

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/347508715>

Açıklamalı Algoritma Örnek Soruları ve Çözümleri

Book · December 2015

CITATIONS

0

READS

207,831

1 author:



Deniz Mertkan Gezin

Trakya University

103 PUBLICATIONS 1,432 CITATIONS

SEE PROFILE

Açıklamalı Algoritma Soruları ve Çözümleri

Yük.Bilg.Müh. Deniz Mertkan GEZGİN

Deniz Mertkan GEZGİN

1978 yılında Kütahya’da doğdu. İlk, orta ve lise eğitimini İzmir’in Selçuk ilçesinde tamamladı. 1995’de kaydolduğu Çanakkale On Sekiz Mart Üniversitesi Mühendislik-Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümünden 1999 yılında mezun oldu. 2001’de Trakya Üniversitesi Teknik Bilimler Meslek Yüksekokulu Bilgisayar Teknolojisi ve Programlama Bölümünde öğretim görevlisi olarak göreve başladı. 2006 yılında Trakya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Bölümünde yüksek lisansını tamamladı. Deniz Mertkan Gezgın, halen aynı üniversitede görev yapmakta ve 2007’de başladığı doktora çalışmasını sürdürmektedir.

Email: mertkan@trakya.edu.tr

Önsöz

Bilgisayar programlamasında ilk safha, problemin analiz edilmesi ve bu analiz doğrultusunda problemin çözülmesine ait işlemlerden oluşur. Bu işlemler programcı tarafından bilgisayar programlama dili ile kodlanmadan önce adım adım yapılmalı ve buna paralel olarak çeşitli şekillerle desteklenmelidir. Kısaca kodlamadan önceki bu işlemlere *Algoritma*, bu algoritmanın birbirleriyle ilişkili şekillerle desteklenmesine de *akış diyagramı* adı verilir. İyi bir programcı aynı zamanda iyi bir algoritma yeteneğine de sahip olmalıdır. Günümüzde bilgisayar mühendisliği veya programcılığı öğrencilerinin ya da bilgisayar programcılığıyla uğraşanların en büyük sorunu algoritma yeteneklerini arttırmadan direkt programlama dilini öğrenmeye çalışmalarıdır. Ancak unutulmamalıdır ki problemin çözümü bilinmezse programlama dilini bilmek de bir anlam ifade etmeyecektir. Bu düşünceden hareketle bilgisayar bölümlerinde okuyan öğrenciler ve bilgisayar programcılığıyla ilgilenenler için çözümlü bir soru bankası oluşturmak istedik. Konuyla ilgili alıştırmaların çözümlerine birtakım yorumlar ekleyerek okuyucuların çözümlemeye çeşitli bakış açıları getirebilmelerini, hızlı ve dinamik programlamaya hazır algoritmalar oluşturabilmelerini amaçladık. Çözdüğümüz problemlerin C ve C# dilinde kodlarını ekleyerek kitabın programlama ve veri yapıları için de bir giriş niteliği taşımasını istedik. Oluşturmaya çalıştığımız bu geniş algoritma soru bankasının bütün okuyucularımıza faydalı olmasını ümit ediyoruz.

Bu kitabın oluşturulmasında desteğini benden esirgemeyen doktora danışman hocam Yrd. Doç. Dr. Ercan Buluş'a, kitabın Türkçe açıklamaların düzenlemesini yapan değerli arkadaşım Arş.Gör. Duygu Dalbudak'a , kitaptaki akış diyagramlarının çizimi ve kitabın grafik tasarımını yapan Bilgisayar Programlama ve Teknolojisi bölümü 2. sınıf öğrencilerinden Beysim Dobrucalı, Aytaç Aruca, Adem Güncan ve Gökhan Hacıoğlu' na ve en önemlisi beni bu yerlere getiren aileme teşekkürü bir borç bilirim.

Deniz Mertkan GEZGİN

Ekim 2008/ Edirne

Sevgili Annem'e ithafen ...

*"Yalan , yalan, yalan pek kolay olmayacak seni unutmak ,
Öyle zor , öyle zor ki seni içimden atmak..."*

İÇİNDEKİLER

Bölüm 1.....	13
Programlama ve Programlama Süreci.....	13
Programlama	14
Programlama Süreci	14
Bölüm 2	16
Algoritma ve Akış Diyagramları	16
Bölüm 3	19
Operatörler.....	19
Operatörler.....	20
Aritmetik Operatörler.....	20
Atama Operatörleri	20
Karşılaştırma Operatörleri ve Mantıksal Operatörler.....	21
Bölüm 4	22
Sayı Algoritma Soru Çözümleri	22
1. Çay demlemenin algoritma ve akış diyagramını çiziniz.	24
2. İki sayının toplamını veren programın algoritma ve akış diyagramını çiziniz.	25
3. Kullanıcının girdiği iki sayının karelerinin toplamını görüntüleyen programın algoritma ve akış diyagramını çiziniz.	29
4. 1'den 100'e kadar olan sayılarının küplerinin toplamını bulan programın algoritma ve akış diyagramını çiziniz	31
5. Doğum tarihi girilen kişinin yaşını hesaplayan programın algoritma ve akış diyagramını çiziniz	33
6. Girilen sayının faktöriyelini bulan programın algoritma ve akış diyagramını çiziniz.	35
7. Çarpma işlemini toplama kullanarak bulan programın algoritma ve akış diyagramını çiziniz.	38
8. Bölme işlemini çıkarma kullanarak yapan programın algoritma ve akış diyagramını çiziniz.....	40
9. Girilen sayının istenilen sayıya göre mod işlemini yaptıran programın algoritma ve akış diyagramını çiziniz.	42

10. Girilen sayının kaç basamaklı olduğunu söyleyen programın algoritma ve akış diyagramını çiziniz.	44
11. Girilen 3 basamaklı bir sayının basamaklarının küpleri toplamı sayının kendine eşit olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.....	46
12. Klavyeden girilen 20 adet sayıdan çift sayıların toplamının tek sayıların toplamına oranını bulan programın algoritma ve akış diyagramını çiziniz.	49
13. 10 ile 1000 arasındaki tam kare sayıları ekrana yazdıran programın algoritma ve akış diyagramını çiziniz.	52
14. Klavyeden girilen 25 adet sayı içerisinde negatif olanların toplamını, çift sayıların çarpımını, 7'ye eşit olanların adetini bulup ekrana yazdıran programın algoritma ve akış diyagramını çiziniz.	54
15. Çarpım Tablosunun algoritma ve akış diyagramını çiziniz.....	57
16. Girilen sayının 5'in kuvveti olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.	60
17. X,Y pozitif olmak üzere, eğer x sayısının çarpanları toplamı y sayısına ve aynı zamanda y sayısının çarpanları toplamı x sayısına eşit ise bu sayılar dost sayılardır. Buna göre girilen iki sayının dost olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.	63
18. Fibonacci serisinin ilk 10 terimini ekrana basan algoritma ve akış diyagramını çiziniz.	67
19. Klavyeden girilen bir sayının negatif, pozitif veya 0 olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.	69
20. Girilen sayının mükemmel sayı olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.	71
21. 1-100 arasındaki çift sayıların toplamının mükemmel sayı olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.	74
22. Herhangi bir sayının herhangi bir dereceden kuvvetini bulan programın algoritma ve akış diyagramını çiziniz.	77
23. Girilen sayının abundant (güçlü) sayı mı ya da Deficient (güçsüz) sayı mı olduğunu bulan programın algoritma ve akış diyagramını çiziniz.	79
24. 1'den 500'e kadar olan tamsayıların toplamını bulan programın algoritma ve akış diyagramını çiziniz.	82
25. Girilen a ve b sayısı 50'den büyük olduğunda $c=a+b$ işlemini yapan değilse bu sayılar uygun değil yazdıran programın algoritma ve akış diyagramını çiziniz.	84

26. 1'den 63'e kadar olan sayılar arasında istenilen sayıyı maksimum 6 seferde bulan programın algoritma ve akış diyagramını çiziniz.	86
27. Girilen decimal (onluk) bir sayının binary (ikilik) bir sayıya dönüştüren programın algoritma ve akış diyagramını çiziniz.	89
28. Binary olarak girilen sayıyı decimal sayıya çeviren programın algoritma ve akış diyagramını çiziniz.	92
29. Verilen yılın artık yıl olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.	95
30. Boyu ile kilosu girilen kişinin şişman mı, zayıf mı yoksa ideal kiloda mı olduğunu gösteren programın algoritma ve akış diyagramını çiziniz.	97
31. Dairenin alanını ve çevresini bulan programın algoritma ve akış diyagramını çiziniz.	100
32. Kenarları A,B,C,D olan bir dörtgenin kare olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.	102
33. Bir uçak 15 dk boyunca düzgün hızlanarak hızı 480 km/s oluyor. Sonra 20 dk sabit hızla gidiyor ve 15 dk boyunca düzgün yavaşlayarak hızı sıfır oluyor. Herhangi bir t anında hızı veren algoritmayı ve akış diyagramını çiziniz.	104
34. Girilen dört basamaklı sayılardan ilk iki basamağı ile son iki basamağının toplamının karesi, sayının kendine eşit olan sayılara orijinal sayı denir. Girilen sayının orijinal olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.	106
35. 1 ile 500 arasındaki tam sayılardan tek sayıların toplamı ile çift sayıların toplamının farkı negatif mi, pozitif mi olduğunu bulan programın algoritma ve akış diyagramını çiziniz.	109
36. 1 ile 500 arasındaki tam sayılardan tek sayıların toplamı ile çift sayıların toplamının farkı negatif mi, pozitif mi olduğunu bulan programın algoritma ve akış diyagramını çiziniz.(Döngü Kullanarak)	112
37. 4 haneli bir sayının birler, onlar, yüzler ve binler hanesini bulan programın algoritma ve akış diyagramını çiziniz.	115
38. Rastgele girilen 50 sayıdan negatif olanların ve pozitif olanların sayısını bulan programın algoritma ve akış şemasını çiziniz.	118
39. Sayı bulmaca oyunu programının algoritma ve akış diyagramı çiziniz.	121
40. 10 ile 200 arasındaki tam sayılardan 3 katının 2 fazlası 5 ile tam bölünebilen sayıları gösteren programın algoritma ve akış diyagramını çiziniz.	124
41. İç açıları verilen üçgenin karar ağacı, algoritma ve akış diyagramını çiziniz.	127

