmek için 2 sayısı ile bölme işlemi yapılır ve sağdan sola doğru kalan sayılar yan yana yazılır.

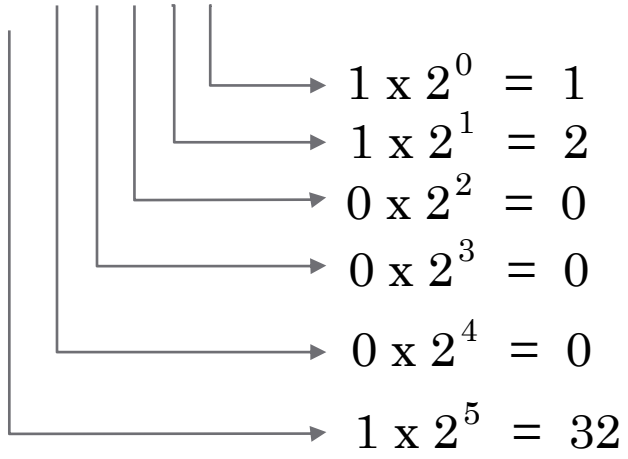
Örnek: 35 sayısının ikilik tabandaki karşılığını hesaplayınız.



$$35 = (100011)_2$$

2 Tabanındaki Sayıyı 10 Tabanına Dönüştürmek

- 2 sayı tabanındaki sayıyı 10 tabanına dönüştürmek için 2 sayı tabanındaki sayının her basamak değerini 2nin kuvvetleri ile çarpılır ve bu değerler toplanır.
- Örnek : $(100011)_2$ sayısının 10 tabanındaki karşılığı nedir?
- 1 0 0 0 1 1

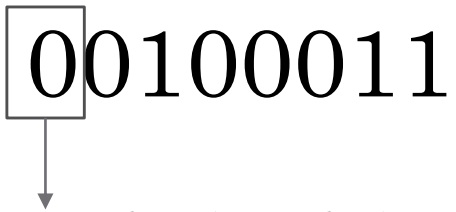


1 + 2 + 32 = 35

Negatif Sayıların Bitsel Gösterimi

Bilgisayarda verilerimiz 8 bit şeklinde gösterilmektedir. Az önce bulduğumuz 100011 sayısı bilgisayar tarafından 8bite tamamlanmaktadır.

Yani 35 sayısı bilgisayarda 00100011 şeklinde saklanmaktadır. Bilgisayarda pozitif ve negatif kavramı olmadığından negatif sayıları saklamak için ilk bit değeri işaret biti olarak ayarlanmaktadır.



İşaret biti(sign bit) 0 ise sayının pozitif olduğunu 1 ise negatif olduğunu göstermektedir.

-35 sayısını bilgisayarda bitsel olarak ifade etmek için farklı yöntemler bulunmaktadır.

1. Yöntem : İşaret biti 1 yapılırsa sayımız -35 olur. Yani 10100011 sayısı -35 olur.
2. Yöntem : 1'e tümleme yöntemi: bitler 0 ise 1e, 1 ise 0'a dönüştürülerek yapılır. Yani 11011100 olacaktır.
3. Yöntem : 2'ye tümleme yöntemi: bilgisayarda ve bilimde en sık kullanılan yöntemdir.

Negatif Sayıların Bitsel Gösterimi

3. Yöntem 2ye tümleme yöntemi en sık kullanılan gösterimdir. Sağdan sola doğru ilk 1 bitini görene kadar sayımız yazılır 1 de dahildir. Sonrasında kalan tüm bitler terse çevrilir yani 1 ise 0, 0 ise 1 e dönüştürülür.

10011000 Sayımızı 2ye tümleme yöntemi ile negatife dönüştürelim.

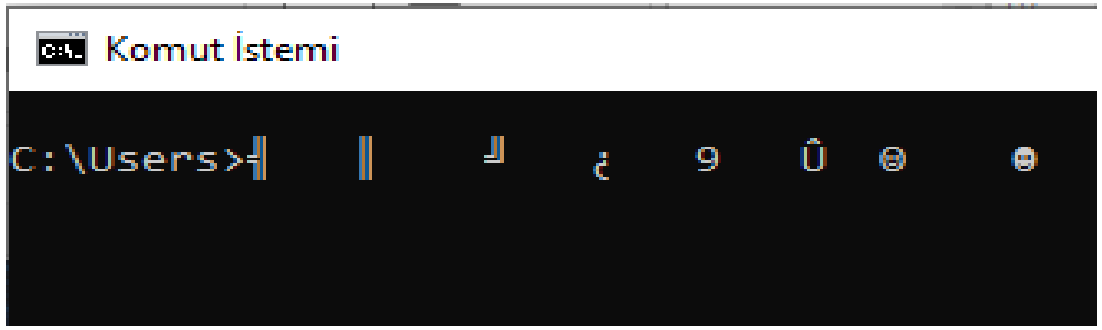
01101000 olarak karşımıza gelecektir.

Bu konu algoritma eğitimimiz için çok iyi bilinmesine gerek yoktur. Bizim yazdığımız programlar derleyiciler tarafından 1 ve 0 lara dönüştürülmektedir. Sizlere genel kültür olması açısından bu bilgiler verildi. Eğer daha detaylı olarak öğrenmek isterseniz internette araştırma yapabilirsiniz.

Karakterlerin 2 lik sayı tabanına dönüştürülmesi

Karakterler sayısal olmayan ifadelerdir. A-Z veya noktalama işaretleri gibi. Her karakterin bilgisayarda bir sayısal karşılığı vardır. Ve ortak dil olarak ASCII sayısal kodlar kabul edilmiştir. Örneğin A karakterinin karşılığı 10 sayı tabanındaki karşılığı 65'tir. Detaylı bilgi için ASCII kodlar şeklinde arama motorlarında arama yapılabilir.

Bizler klavyeden A karakterine bastığımız zaman otomatik olarak ikili sayı tabanına dönüşmektedir.

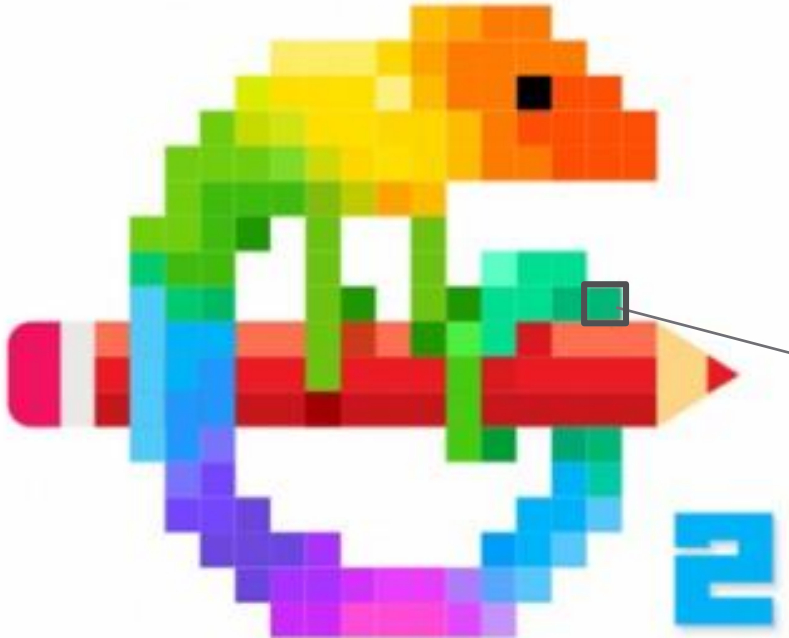


Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
?	63	0077	0x3f	_	95	0137	0x5f				

Renklerin Sayısal Karşılıkları

- Bilgisayarda oluşturulan görüntüler dijital görüntülerdir. Yine 0 ve 1 (bitler) lerden oluşmaktadır. Her rengin bir sayısal karşılığı bulunmaktadır. RGB, CMYK, Hexadecimal sayısal cinsinden karşılıkları bulunmaktadır.

Renk & İsim	R-G-B Eşdeğeri	Hexadecimal # (Kodu)
Kırık Beyaz / Broken White	242 242 242	# f2f2f2
Hayalet Beyaz / Ghost White	248 248 255	# f8f8ff
Beyaz Duman / White Smoke	245 245 245	# f5f5f5
Gainsboro Beyazı / Gainsboro	220 220 220	# dcdcdc
Beyaz / White	255 255 255	# ffffff

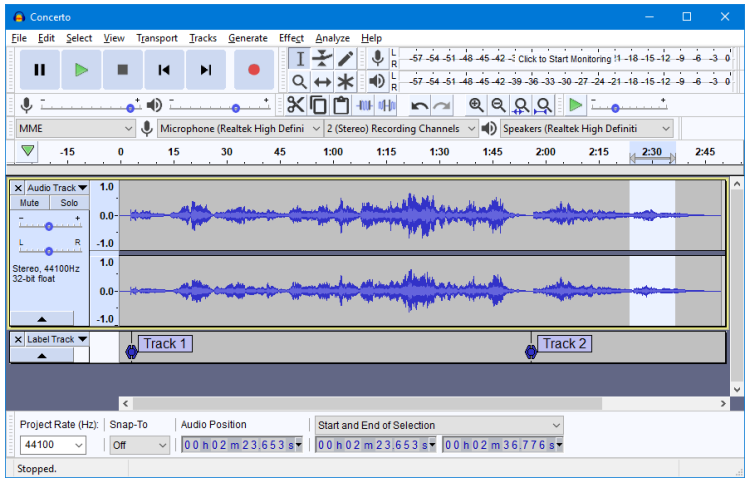


Bir resmi oluşturan en küçük birime pixel ismi verilmektedir. Ve pixel dediğimiz kutucuklar renkler ile doludur.

Pixel

Programların Amacı Nedir?

- Bilgisayarda kullanmış olduğumuz programlar aslında bir sorunu veya bir problemi çözmek için oluşturulan kodlar bütünüdür.
- Örneğin resimler üzerinde işlem yapabilmek için Photoshop Programı
- Video izlemek için Windows Media Player veya GOM Player Programı
- Ses düzenlemek için Audacity gibi programlar aslında bir problemi ortadan kaldırmak için üretilmiş programlardır.



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.

Bilgisayarlara ne yapmaları gerektiğini söylememizi sağlayan özel bir dil

Problem Nedir?

Karşılaşılabilecek soruna veya çözülmesi gereken duruma problem denir.

Örneğin: İlk kez gideceğiniz yerde yolu kaybetmeniz, telefonunuzun kitlenmesi, online alışveriş sitesinden aldığınız ürünün yanlış gelmesi gibi durumlar aslında günlük hayatta karşılaştığımız problemlerdir. Bu problemleri aşmak için farklı çözüm yolları deneriz.

Bir problemi çözmesi beklenen alternatif yollar arasından en doğru olanı seçebilmeye problem çözme denir.

Problemi ortaya koyma aşamasından, problemin çözümünün tamamlanmasına kadar geçen zaman ise problem çözme süreci olarak adlandırılır.

Problemleri çözmek için genellikle iki farklı yöntem kullanılır:

1. Deneme yanılma ya da tahminde bulunma yoluyla çözme

2. Algoritma geliştirme yoluyla çözme

Problem Çözme Süreci

1. Problemi tanımlama:

Problemin ne olduğu belirgin bir şekilde ortaya konulmalıdır.

2. Problemi anlama:

Problemin kaynağının ne olduğu belirlenmelidir. Bir problem ne kadar iyi anlaşılırsa çözümü o kadar kolay olacaktır.

3. Alternatif çözüm yollarını belirleme:

Problemi çözmesi beklenen tüm alternatifler sıralanmalıdır.

4. En uygun çözümü seçme:

Bir önceki adımda belirlenen alternatifler arasından en uygun olanının seçilmesi gerekir.

5. Çözümü uygulama:

Bir önceki adımda belirlenen çözüm yöntemi kullanılarak problemi çözme işi gerçekleştirilir.

6. Çözümü test etme:

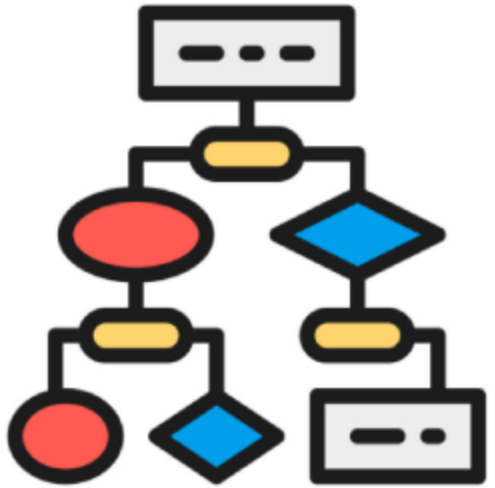
Uygulanan çözümün beklentileri yerine getirip getirmediği test edilmelidir.

Algoritma Kavramı

- Algoritma kelimesi bir İslam Bilgini olan **El-Harezmi'nin** isminin Latince karşılığından gelmektedir. El-Harezmi matematik, gök bilim ve coğrafya alanlarında çalışmış, cebirin temelini oluşturmuş, bugünkü bilgisayar bilimi ve elektroniğin temeli olan 2'lik (binary) sayı sistemini ve 0'ı (sıfır) bulmuş önemli bir bilim insanıdır.

- Programlamanın öğrenilebilmesi için öncelikle algoritmanın ne olduğuna ve nasıl geliştirilmesi gerektiğine cevap bulunmalıdır.

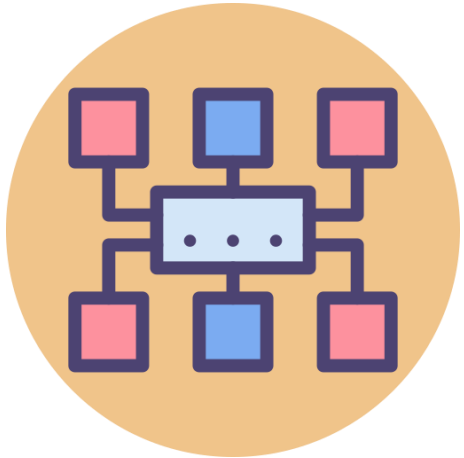
- Problem çözme yöntemlerinden biri olan algoritma geliştirmek kodlamaya atılan ilk adımdır. Algoritma mantığı iyice kavrandıktan sonra bu mantık ile birlikte bir programlama dili kullanılarak yazılım geliştirme süreci başlar.



Algoritma Kavramı

- **Algoritma,** belirli bir mantığı olan, farklı düşünebilmeyi ve problem çözmeyi öğretmek için tasarlanan bir yoldur. Başka bir ifadeyle bir problemi çözmeye giden yolun basit, net ve belirli bir sıraya göre tasarlanmış hâlidir.

1. Açık ve net olmalıdır.
2. Kullanılacak olan girdiler iyi tanımlanmış olmalıdır
3. Çıktılar açık ve anlaşılır olmalıdır.
4. Algoritmalar hızlı olmalıdır.
5. Sonlu ve uygulanabilir olmalıdır.



- Algoritmaların program haline getirilmesi için programlama dilleri kullanılır.
- Programlama dilleri kullanılarak yazılımlar geliştirilir.

Algoritma Örnekleri

- Günlük yaşamda karşılaştığımız problemleri bilerek veya farkında olmadan adım adım çözmeye çalışırız.
- Örneğin yazı yazarken kaleminizin ucu kırıldığında şu adımları takip ederek bu sorunu çözersiniz.

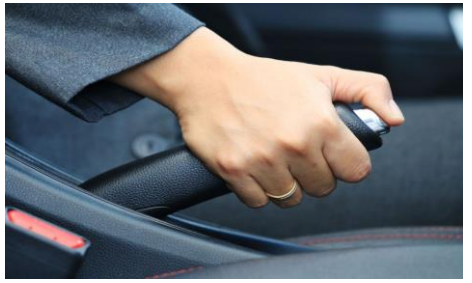
1. Kalemtıraşı çıkar.
2. Kalemi al.
3. Çöp kovasının yanına git.
4. Kalemin ucunu aç.
5. Sırana geri dön.
6. Yazmaya devam et.

Çay Demleme Algoritması

Adım 1: Başla.
Adım 2: Mutfağa git.
Adım 3: Çaydanlığa su koy.
Adım 4: Çaydanlığı ocağa koy.
Adım 5: Ocağı yak.
Adım 6: Su kaynadı mı, kontrol et.
Adım 7: Cevap evet ise demliğe çay koy ve demle. Cevap hayır ise adım 6‘ya git.
Adım 8: 10 dakika bekle.
Adım 9: Çayı doldur.
Adım 10: Bitir.

Arabayı Çalıştırıp Yola Çıkma

Adım 1: Başla
Adım 2: Sürücü koltuğuna geç.
Adım 3: Emniyet kemerini tak.
Adım 4: Aynaları kontrol et.
Adım 5: Anahtarı tak.
Adım 6: Kontağı çevir.
Adım 7: El frenini indir.
Adım 8: Vitese geç.
Adım 9: Gaza bas.
Adım 10: Bitir.

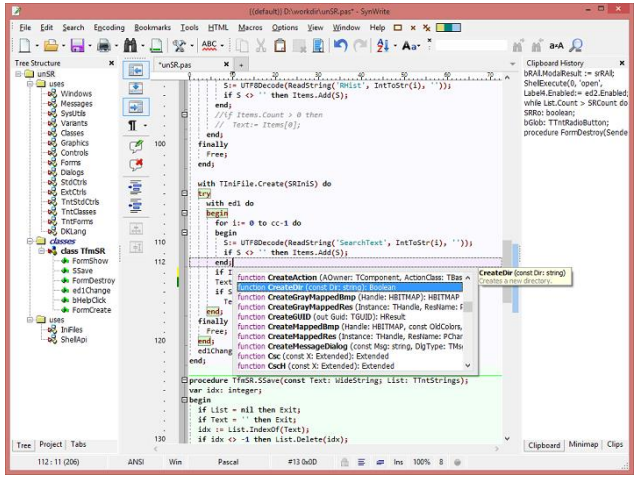


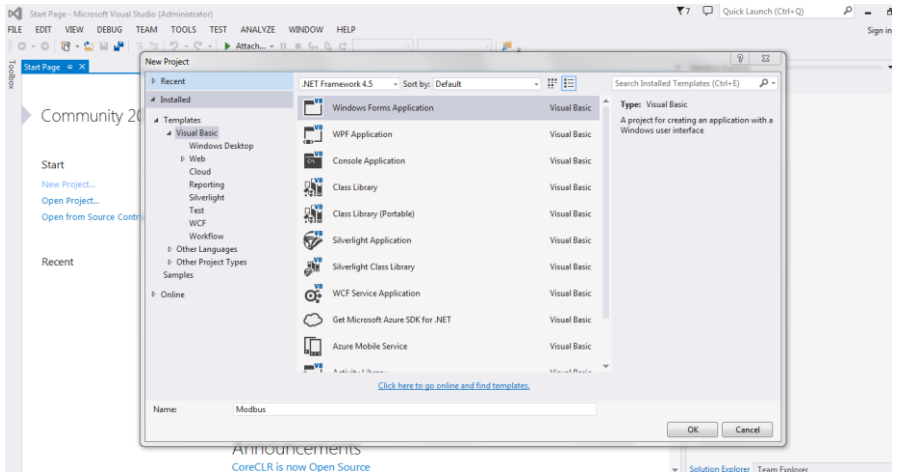
Programlamaya İlişkin Kavramlar

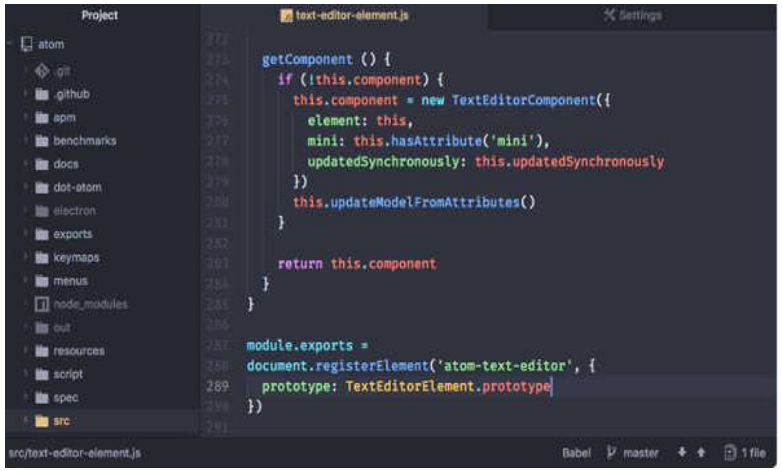
1. **Kaynak Kod:** Herhangi bir yazılımın, makine diline dönüştürülerek işlenip yorumlanmasından önce insanların okuyabildikleri ve üzerinde çalışabildikleri kodlar olarak tanımlanabilmektedir.

```
#include<stdio.h>
main()
{
      printf("HelloWorld")
}
```

2. **Kod Editörü:** Yazılım geliştirmek isteyen kullanıcıların bu yazılımları ortaya çıkarmak için kod yazarak üzerinde çalıştığı platformdur. Her yazılımcının **kod yazmak** için bir düzenleyiciye (kod editörüne) ihtiyacı vardır. Bazı kod editörleri yalnızca bir veya iki dili desteklemektedir. Bazı kod editörleri ise çoklu dilleri ve platformları da desteklemektedir.







Programlamaya İlişkin Kavramlar

3. Amaç Program: Kaynak programın derlenmesi ile elde edilen makine diline dönüştürülmüş programa amaç program (object program) denir. Programa dili ile yazılan kaynak programları makine dili ile yazılan amaç programlara çevirmek için derleyiciler kullanılır.

4. Derleyici (Compiler): Geliştiricilerin herhangi bir programlama dilini kullanarak yazdığı kaynak kodu bilgisayarın anlayabileceği makine diline yani 0 ve 1'lere çeviren aracı yazılımdır. Derleyici sayesinde geliştiriciler farklı programlama dillerini kullanarak aynı işlevi yerine getiren yazılımlar üretebilirler.

5. Yorumlayıcı (interpreter): Yüksek seviyeli programlama dili ile yazılmış bir programı adım adım makine diline çeviren ve makine dilindeki talimatları çalıştıran programdır.

Derleyici ve Yorumlayıcı Farkı

Derleyiciler, yorumlayıcılara göre daha hızlıdır. Çünkü yorumlayıcılar ilk kod satırından son kod satırına kadar her satırını teker teker yorumlar ve kodun karşılığındaki işlemi gerçekleştirir. Derleyiciler ise kodların tamamını bilgisayar diline çevirir. Eğer hata varsa, tüm hataları programcıya bildirir.

Algoritma Gösterim Şekilleri

1. Düz yazı ile gösterim,
2. Sözde kod(pseudo code) ile gösterim,
3. Akış şeması ile gösterim.

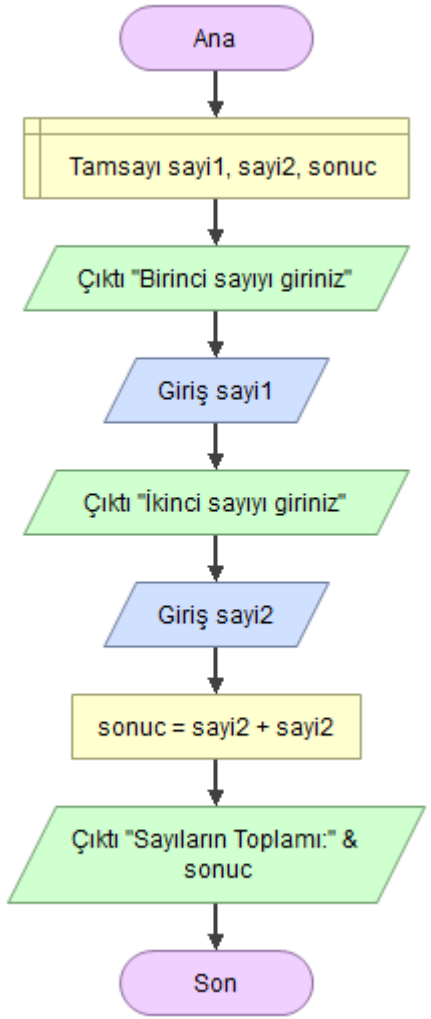
Düz Yazı İle Gösterim

Adım1 : Başla
Adım2 : Birinci Sayıyı Oku
Adım3 : İkinci Sayıyı Oku
Adım4 : Birinci Sayı ve İkinci Sayıyı Topla
Adım5 : Toplamı Ekrana Yaz
Adım6 : Bitir

Pseudo Code(Sözde Kod) ile Gösterim

A1 : Başla
A2 : sayi1 oku
A3 : sayi2 oku
A4 : sonuc = sayi1 + sayi2
A5 : Yaz sonuc
A6 : Bitir

Akış Şeması İle Gösterim



Kullanılan Operatörler – 1. Matematiksel

Matematiksel işlemleri yapmak için kullanılan operatörler.

Operatör	İşlem
Toplama +	3+2
Çıkarma –	9-7
Çarpma *	6*3
Bölme /	9/3
Mod %	10%3 Kalanı Bulma, İşlem Sonucu 1'dir
Üs Alma ^	4^3

Matematiksel Operatörlerde İşlem Önceliği

Matematiksel işlemleri algoritmada her zaman yazıldığı gibi kullanamayız. Bilgisayar mantığına göre matematiksel ifadelerin tek satır halinde yazılmaları gerekir. Algoritmada işlem öncelik sırası kuralları aşağıda verilmiştir. Parantez kullanılarak işlem öncelik sıraları değiştirilir. İç içe kullanılan parantezlerde öncelik en içtekindedir. Aynı işlem önceliğine sahip elemanlarda işlem soldan sağa doğrudur.

İşlem öncelik sırası kuralları			
Sıra	Tanım	Matematik	**Bilgisayar**
1	Parantezler	(( ))	**(( ))**
2	Üs Almak	a^n	**a^n**
3	Çarpma ve Bölme	ab a/b	**a*b a/b**
4	**Toplama ve Çıkarma**	**a+b a-b**	**a+b a-b**

$$\frac{CD}{AD}+B+\frac{CD}{A} \longrightarrow \text{(C*D/(A*D))+B+C*D/A}$$

$$C+\frac{BA}{A}-B^{C}(B-C)^{B} \longrightarrow \text{C+B*A/A-B^C*(B-C)^B}$$

Kullanılan Operatörler – 2. Karşılaştırma

* Program içinde ifadelerin karşılaştırılması için kullanılır. Karşılaştırma ifadelerinin sonucu ya **TRUE(1) - DOĞRU** yada **FALSE(0) - YANLIŞ** olur. Üçüncü bir ihtimal yoktur.

Operatör	İşlem
Eşit Mi ==	3==3 işlem Sonucu **True,** 3==4 İşlem Sonucu **False**
Eşit Değil mi != <>	4!=5 **True,** 5!=5 **False**
Büyük mü >	6 > 4 **True,** 6 < 4 **False**
Büyük veya Eşit mi >=	5 >=3 **True**, 5 >=5 **True**, 5>=6 **False**
Küçük mü <	6 < 8 **True** 6 < 4 **False**
Küçük veya Eşit mi <=	5<=6 **True**, 5<=5 **True**, 5<=4 **False**

Kullanılan Operatörler – 3. Mantıksal

Birden fazla karşılaştırma ifadesini birleştirmek için kullanılır.

Operatör	İşlem
Ve &&	Birden fazla karşılaştırma ifadesinin hepsinin doğru olması durumunda True Sonucunu verir. Şartlardan bir tanesinin yanlış olması durumunda False sonucunu verir
Veya ||	Birden fazla koşuldan bir tanesinin doğru olması durumunda True sonucunu verir. Şartların tümünün yanlış olması durumunda False sonucunu verir.
Değil !	True ifadesini False, False ifadesini True yapar. İşlemi tersine çevirir

Ve(&&)			Veya(||)			Değil(!)	
a	**b**	**a and b**	**a**	**b**	**a or b**		
1	1	1	1	1	1	1	0
0	1	0	0	1	1	0	1
1	0	0	1	0	1		
0	0	0	0	0	0		

Örnek: Facebook hesabınıza giriş yaparken kullanıcı adınızın ve şifrenizin doğru olması denetlenir.

(kullanıcıAdi = Ahmet && şifre = 123)

Eğer girilen bilgilerden birisi yanlış olursa sistem False sonucunu döndürür ve sizi hatalı giriş sayfasına yönlendirir

Değişkenler

Kullanıcı tarafından girilen veya program tarafından üretilen verilerin bilgisayarda tutulmasını sağlayan yapılardır. Saklayacağı veri türüne göre değişken tipleri bulunmaktadır.

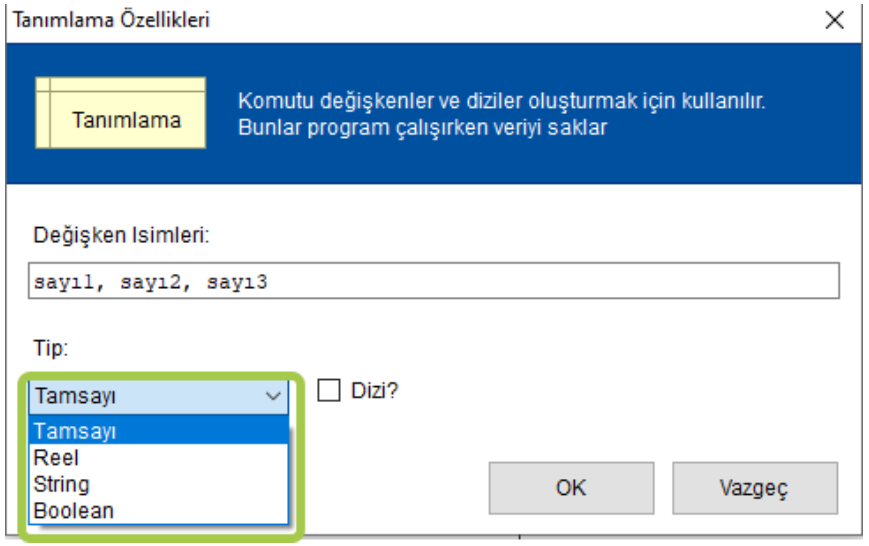
Program içerisinde verilerimiz RAM Bellek üzerinde tutulmaktadır. Ve tutulan bilginin türüne göre farklı tiplerde değişkenler bulunmaktadır. Program içerisinde bir değişken oluşturduğumuz zaman RAM Bellek üzerinde değişken adı ile bir alan açılmaktadır. Ram Bellek, bankalardaki kiralık kasalara benzetebiliriz. Değerli eşyalarımızı orada saklayabilir ve ihtiyacımız olduğu zaman alıp kullanabiliriz.

ad = "Bilişim Dünyası" sayi = 10 derece = 38.6 islem = TRUE

Değişken Tipleri

Programlama dillerinde saklanacak veriler tiplerine göre değişiklik göstermektedir. Algoritma eğitiminde temel olarak 4 farklı başlık altında inceleyeceğiz. Fakat programlama dili eğitimine geçtiğimizde bu değişken tiplerindeki farklılıkları inceleyeceğiz.

Değişken Tipi	Açıklama	Örnek
Tamsayı	Tamsayı türündeki verileri saklamak için kullanılır.	sayi = 10
Reel	Ondalıklı sayıları saklamak için kullanılır.	vergi = 235.26
String	Metinsel ifadeleri saklamak için kullanılır.	ad = "Ahmet"
Boolean	Sadece TRUE yada FALSE değerlerini saklamak için kullanılır.	islem = TRUE



NOT: sayi = 10 ifdesindeki = iki ifadenin eşit olduğunu göstermez. Sağ taraftaki değerin sol taraftaki değişken ismine yazıldığını gösterir. Ram bellek üzerinde sayi adında bir alan oluşturulur ve bu alana 10 sayısının yazıldığını gösterir.

Değişken Tanımlama Kuralları

1-Değişken ismi içerisinde boşluk kullanılmaz.
Doğru — Adi_Soyadi Yanlış — Adi Soyadi

2-Değişken isimleri rakam veya özel karakterler ile başlayamaz, harf ile başlamalıdır.
Doğru — Sozlu1 Yanlış — 1.Sozlu

3-Programlama dilinin kullandığı komut satırları değişken ismi olarak kullanılamaz.
İnt, void, string, try

4-Değişken isminin 255 karakterden fazla olmaması gerekir.

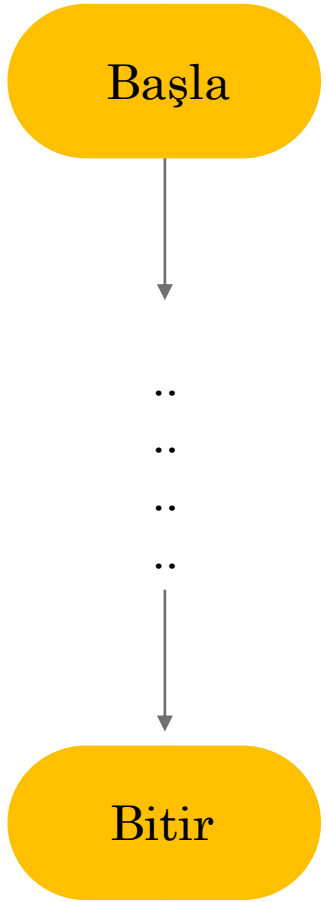
5-Değişken ismi içerisinde harf, rakam ve alt çizgi dışındaki karakterler kullanılmamalıdır.
Doğru — Adi_Soyadi Yanlış — Adi+Soyadi

6- Birden fazla kelimeden oluşan değişken isimlendirmede kabul görmüş kural «camel case(deve hörgücü)». İlk kelime küçük harfle başlar diğer kelimelerin ilk harfi büyük yazılır.
adSoyadi, okulNo, musteriNo, ogrenciAdSoyad

Akış Şemalarını Oluşturmak

1. Başlama - Bitirme

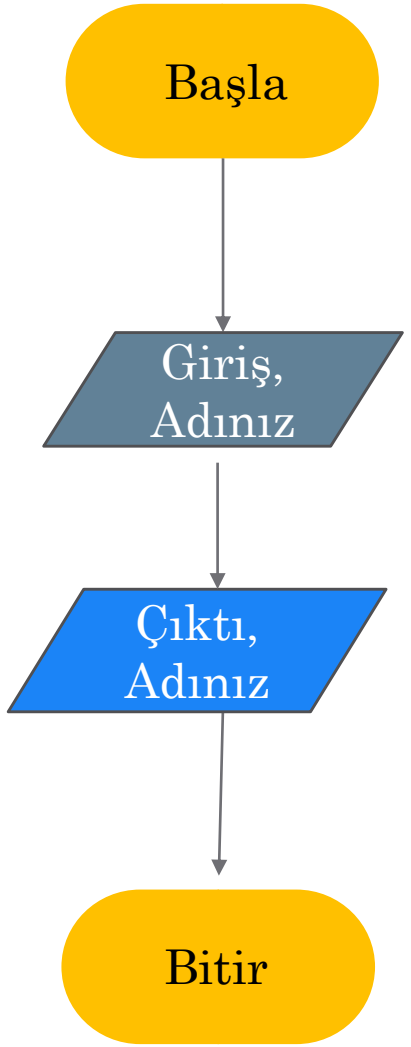
Algoritmaların bir başlangıcı ve sonu olduğunu söylemiştik.
Başlangıç ve bitişi göstermek için kullanılır.



Akış Şemalarını Oluşturmak

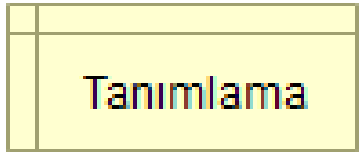
2. Bilgi Girişi ve Çıkışı

Kullanıcıdan bilgi almak ve kullanıcıya bilgi göstermek için kullanılır. Yandaki akış diyagramında kullanıcıdan ad bilgisini alarak, ekranda ad bilgisini göstermektedir.

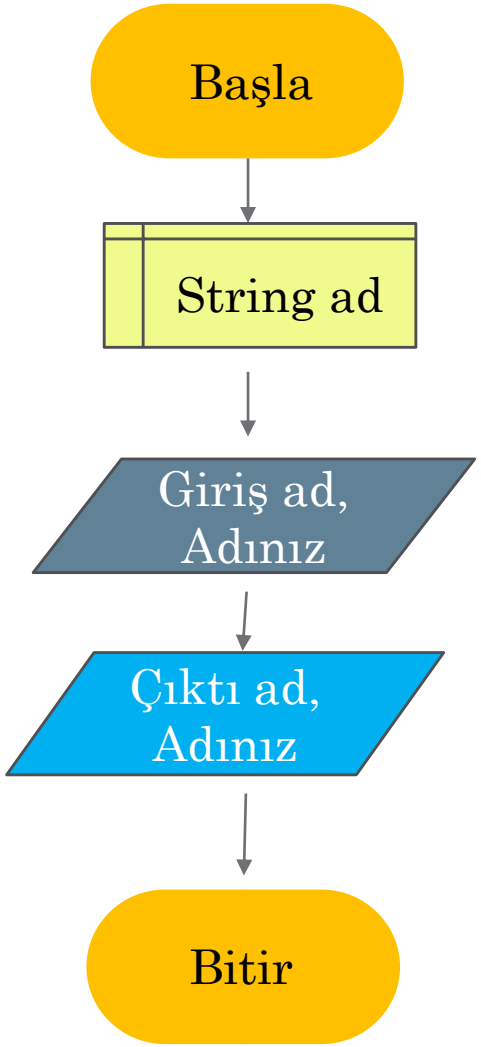


Akış Şemalarını Oluşturmak

3. Değişken Tanımlama



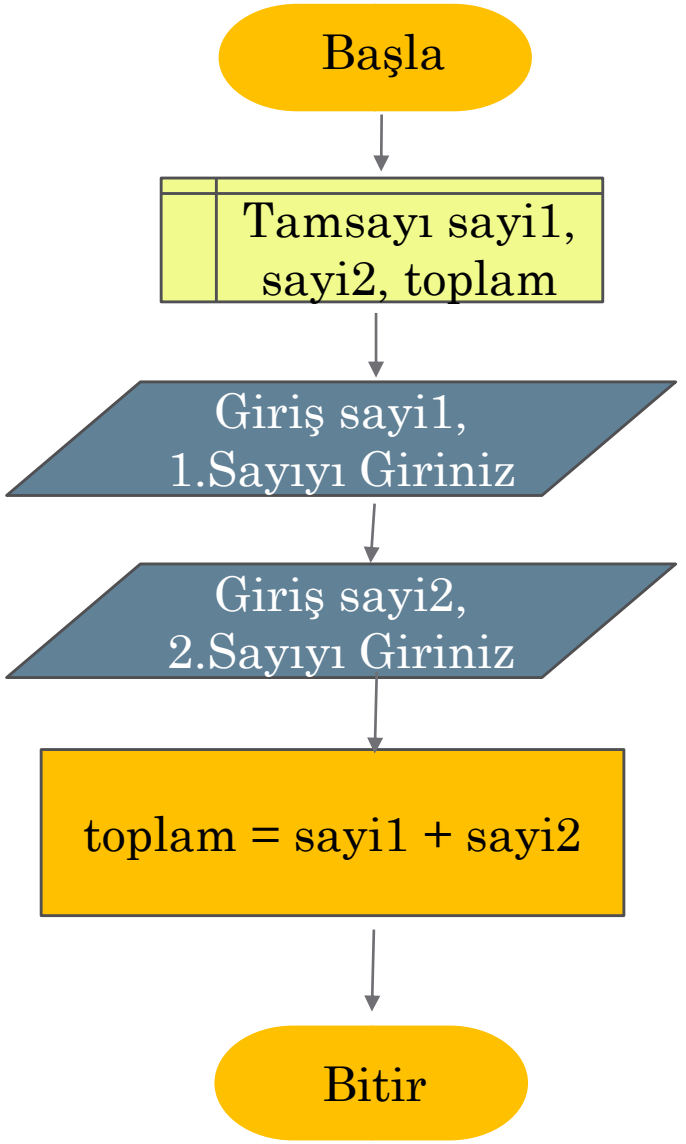
Kullanıcı tarafından girilen bilgilerin veya program tarafından üretilen bilgilerin saklanması için değişken yapıları kullanılır. Değişken tanımlamak için yukarıdaki şekil kullanılır. Kullanıcıdan alınan ad bilgisini programda tutabilmek için ad adında bir değişken tanımlanmıştır.



Akış Şemalarını Oluşturmak

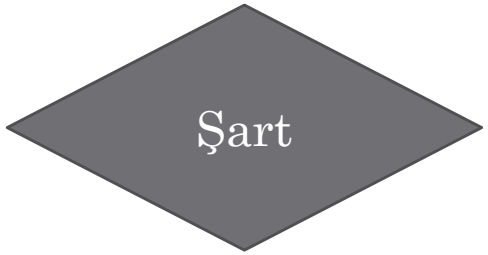
4. Atama Veya İşlem Yapma

Program tarafından matematiksel işlem yapılmasını sağlar. Örnekte girilen iki sayının toplama işleminin yapılmasını sağlar.

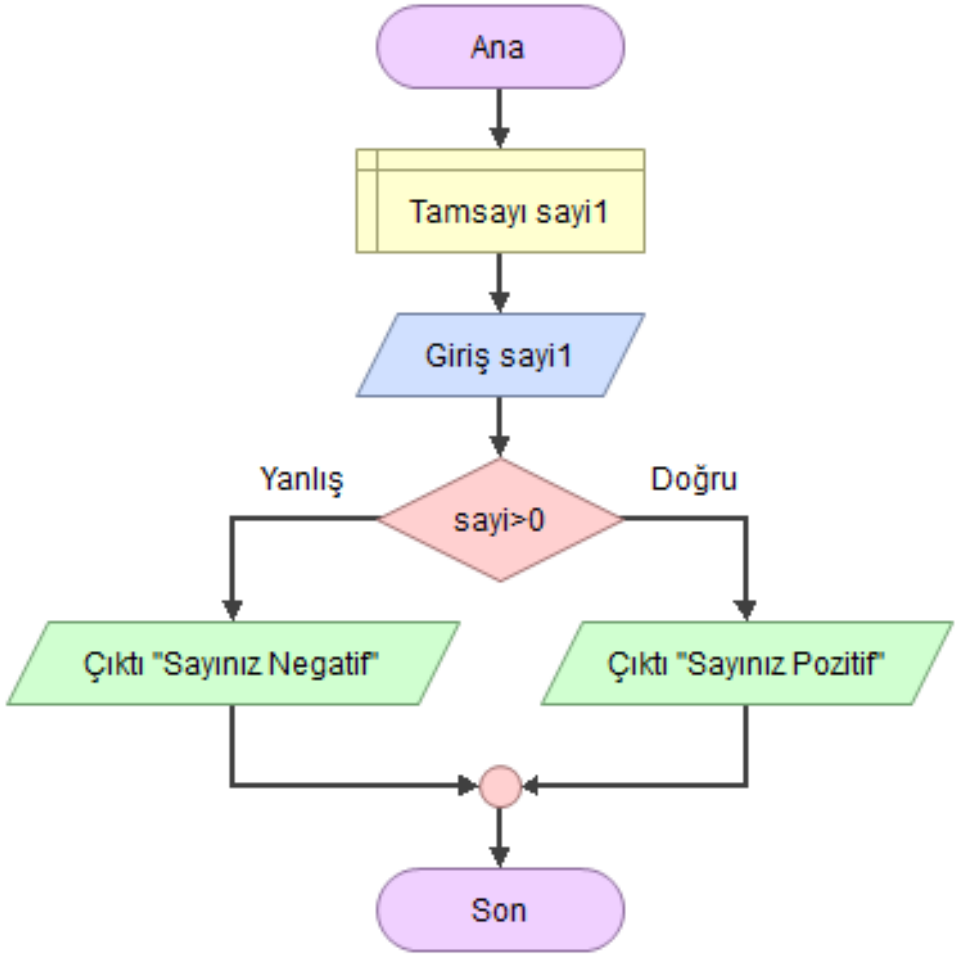


Akış Şemalarını Oluşturmak

4. Atama Veya İşlem Yapma



Programda belli koşullara göre işlem yapılmasını sağlar. Karşılaştırma ifadeleri kullanılarak programın dallanmasını sağlar.
Örnekte: Girilen sayının 0 dan büyük olması durumunda(sayi>0 şartı) ekrana «Sayınız Pozitif» 0 dan küçük olması durumunda «Sayınız Negatif» mesajını verecektir.