42. Girilen sayının yaklaşık olarak karekökünü hesaplayan programın algoritma ve akış diyagramını çiziniz.	131
43. Obek ve Okek bulan programın algoritma ve akış diyagramını çiziniz.	133
44. 1 ile 25 arasındaki tam sayıların karelerinin çarpımını bulan programın algoritma ve akış diyagramını çiziniz.	136
45. $ax^2 + bx + c = 0$ tipindeki bir denklemin köklerini bulan programın algoritma ve akış diyagramı çizini	138
46. 1-100 arasında kaç asal sayı vardır gösteren programın algoritma akış diyagramını çiziniz.	141
47. 10-100 arasındaki asal sayıları gösteren programın algoritma akış diyagramını çiziniz.	145
48. Girilen sayının smith sayısı olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.	148
49. Dışarıdan iki dik kenarı girilen üçgenin hipotenüsünü hesaplayan programın algoritma ve akış diyagramını çiziniz.	152
50. Dışarıdan iki kenarı ve aradaki açısı girilen üçgenin alanını hesaplayan programın algoritma ve akış diyagramını çiziniz.	154
51. Dışarıdan yarıçapı girilen kürenin alanını ve hacmini hesaplayan programın algoritma akış diyagramını çiziniz.	156
52. Girilen bir tam sayının hanelerindeki en büyük sayıyı bulan algoritma ve akış diyagramını çiziniz.	158
53. Ekrandan girilen bir sayı eğer 5-10 arasında ise girilen sayının karesini alıp gösteren, eğer 5'ten küçük ise faktöriyelini alan, 10'dan büyük ise sayıyı ikiye bölüp bir eksiğini yazan programın algoritma ve akış diyagramını çiziniz.	161
54. Dışarıdan 'Derece' cinsinden girilen açıyı ; 'Radyan' ve 'Grad' cinsine çeviren programın algoritma ve akış diyagramını çiziniz.	164
55. Arka arkaya girilen rastgele 10 tam sayının ortalaması ile bu sayılardan en büyük ve en küçük olanın ortalamasını bularak elde edilen bu iki ortalamanın farkını bulan programın algoritma ve akış diyagramını çiziniz.	166
56. 1 k sayısı tek ise 3 ile çarpılıp 1 ekleniyor çift ise 2 ile bölünüyor işlem k sayısı 1 olana kadar devam ediyor bu işlemin kaç adım sürdüğünü, işlem sırasında k sayısının aldığı max değeri k sayısının hangi sayıdan sonra hep çift olarak 1'e ulaştığını bulan programın algoritma ve akış diyagramını çiziniz.	169
Bölüm 5	173

Seri Algoritma Soru Çözümleri	173
57. e^x fonksiyonunun seriye açılımı aşağıdadır. Buna göre; dışarıdan girilen x ve N değerine göre; e^x 'i hesaplayan programın algoritma ve akış diyagramını çiziniz.	174
58. x ve n değişkenine göre yandaki işlemi yapan programın algoritmayı ve akış diyagramını çiziniz.	177
59. $1-1/3+1/5-1/7+1/9-1/11+...$ serisinin n tane terim için toplamını hesaplayan programın algoritma ve akış diyagramını çiziniz.	180
60. $\cos(x)$ fonksiyonu seriye aşağıdaki gibi açılmaktadır. Buna göre dışarıdan girilen x değerinin cosinüs'ünü hesaplayan programın algoritmasını ve akış diyagramını çiziniz.	182
61. Aşağıda verilen işlemin sonucunu girilen değer için hesaplayan programın algoritma ve akış diyagramını çiziniz.	185
Bölüm 6	188
Dizi Algoritma Soru Çözümleri	188
Dizi Kullanımı ve Mantiğı	189
62. 10 elemanlı bir sayı dizisini girişini yapan algoritma ve akış diyagramını çiziniz.	191
63. Fibonacci serisinin ilk 10 terimini dizi kullanarak bulan programın algoritma ve akış diyagramı çiziniz.	193
64. Girilen bir kelimenin uzunluğunu bulan programın algoritma ve akış diyagramını çiziniz.	195
65. Bir sayı dizisinin en büyük elemanını bulan programın algoritma ve akış diyagramı çiziniz.	197
66. Girilen kelimeyi tersten yazdıran programın algoritmasını ve akış diyagramını çiziniz.	199
67. Bir decimal sayıyı binary (10'luk-2'lik) sayıya çeviren programın algoritmasını ve akış diyagramını çiziniz.	201
68. 10 elemanlı bir sayı dizisinde en küçük elemanın bu dizinin kaçınıcı elemanı olduğunu bulan programın algoritma ve akış diyagramını çiziniz.	204
69. N elemanlı bir dizi bulunmaktadır. Klavyeden girilen sayılar diziye, bir tane baştan bir tane sondan olmak üzere yerleştirilmektedir. Örneğin ilk sayı birinci elemana, ikinci sayı N'inci elemana, üçüncü sayı ikinci elemana, dördüncü sayı N-1'inci elemana... şeklinde yerleştirilmektedir. N tane sayıyı klavyeden okuyup diziye yerleştiren ve diziyi ekranda görüntüleyen programın algoritma ve akış diyagramını çiziniz.	207
70. Aşağıdaki Çıktıyı Veren Programın Algoritma ve Akış Diyagramını Çiziniz.	210

71. Bir sınıftaki 50 öğrencinin bir dersten aldıkları yıl sonu notları veriliyor. Başarı notu 50 olduğuna göre kaç öğrencinin başarılı olduğunu bulan programın algoritma ve akış diyagramını çiziniz.	213
72. 10 elemanlı bir sayı dizisinin ortalaması tam sayı ise bu sayıdan dizide kaç tane olduğunu veren programın algoritma ve akış diyagramını çiziniz.	216
73. İstenildiği kadar elemandan oluşan bir sayı dizisinde negatif ve pozitif elemanların sayısını bulan algoritma ve akış diyagramını çiziniz.	219
74. 10 elemanlı bir dizinin elemanlarından hem 4'e hemde 5'e bölünen sayıları bulan programın algoritma ve akış diyagramı çiziniz.	223
75. Girilecek 10 adet sayıyı bir diziye aktararak bu dizideki sayıların ortalamasını bulduran programın algoritmasını ve akış diyagramını çiziniz.	225
76. Girilen cümlede, girilen karakterden kaç tane olduğunu bulan programın algoritması ve akış diyagramını çiziniz.	227
77. Tersinden ve düzünden okunuşu aynı olan yazılara polindrom denir. Bir yazının polindrom olup olmadığını bulan programın algoritmayı ve akış diyagramını çiziniz.	230
78. 5 elemanlı dizinin elemanlarının toplamlarını bulan algoritma ve akış diyagramını çiziniz.	233
79. Bir dizide dışarıdan girilen bir sayının, dizinin elemanlarından kaç tanesinden küçük olduğunu bulan programın algoritma ve akış diyagramını çiziniz.	235
80. Bir dizide dizi elemanlarının sondan başa gelecek şekilde düzenlenmesini sağlayan algoritma ve akış diyagramının çiziniz.	238
81. İkili bir düzende aynı basamaklı iki sayı verildiğinde bunların toplamını yapan programın algoritma ve akış diyagramını çiziniz.	241
82. 10 elemanlı bir sayı dizisinin en büyük ve en küçük elemanlarını ve yerini bulan programın algoritma ve akış diyagramını çiziniz.	244
83. Bir dizinin ikinci büyük elemanını bulan programın algoritma ve akış diyagramını çiziniz.	247
84. Eleman değerleri verilmiş 7 elemanlı bir sayı dizisinde tekrarlanan sayıların ilk yazılanı dışında kalanları kaldırarak başa doğru öteleyen programın algoritma ve akış diyagramını çiziniz.	250
85. Klavyeden girilen maksimum 20 karakterli kelimedeki sesli harflerin kelimenin toplam karakter sayısına göre yüzde oranını hesaplayan programın algoritma ve akış diyagramını çiziniz.	253
86. Tam sayılardan oluşan bir dizi veriliyor, bu dizi elemanlarından kaç tanesinin bir basamaklı, kaç tanesinin iki basamaklı, kaç tanesinin de üç basamaklı olduğunu bulan programın algoritma ve akış diyagramını çiziniz.	257

Açıklamalı Algoritma Soruları ve Çözümleri

87. 50 elemanlı bir dizinin 1. ve 2. elemanları toplamının yarısı başka bir dizinin ilk elemanı, 3. ve 4. elemanları toplamının yarısı 2. elemanı şeklindedir. Bu işlemi yapan programın algoritma ve akış diyagramını çiziniz.	261
88. İki boyutlu olarak oluşturulan matrise $matris[i,j]$ dışarıdan değer girilen programın algoritma ve akış diyagramı çiziniz.	264
89. $[2 \times 2]$ tipindeki bir kare matrisin transpozisini veren algoritma ve akış diyagramını çiziniz.	267
90. İki boyutlu bir diziyi , tek boyutluya çeviren programın algoritma ve akış diyagramını çiziniz.	270
91. İki kare $[3 \times 3]$ matrisin toplamını yapan programın algoritma ve akış diyagramını çiziniz.	273
92. 5 adet öğrencinin numarası, vize, final notunu tutup matriste diğer sütuna atan ve listeleyen programın algoritma ve akış diyagramını çiziniz.	277
Bölüm 6	280
Arama ve Sıralama Algoritmaları Soru ve Çözümleri.....	280
93. Sıralı arama (sequential search) algoritması ile girilen bir sayıyı dizideki yerini bulan programın algoritma ve akış diyagramını çiziniz.	281
94. N elemanlı bir dizide ikili (binary) arama algoritması ile girilen bir sayıyı arayan algoritma ve akış diyagramını çiziniz.	284
95. Bubble (kabarcık) sıralama algoritması ile bir dizinin sıralanması programının algoritma ve akış diyagramını çiziniz.	287
96. Selection sort (seçme sıralama) algoritması ile bir dizinin sıralanması programının algoritma ve akış diyagramını çiziniz.	290
97. Insertion Sort (ekleme sıralaması) algoritmasını kullanarak sıralama yapan programın algoritma ve akış diyagramını çiziniz.	293
98. Shell sort (kabuk sıralaması) algoritmasını yapan programın algoritma ve akış diyagramını çiziniz.	296
99. Quick Sort (hızlı sıralama) algoritmasını yapan programın algoritma ve akış diyagramını çiziniz.	299
100. Merge Sort (birleşme sıralaması) algoritmasını yapan programın algoritma ve akış diyagramını çiziniz.	304
KAYNAKLAR	316
Kitaplar	316
Web Adresleri.....	316

Bölüm 1

Programlama ve Programlama Süreci

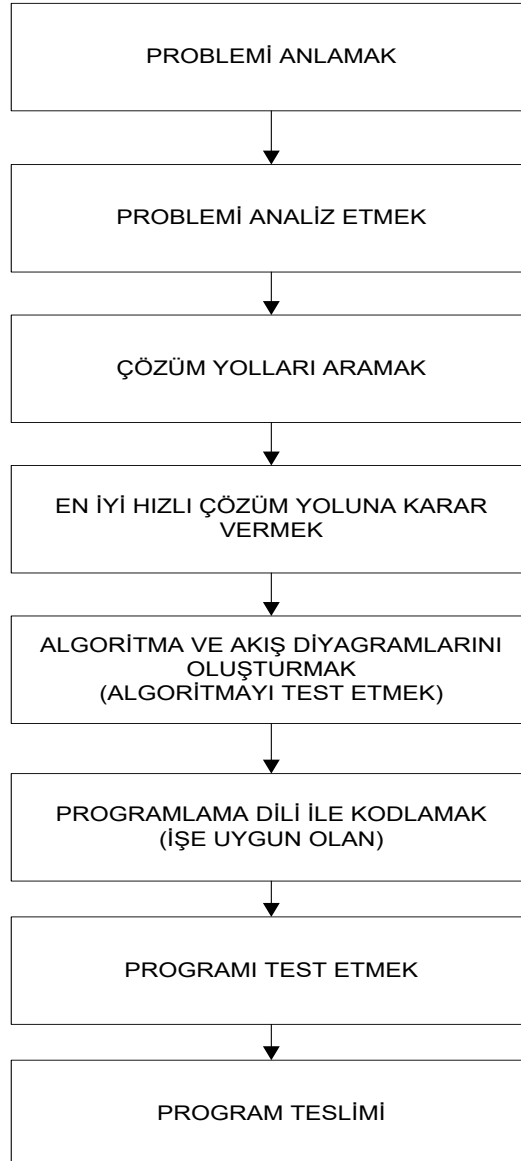
Programlama

Herhangi bir problemin bir programlama dili kullanılarak çözülmesi için yazılan kod satırlarına *programlama* denir. Programlama, bir düşünce sanatıdır ve kullanılacak programlama dilinden bağımsız bir düşünme tekniğidir. Kullanılan programlama dilinin bu işleme katkısı ise makine dilinde çok komutla yapılabilecek işlemleri, tek komutta yapma imkanı vermesidir. Ancak, yapılacak işe göre kullanacağınız programlama dilinde, hangi komutları hangi sıra ve formatta yazacağınızı bilmek, daha doğrusu düşünebilmek programlama sanatıdır. Programlamayı öğrenmeyi hedefleyenler program komutlarını iyi bilmeli ve programlama konusunda meraklı olmalıdırlar. Ne kadar çok uygulama yapabilirse o derece programcının aktifliği ve becerisi artar. Bir programı yazarken geçilmesi gereken belirli süreçler vardır. Bu sebeple öncelikle programlama süreci hakkında bilgi vermenin doğru olacağını düşünüyoruz.

Programlama Süreci

Programlama süreci, bir programı oluştururken geçilen evrelerin tümüne verilen addır. Bilgisayar programı yazma süreci bir çok adımdan oluşur. Problemin analizinden programın dağıtım aşamasına kadar tek tek ele alınması gereken bu adımların güçlü ve düzgün bir şekilde oluşturulması programın sağlamlığını ve esnekliğini arttırır. Bir programı yazmadan önce bu programın yazılmasını isteyen firma ya da kişinin ne istediğinin tam olarak anlaşılması gerekir. Problem analiz edilmeli, nasıl çözülebilir diye düşünülmeli ve çözüm için tek bir yol değil alternatif yollar da aranmalıdır. En hızlı ve sağlam çözüm yani ileride veri artışı ya da güncellemelere açık çözümün seçilmesine dikkat edilmelidir. Bu işlemlerden sonra algoritma ve akış diyagramı tasarlanmalıdır. Algoritma test edilmeli, istenilen programa uygun olacak şekilde veri tabanı programı, matematik programı ve güvenlik gibi konulara göre bir programlama dili seçilmelidir. Yapılan program da test edilmeli ve sonunda dağıtım yapılmalıdır. Öğrencilerin programlama süreciyle ilgili şekil ve akış diyagramlarını daha etkin olarak öğrenebilmeleri için bu süreçlerin şemayla da gösterilmesinin uygun olacağını düşünüyoruz.

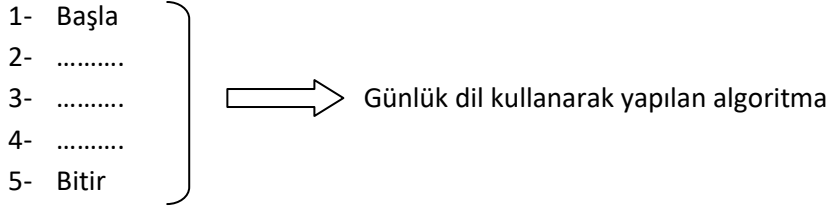
PROGRAM YAZMA SÜRECİ



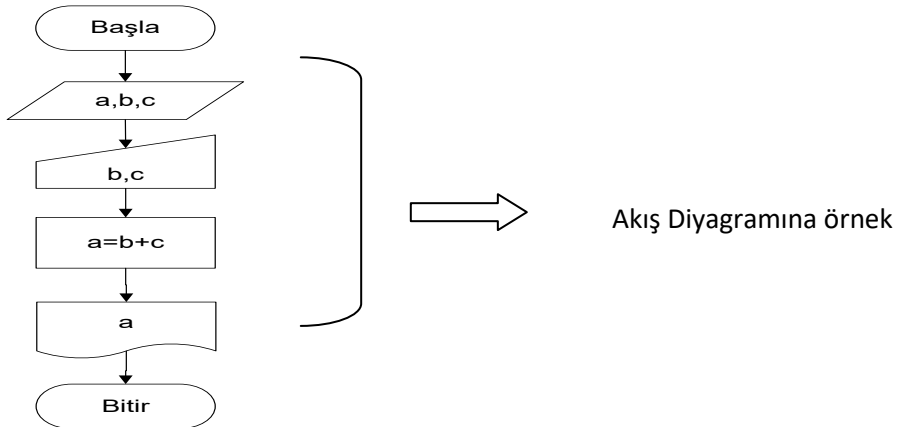
Bölüm 2

Algoritma ve Akış Diyagramları

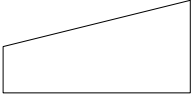
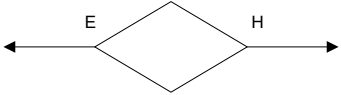
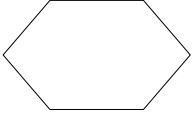
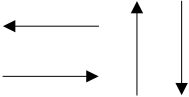
Algoritma , bir programı yazmadan önce izlenmesi gereken yol, rota olarak da kabul edilir. Algoritma sözcüğü, 9. yüzyılda yaşamış ünlü matematikçi **el Harezmi**'nin adından gelmektedir. Cebir ve algoritmalarla ilgili dünyanın ilk kitabını yazan el Harezmi (Al-Khwārizmī)'nin adı, Batılılar tarafından **algorizma** şeklinde telaffuz edilmiş ve daha sonra bu kavrama isim olarak verilmiştir. Bir algoritma, içinde bulunduğumuz **başlangıç** durumundan hedefimiz olan **bitiş** durumuna ulaşmamız için kullanılır. Algorizmanın bize ne yapılması gerektiğini **adım adım** ve **açık** bir biçimde anlatması gerekir ve çeşitli yöntemlerle ifade edilebilir. Bu yöntemler iki grupta incelenebilir: Birincisi yapılacak işlemin adımlar halinde, **gündelik dil** kullanılarak ifade edildiği yöntemdir. İkincisi, grafiksel olarak ifade edilen yöntemdir. Görsel olarak anlamayı kolaylaştırmayı sağlayan bu grafiklere **akış diyagramı** ya da **akış şeması** adı verilir. Gündelik dil kullanılarak bir problemi çözerken analiz safhasından sonra numaralanmış satırlar halinde işlemler oluşturulur. Aşağıda kısa bir başlangıç örneği verilmiştir:



Yukarıdaki örnekte 5 adımlık bir algoritma dizilişi gösterdik. Her algoritma Başla ile başlar , Bitir ile sonlanır. Ara adımlarda (bu örnekte 2,3,4) işlemler, döngüler, şart ifadeleri yürütülür. Algoritma dizilimi öğrenciler için sıkıcı ve anlaşılması zor olan bir yapı olarak görülebilir. Çünkü yapılan hata kontrollerinde program dallanmaları görülmeyebilir ya da karıştırılabilir. Bunun için akış diyagramı daha kolaydır. Fakat unutulmamalıdır ki akış diyagramını çizerken gündelik dil kullanarak oluşturduğumuz algoritma esas alınır.



Akış diyagramında kullanılan Semboller

	Başla Bitir
	Değişken Tanımlama
	Veri Giriş İşlemleri
	İşlem
	Eğer (karar)
	Döngü
	Yazdır , Ekran Çıktısı
	Akış Yönü
	Bağlantı

Bölüm 3

Operatörler

Operatörler

Operatörler, değişkenler veya sabitler üzerinde matematiksel ve karşılaştırma işlemleri yapan simgelerdir. Bu kitapta hem problem çözümlerinde kullanılması hem de ileride programlama dillerinin desteklediği operatörleri öğrenmenin kolay olabilmesi için operatörlere yer verilmiştir.

Aritmetik Operatörler

Değişken veya sabitler üzerinde temel aritmetik işlemleri gerçekleyen operatörlerdir.

+	Toplama
-	Çıkarma
*	Çarpma
/	Bölme
%	Artık Bölme(Mod)

Atama Operatörleri

Bu operatörler bir değişkene , bir sabit eşitlemek için kullanılır. Değişkene atama işlemi önemli bir konu olup öğrencilerin algoritma yazarken hataya düştükleri bir noktadır.

Örnek:

$X=3$ *Burada x değişkenine 3 değerini atadık. 3 burda değişkenin içindeki veridir.*

$3=X$ *Bu yanlış bir ifadedir. Çünkü veriye bir değişken atayamayız.*

Genel yapı aşağıdaki gibidir.Bazen operatörler birleşik atama işlemlerinde de kullanılır.

değişken = değişken [operatör] ifade;

şeklinde ise, daha kısa bir biçimde

değişken [operatör]= ifade;

olarak yazılabilir.

=	Atama
+=	Ekleyerek atama
-=	Eksilterek atama
*=	Çarparak atama
/=	Bölerek atama
%=	Bölüp kalanı atama
++	Bir arttırma
--	Bir azaltma

Karşılaştırma Operatörleri ve Mantıksal Operatörler

Sayısal değerleri veya karakterleri karşılaştırmak için kullanılır. Bu ifadeler karar (if) mekanizmalarında işimize çok yarayan ifadelerdir.

>	Büyüktür
<	Küçüktür
==	Eşittir
>=	Büyük-Eşittir
<=	Küçük-Eşittir
!=	Eşit değil
&&	Mantıksal VE
	Mantıksal VEYA

Bölüm 4

Sayı Algoritma Soru Çözümleri

Bu bölümde kolaydan zora doğru birden çok algoritma çözülmüştür. Amacımız iyi derecede problem çözme ve matematik zekasına sahip olanların konuyla ilgili bol alıştırma yapmaları ve algoritma çözerken nasıl bir mantığın takip edilmesi gerektiğini kavramalarıdır. Ezberleme sisteminin doğru olmayacağı düşüncesiyle bu kitaptaki örnekleri ezberlemek yerine soruları çözerken , sorunun çözümüne nasıl yaklaşıldığını, hangi metotların geliştirildiğini anlayarak ileride yazılacak programların algoritma safhalarını geliştirmenin daha doğru olacağını düşünüyoruz. Performansın arttırılabilmesi açısından algoritma çözümlerinde dikkatinizi soruya ve çözüme odaklamalı, bunun için de sessiz bir ortamda çalışmalısınız. Akış diyagramlarını ve şekilleri düzgün bir şekilde çizmek, çalışma esnasında geriye dönüşlerde, yapılan işlemleri hatırlamak için açıklama satırları koymak konunun daha kolay anlaşılmasını sağlayacağından önemlidir. Biz, bu kitapta kullandığımız örneği zihninizden verdiğimiz değerlerle test edeceğiz. Sizlerin de yaptığınız problemlerde test aşamasını unutmamanızı önemle tavsiye ederiz. Şimdiden iyi çalışmalar...

1. Çay demlemenin algoritma ve akış diyagramını çiziniz.