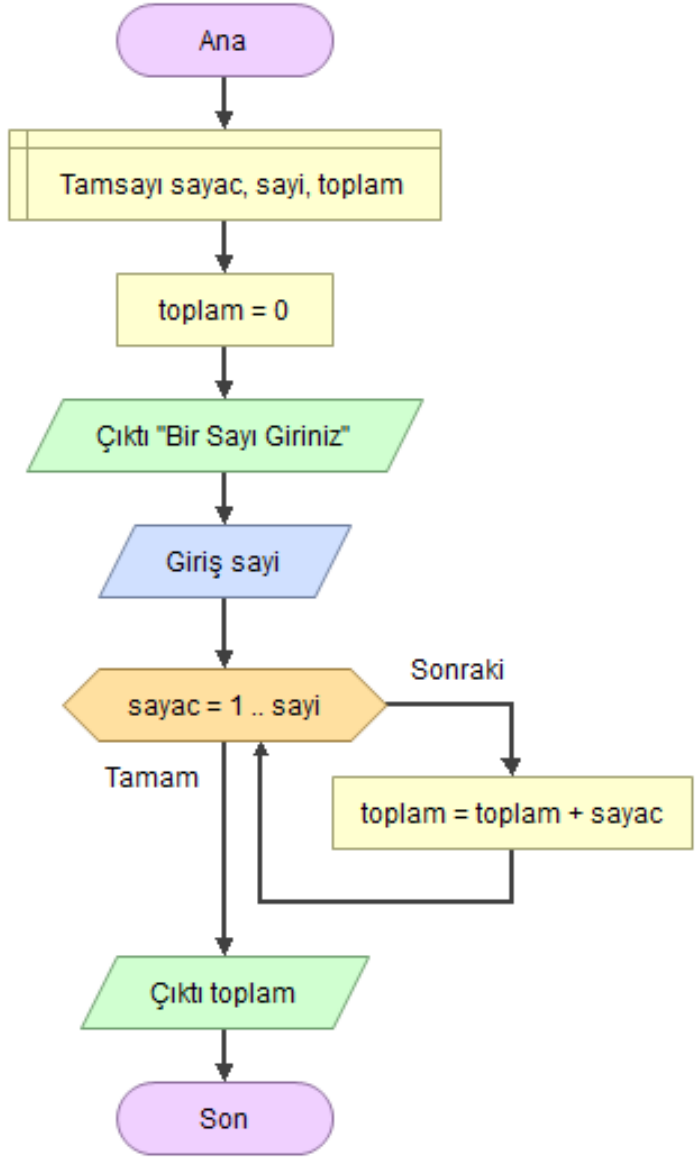


Akış Şemalarını Oluşturmak

5. Tekrarlayan İşlemler - Döngüler

Döngü

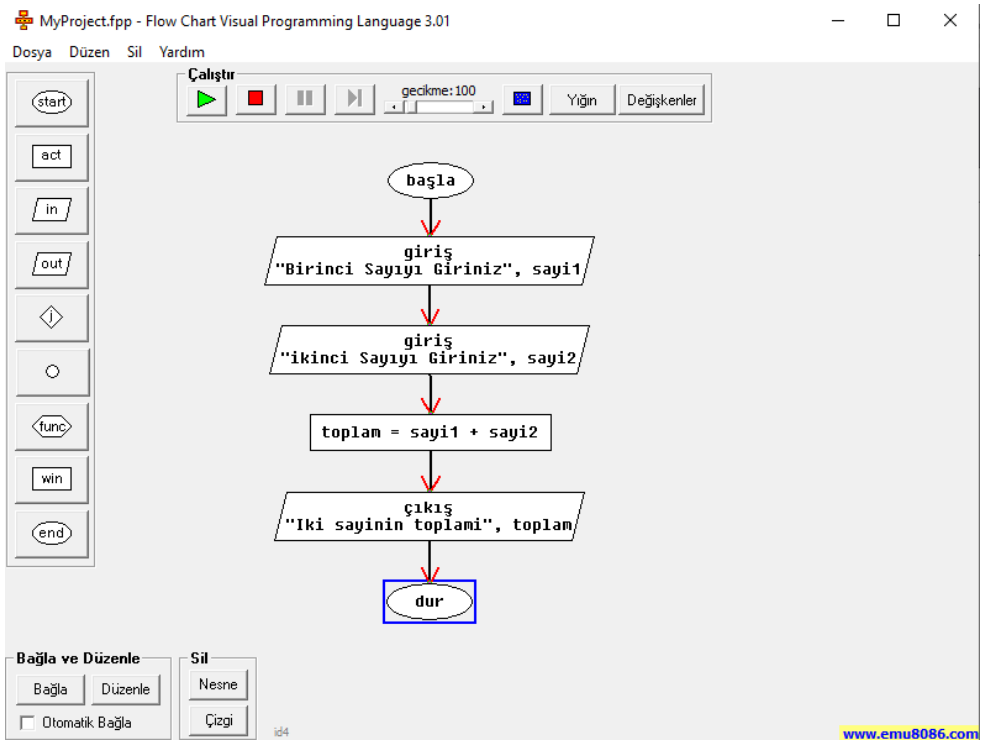
Programda tekrarlanması istenen işlemler için döngü oluşturur.
Örnekte, kullanıcıdan alınan sayıya kadar olan sayıları toplayan bir algoritma yazılmıştır. Sayac adındaki değişken ile 1 den başlayarak kullanıcının girdiği sayıya kadar bir döngü oluşturulur ve gelen her sayı toplam değişkeninde toplanır.



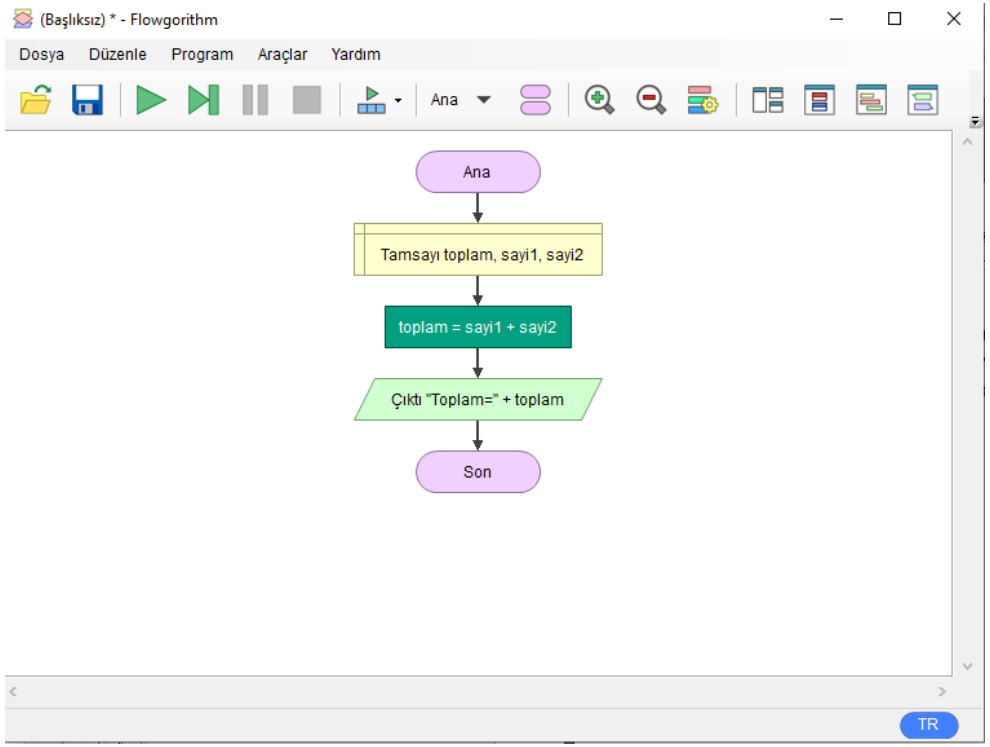
Akış Diyagramları Oluşturmak

Akış şemaları oluşturmak ve oluşturulan akış şemalarını test etmemizi sağlayan iki programdan bahsedeceğim. Normalde akış şemaları şekillerden ibaret olduğu için bu şekilleri oluşturacak birçok program mevcuttur. Fakat akış diyagramlarını oluşturan her program oluşturduğumuz akış diyagramlarını çalıştırmaz.

1. Flow Chart Visual Programing Language: Akış diyagramları oluşturup test etmemizi sağlayan ilk programlardan bir tanesidir.



2. Flowgorithm: Akış diyagramları oluşturup test etmemizi sağlayan güncel programdır. Biz derslerimizde bu programı kullanacağız.



Algoritma Örnekleri

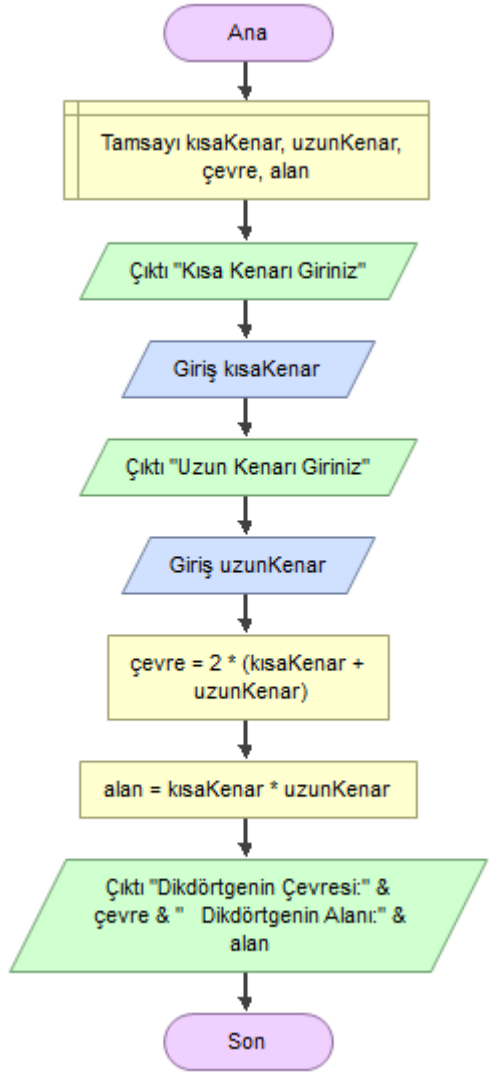
Soru : Kısa ve uzun kenarı girilen dikdörtgenin alanını ve çevresini hesaplayan algoritma ve akış şemasını tasarlayınız.

Düz Yazı İle

A1: Başla
A2: Değişkenleri tanımla kısaKenar, uzunKenar, çevre,alan
A3: Kısa Kenar Giriniz, kısaKenar
A4: Uzun Kenar Giriniz, uzunKenar
A5: Çevreyi Hesapla, 2*(kısaKenar + uzunKenar)
A6: Alan Heapla, kısaKenar * uzunKenar
A7: Ekranda Göster, çevre, alan
A8: Bitir

PSEUDO KOD(Sözde kod)

A1: Başla
A2: SAYISAL kısaKenar, uzunKenar, çevre, alan
A3: Oku kısaKenar
A4: Oku uzunKenar
A5: çevre = 2*(kısaKenar + uzunKenar)
A6: alan = kısaKenar * uzunKenar
A7: Yaz çevre, alan
A8: Bitir

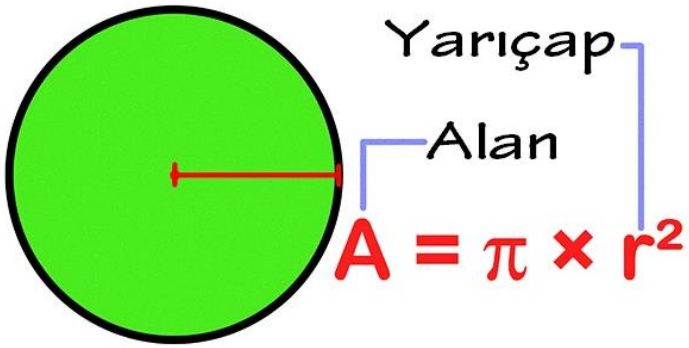


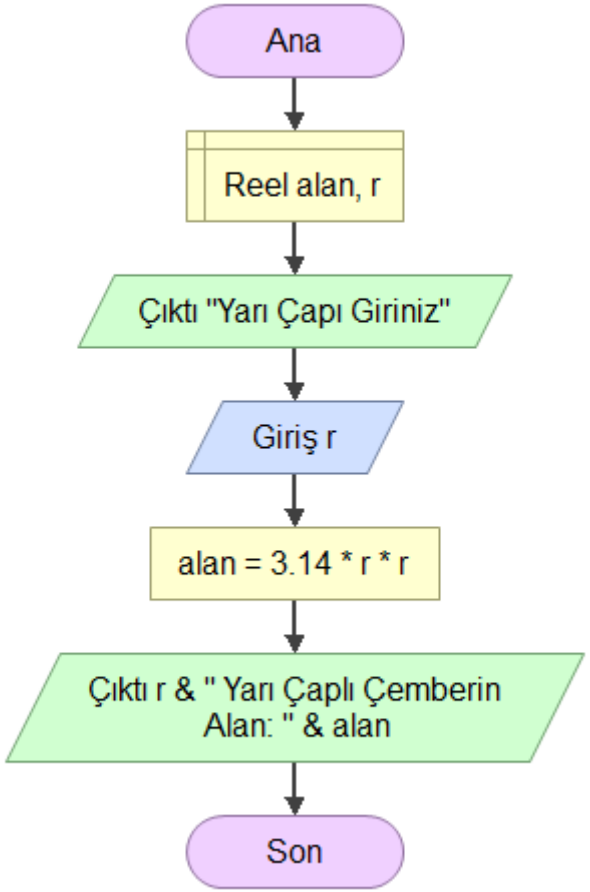
Algoritma Örnekleri

Soru : Yarı çapı verilen dairenin alanını hesaplayan algoritma ve akış şemasını tasarlayınız

Düz Yazı İle
A1: Başla
A2: Değişkenleri tanımla r, alan
A3: Yarı Çapı Girin, r
A4: Alanı Hesapla, 3.14 * r *r
A5: Ekranda Göster, alan
A6: Bitir

PSEUDO KOD(Sözde kod)
A1: Başla
A2: SAYISAL r, alan
A3: Oku r
A4: alan = 3.14 * r * r
A5: Yaz, alan
A6: Bitir





Algoritma Örnekleri

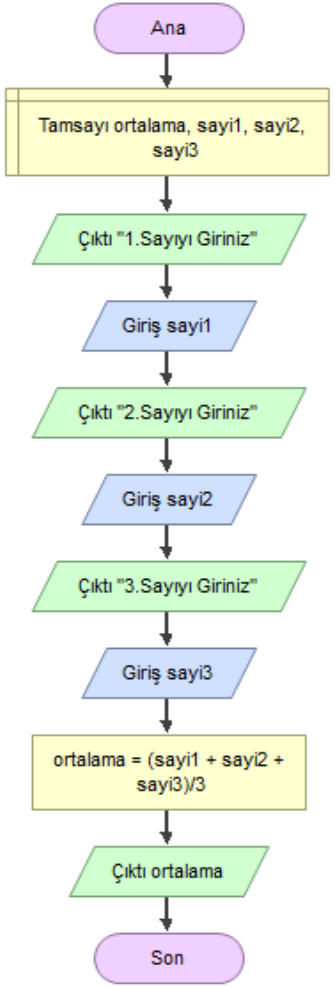
Soru : Girilen 3 sayının ortalamasını hesaplayan programın algoritmasını yazınız.

Düz Yazı ile Gösterim

A1: Başla
A2: Birinci Sayıyı Giriniz, Sayi1
A3: İkinci Sayıyı Giriniz, Sayi2
A4: Üçüncü Sayıyı Giriniz, Sayi3
A5: Üç Sayıyı Topla ve 3'e Böl, ortalama
A6: Ekranda Ortalamayı Göster
A7: Bitir

Pseudo Code ile Gösterim

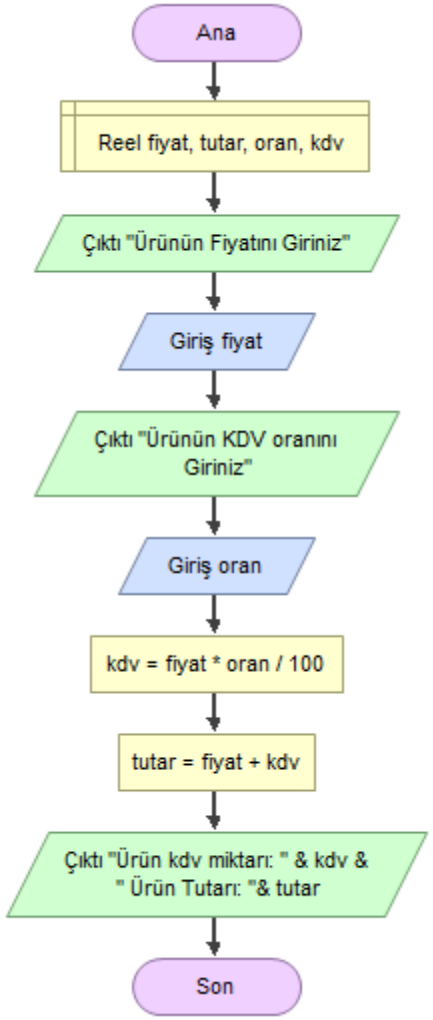
A1: Başla
A2: Tamsayı sayi1, sayi2, sayi3
A3: Reel ortalama
A4: Oku, Sayi1, Sayi2, Sayi3
A5: Ortalama = (Sayi1 + Sayi2 + Sayi3) /3
A6: Yaz Ortalama
A7: Bitir



Algoritma Örnekleri

Soru : Fiyatı ve kdv oranı girilen ürünün toplam fiyatını ve kdvsini hesaplayan algoritma ve akış şemasını tasarlayınız.

A1: Başla
A2: Sayısal fiyat, oran, kdv, tutar
A3: Yaz "Ürünün Fiyatını Giriniz"
A4: Oku fiyat
A5: Yaz " KDV oranını giriniz "
A6: Oku oran
A7: kdv = fiyat * oran/100
A8: tutar = fiyat + kdv
A9: Yaz tutar, kdv
A10: Bitir

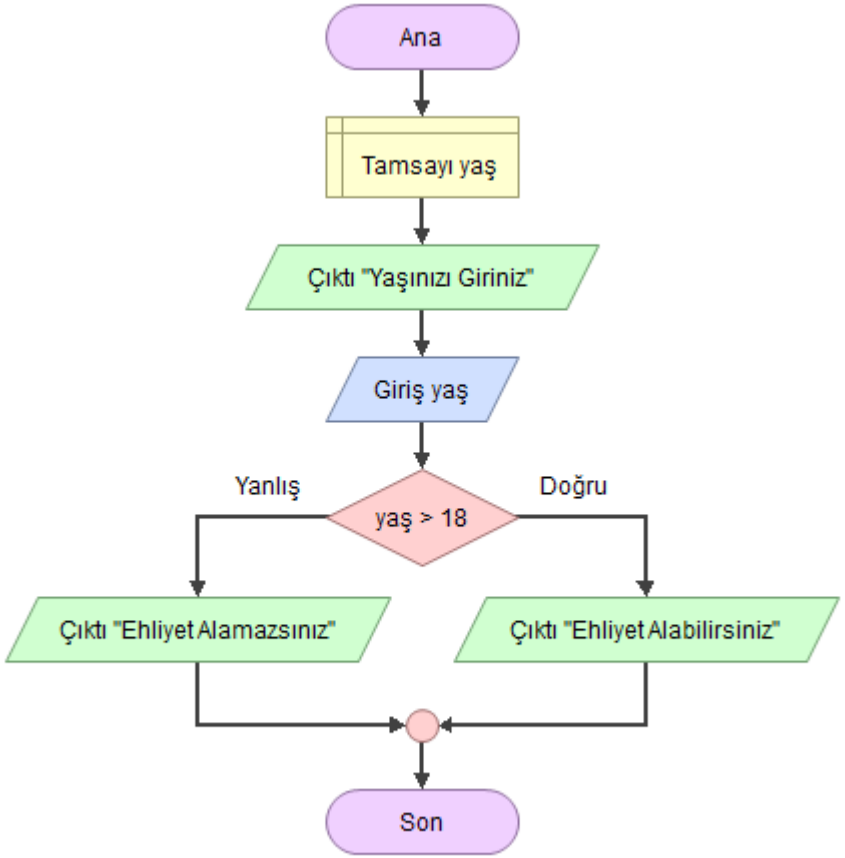


Algoritma Örnekleri - Koşullar

Soru : Yaşı girilen kişinin yaşı 18 den büyük ise ehliyet alabilirsiniz, değilse ehliyet alamazsınız mesajlarını yazdıran algoritma ve akış şemasını tasarlayınız.