Algoritma:

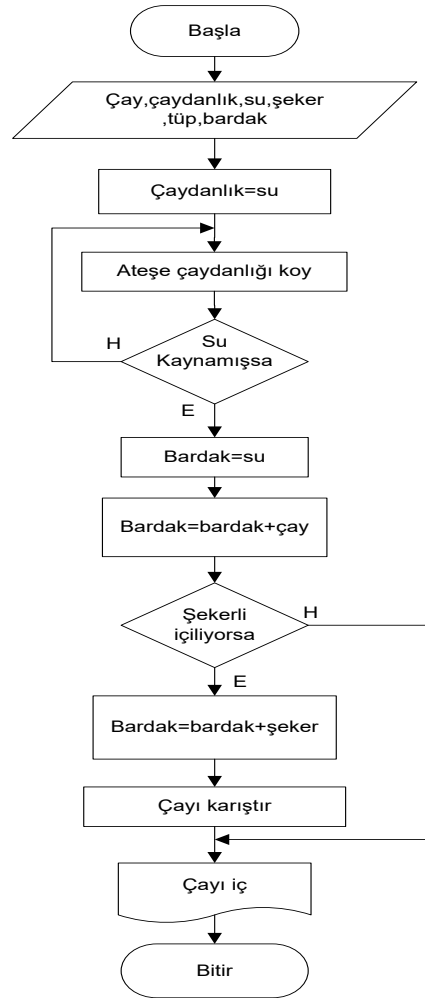
- 1- Başla
- 2- Çay,çaydanlık,su,şeker,tüp,bardak
- 3- Çaydanlığa suyu koy, (çaydanlık=su)
- 4- Çaydanlığı ateşe koy
- 5- Eğer su kaynamışsa bardağa su koy (bardak=su)
- 6- Bardağa çayı koy (bardak=bardak+çay)
- 7- Eğer şekerli içiliyorsa bardağa şeker koy (bardak=bardak+şeker)
- 8- Çayı karıştır
- 9- Çayı iç
- 10- Bitir

Açıklama:

Yaşamın her yerinde problem çözme olduğu düşüncesiyle bu soruyu özellikle ilk soru olarak koyma ihtiyacı duyduk. Algoritma da problem çözme adımlarını oluşturduğu için hayatımızın her anında algoritma vardır. Çay yapmanın birden çok yöntemi olabileceği gibi problem çözerken de karşımıza birden çok çözüm çıkabilir. Hatta bunların hepsi doğru da olabilir. Amaç böyle durumlarda az kaynakla çok iş yapmaktır. Algoritmanın hızının yüksek olması programın hızının da yüksek olmasını sağlar. Bunlara dikkat etmeliyiz. Bardak=su

ifadesi, bardak değişkenine suyu koy anlamına gelir. Burada bardak değişkeni , su ise veridir.

Akış diyagramı :



2. İki sayının toplamını veren programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- a,b,c
- 3- b,c gir
- 4- $a=b+c$
- 5- Yazdır a
- 6- Bitir

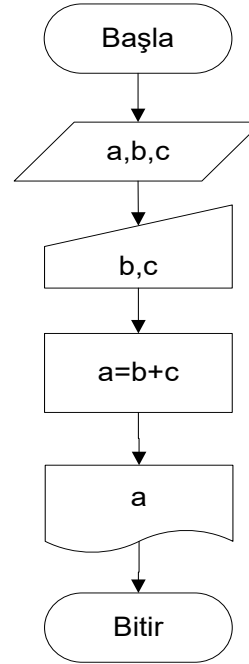
Algoritma Testi :

Adım	A	B	C
1	0	5	3
Sonuç	8	5	3

Açıklama:

Bu soru basit olmasına rağmen hem değişken kavramına giriş yapmak hem de yapılacak iki örnek arasındaki farkı gösterebilmek açısından önemlidir. İlk örnekte üç değişken kullanılarak soru çözülecek; ikinci örnekte ise aynı soru iki değişken kullanılarak çözülecektir. Böylece algoritma soruları çözüldükçe yeni kavramlar gelişecek, program bilgisayar hafızasından daha kısa ve daha az yer alacak, az kaynakla çok iş yapılacaktır.

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int a,b,c;
    printf("1.sayıyı giriniz");
    scanf("%d",&b);
    printf("2.sayıyı giriniz");
    scanf("%d",&c);
    a=b+c;
    printf("toplam :%d ",a);
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int a, b, c;
            Console.Write("1.Sayıyı Giriniz = ");
            b = Convert.ToInt32(Console.ReadLine());
            Console.Write("2.Sayıyı Giriniz = ");
            c = Convert.ToInt32(Console.ReadLine());
            a = b + c;
            Console.Write("Toplam = " + a);
            Console.ReadLine();
        }
    }
}
```

Algoritma:

- 1- Başla
- 2- a,b
- 3- a, b gir
- 4- $a=a+b$
- 5- Yazdır a
- 6- Bitir

Açıklama:

Bir önceki soruda da bahsedildiği gibi üç adet değişken kullanarak yapılan diğer örnek yerine bu çözümde iki adet değişken kullandık. Böylece hafızadan daha az yer alarak işlemimizi yaptık. Burada aslında önemli nokta soruyu çözerken birden fazla çözüm yolunun olabilmesidir. Hepsi doğru olmasına rağmen algoritma analizi yapıldığında az değişken ya da az değer (if) kullanarak işlemimizi yapmalıyız. Ancak öğrenciler öncelikle soruları, alıştırmaları çözmelidirler.

3 x 4 byte (integer)= 12

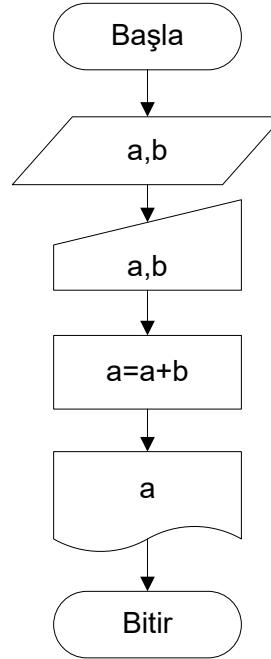
2 x 4 byte (integer)= 8

4 byte kazancımız vardır.

Algoritma Testi :

Adım	A	B
1	7	5
Sonuç	12	5

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

int main()
{
    int a,b;
    printf("1.sayıyı giriniz");
    scanf("%d",&a);
    printf("2.sayıyı giriniz");
    scanf("%d",&b);
    a=a+b;
    printf("toplam :%d ",a);
    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int a, b;
            Console.Write("1.Sayıyı Giriniz = ");
            b = Convert.ToInt32(Console.ReadLine());
            Console.Write("2.Sayıyı Giriniz = ");
            c = Convert.ToInt32(Console.ReadLine());
            a = a + b;
            Console.Write("Toplam = " + a);
            Console.ReadLine();
        }
    }
}
```

3. Kullanıcının girdiği iki sayının karelerinin toplamını görüntüleyen programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- sayi1,sayi2,top
- 3- sayi1,sayi2 gir
- 4- $top=(sayi1^2)+(sayi2^2)$
- 5- **yazdır top**
- 6- Bitir

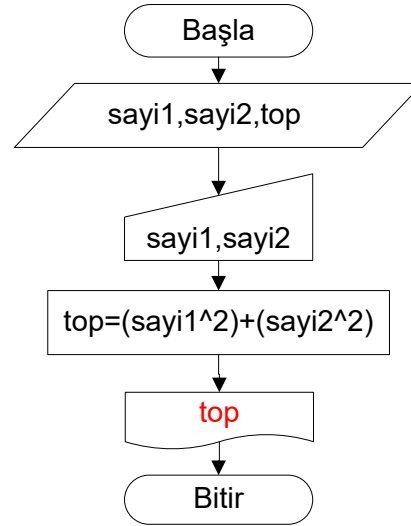
Açıklama:

Bu soru algoritma açısından önemli olduğu kadar programlama dili komutları açısından da önemlidir. Soruda İki sayı kullanıcıdan isteniyor. Bunların kareleri toplamı **top** değişkenin içine atılıyor. Burada bilinmesi gereken $\wedge$ işareti programlama dillerinin bazılarında üs anlamına gelmektedir. Burada algoritma çözüyor olduğumuzu da göz önüne alarak sayının karesinin de yazılabileceğini ve günlük dil kullanılabileceğini unutmamalıyız. Ancak daha kısa olması açısından biz üs işaretini ($\wedge$) kullanıyoruz. Bazı programlama dillerinde üs alma fonksiyonları vardır. Bunun için gerekli header(ör: math.h) dosyasını programınıza eklemeniz gerekir. Örnek: $\text{pow}(3,2) \rightarrow 9$ dur. Şimdi yeni sorularla devam edelim.

Algoritma Testi :

Adım	Sayi1	Sayi2	top
1	3	5	0
Sonuç	3^2	5^2	34

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

int main()

{
    int sayi1,sayi2,top=0;
    scanf("%d%d",&sayi1,&sayi2);
    top=sayi1^2+sayi2^2;
    printf("%d",top);
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi1, sayi2;
            int toplam = 0;
            Console.Write("1.Sayıyı Giriniz = ");
            sayi1=Convert.ToInt32(Console.ReadLine());
            Console.Write("2.Sayıyı Giriniz = ");
            sayi2=Convert.ToInt32(Console.ReadLine());
            toplam = (sayi1 * sayi1) + (sayi2 * sayi2);
            Console.Write("Toplam = " + toplam);
            Console.ReadLine();
        }
    }
}
```

4. 1'den 100'e kadar olan sayılarının küplerinin toplamını bulan programın algoritma ve akış diyagramını çiziniz

Algoritma:

- 1- Başla
- 2- $sayi=1, toplam=0$
- 3- $toplam=toplam+(sayi^3)$
- 4- $sayi++$
- 5- Eğer $sayi=101$ ise devam et,
değilse 3'e git
- 6- Yazdır toplam
- 7- Bitir

sebebi de 4.adımda sayacı baştan arttırmamızdır.

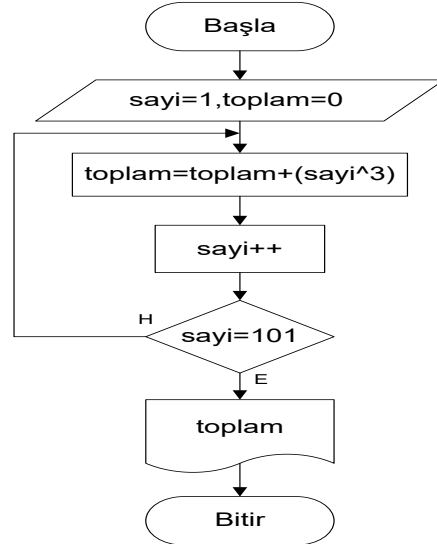
Algoritma Testi:

Adım	S1	Toplam
1	1	1
2	2	9
3	3	36
...	...	...
Sonuç	100	26532801

Açıklama:

İlk defa bu soruda döngü ve sayaç kullanıyoruz. Eğer çözeceğiniz soru 1...100 arası gibi bir ifade ile başlıyorsa bu programdaki işlemler sayaç 100 olana kadar işlem döndürülecek anlamına gelmektedir. Yani döngü kurulacaktır. Bu soruda **$toplam=toplam+(sayi^3)$** işlemi 100 defa yapılacak, sayaç olarak da **sayi** değişkeni seçilecektir. Bu soruda sayı değişkeni seçilmiştir fakat farklı sorularda farklı sayaç değerleri seçilebilir. Döngü işlemini yapan kısım, 5. adımdır. Şartlar uymadığı takdirde **sayi=101** olmadığı sürece 3. adıma dallanma yapılıyor. Bu durum da programı bir döngüye sokuyor. Bir de burada dikkati çeken **sayi=101** şartıdır. Neden 100 değil de 101'dir? Bunun

Akış Diyagramı :



C Kod:

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int sayi,top=0;
    for(sayi=1;sayi<=101;i++)
        top=top+(sayi*sayi*sayi);
    printf("1-100 arasi toplam :%d \n",top);
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i;
            int toplam = 0;
            for (i = 1; i <= 101; i++)
            {
                toplam = toplam + (i * i * i);
            }
            Console.WriteLine("1-100 arası toplam = " + toplam);
            Console.ReadLine();
        }
    }
}
```