Belli bir şart doğrultusunda programın akışını değiştiren yapılara koşullu ifadeler denir.

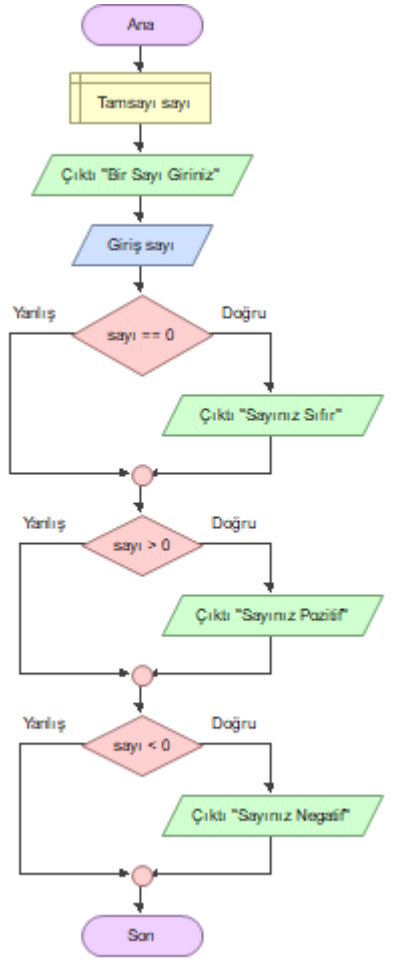
A1: Başla
A2: Sayısal yaş
A3: Yaz "Yaşınızı Giriniz"
A4: Oku yaş
A5: Eğer yaş > 18 ise Yaz " Ehliyet Alabilirsiniz",
Değilse Yaz "Ehliyet Alamazsınız"
A6: Bitir



Algoritma Örnekleri

Soru : Girilen sayı 0 dan büyük ise “pozitif” küçük ise “negatif” sıfıra eşit ise ”sıfır” mesajını verdiren algoritma ve akış şemasını tasarlayınız.

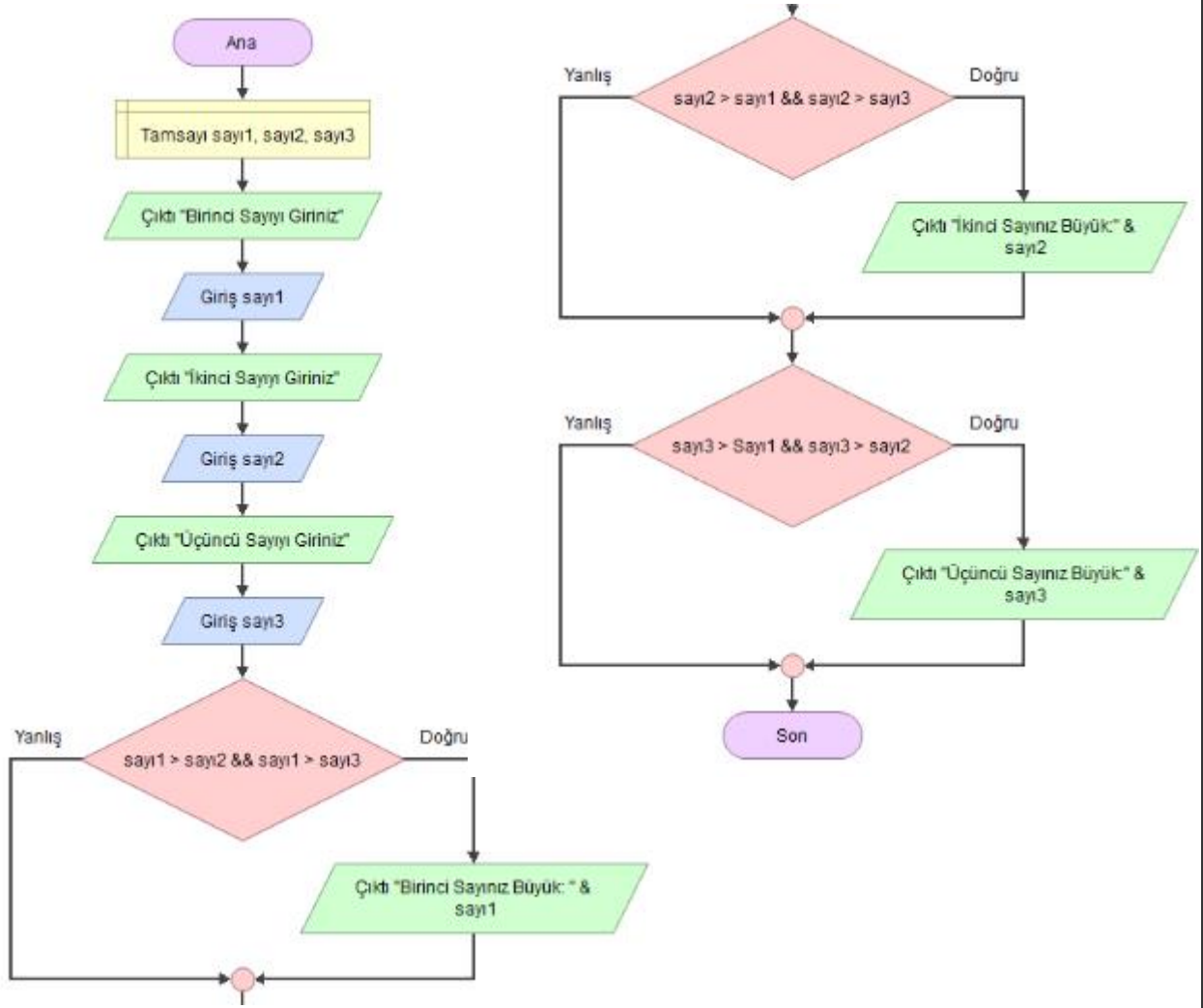
A1: Başla
A2: Sayısal sayı
A3: Yaz "Bir Sayı Giriniz"
A4: Oku sayı
A5: Eğer sayı == 0 ise Yaz " Sayınız Sıfır",
A6: Eğer sayı > 0 ise Yaz "Sayınız Pozitif "
A7: Eğer sayı < 0 ise Yaz "Sayınız Negatif"
A6: Bitir



Algoritma Örnekleri

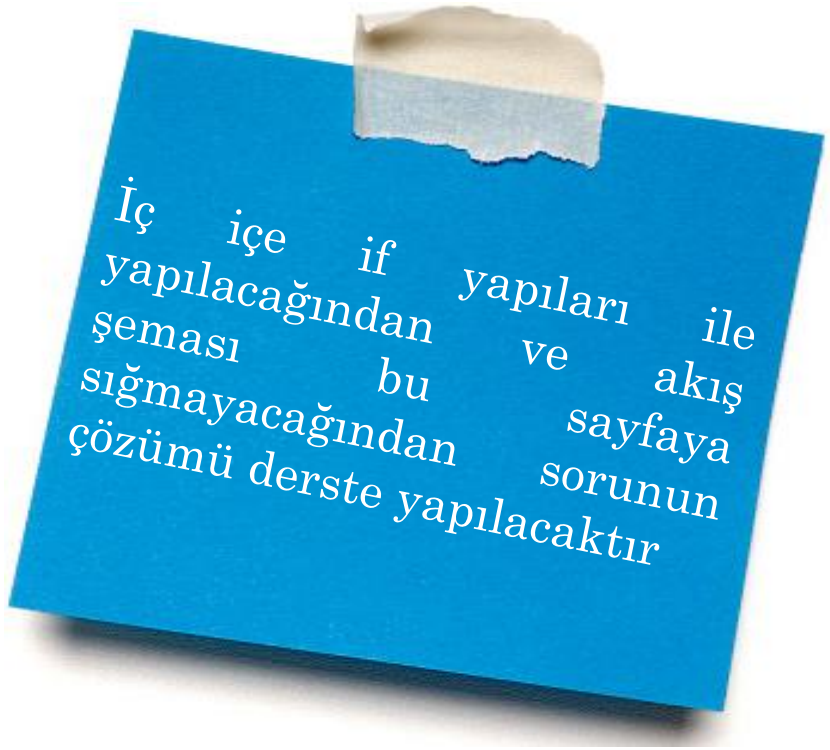
Soru : Kullanıcının girdiği 3 sayıdan büyük olanını yazdıran algoritma ve akış şemasını tasarlayınız.

A1: Başla
A2: Sayısal sayı1, sayı2, sayı3
A3: Yaz " Sayıları Giriniz"
A4: Oku sayı1, sayı2, sayı3
A5: Eğer sayı1 >sayı2 ve sayı1 > sayı3 ise
Yaz " En büyük Sayı1 ",
A6: Eğer sayı2 >sayı1 ve sayı2 > sayı3 ise
Yaz " En büyük Sayı2 ",
A7: Eğer sayı3 >sayı1 ve sayı3 > sayı2 ise
Yaz " En büyük Sayı3 ",
A8: Bitir



Algoritma Örnekleri

Soru : Bir firma işe alımlarda 40 yaş altı kişileri tercih etmektedir. Bu şartı sağlayan kişilerde de sürücü belgesi olan üniversite mezunlarını tercih etmektedir. Buna göre kullanıcıya önce yaşı sorulsun. Yaşı 40 üstü olanlara "**Üzgünüz, kriterlerimize uymuyorsunuz.**" uyarısı verilerek programdan çıkılırken; yaş şartı uyanlara diğer iki soruyu sorarak işe alınıp alınmadıklarını çıktı olarak veren kodu yazınız.



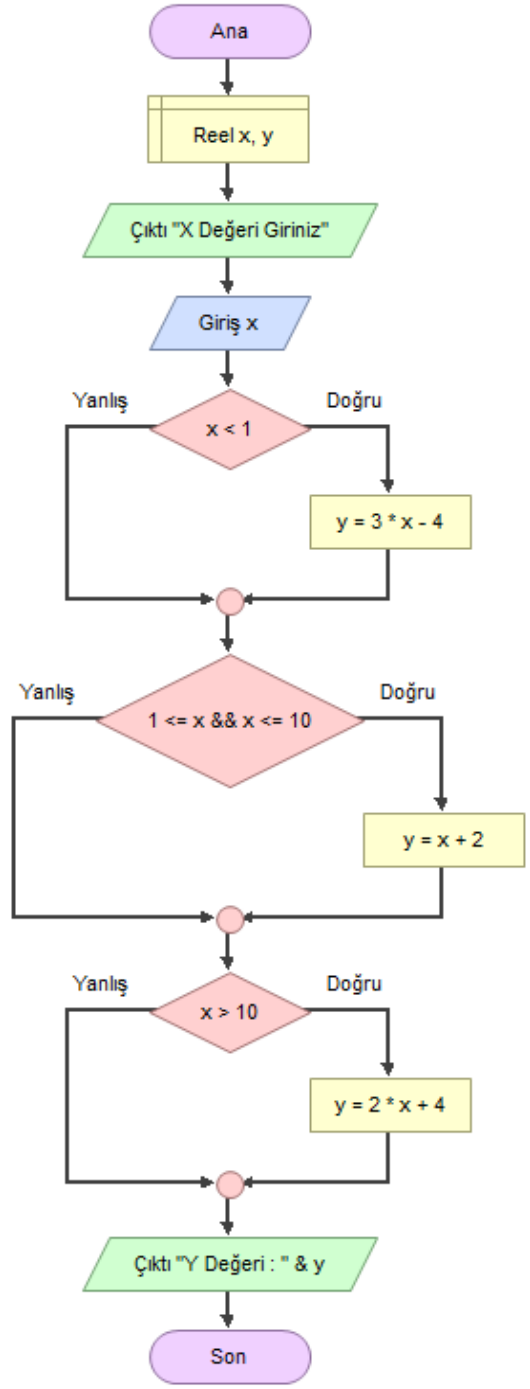
Algoritma Örnekleri

Soru : Aşağıda verilen parçalı fonksiyonda, klavyeden girilen x değeri için y'yi hesaplayıp ekrana yazdıran programın satır kodunu ve akış diyagramını geliştiriniz.

$$f(x) = \begin{cases} 3x - 4, & x < 1 \\ x + 2, & 1 \le x \le 10 \\ 2x + 4, & x > 10 \end{cases}$$

Bu fonksiyonun çözümünde klavyeden girdiğimiz x değerlerini kontrol etmemiz gerekir. Örneğin x değeri 1 den küçükse **3x - 4** formülünü kullanarak y değerini hesaplarız. Veya girdiğimiz x değeri 1 ile 10 arasında ise y değeri **x + 2** şeklinde hesaplanır.

A1: Başla
A2: Sayısal x, y
A3: Oku x,
A4: Eğer x < 1 ise y = 3 * x – 4
A5: Eğer (1 <= x && x <= 10) ise y = x + 2
A6: Eğer x > 10 ise y = 2 * x + 4
A7: Yaz y
A8: Bitir.

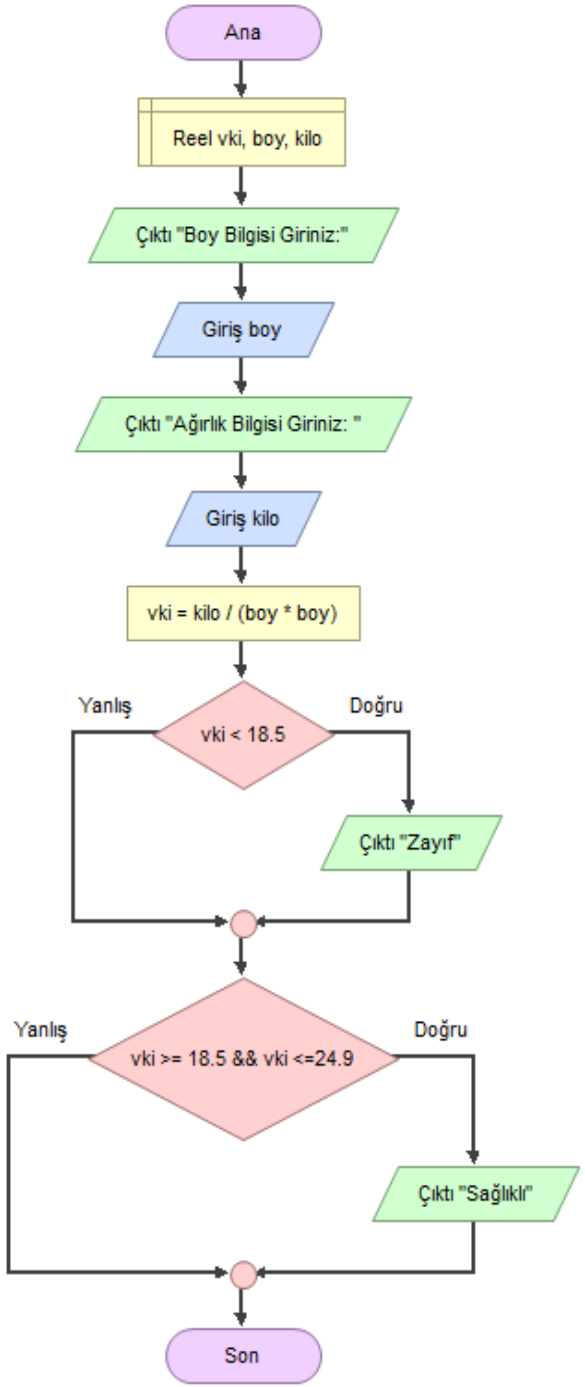


Algoritma Örnekleri

Soru : Obeziteyi hesaplamak için tüm dünyada **Vücut kitle indeksi** (VKİ) hesaplaması kullanılır. VKİ değeriniz kilogram olarak ağırlığınızın, metre cinsinden boy uzunluğunun karesine bölünmesiyle (kg/m²) elde edilir. Girilen boy ve ağırlık değerlerine göre VKİ değerini hesaplayınız ve sonucu aşağıdaki tablo değerlerine göre Kilo Kategorisini ekrana yazdırın.

Kilo Kategorisi	VKİ (kg/m²)
Zayıf	< 18.5
Sağlıklı	18.5-24.9
Fazla Kilolu	25-29.9
I. Derece Obezite	30-34.9
II. Derece Obezite	35-39.9
III. Derece Obezite	≥40

A1: Başla
A2: Sayısal vki, boy, kilo
A3: Oku boy, kilo
A4: vki = kilo / (boy * boy)
A5: Eğer vki < 18.5 Yaz Zayıf
A6: Eğer vki >=18.5 && vki <= 24.9 Yaz Sağlıklı
A7: Eğer vki >= 25 && vki <= 29.99 Yaz Fazla Kilolu
A8: Eğer vki >= 30 && vki <= 34.9 Yaz 1. Derece Obezite
A9: Eğer vki >= 35 && vki <= 39.9 Yaz 2. Derece Obezite
A10: Eğer vki >= 40 Yaz 3.Derece Obezite
A11: Bitir



Algoritma Örnekleri

Soru: $ax^2 + bx + c = 0$ Şeklindeki 2. dereceden bir denklemin köklerini bulmak için gerekli olan algoritmayı ve akış diyagramını hazırlayınız (Not: a, b ve c değerleri kullanıcı girecektir).

Tek bilinmeyenli ikinci derecede denklemin **diskriminant**ı şöyle tanımlanmaktadır.

$$\Delta = b^2 - 4ac$$

a) $\Delta > 0$ yani Δ(delta) pozitif ise, denklemin farklı iki gerçel kökü vardır. x_1 ve $x_2$olarak ifade edilen bu iki kök şu formül kullanılarak bulunur.

$$x_1 = \frac{-b + \sqrt{\Delta}}{2a} \quad \text{ve} \quad x_2 = \frac{-b - \sqrt{\Delta}}{2a}$$

b) $\Delta = 0$ yani Δ sıfıra eşit ise, denklemin, değerleri birbirleriyle çakışan, yani birbirine eşit, iki gerçel kökü vardır.

$$x_1 = x_2 = -\frac{b}{2a}$$

c) $\Delta < 0$ yani Δ negatif ise, denklemin gerçel kökü yoktur yani denklemin çözümü bulunamaz.

Algoritma Örnekleri

A1: Başla
A2: Sayısal a, b, c, diskriminant, x1, x2
A3: Oku a, b, c
A4: diskriminant = b * b – 4 * a *c

$$\Delta = b^2 - 4ac$$

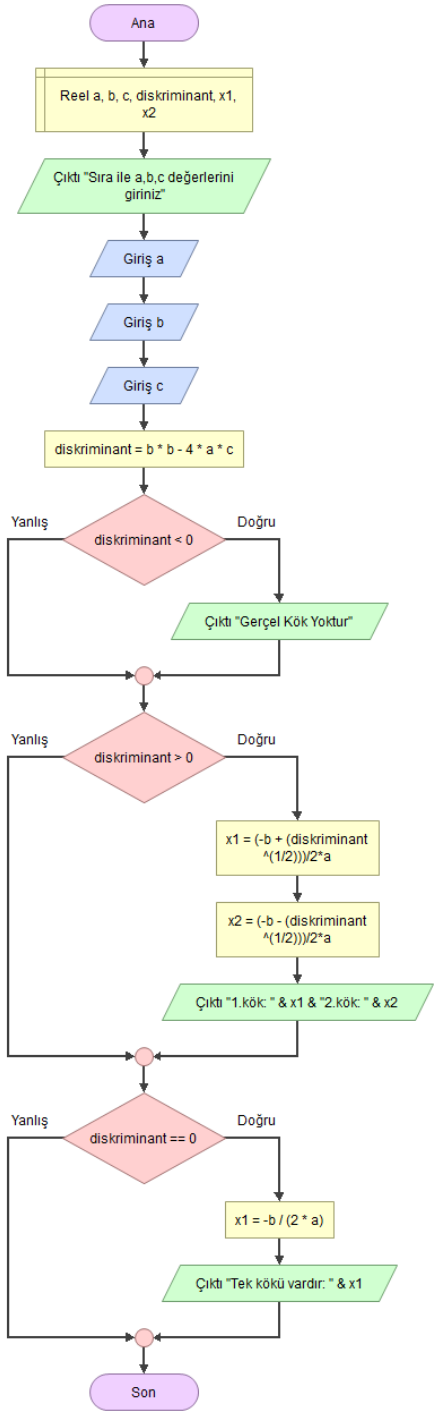
A5: Eğer diskriminant < 0 ise YAZ Gerçel Kökü Yoktur
A6: Eğer diskriminant >0 ise
    x1 = (-b + (diskriminant ^(1/2))) / (2 * a)
    x2 = (-b - (diskriminant ^(1/2)))/ (2 * a)
    Yaz x1, x2

$$x_1 = \frac{-b+\sqrt{\Delta}}{2a} \quad \text{ve} \quad x_2 = \frac{-b-\sqrt{\Delta}}{2a}$$

A7: Eğer diskriminant == 0 ise
    x1 = -b / (2 * a)
    YAZ x1

$$x_1 = x_2 = -\frac{b}{2a}$$

A8: Bitir

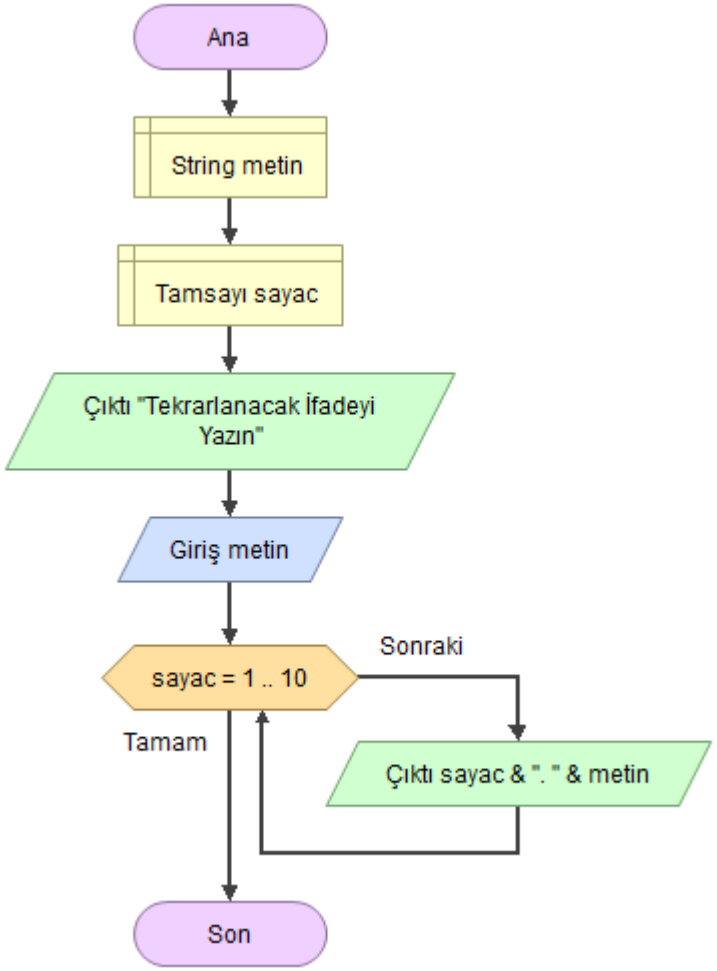


Algoritma Örnekleri

Soru : Girilen bir ifadeyi 10 defa ekranda yazmasını sağlayan algoritma ve akış diyagramını çiziniz.