5. Doğum tarihi girilen kişinin yaşını hesaplayan programın algoritma ve akış diyagramını çiziniz

Algoritma:

- 1- Başla
- 2- dyili, yıl, yas
- 3- dyili,yıl gir
- 4- $yas=yıl-dyili$
- 5- Yazdır yas
- 6- Bitir

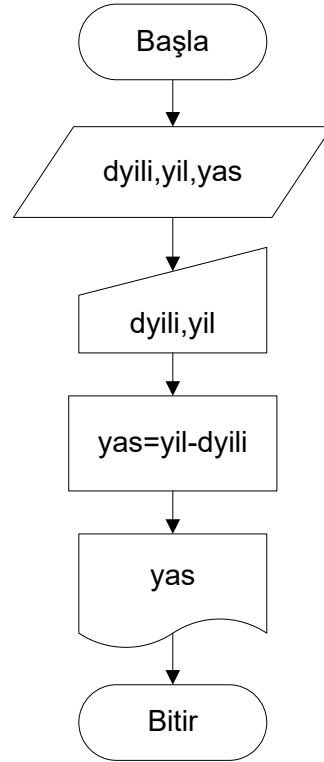
Açıklama:

Başlangıç düzey için kitaba konulmuş bir soru.Bu soruda da değişken kavramını açıklayalım. Değişken olarak bu soruda **dyili**, **yıl**, **yas** olmak üzere üç değişken aldık. Değişkenler, içinde tipine göre değer taşıyan ve bellekte yer tutan yapılardır. **Yıl** değişkeni sistem tarihi olarak da alınabilir.

Algoritma Testi :

Adım	dyili	Yıl	Yas
1	1978	2008	0
Sonuç	1978	2008	30

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int dyili,yil,yas;
    printf("bugünün yılını giriniz :");
    scanf("%d",&yil);
    printf("dogum yılını giriniz :");
    scanf("%d",&dyili);
    yas=yil-dyili;
    printf("yasiniz :%d ",yas);
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int dyili, yas;
            DateTime yil;
            yil = DateTime.Now;
            Console.Write("Doğum yılını giriniz = ");
            dyili = Convert.ToInt32(Console.ReadLine());
            yas = yil.Date.Year - dyili;
            Console.Write("Yaş = " + yas);
            Console.ReadLine();
        }
    }
}
```

6. Girilen sayının faktöriyelini bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- sayi , fak=1
- 3- sayi gir
- 4- fak=fak*sayi
- 5- sayi --
- 6- Eğer sayi=1 ise devam et ,
değilse 4'e git
- 7- Yazdır fak
- 8- Bitir

döngü içerisinde çalışması gerekir. Fak=Fak*i formülü mutlaka bilinmelidir.

Algoritma Testi:

Adım	Sayi	fak
1	5	1
2	4	20
3	3	60
4	2	120
Sonuç	1	120

Açıklama:

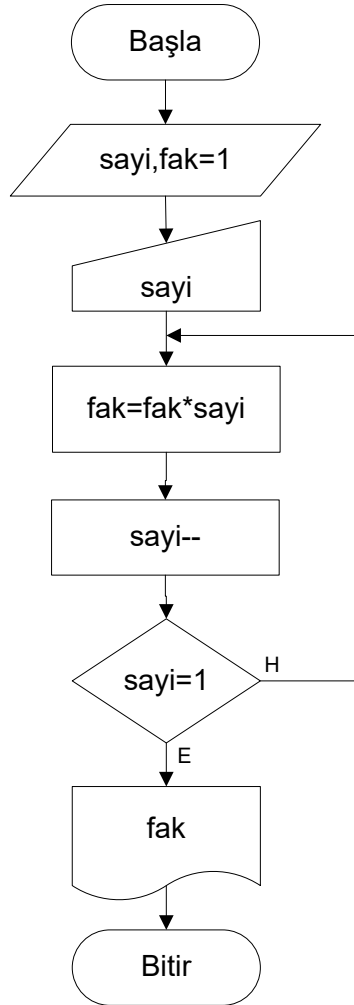
Bu örnekte faktöriyel işleminin algoritmasını yaptık. Burada döngü kavramı karşımıza çıktı. Program **sayi=1** olana kadar devamlı dallanacak, şartımız doğru olduğunda programı bitirecektir. Mesela C Programlama dili For döngüsüne çok uygundur. Bu soruda kullandığımız yeni operatör ve formülleri tanıyalım.

(=) → Eşit mi ? (--) Bir eksiltme

Faktöriyel Örnek → $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$

Faktöriyel Sorusu döngüleri anlatmak için en çok kullanılan örneklerdendir. Tüm kitap ve dokümanlarda bulunur ve bulunmalıdır. Bu programın girilecek faktöriyel sayısı kadar

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int i,sayi,fak=1;
    printf("bir sayı giriniz : ");
    scanf("%d",&sayi);
    for(i=sayi;i>1;i--)
        fak=fak*i;
    printf("Faktoriyel : %d",fak);
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i, sayi;
            int fak = 1;
            Console.Write("Sayıyı giriniz = ");
            sayi = Convert.ToInt32(Console.ReadLine());
            for (i = sayi; i > 1; i--)
            {
                fak = fak * i;
            }
            Console.Write("Faktoriyel = " + fak);
            Console.ReadLine();
        }
    }
}
```

7. Çarpma işlemini toplama kullanarak bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başlat
- 2- $sayi1, sayi2, sayac=0$
- 3- $sayi1, sayi2$ gir
- 4- Eğer $sayi2 > 0$ ise
 $sayac = sayac + sayi1$, $sayi2--$, 4'e git,
değilse devam et
- 5- Yazdır $sayac$
- 6- Bitir

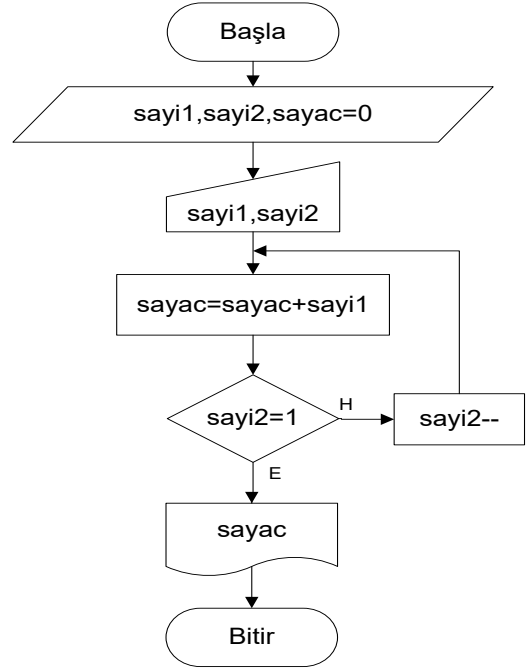
Açıklama:

Bu soruda çarpma işlemi kullanmadan çarpma işlemini toplama ile yapacağız. Yapmamız gereken, iki sayı girdiğimizde ikisinden birini döngü için kullanmaktır. Yani **$sayi2=3$** ise, program 3 defa dönecek anlamına gelir. Döngü, her dönmesinde sayaca **$sayi1$** eklenir (**$sayac = sayac + sayi1$**). Başlangıç değeri **$sayac=0$** vermemizin amacı da 3. Adımda **$sayac = sayac + sayi1$** $sayac$ değerinin belli olması gerektiridir. Yoksa program bu ifadenin değerini yorumlayamaz.

Algoritma Testi:

Adım	Sayi1	Sayi2	sayac
Baş.Değ.	5	3	0
1	5	3	5
2	5	2	10
Sonuç	5	1	15

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    int sayi1,sayi2,sayac=0;

    scanf("%d",&sayi1);

    scanf("%d",&sayi2);

    while(sayi2>0)
    {
        sayac=sayac+sayi1;

        sayi2 --;

    }

    printf("%d",sayac);

    getch();

}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi1, sayi2;

            int sayac = 0;

            Console.Write("1.Sayıyı giriniz = ");

            sayi1 = Convert.ToInt32(Console.ReadLine());

            Console.Write("2.Sayıyı giriniz = ");

            sayi2 = Convert.ToInt32(Console.ReadLine());

            while (sayi2 > 0)
            {

                sayac = sayac + sayi1;

                sayi2 -= 1;

            }

            Console.Write("Sonuç = " + sayac);

            Console.ReadLine();

        }
    }
}
```

8. Bölme işlemini çıkarma kullanarak yapan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- bolunen,bolen,sayac=0
- 3- bolunen,bolen gir
- 4- bolunen=bolunen-bolen
- 5- Eğer bolunen<bolen ise 6'ya git,
değilse sayac++ , 4'e git
- 6- Yazdır sayac
- 7- Bitir

Açıklama:

Bu soruda bölme işlemi kullanmayıp, sadece çıkarma yaparak bölme işlemini yapacağız. **Bölünen**'den **bölen**'i çıkartıp, bölünen bölenden ufak olduğunda **bolunen<bolen** sayacı yazdırmak mantığıyla soru çözülecektir.

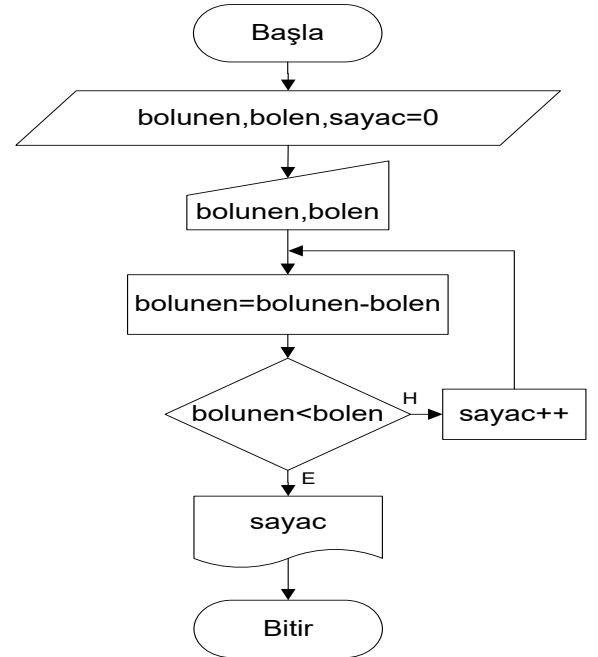
$12/3=4 \rightarrow$ Normal sonuç

$12-3=9$ yandaki işlem bölünen<bölen olana kadar tekrar eder.

Algoritma Testi:

Adım	Bolunen	Bolen	Sayac
Baş.Değ.	12	3	0
1	9	3	1
2	6	3	2
3	3	3	3
Sonuç	1	3	4

Akış Diyagramı:



C Kod

```
#include <stdio.h>
#include <conio.h>

int main()
{
    int bolunen,bolen,sayac=0;

    printf("Bolunecek sayıyı giriniz : ");

    scanf("%d",&bolunen);

    printf("Bölen sayıyı giriniz : ");

    scanf("%d",&bolen);

    for(sayac=1;bolunen>bolen;sayac++)

        bolunen=bolunen-bolen;

    printf("Bölme işleminin sonucu : %d",sayac);

    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int bolunen, bolen;

            int i = 0;

            Console.Write("Bölünecek sayıyı giriniz = ");
            bolunen = Convert.ToInt32(Console.ReadLine());

            Console.Write("Bölen sayıyı giriniz = ");
            bolen = Convert.ToInt32(Console.ReadLine());

            for (i = 1; bolunen > bolen; i++)
            {
                bolunen = bolunen - bolen;
            }

            Console.Write("Bölme işleminin sonucu = " + i);

            Console.ReadLine();
        }
    }
}
```

9. Girilen sayının istenilen sayıya göre mod işlemini yaptıran programın algoritma ve akış diyagramını çiziniz.

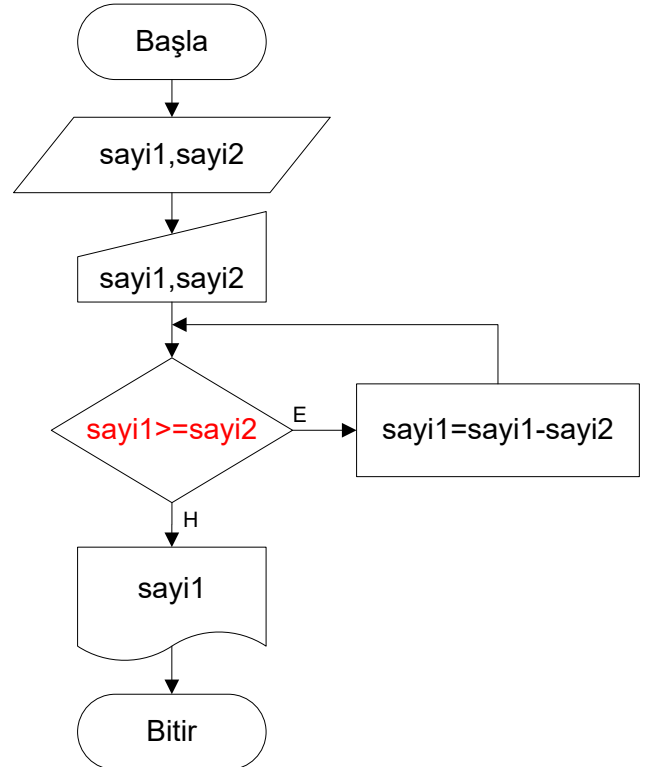
Algoritma:

- 1- Basla
- 2- sayi1,sayi2
- 3- sayi1, sayi2 gir
- 4- Eğer $\text{sayi1} \geq \text{sayi2}$ değilse 5'e git,
ise $\text{sayi1} = \text{sayi1} - \text{sayi2}$, 4'e git
- 5- Yazdır sayi1
- 6- Bitir

Algoritma Testi:

Adım	Sayı1	Sayı2
Baş.Değ.	7	3
1	4	3
Sonuç	1	3

Akış Diyagramı:



Açıklama:

Programlama dili ile kod yazdığımızda Mod fonksiyonunu kullanırız. Fakat algoritma alıştırmalarında öğrencilerin daha geniş düşünebilmelerini sağlamak için bu soruda mod işlemini mod fonksiyonu kullanmadan nasıl yapacağımızı gösterdik.

$$7 \text{ Mod } 3 = 1 \quad 7 \% 3 = 1$$

% işareti (Mod) içinde kullanılabilir.

İki sayı girdik. İlk sayı, ikinci sayıdan küçük olduğu sürece programımız döngü içinde olur. **sayi1 sayi2'**den küçük olduğunda (**sayi1<sayi2**) sonucumuz sayi1 olarak karşımıza çıkacaktır.

C Kod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    int sayi1,sayi2;

    printf("1.sayıyı giriniz");

    scanf("%d",&sayi1);

    printf("2.sayıyı giriniz");

    scanf("%d",&sayi2);

    while(sayi1>=sayi2)

        sayi1=sayi1-sayi2;

    printf("%d",sayi1);

    getch();
}
```

C# Kod:

```
using System;

namespace dmng
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi1, sayi2;

            int i = 0;

            Console.Write("1. sayıyı giriniz = ");

            sayi1 = Convert.ToInt32(Console.ReadLine());

            Console.Write("2. sayıyı giriniz = ");

            sayi2 = Convert.ToInt32(Console.ReadLine());

            while (sayi1 >= sayi2)
            {
                sayi1 = sayi1 - sayi2;
            }

            Console.Write("Sonuç = " + sayi1);

            Console.ReadLine();
        }
    }
}
```

10. Girilen sayının kaç basamaklı olduğunu söyleyen programın algoritma ve akış diyagramını çiziniz

Algoritma:

- 1- Başla
- 2- sayi, sayac=1
- 3- sayi gir
- 4- Eğer sayi≤9 ise 6'ya git,
değilse devam et
- 5- sayi=sayi/10,sayac++ 4'e git
- 6- Yazdır sayac
- 7- Bitir

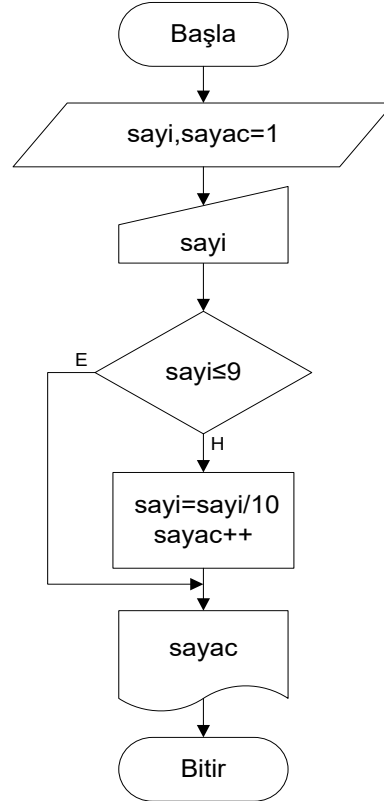
Açıklama:

Bu tür sorularda yani basamak sorularında, sayıyı devamlı 10'a bölerek soruyu çözebilirsiniz. Girilen sayı, her defasında 10'a bölünerek, tam kısmı 10'dan küçük oluncaya kadar program bir döngü vasıtasıyla devam eder. Şartımız 4. adımda sağlandığında, elimizdeki sayaç bize sayımızın kaç basamaklı olduğunu gösterir.

Algoritma Testi:

Adım	Sayı	Sayac
Baş.Değ.	111	1
1	11	2
2	1	3
Sonuç	1	3

Akış Diyagramı:



C Kod:

```
#include <conio.h>

#include <stdio.h>

int main()
{
    float sayi;

    int i=0;

    printf("Bir sayı giriniz \n");

    scanf("%f",&sayi);

    Do
    {
        sayi=sayi/10;

        i++;

    } while(sayi>=9)

    printf(" Sayı %d basamaklı",i);

    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            float sayi;

            int i = 1;

            Console.Write("Bir sayı giriniz = ");

            sayi = Convert.ToInt32(Console.ReadLine());

            while (sayi > 9)
            {
                sayi = sayi / 10;

                i++;

            }

            Console.Write("Sayı " + i + " basamaklı");

            Console.ReadLine();

        }
    }
}
```

11. Girilen 3 basamaklı bir sayının basamaklarının küpleri toplamı sayının kendine eşit olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

Şimdi örnek değer vererek algoritmayı test edelim.

- 1- Başla
- 2- x,sayi, **top=0**,bas
- 3- sayi gir
- 4- x=sayi
- 5- bas=sayi%10
- 6- top=top+(bas*bas*bas)
- 7- sayi=sayi/10
- 8- **Eğer sayi<10 ise**
top=top+(sayi^3),
değilse 4'e git
- 9- Eğer (x=sayi) ise yazdır "eşit",
değilse yazdır "eşit değil"
- 10- Bitir

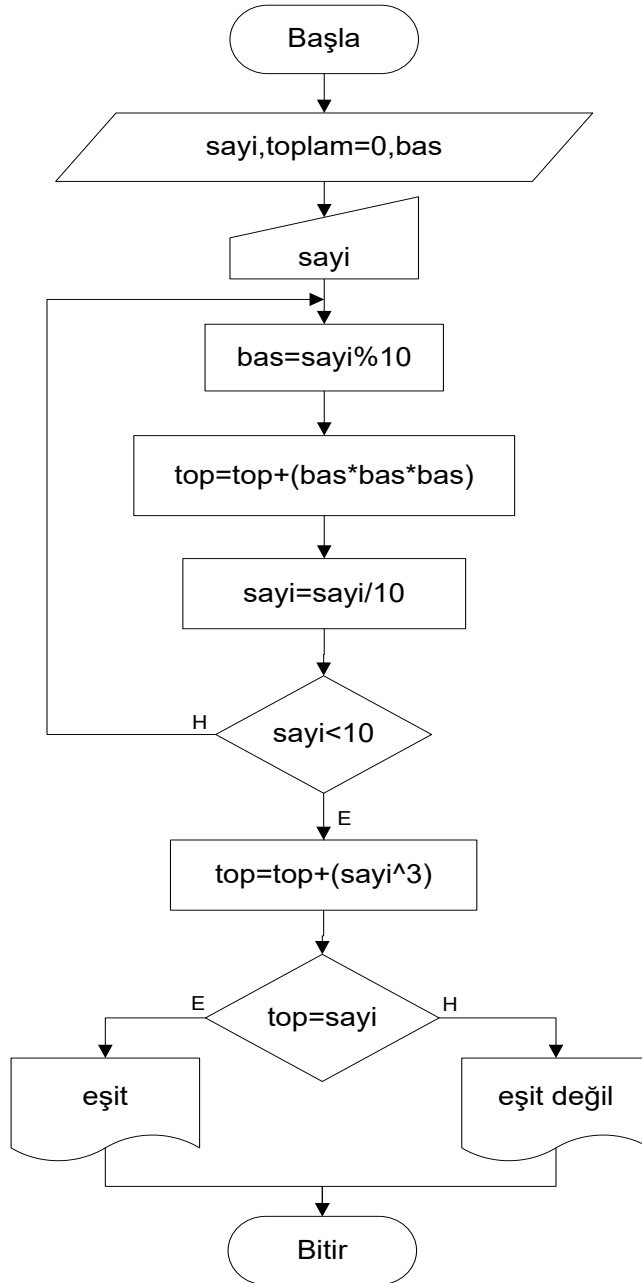
Algoritma Testi:

Adım	Sayı	Top	Bas	K
Baş. Değ.	101	0	1	101
1	101	1	1	101
2	10	1	0	101
3	1	2	1	101
Sonuç	Eşit Değildir			

Açıklama:

Bu soruda ilk önce yapmamız gereken basamak değerlerini bulmaktır. Bunun için sayıyı devamlı 10'a bölmeliyiz her zaman kullandığımız gibi % işareti bölme işleminde kalanı veriyor / işareti ise bölümün tam kısmını veriyor. Üs alma işareti olarak da ^ işaretini kullanıyoruz. Soruda döngü işlemleri ve şart işlemleri ön plana çıkıyor.