Program içerisinde tekrarlanmasını istediğimiz işlemler olacaktır. Bu durumda döngüler kullanılır. Belli bir sayıda işlemlerin tekrarlanmasını sağlar.

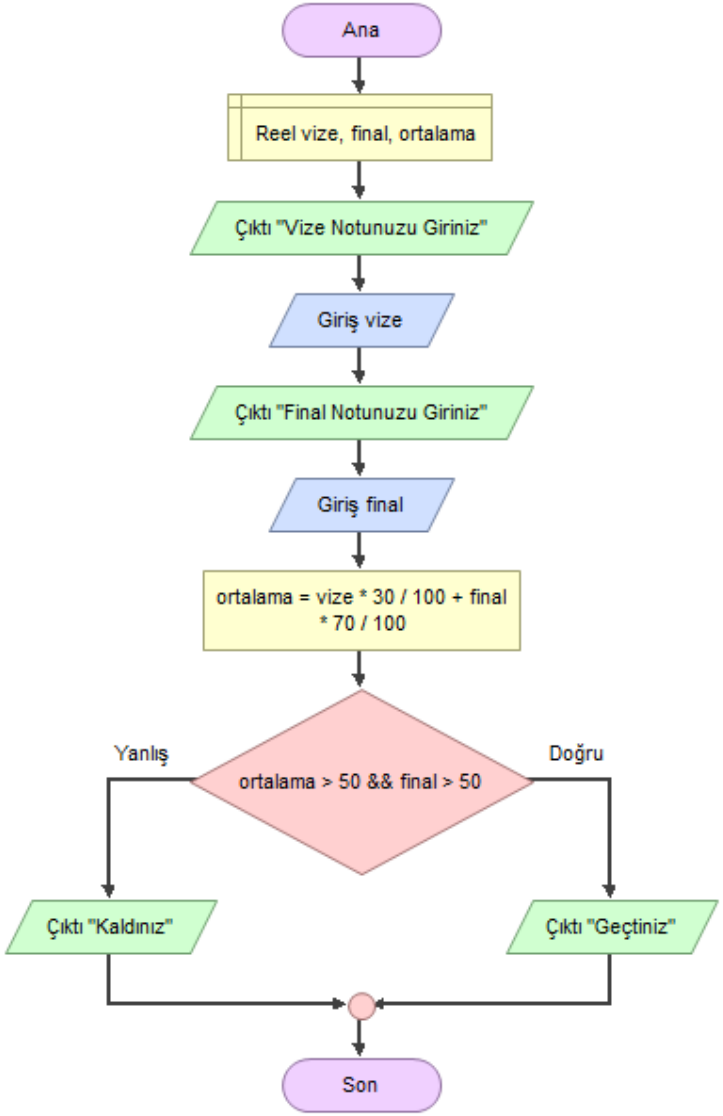
A1: Başla
A2: String metin, Sayısal sayac
A3: Oku, metin
A4: Döngü(sayac = 1 to 10 step 1)
A5: Yaz sayac, metin
A6: Döngü Sonu
A7: Bitir



Algoritma Örnekleri

Soru : Üniversite için vize final sınavları yapılmaktadır. Bir öğrencinin dersten geçme şartı vizenin %30 ile final notunun %70 in toplamı 50 ve üzeri ve final notunun 50 ve daha yüksek olmasıdır. Buna uygun algoritmayı oluşturunuz.

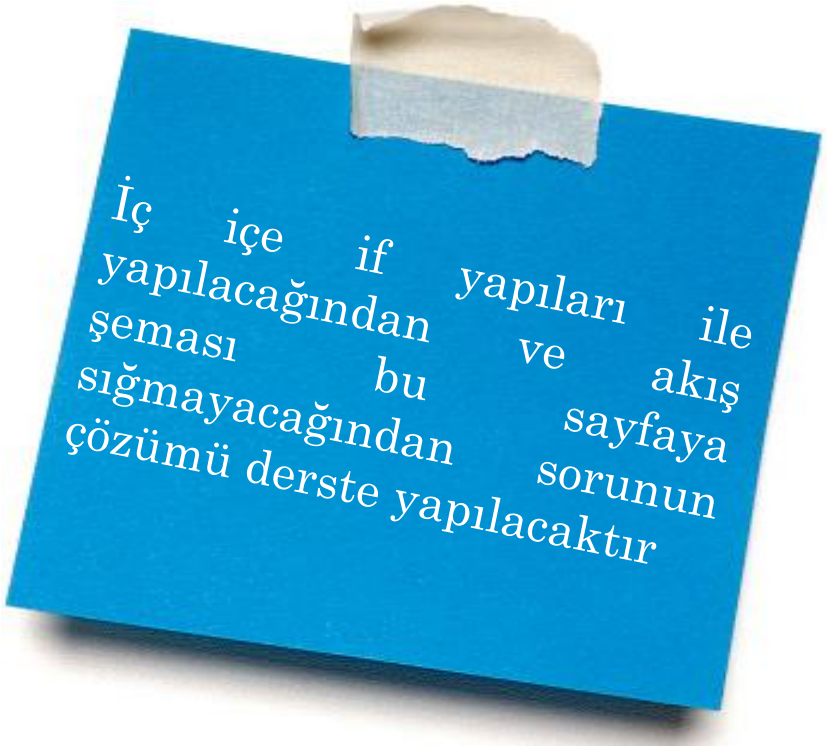
A1: Başla
A2: Sayısal vize, final, ortalama
A3: Oku vize, final
A4: ortalama = vize * 30 / 100 + final * 70 / 100
A5: Eğer ortalama > 50 && final > 50 ise Yaz Geçti
A6: Değilse Yaz Kaldı
A7: Bitir



Algoritma Örnekleri

Soru : 100 lük sistemde girilen notu 5 lik sisteme çeviren programın algoritmasını ve akış diyagramını oluşturunuz

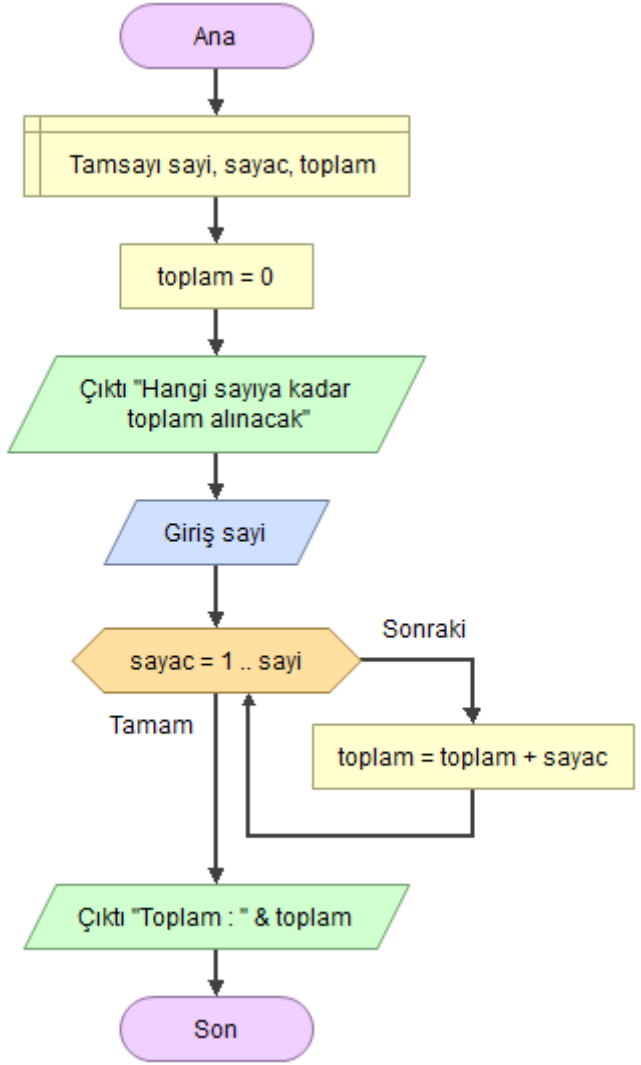
A1: Başla
A2: Sayısal sayi
A3: Oku sayi
A4: Eğer(sayi >= 85 && sayi <=100) ise yaz 5
A5: Değilse Eğer(sayi >= 70 && sayi <= 84) Yaz 4
A6: Değilse Eğer(sayi >=55 && sayi <=69) Yaz 3
A7: Değilse Eğer(sayi >= 45 && sayi <=54) Yaz 2
A8: Değilse Eğer(sayi > 0 && sayi <=44) Yaz 1
A9: Bitir.



Algoritma Örnekleri

Soru : Girilen sayıya kadar olan sayıları toplayan algoritma ve akış diyagramını hazırlayınız

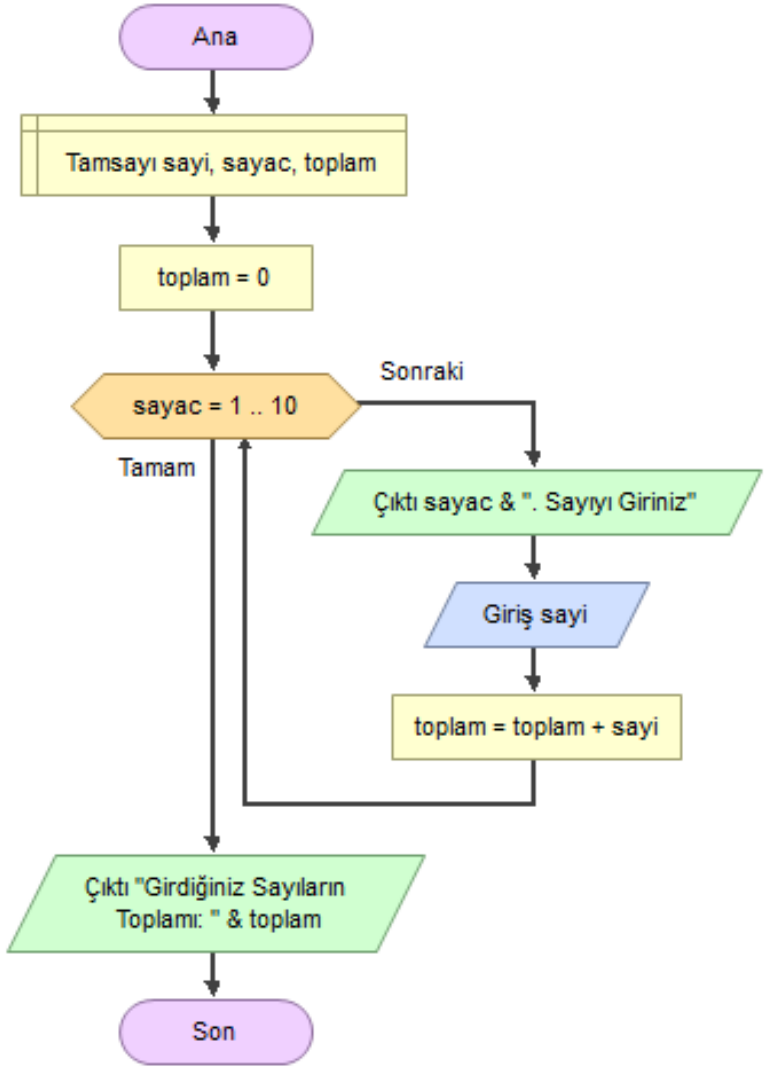
A1: Başla
A2: Sayısal sayi, sayac, toplam
A3: toplam=0
A4: Oku sayi
A5: Döngü(sayac = 1 to sayi)
A6: toplam = toplam + sayaç
A7: Döngü Sonu
A8: Yaz toplam
A9: Bitir



Algoritma Örnekleri

Soru : Klavyeden girilen 10 tane sayıyı toplayan algoritma ve akış diyagramını hazırlayınız

A1: Başla
A2: Sayısal sayi, sayac, toplam
A3: toplam=0
A4: Döngü(sayac = 1 to 10)
A5: Oku sayi
A6: toplam = toplam + sayi
A7: Döngü Sonu
A8: Yaz toplam
A9: Bitir

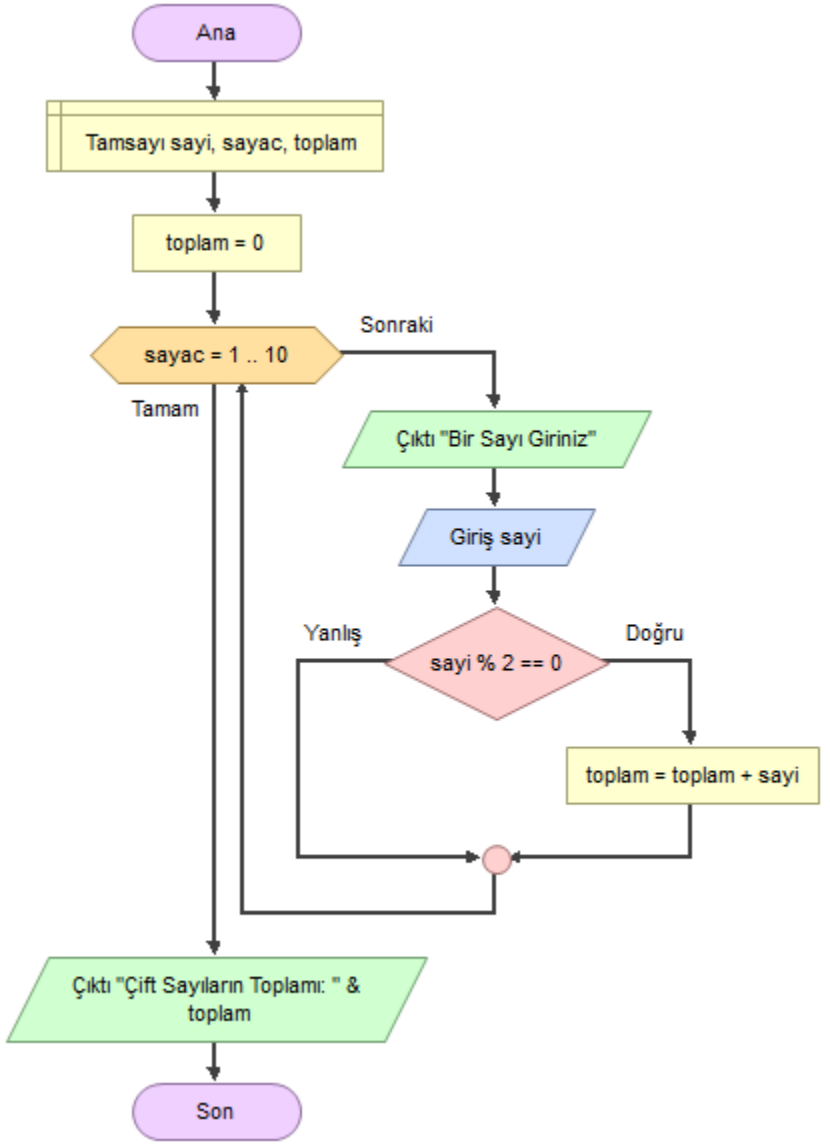


Algoritma Örnekleri

Soru : Klavyeden girilen 10 tane sayıdan çift olanların toplamını hesaplayan algoritma ve akış diyagramını hazırlayınız

Bir sayının tek mi çift mi olduğunu anlamak için o sayının 2 ile bölümünden kalana bakarız. Kalan 0 ise sayımız çift, kalan 1 ise sayımız tektir. Kalanı bulmak için % kullanılır

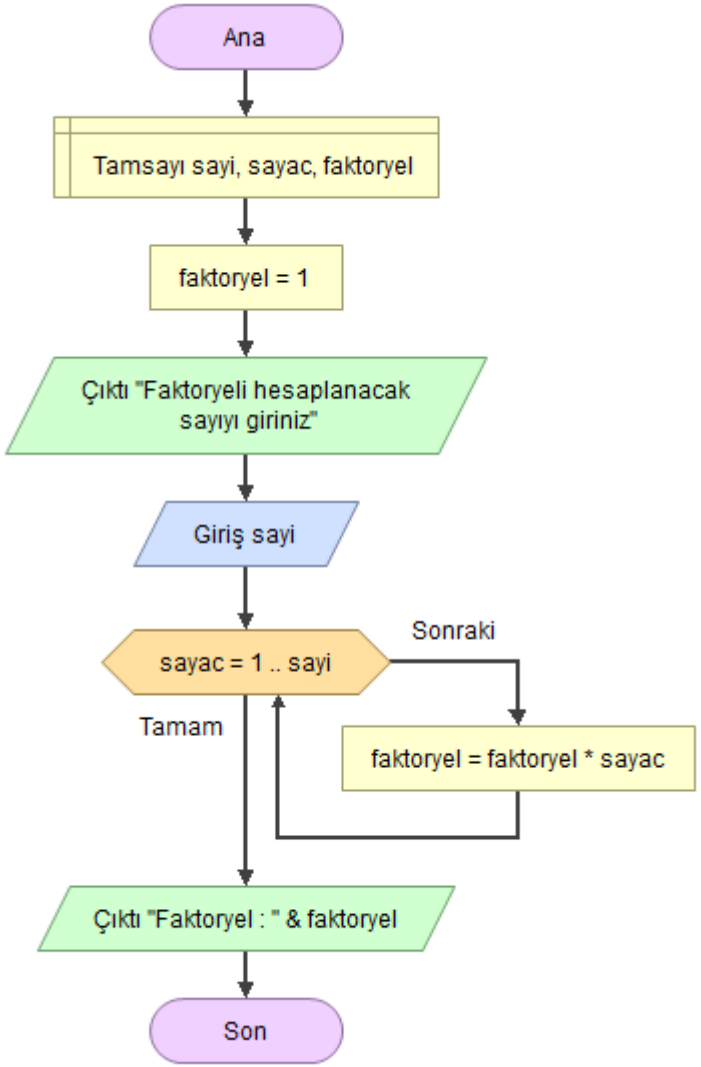
A1: Başla
A2: Sayısal sayi, sayac, toplam
A3: toplam=0
A4: Döngü(sayac = 1 to 10)
A5: Oku sayi
A6: Eğer sayi %2 == 0 İse toplam = toplam + sayi
A7: Döngü Sonu
A8: Yaz toplam
A9: Bitir



Algoritma Örnekleri

Soru : Girilen sayının faktöriyelini hesaplayan algoritma ve akış diyagramını hazırlayınız

A1: Başla
A2: Sayısal sayi, sayac, faktoriyel
A3: faktoriyel = 1
A4: Oku sayi
A5: Döngü(sayac = 1 to sayi)
A6: faktoriyel = faktoriyel * sayac
A7: Döngü Sonu
A8: Yaz faktoriyel
A9: Bitir