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

int main()
{
    int sayi,top=0,bas;

    printf("Bir sayı giriniz \n");

    scanf("%d",&sayi);

    dnz:

    bas=sayi%10;

    top=top+bas^3;

    if(sayi<10)
    {
        top=top+sayi^3;

        goto dnz;
    }

    if(top==sayi)

        printf("esit");

    else

        printf("esit degil");

    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi,bas;
            int top=0;

            Console.Write("Bir sayı giriniz = ");
            sayi = Convert.ToInt32(Console.ReadLine());
            dnz:
            bas = sayi % 10;
            top = top + (bas * bas * bas);

            if (sayi < 10)
            {
                top = top + (sayi * sayi * sayi);
                goto dnz;
            }
            if (top==sayi)
            {
                Console.WriteLine("Eşit");
            }
            else
            {
                Console.WriteLine("Eşit değil");
            }
            Console.ReadLine();
        }
    }
}
```

12. Klavyeden girilen 20 adet sayıdan çift sayıların toplamının tek sayıların toplamına oranını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma :

- 1- Başla
- 2- tek,cift,sayi,i=1, oran
- 3- sayi gir
- 4- Eğer $\text{sayi} \% 2 = 0$ ise $\text{cift} = \text{cift} + \text{sayi}$,
değilse $\text{tek} = \text{tek} + \text{sayi}$
- 5- Eğer $i = 20$ ise devam et,
değilse $i++$ 3'e git
- 6- $\text{oran} = \text{cift} / \text{tek}$
- 7- Yazdır oran
- 8- Bitir

için statik çözümler yerine dinamik çözümler üretmeliyiz.

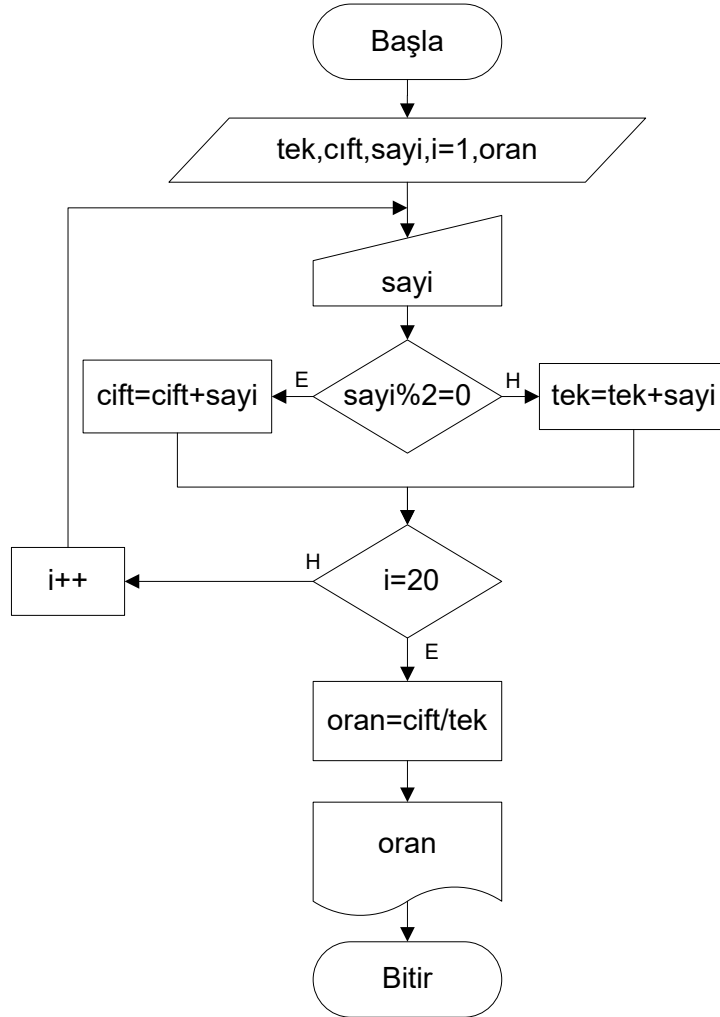
Algoritma Testi:

Adım	Sayı	cift	Tek	Oran
Baş.Değ.	-	0	0	-
1	4	4	0	-
2	5	4	5	-
3	10	14	5	-
4	6	20	5	-
Sonuç	6	20	5	4

Açıklama:

Bu soruda döngü ve şart mekanizması yine önümüze çıkıyor. 20 defa sayı girmek yerine programda 20 defa çalışacak bir döngü kurmalıyız. Eğer mekanizması, (%) kalanın tek mi çift mi olduğunu saptar ve bunları boş değişkenlere atar. Kalan tek ise **tek** değişkenine, çift ise **cift** değişkenine toplanarak atılır (Bu değişkenlere akılda daha kolay tutulabilmesi için kese de diyebiliriz). Burada tek ve cift değişkenlerinin değerleri başta 0 olmak zorundadır. Buna dikkat etmeliyiz, yoksa $\text{cift} = \text{cift} + \text{sayi}$ işlemini yapamayız. Bu sorunun algoritma testini 20 sayı için değil 4 sayı için yapıyoruz. Çünkü algoritmayı kurduğunuzda algoritma dinamik ise sayı 4 olsun 1004 olsun fark etmez. Bunu

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

int main()
{
    int sayi,tek=0,cift=0,i;

    float oran;

    for(i=1;i<=20;i++)
    {
        scanf("%d",&sayi);

        if(sayi%2==0)

            cift=cift+sayi;

        else

            tek=tek+sayi;

    }

    oran=cift/tek;

    printf("oran : %f ",oran);

    getch();

}
```

C# Kod

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int tek = 0;
            int cift = 0;
            int i,sayi;
            float oran;
            for (i = 1; i <= 20; i++)
            {
                Console.Write(i + ".sayıyı giriniz = ");
                sayi = Convert.ToInt32(Console.ReadLine());
                if ((sayi % 2) == 0)
                {
                    cift = cift + sayi;
                }
                else
                {
                    tek = tek + sayi;
                }
            }
            oran = cift / tek;
            Console.WriteLine("Oran = " + oran);
            Console.ReadLine();
        }
    }
}
```

13. 10 ile 1000 arasındaki tam kare sayıları ekrana yazdıran programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- sayac=4
- 3- Eğer $(\text{sayac} * \text{sayac}) \leq 1000$ ise
yazdır sayac*sayac , değilse 5'egit
- 4- sayac++
- 5- Bitir

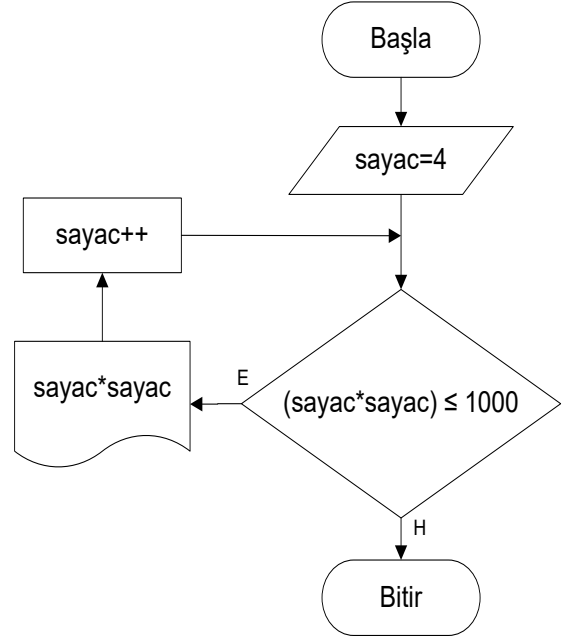
Açıklama:

Bu soru kolay bir örnek olarak karşımıza çıkıyor. 10 ile 100 arasındaki tam kare sayılar 16 ile başlayacağından sayac da 4'ten başlar. 4'ten küçük olanlara gerek yoktur. Çünkü $3*3$ örnek olarak 9'dur. 10 ile 100 arasında değildir. Şimdi Algoritmayı test edelim.

Algoritma Testi:

	Sayac	Sayac*Sayac
1	4	$16 > 10$
2	5	$25 > 10$
:	:	:
Sonuc	10	$100 > 10$

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
main(void)
{
    int sayac;
    for(sayac=4;sayac*sayac<=1000;sayac++)
    {
        printf("%lf\n",pow(sayac,2));
    }
    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 4;

            for (i = 4; i <= 1000; i++)
            {
                if ((i * i) <= 1000)
                {
                    Console.WriteLine(Math.Pow(i,2));
                }
            }
            Console.ReadLine();
        }
    }
}
```

14. Klavyeden girilen 25 adet sayı içerisinde negatif olanların toplamını, çift sayıların çarpımını, 7'ye eşit olanların adetini bulup ekrana yazdıran programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $i=1, \text{sayi}, \text{neg}=0, \text{cift}=0, \text{esit}=0$
- 3- sayı gir
- 4- Eğer $\text{sayi} < 0$ ise $\text{neg} = \text{neg} + \text{sayi}$,
değilse devam et
- 5- Eğer $\text{sayi} \% 2 = 0$ ise $\text{cift} = \text{cift} * \text{sayi}$,
7'ye git, değilse devam et
- 6- Eğer $\text{sayi} = 7$ ise $\text{esit} = \text{esit} + 1$,
değilse devam et
- 7- Eğer $i = 25$ ise devam et,
değilse $i++$ 3'e git
- 8- Yazdır "negatif:" neg, "cift:" cift, "
7'ye esit:" esit
- 9- Bitir

Açıklama:

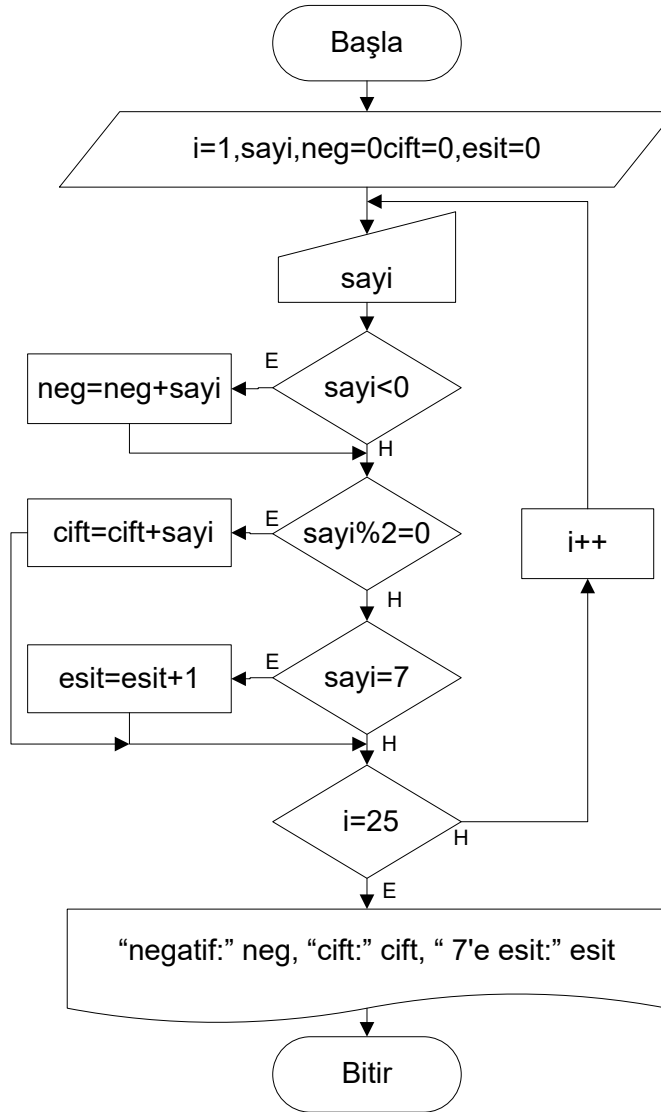
Bu soru basit olmakla beraber, programlama dilinde çok önemli ve çok kullandığımız if yapılarını anlatıyor. Tabii ki algoritmada if yerine eğer kullanıyoruz. Çünkü algoritmada program dili komutları kullanmıyoruz. Yine bir döngümüz var 25 tane

sayı girdiriyor ve bu sayıları 3 şart ile kıyaslayıp hangisine uyarsa o değişkenin içine atıp toplattırıyoruz. Bu soru daha çok alıştırmaya sorusu niteliğindedir. Soru basittir ama bize eğer(karar) mekanizmasına iyice alışmamız gerektiğini anlatmaktadır. Buna örnek için içe for döngülerinde matris işlemlerini örnek verebiliriz. Bu testi yaparken programın dinamik olduğunu göz önüne alarak biz 5 sayı için yapmayı tercih ettik. İstenilirse 25 için de, 5 için de N kez için de olabilir. 😊