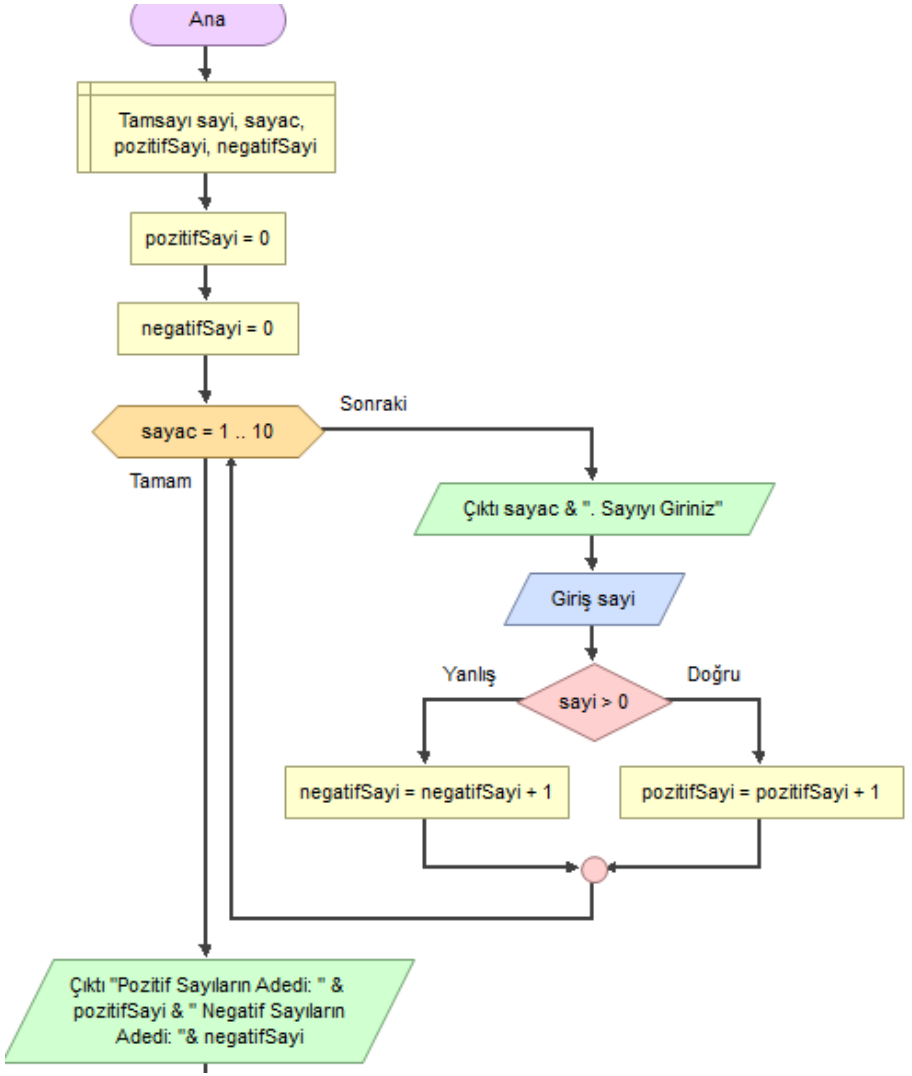


Faktöryel : Bir sayının faktöriyeli, 1'den o sayıya kadar olan tüm doğal sayıların çarpımıdır.
5! = 1 x 2 x 3 x 4 x 5 = 120

Algoritma Örnekleri

Soru : Girilen 10 adet sayının kaç tanesinin pozitif kaç tanesinin negatif olduğunu bulan algoritma ve akış diyagramını hazırlayınız

A1: Başla
A2: Sayısal sayi, sayac, pozitifSayi, negatifSayi
A3: pozitifSayi = 0, negatifSayi = 0
A4: Döngü(sayac = 1 to 10 step 1)
A5: Oku sayi
A6: Eğer sayi>0 ise pozitifSayi = pozitifSayi + 1
Değilse negatifSayi = negatifSayi+1
A7: Döngü Sonu
A8: Yaz pozitifSayi, negatifSayi
A9: Bitir

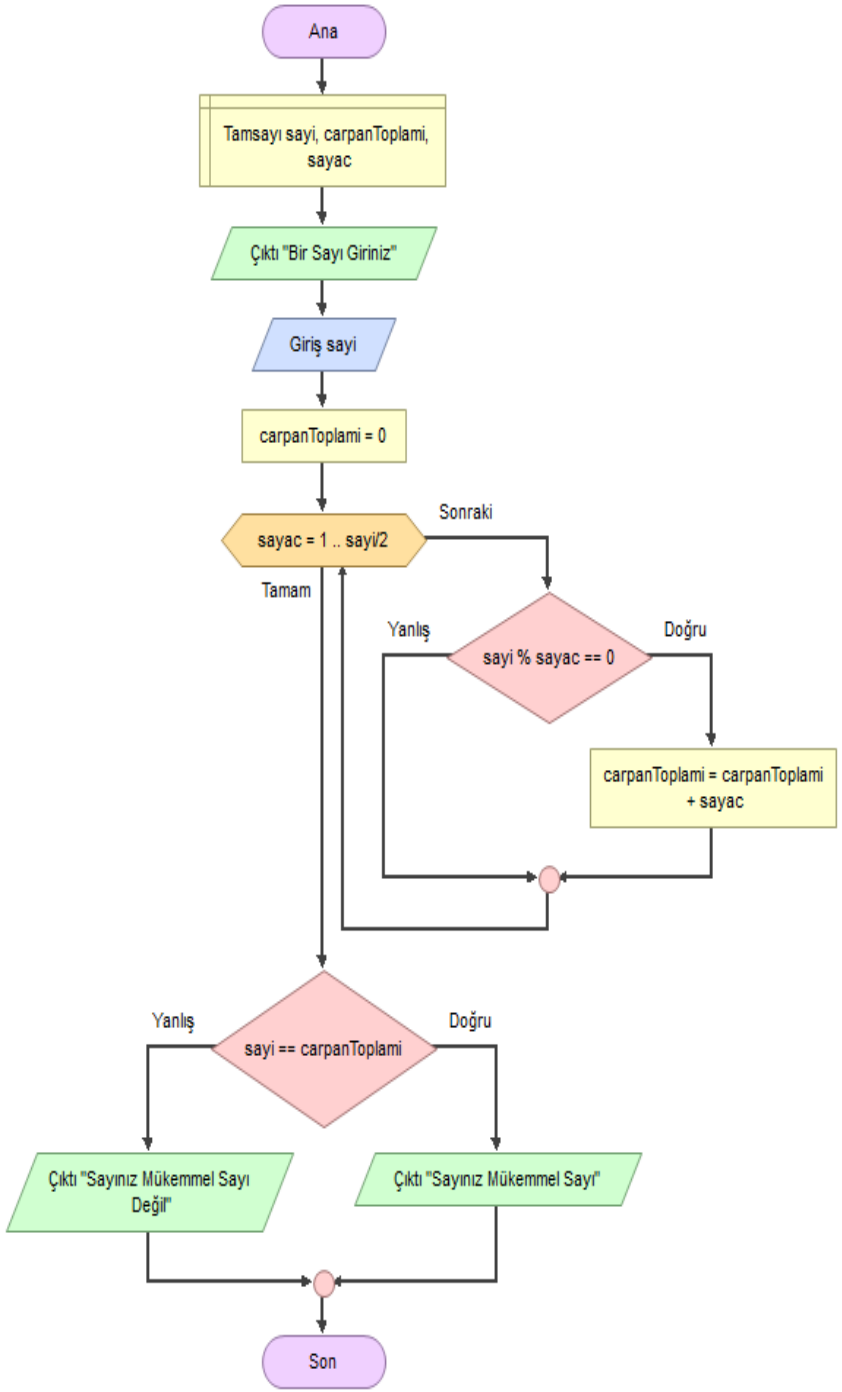


Algoritma Örnekleri

Soru : Girilen sayının mükemmel sayı olup olmadığını bulan algoritma ve akış diyagramını hazırlayınız

Mükemmel Sayı: Kendisi hariç pozitif tam bölenlerinin toplamı, sayının kendisine eşit olan sayıya denir. Örneğin 28 sayısının pozitif tam bölenlerinin toplamı:
1 + 2 + 4 + 7 + 14 =28

A1: Başla
A2: Sayısal sayi, sayaç, carpanToplami
A3: Oku sayi
A4: carpanToplami = 0
A5: DÖNGÜ(Sayac = 1 to sayi/2 step 1)
A6: EĞER sayi % sayac == 0 ise
carpanToplami = carpanToplami + sayac
A7: DÖNGÜ SONU
A8: Eğer sayi == carpanToplami ise Yaz Sayınız Mükemmel
DEĞİLSE YAZ Sayınız Mükemmel Değil
A9: Bitir

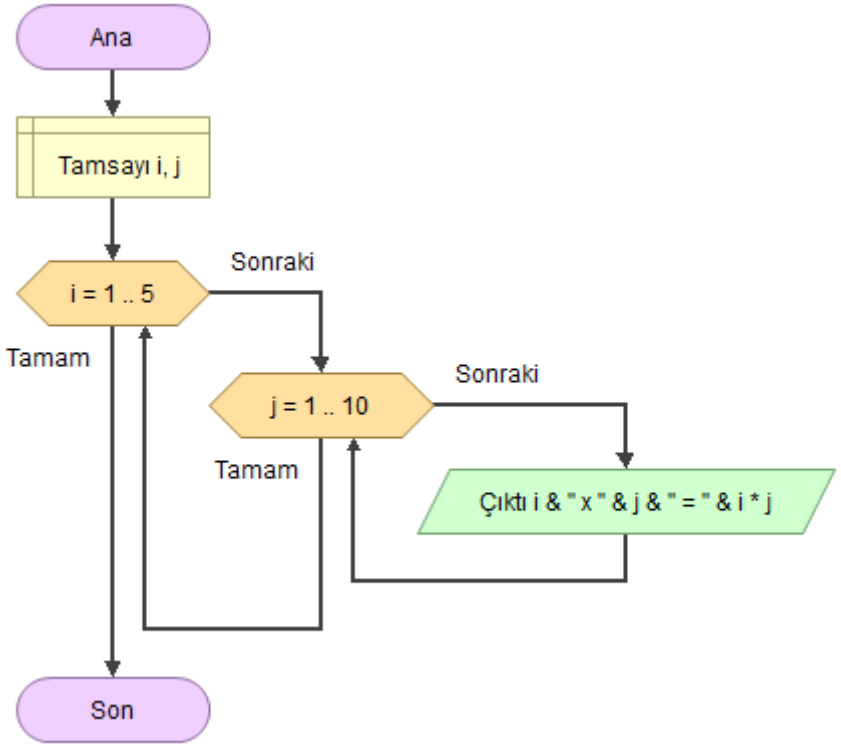


Algoritma Örnekleri

Soru : 5 e kadar olan çarpım tablosunu ekrana yazan algoritma ve akış diyagramını hazırlayınız.

Çıktı:
1 x 1 = 1
1 x 2 = 2
…
…
…
Şeklinde Sonuç Vermeli. Bu uygulamamızda iki tane döngü iç içe kullanılacaktır. Dıştaki döngümüz bir kere çalıştığında içteki döngümüz 10 defa çalışacaktır.

A1: Başla
A2: Sayısal i, j
A3: Döngü(i = 1 to 5)
A4: Döngü(j = 1 to 10)
A5: Çıktı i & " x " & j & " = " & i * j
A6: Döngü j Sonu
A7: Döngü i Sonu



Döngü Tipleri For - While

Programlamada bazen bir dizi ifadenin birden çok kez tekrarlanması gerekir. Bu görevleri gerçekleştirmek için tekrar kontrol yapıları vardır. Bütün programlama dillerinde kullanılan iki farklı döngü bulunmaktadır. Kullanım alanlarına göre farklılık göstermektedir. Şimdi bu iki döngü arasındaki farkları görelim.

For

1. Tekrar sayısının bilindiği durumlarda kullanılır.

2. Döngü içindeki tekrarı sağlayan değişkenin değeri otomatik artar.

3. Döngüden, program içinde belirlenen üst limite ulaşıldığında çıkılır.

While

1. Tekrar sayısının bilinmediği durumlarda kullanılır.

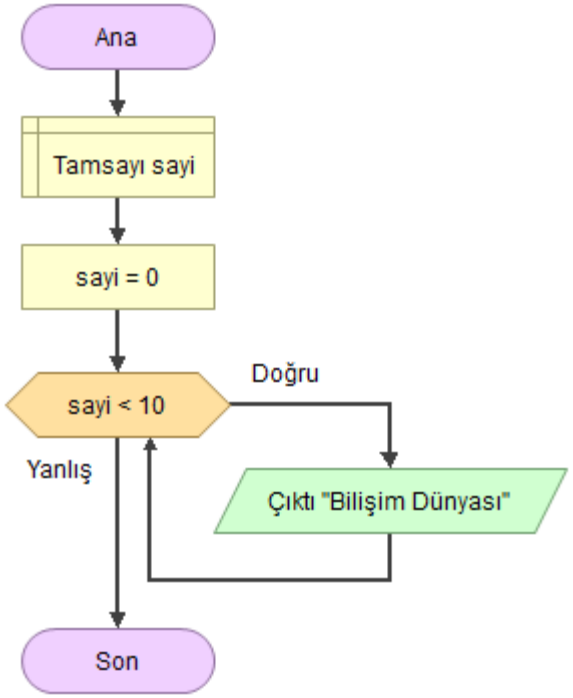
2. Döngü içindeki değişkenin değeri otomatik artmaz. Döngü içinde, programcının ekleyeceği kod satırı ile arttırılır.

3. Bir koşula bağlı olarak çalışır. Koşul sonucu TRUE olduğu sürece döngü çalışır, FALSE olduğunda ise döngüden çıkılır.

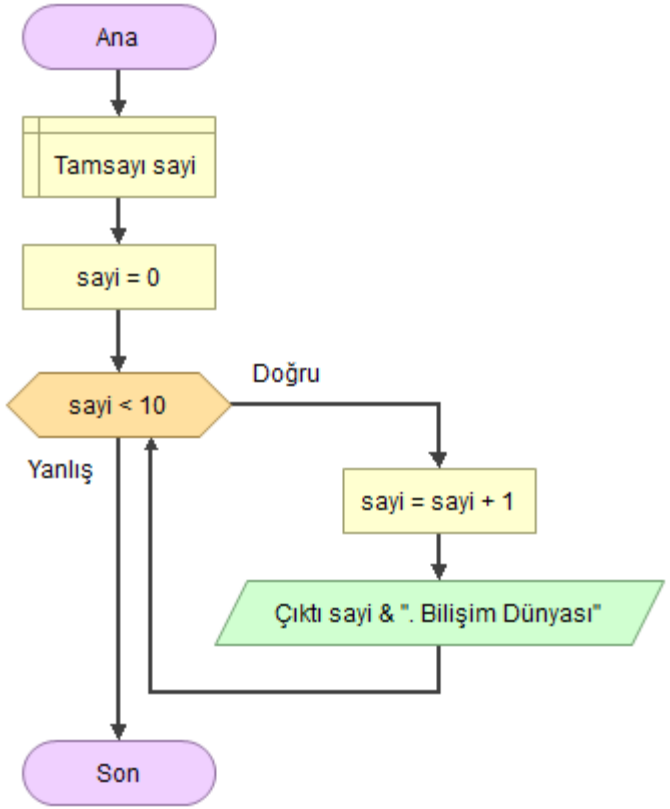
Döngü Tipleri While

While döngüsünde tanımlanan şart gerçekleştiği sürece çalışmaya devam etmektedir. For döngüsünde olduğu gibi sayac değişkenimiz otomatik artmayacaktır. Döngü içinde sayac değişkenin değeri arttırılmalıdır. Aksi takdirde döngümüz sonsuz döngü olacaktır.

Sonsuz Döngü



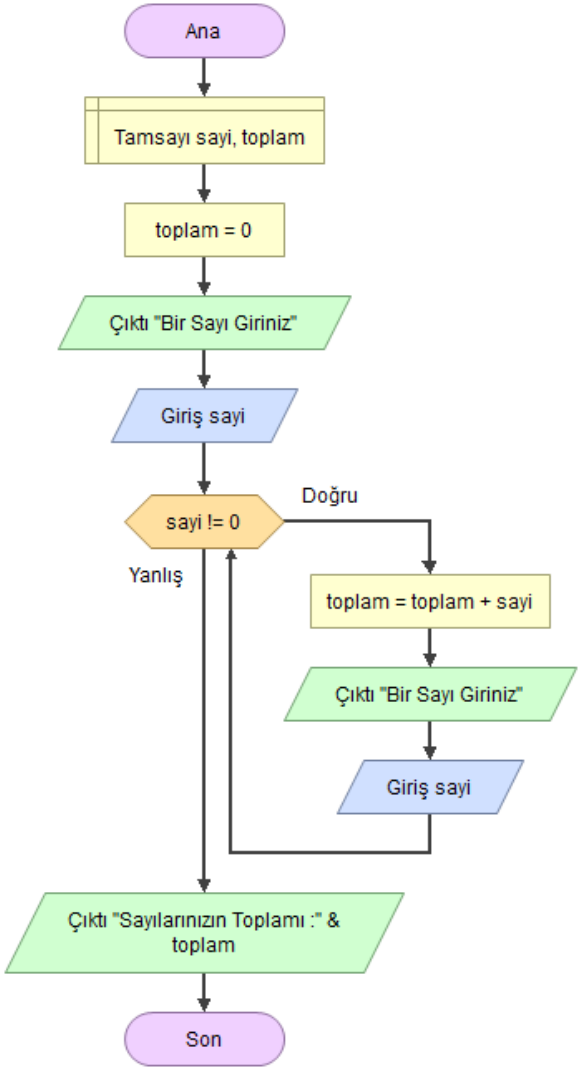
10 defa çalışan döngü



Döngü Tipleri While

Soru : Klavyeden 0 sayısı girilene kadar, girilen tüm sayıları toplayan programın algoritmasını ve akış şemasını hazırlayınız.

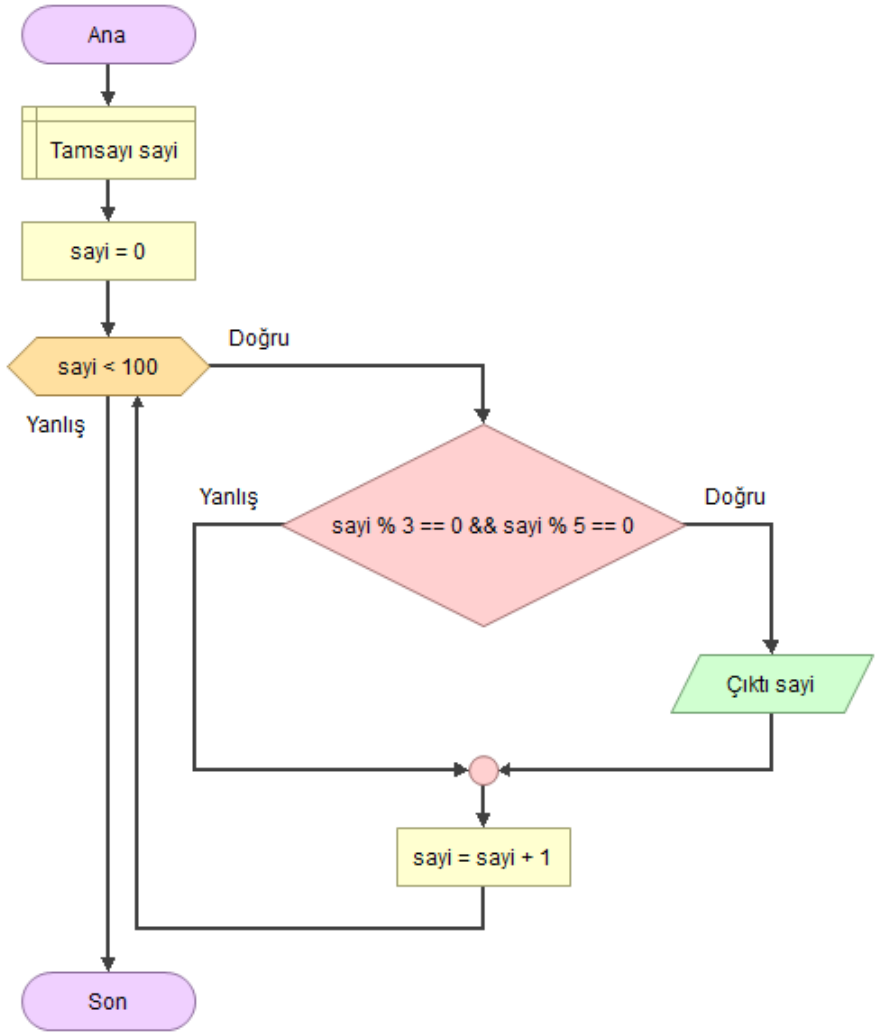
A1: Başla
A2: Sayısal sayi, toplam
A3: Oku sayi
A4: toplam = 0
A5: DÖNGÜ (sayi ! = 0)
A6: toplam = toplam + sayi
A7: Oku sayi
A8: DÖNGÜ SONU
A9 Yaz toplam
A10: Bitir



Algoritma Örnekleri

Soru : 0-100 arasında 3 e ve 5 e tam bölünen sayıları listeleyen programı while döngüsü kullanarak algoritma ve akış şemasını hazırlayınız.

A1: Başla
A2: Sayısal sayi
A3: sayi = 0
A4: Dongü(sayi < 100)
A5: Eğer(sayi % 3 == 0 && sayi %5 ==0) ise Yaz sayi
A6: sayi = sayi + 1
A7: Döngü Sonu
A8: Bitir

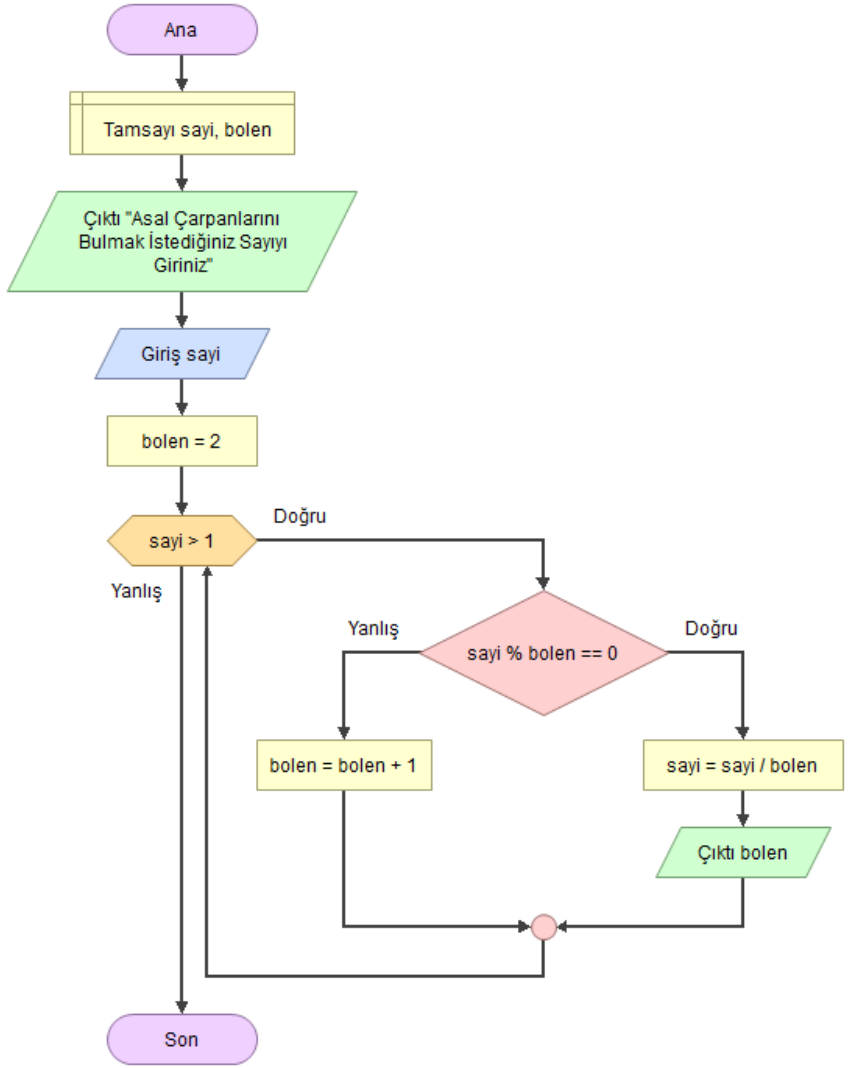


Algoritma Örnekleri

Soru : Girilen sayının asal çarpanlarını bulan algoritma ve akış diyagramını hazırlayınız

Bir şart gerçekleştiği sürece kodun tekrarlanmasını sağlayan döngü While döngüsüdür. Bu örnekte tekrar sayısını bilemeyeceğimiz için bir şarta bağlı olarak tekrarlama işlemlerini yapacağız.

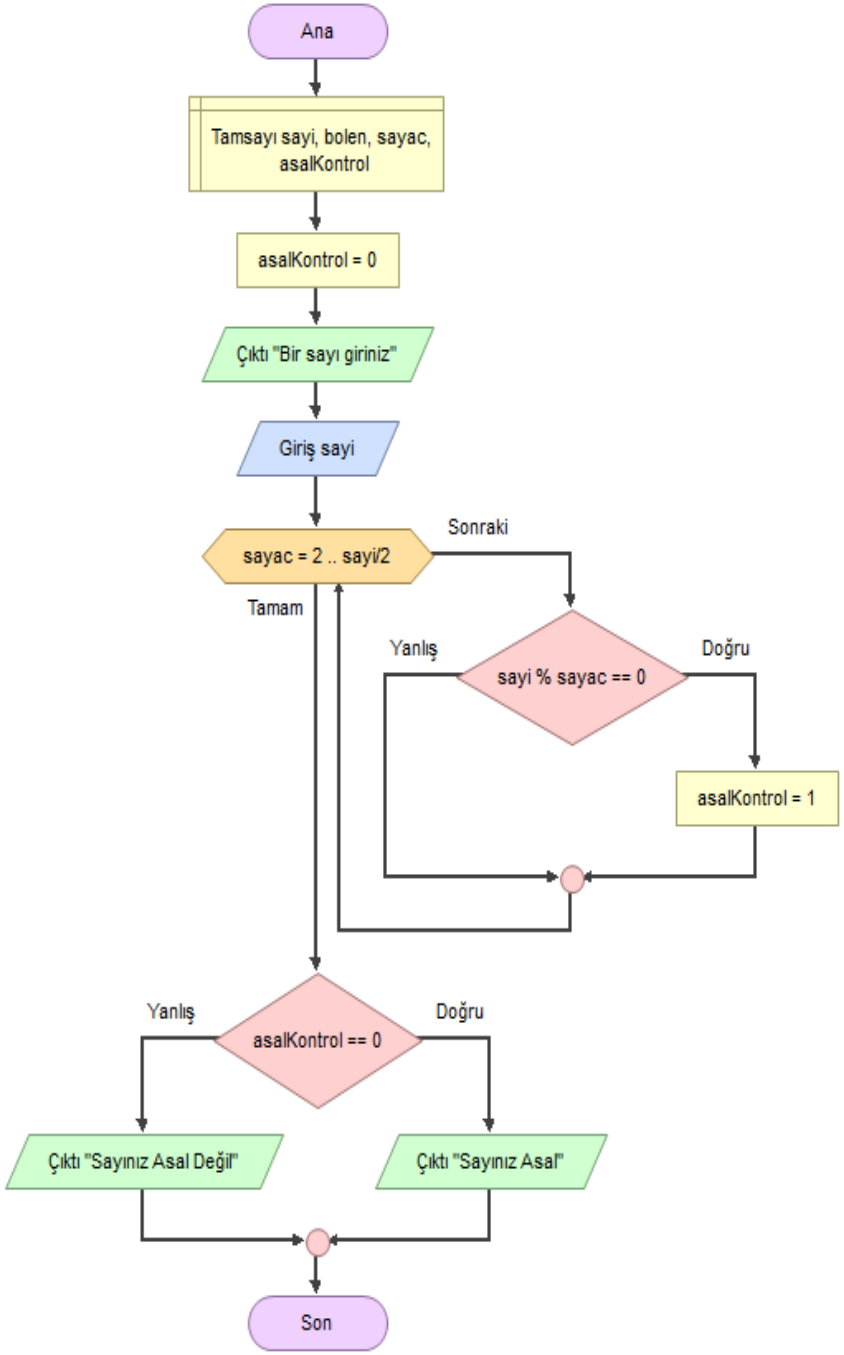
A1: Başla
A2: Sayısal sayi, bolen
A3:Oku sayi
A4: bolen =2
A5: DÖNGÜ while(sayi > 1)
A6: EĞER sayi % bölen == 0 ise
sayi = sayi / bölen, Yaz bolen
DEĞİLSE
bölen = bölen+1
A7: DÖNGÜ SONU
A8: Bitir



Algoritma Örnekleri

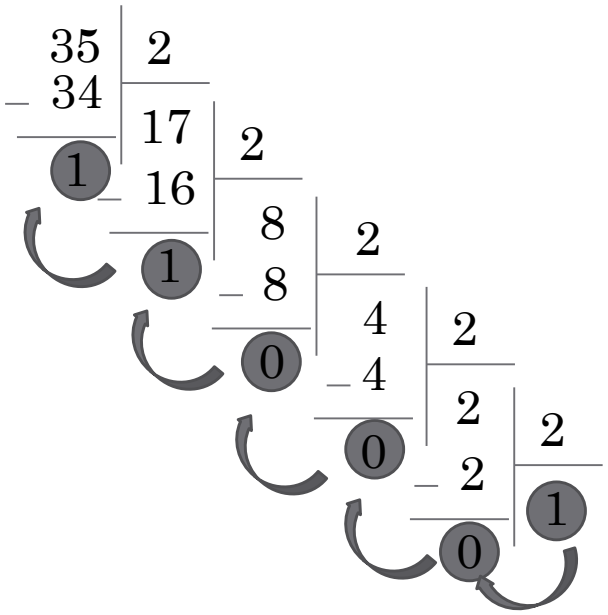
Soru : Girilen sayının asal olup olmadığını bulan algoritma ve akış diyagramını hazırlayınız

A1: Başla
A2: Sayısal sayi, sayaç, asalDurumKontrol = 0
A3: Oku sayi
A4: DÖNGÜ(Sayac = 2 to sayi/2 step 1)
A5: EĞER sayi % sayac == 0 ise
asalDurumKontrol = 1
A6: DÖNGÜ SONU
A7: Eğer asalDurumKontrol == 0 ise Yaz Sayınız Asal
DEĞİLSE Yaz Sayınız Asal Değil,
A8: Bitir



Algoritma Örnekleri

Soru: 10 tabanındaki bir sayıyı 2 tabanına çeviren algoritma ve akış diyagramını hazırlayınız.



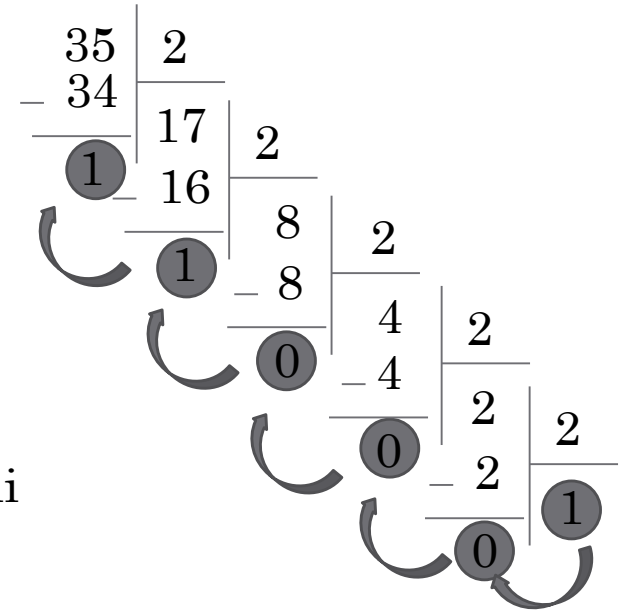
Bu sorunun çözümünü kağıt üstünde nasıl yapılacağını görmüştük. Girilen sayının 2 ile bölümünden kalan kısımları alıp yan yana ekliyorduk. Bölme işleminde bölüm kısmının küsuratlı olmaması için aşağı yuvarlama işlemi yapılmalı. Örneğin 35 sayısını 2 ye böldüğümüz zaman 17.5 gibi bir sonuç olmamalı. Bu yüzden 17.5 sayısını aşağı yuvarlamamız gerekiyor. Bunun için her programlama dilinin bir metodu(komutu) vardır. Flowgorithm programında Int metodu ile bu işlemi yapabiliriz. Bölme işleminde kalan kısmını ise metinsel ifadelerde birleştirme operatörü(&) ile yan yana getirebiliriz. Burada döngüsel bir işlem vardır. Ve girilen sayı 1 olana kadar bu bölme işlemi devam edecektir. Burada While döngüsünü kullanabiliriz.

$$35 = (100011)_2$$

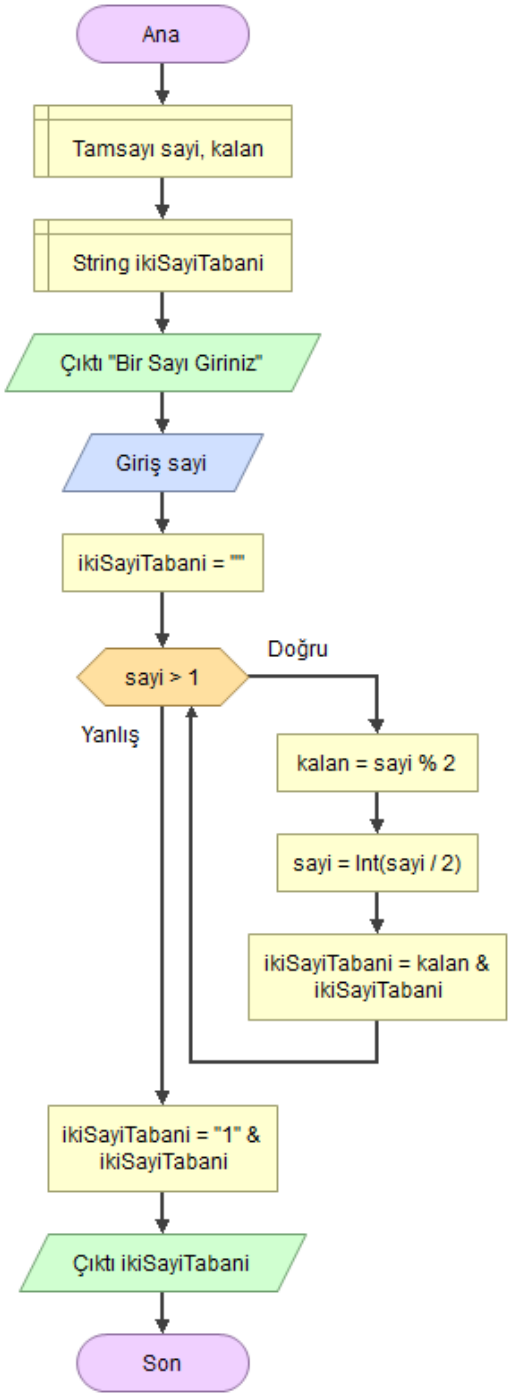
Algoritma Örnekleri

Soru: 10 tabanındaki bir sayıyı 2 tabanına çeviren algoritma ve akış diyagramını hazırlayınız.

A1: Başla
A2: Sayısal sayi, kalan
A3: String ikiSayiTabani
A4: Oku sayi
A5: Döngü WHILE(sayi >1)
kalan = sayi % 2
sayi = Tam Kısmı(sayi / 2)
ikiSayiTabani = kalan & ikiSayiTabani
A6: Döngü Sonu
A7: ikiSayiTabani = "1" & ikiSayiTabani
A8: Yaz ikiSayiTabani
A9: Bitir



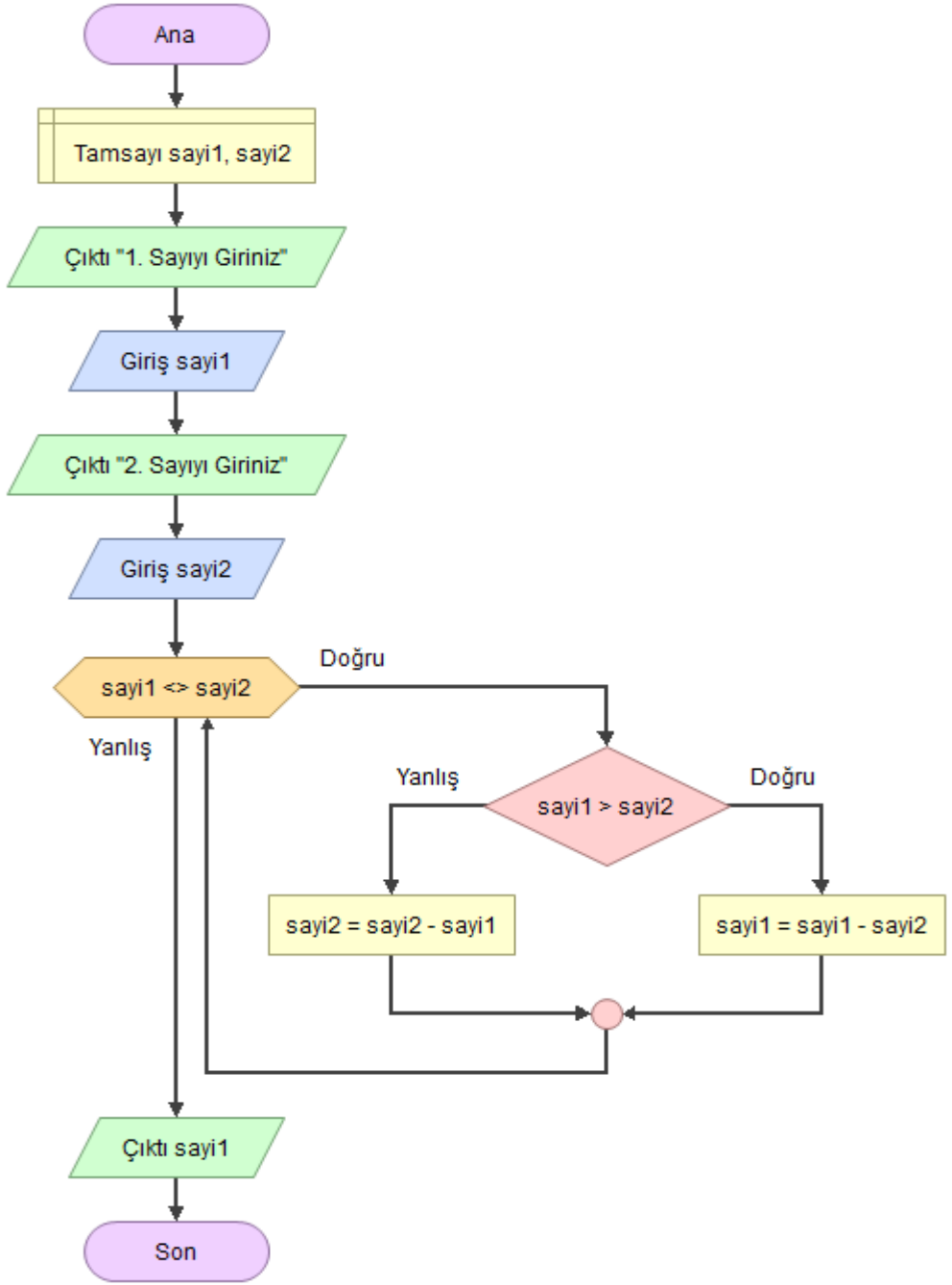
$$35 = (100011)_2$$



Algoritma Örnekleri

Soru: Girilen iki sayının ortak bölenlerinin en büyüğünü(OBEB)'ini bulan programın algoritma ve akış diyagramını oluşturunuz.

A1: Başla
A2: Sayısal sayi1, sayi2
A3: Oku sayi1, sayi2
A4: Döngü WHILE(sayi1 <> sayi2)
    EĞER sayi1 > sayi2 ise
        sayi1 = sayi1 – sayi2
    DEĞİLSE
        sayi2 = sayi2 – sayi1
A5: Döngü Sonu
A6: Bitir



Algoritma Örnekleri

Soru : Girilen sayı kadar asal sayı bulan algoritma ve akış diyagramını hazırlayınız

Daha önceki derslerimizde girilen sayının asal olup olmadığını tespit eden programın algoritmasını hazırlamıştık. Şimdi ise girilen sayı kadar asal sayı bulunmasını sağlayacağız. Önce bir sayı girdik ve asal sayı olup olmadığını tespit etmek için bir döngü kullandık ve girdiğimiz sayıyı bölmeye çalıştık. Şimdi ise sayıyı biz girmeyeceğiz. Sayıları, ikinci bir döngü kullanarak ortaya getireceğiz. Ve bu döngümüzün tekrar sayısı kesin olmayacak. Çünkü kullanıcı 5 tane asal sayı istediğinde 2,3,5,7,11 sayılarını getirmesi için döngü 11 defa dönecek, kullanıcı 7 tane asal sayı istediğinde 2,3,5,7,11,13,17 sayılarını getirmesi için döngü 17 defa çalışmış olacak.

Tekrar sayısını bilemediğimiz durumlarda döngümüze bir başlangıç ve bitiş değeri veremeyiz. Bu durumda döngümüze bir şart ekleriz ve bu şart doğru olduğu sürece döngünün çalışmasını sağlarız. Bu döngü **WHILE** Döngüsüdür.

Sorunun Çözümünü Dersimizde Yapacağız

Algoritma Örnekleri

Soru : İstenilen sayıda mükemmel sayı bulan algoritma ve akış diyagramını hazırlayınız

Daha önceki derslerimizde mükemmel sayının tanımını ve nasıl tespit edildiğini görmüştük. Sadece girdiğimiz sayının mükemmel olup olmadığını gördük. Şimdi ise biz sadece kaç adet mükemmel sayı bulmak istediğimizi gireceğiz ve program 2 den başlayarak tek tek sayıları kontrol edecek ve her bulduğu mükemmel sayıyı ekrana yazacak. Ve istediğimiz adet kadar mükemmel sayıyı bulduktan sonra program sona erecek. Bu programda kaç tane sayının kontrol edileceğini bilemeyiz. Bu yüzden aranacak sayıların üst sınırı bilmediğimiz için while döngüsü ile bir şart kullanarak yaparız. Şart olarak **istenenMukemmelSayi > bulunanMukemmelSayi** şeklinde oluşturabiliriz.

Programı kendiniz yapmayı deneyin. Yapamadığınız takdirde dersi izleyiniz.

Sorunun Çözümünü Dersimizde Yapacağız

DİZİLER

Değişkenler üzerinde sadece tek bir veriyi saklayabildiğini gördük. Peki çok sayıda bilgiyi saklamak istediğimiz zaman her bilgi için bir değişken mi tanımlayacağız? Tabi ki hayır.
Birden fazla veriyi saklamamızı sağlayan veri tiplerine dizi(array) denir.
Bir liste veriyi tek bir değişkende saklamamıza olanak sunar.
Örneğin bir sınıftaki öğrencilerin notlarını saklamak istesek her öğrencinin notunu ayrı değişkenlerde saklamamız neredeyse imkansız olacaktır. Bu durumda dizileri kullanabiliriz.

İndis Numaraları	0	1	2	3	4	5	6	7	8	9
ogrenciAdi[ ]	A	B	C	D	E	F	G	H	I	J
ogrenciNot[ ]	65	95	80	85	70	60	50	80	85	90

Düşünelim

1 öğrencinin 10 tane dersi varsa, her dersten 2 yazılı, 2 sözlü , 1 proje ve Not Ortalaması (Her bir ders için) kaç değişken tanımlamak gerekir?
1 Öğrenci için -> 10 Ders x (2+2+1+1) = 60 tane değişken
1 Sınıf için (30 Öğrenci) -> 60 x 30 = 1800 tane değişken
1 Okul için (12 Şube) -> 1800 x 12 = 21.600

DİZİ TANIMLAMA

Dizileri tanımlamak için [ ] işaretleri kullanılır. Dizi içindeki her elemanın bir sıra numarası(indis) vardır. Ve bu elemanlara erişmek için indis numarası kullanılır. İndis numaraları 0 dan başlar 1, 2, 3 … şeklinde devam eder.

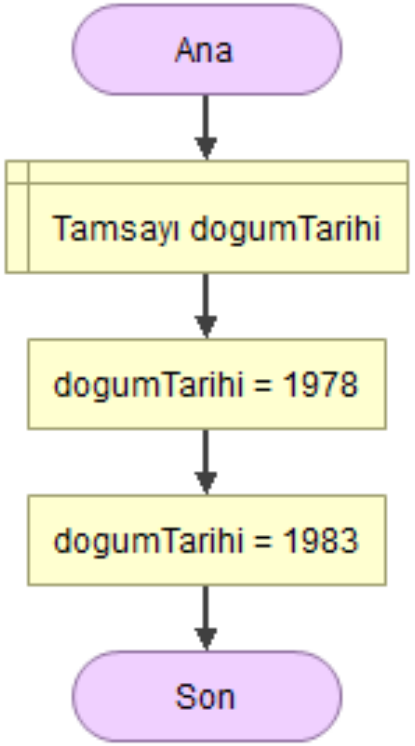
Bir dizi tanımlanırken orta seviyeli programlama dillerinde(C / C++ vb.) kaç elemandan oluşacağı belirtilmelidir.
Bir dizi de aynı türden elemanlar bulunmalıdır. Sayısal olacaksa sadece sayılar eklenmeli. String olacaksa sadece metinsel veriler girilmeli. Bazı programlama dillerinde veri türü farketmeksizin bir dizi içinde saklanabilir (Python, javascirpt..).

Dizi içindeki verilere erişmek için döngüler kullanılmaktadır.

- Dizi boyutu önceden belirtilmelidir.
- Dizi elemanlarına erişmek için **[ ]** işaretleri kullanılır.
- Dizinin ilk elemanı 0. elemandır.
- DÖNGÜ ile birlikte kullanılırlar.

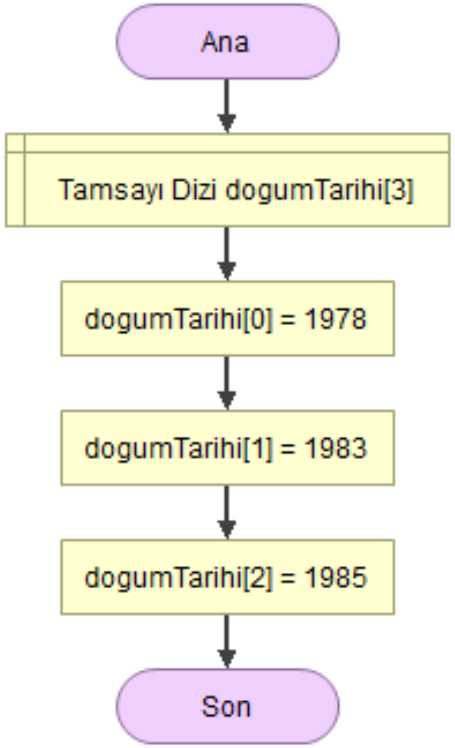
Dizi ve Değişken Özellikleri

DEĞİŞKEN



dogumTarihi isimli değişkende en son atanan 1983 sayısı saklı kalır. 1978 sayısı bellekten silinir.

DİZİ



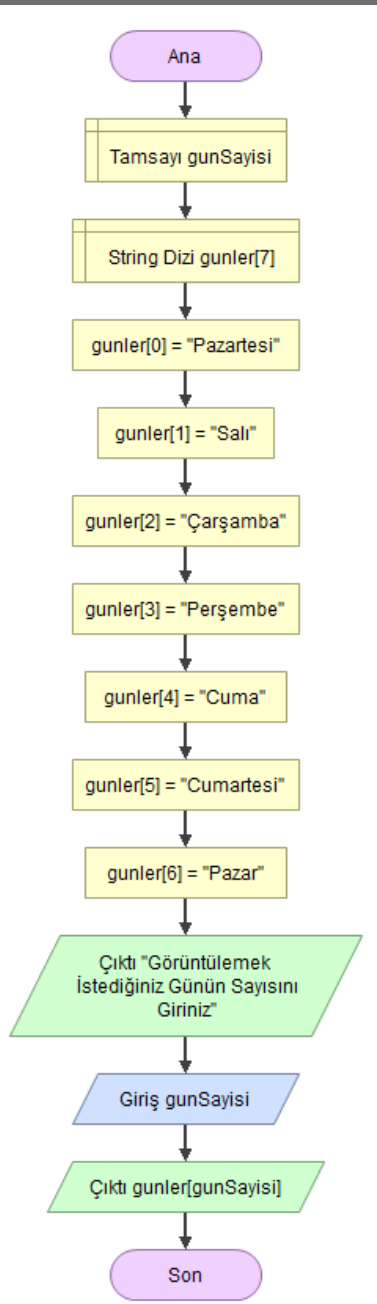
dogumTarihi isimli dizide sırası ile tüm veriler saklı kalır.
Yaz dogumTarihi[0] dediğimizde ekrana 1978 yazacaktır.
Veya
Yaz dogumTarihi[1] dediğimizde ekrana 1983 yazacaktır.

Dizi ve Döngü Örnekleri

Bir dizi oluşturulduktan sonra, programlama dillerinde veriler tek satırda girilebilirken flowgorithm programında tek tek girilmektedir. Şimdi basit örneklerle dizi mantığını anlamaya çalışalım.
Programlama dillerinde dizi kullanımı(c, c++, javascript, php vb.)

gunler = ["Pazartesi", "Salı", "Çarşamba", "Perşembe", "Cuma", "Cumartesi", "Pazar"]

Buradaki verilere erişmek için indis numaralarından faydalanacağız. Örneğin günler[0] dizi elemanını çağırdığımız zaman Pazartesi günü gelecektir. Şimdi bu yapının akış şemasını hazırlayalım ve kullanıcının girdiği sayıya göre günü ekranda görüntüleyelim

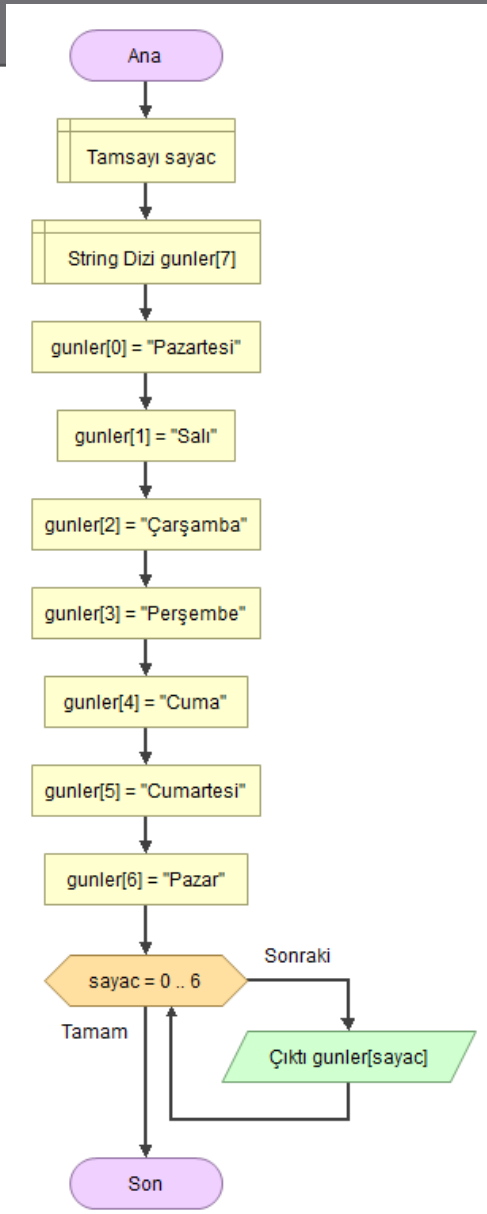


Dizi ve Döngü Örnekleri

Girmiş olduğumuz dizinin tüm elemanlarını listelemek istersek en sık kullanılan yöntemlerden biri olan for döngüsünü kullanabiliriz.

Döngümüz 0 dan başlayarak 6 ya kadar dönecektir. Ve her döndüğünde gunler[sayac] dizindeki elemanı ekranda gösterecektir.

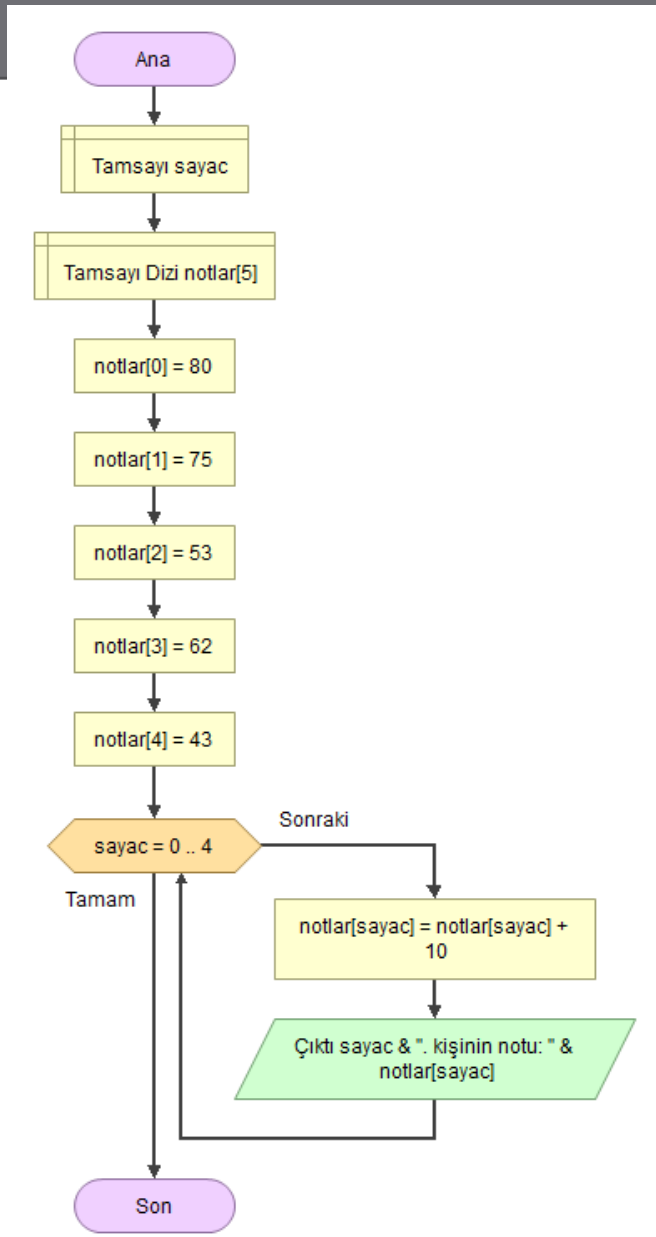
Genellikle veri tabanından bilgilerin tamamı dizilere aktarılır ve ekranda dizi değişkeni kullanılarak gösterilir. Sadece gösterme işlemi için kullanılmaz dizi içinde bulunan veriler istenildiği takdirde yeniden sıralanır veya içindeki veriler güncellenebilir.



Dizi ve Döngü Örnekleri

Dizi içinde bulunan 5 öğrencinin notlarına 10 puan eklenmesini sağlayan algoritma ve akış diyagramını hazırlayınız.

A1: Başla
A2: Sayısal sayac
A3: Dizi notlar
A4: notlar = [80, 75, 53, 62, 43]
A5: DÖNGÜ (sayac = 0 to 4)
A6: notlar[sayac] = notlar[sayac] + 10
A7: YAZ notlar[sayac]
A8: DÖNGÜ SONU
A9: Bitir

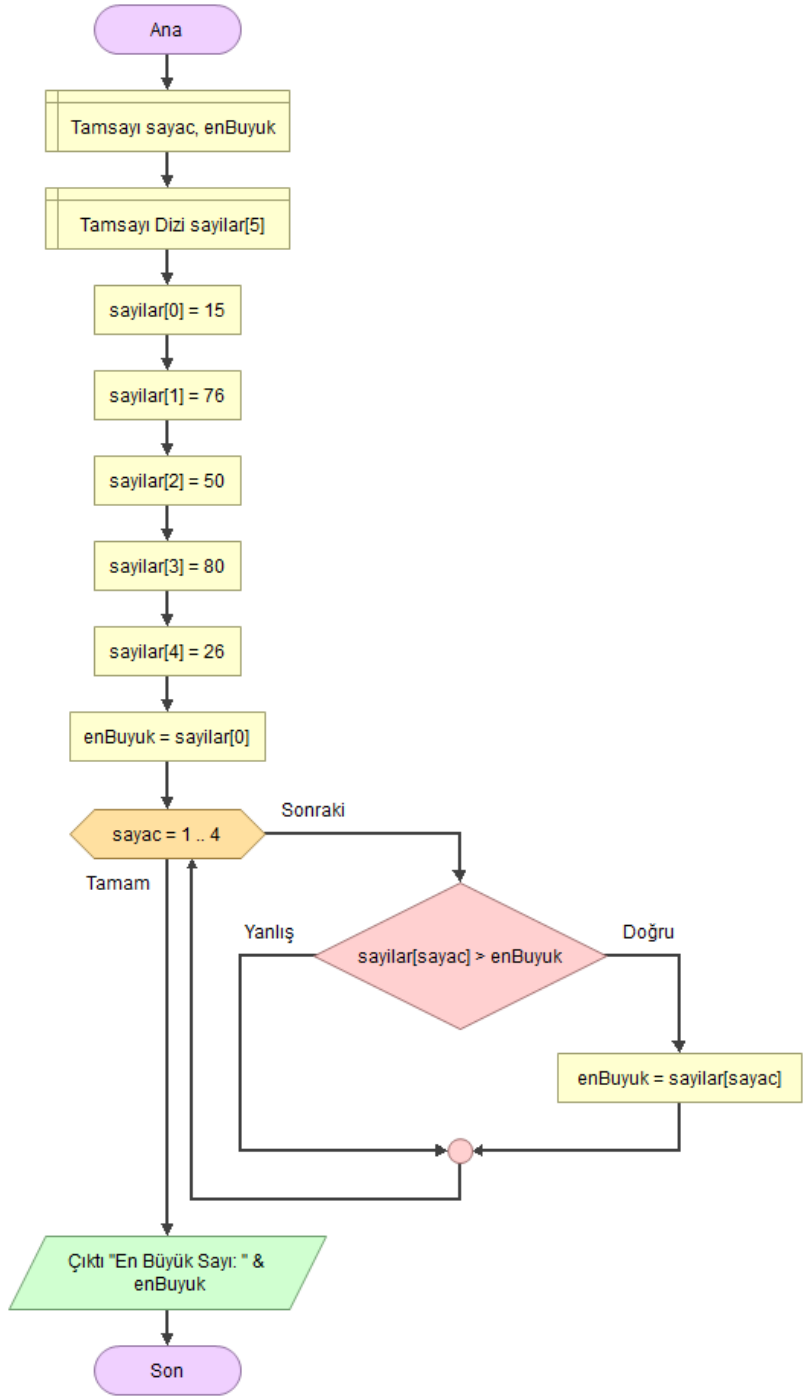


Dizi ve Döngü Örnekleri

Dizi içinde bulunan 5 sayıdan en büyük olanını bulmayı sağlayan algoritma ve akış diyagramını hazırlayınız.

Yöntem: Dizi içinde bulunan ilk elemanın en büyük sayı olduğunu varsayıyoruz ve sonraki elemanlar ile bu sayıyı karşılaştırıyoruz. Eğer daha büyük sayı varsa enBuyuk değişkenine bu sayıyı alıyoruz ve diğer elemanlar ile karşılaştırmaya devam ediyoruz.

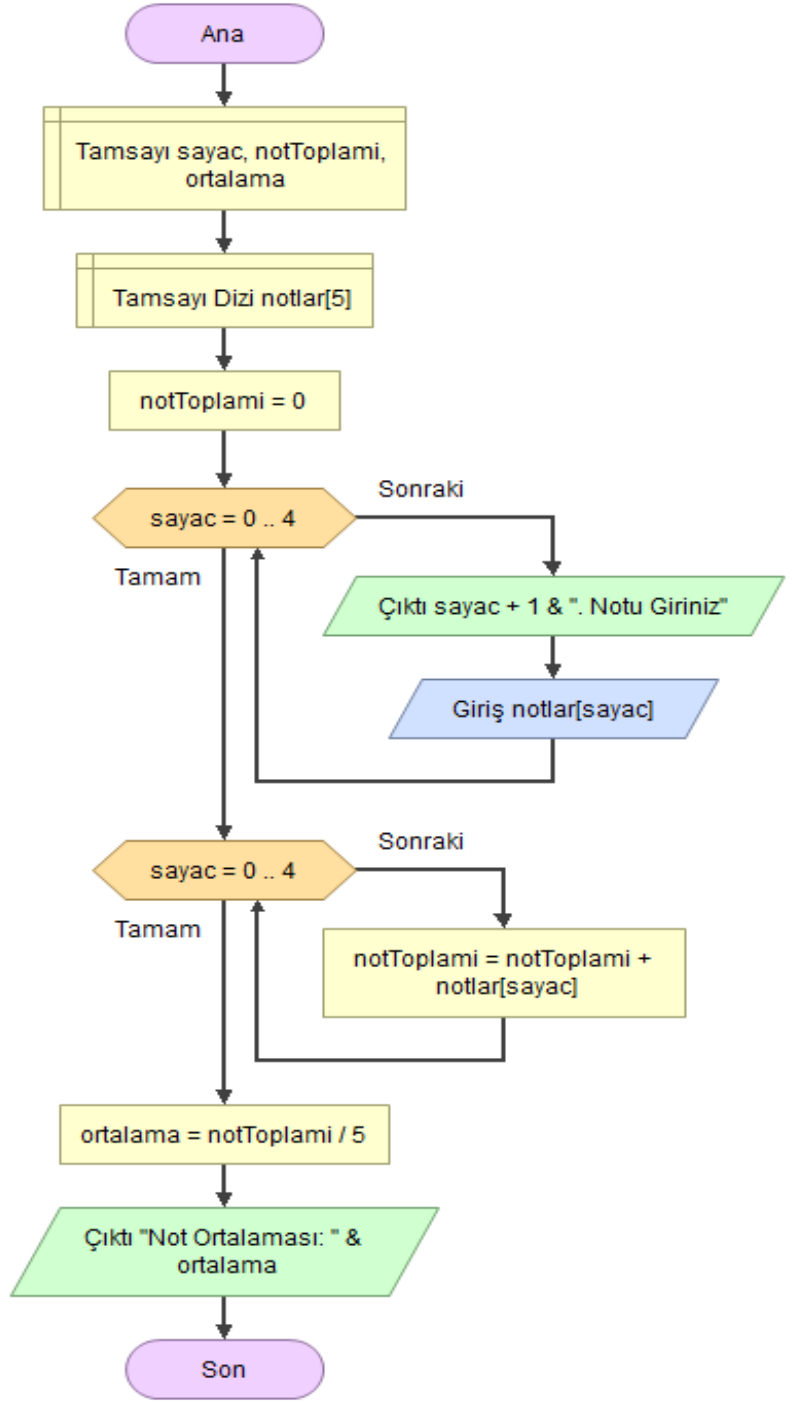
A1: Başla
A2: Sayısal sayac, enBuyuk
A3: Dizi sayilar
A4: sayilar = [15, 76, 50, 80, 26]
A5: enBuyuk = sayilar[0]
A6: DÖNGÜ (sayac = 1 to 4)
A7: Eğer sayilar[sayac] > enBuyuk ise
    enbBuyuk = sayilar[sayac]
A8: DÖNGÜ SONU
A9: YAZ enBuyuk
A10: Bitir



Dizi ve Döngü Örnekleri

Soru: Bir öğrenciye ait klavyeden girilen 5 notun ortalamasını bulan programın algoritma ve akış diyagramını hazırlayınız.

A1: Başla
A2: Sayısal sayac, notToplami, ortalama
A3: Dizi notlar
A4: notToplami = 0
A5: DÖNGÜ (sayac = 0 to 4)
A6: Oku notlar[sayac]
A7: DÖNGÜ SONU
A8: DÖNGÜ (sayac= 0 to 4)
A9: notToplami = notToplami + notlar[sayac]
A10:DÖNGÜ SONU
A11: ortalama = notToplami / 5
A12: YAZ ortalama
A13: Bitir



Dizi ve Döngü Örnekleri

Soru: Dizi içinde bulunan 5 tane sayıyı küçükten büyüğe sıralayan programın algoritma ve akış diyagramını hazırlayınız.

sayi= [21, 14, 36, 27, 11]
Sıralamak için nasıl bir yöntem izlemeliyiz.
Sayılarımızı dizi içinde yan yana olanları karşılaştırarak küçük olanları sol tarafa büyük olanları sağ tarafa almalıyız. Bu işlemi dizideki eleman sayısı kadar tekrarlamalıyız. İç içe döngüleri kullanarak bu işlemi yapacağız.

[redacted]

21 14 36 27 11

[redacted]

Dizi ve Döngü Örnekleri

Soru: Dizi içinde bulunan 5 tane sayıyı küçükten büyüğe sıralayan programın algoritma ve akış diyagramını hazırlayınız.

A1: Başla
A2: Sayısal geçici, i , j, k
A3: Dizi sayilar[5]
A4: sayilar[21, 14, 36, 27, 11]
A5: Döngü(i = 0 to 3)
A6: Döngü(j = 0 to 3)
A7: Eğer sayilar[j] > sayilar[j+1] ise
A8: geçici = sayilar[j]
A9: sayilar[j] = sayilar[j+1]
A10: sayilar[j+1] = geçici
A11: Döngü Sonu j
A12: Döngü Sonu i
A13: Döngü (k = 0 to 4)
A14: Yaz sayilar[k]
A15: Döngü Sonu
A16: Bitir