Algoritma Testi:

Adım	Sayı	Neg	cift	(eşit)
Baş.Değ.	-	0	0	0
1	4	0	4	0
2	-1	-1	4	0
3	-2	-3	2	0
4	7	-3	2	7
Sonuç	8	-3	10	7

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

main(void)
{
    int i,sayi,negatif=0,cift=0,bol=0;

    for(i=1;i<=25;i++)
    {
        scanf("%d",&sayi);

        if(sayi<0)

            negatif=negatif+sayi;

        if(sayi%2==0)

            cift=cift*sayi;

        if(sayi==7)

            bol++;

    }

    printf("negatif :%d \n,cift :%d\n,7'e esit :
    %d",negatif,cift,bol);

    getch();

}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 1;
            int sayi;
            int neg = 0, cift = 1, bol = 0;
            for (i = 1; i <= 5; i++)
            {
                Console.Write(i + ".sayıyı giriniz = ");
                sayi = Convert.ToInt32(Console.ReadLine());
                if (sayi < 0)
                {
                    neg = neg + sayi;
                }
                else if ((sayi % 2) == 0)
                {
                    cift = cift * sayi;
                }
                else if (sayi == 7)
                {
                    bol++;
                }
            }
            Console.WriteLine("Negatif sayıların toplamı = " + neg);
            Console.WriteLine("Çift sayıların çarpımı = " + cift);
            Console.WriteLine("7'e eşit olanların adeti = " + bol);
            Console.ReadLine();
        }
    }
}
```

15. Çarpım Tablosunun algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $i=1, j=1$
- 3- Yazdır $i*j$
- 4- Eğer ($i=10$) ise 6'ya git ,
değilse devam et
- 5- Eğer ($j=10$) ise $j=1$,
değilse $i++$, $j++$ 3'e git
- 6- Bitir

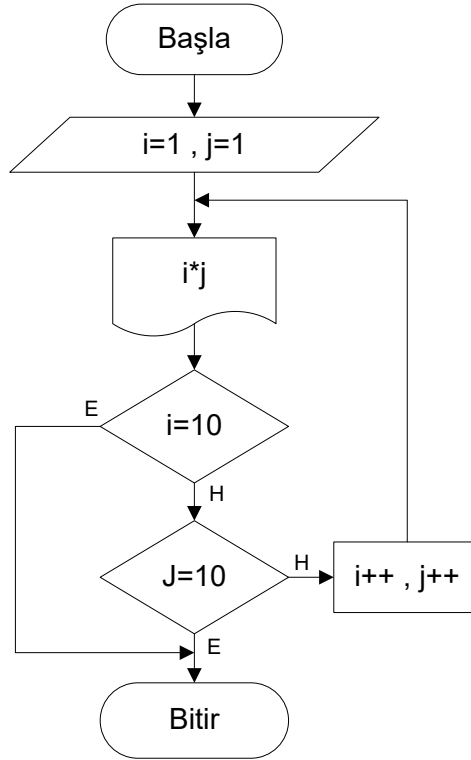
Algoritma Testi:

Adım	i	J	$i*j$
Baş.Değ.	1	1	-
1	1	1	1
2	1	2	2
3	1	3	3
4	1	4	4
...	...	...	...
10	1	10	10
11	2	1	2
12	2	2	4
i değeri 10 oluncaya kadar testimiz devam edecektir.			

Açıklama:

Çarpım tablosu sorusu bize iç içe döngü olayını öğretmek için birebirdir. C programlama dilinde for döngüleri ile çok kullanılır. Bunun için iki değişken aldık: Bunlar i ve j değişkenleridir. Burada i bir defa çalıştığında j 10 defa çalışacaktır. İşlemin bütün mantığı buna dayanmaktadır. Bu soruyu karar mekanizmaları ile çözdük ama döngü şekli ile de akış diyagramımızı çizebiliriz. Döngü şeklimiz Akış diyagramlarında bulunmaktadır. Bunu için kitabın ilk sayfalarında akış diyagramı ile ilgili bölümlere tekrar bakılabilir.

Akış Diyagramı:



C Kod

```
#include <stdio.h>
#include <conio.h>
main(void)
{
    int i ,j;
    for(i=1;i<=10;i++)
    {
        printf("\n");
        for(j=1;j<=10;j++)
        {
            printf ("%d * %d = %d \n",i,j,i*j);
        }
    }
    getch();
}
```

C# Kod

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i, j;
            for (i = 1; i <= 10; i++)
            {
                Console.WriteLine();
                for (j = 1; j <= 10; j++)
                {
                    Console.WriteLine(i + "*" + j + "=" + i * j);
                }
            }
            Console.ReadLine();
        }
    }
}
```

16. Girilen sayının 5'in kuvveti olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- sayi
- 3- sayi gir
- 4- Eğer $\text{sayi} \% 5 = 0$ ise $\text{sayi} = \text{sayi} / 5$
4'e gir, değilse devam et
- 5- Eğer $\text{sayi} = 1$ ise yazdır
"5'in kuvvetidir"
değilse yazdır "5'in kuvveti değildir"
- 6- Bitir

Açıklama:

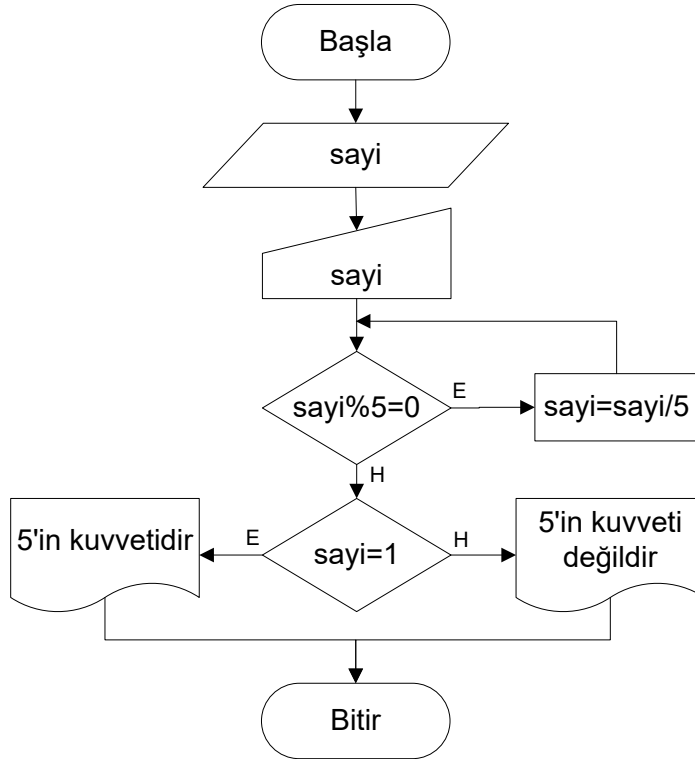
Bu soruda girilen sayının 5'in katı olup olmadığına bakıyoruz. Bu soruda dayandığımız mantık sayıyı devamlı 5'e böldürmektir. Kalan 0 olmazsa zaten 5'in katı değildir ve programdan çıkarılır. Fakat sayıyı devamlı 5'e bölüp kalanı 0 bulduğumuzda ve en son artık sayımız azala azala 1'e eşit duruma geldiyse, o zaman 5'in katıdır, deriz. Algoritma testinde bu açıklama daha iyi anlaşılacaktır.

Algoritma Testi:

Adım	Sayı	$\text{sayi} \% 5$
1	25	0
2	5	0
sonuc	1	1
sonuc	$\text{sayi} = 1 \rightarrow 5$ 'in tam kuvvetidir.	

Adım	Sayı	$\text{sayi} \% 5$
1	10	0
Sonuc	2	2
Sonuc	$\text{sayi} = 2 \rightarrow 5$ 'in tam kuvveti değildir.	

Akış Diyagramı:



C Kod

```
#include <stdio.h>

#include <conio.h>

main(void)
{
    int sayi ;

    scanf("%d",&sayi);

    dnz:

    if(sayi%5==0)
    {
        sayi=sayi/5;

        goto dnz;
    }

    if(sayi==1)
        printf("5'in kuvvetidir");
    else
        printf("5'in kuvveti degildir");

    getch();
}
```

C# Kod

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi = 0;
            Console.Write("Sayı giriniz = ");
            sayi = Convert.ToInt32(Console.ReadLine());
            dnz:
            if (sayi % 5 == 0)
            {
                sayi = sayi / 5;
                goto dnz;
            }
            else if (sayi == 1)
            {
                Console.WriteLine("5'in kuvvetidir");
            }
            else
            {
                Console.WriteLine("5'in kuvveti değildir");
            }
            Console.ReadLine();
        }
    }
}
```

17. X,Y pozitif olmak üzere, eğer x sayısının çarpanları toplamı y sayısına ve aynı zamanda y sayısının çarpanları toplamı x sayısına eşit ise bu sayılar dost sayılardır. Buna göre girilen iki sayının dost olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $x, y, xcarpan=0, ycarpan=0, i=1, temp=0$
- 3- x, y gir
- 4- $i=x/2$
- 5- Eğer $x\%i=0$ ise $xcarpan=xcarpan+i$, değilse devam et
- 6- Eğer $i=0$ ise devam et, değilse $i--$ 5'e git
- 7- $i++$
- 8- Eğer $y\%i=0$ ise $ycarpan=ycarpan+i$, değilse devam et
- 9- Eğer $i=x/2$ ise devam et, değilse $i++$ 8'e git
- 10- Eğer $xcarpan=y$ & $ycarpan=x$ ise yazdır "Bu sayılar dosttur", değilse yazdır "Bu sayılar dost sayı değildir"
- 11- Bitir

Açıklama:

Dost sayılar MÖ 500'lü yıllarda Pisagor okulunda sayı mistizmi üzerine araştırma yapan matematikçilerin yarattıkları bir sayı grubudur. Böyle soruları çözmek bizim algoritma konusunda gelişmemizi sağlar. Burada 2 adet sayaç kullanmamız gerekmektedir, sayaçlarımız ycarpan ve xcarpan'dır. Bu değişkenleri başta 0 olarak tanımlamalıyız. Buna dikkat edilmelidir. Örneğin eğer biz değişkenleri tanımlarken xcarpan=0 olarak tanımlamasaydık algoritmada 5. adımda bulunan xcarpan=xcarpan+i işlemi çalışmazdı i değeri ile hangi değeri toplayacağını derleyici bilemez ve hata döndürür. Burada 4. adımda X sayısını 2'ye bölmek de önemlidir. Çünkü bir sayıyı kendi hariç bölebilecek en büyük sayı yarısı kadardır.

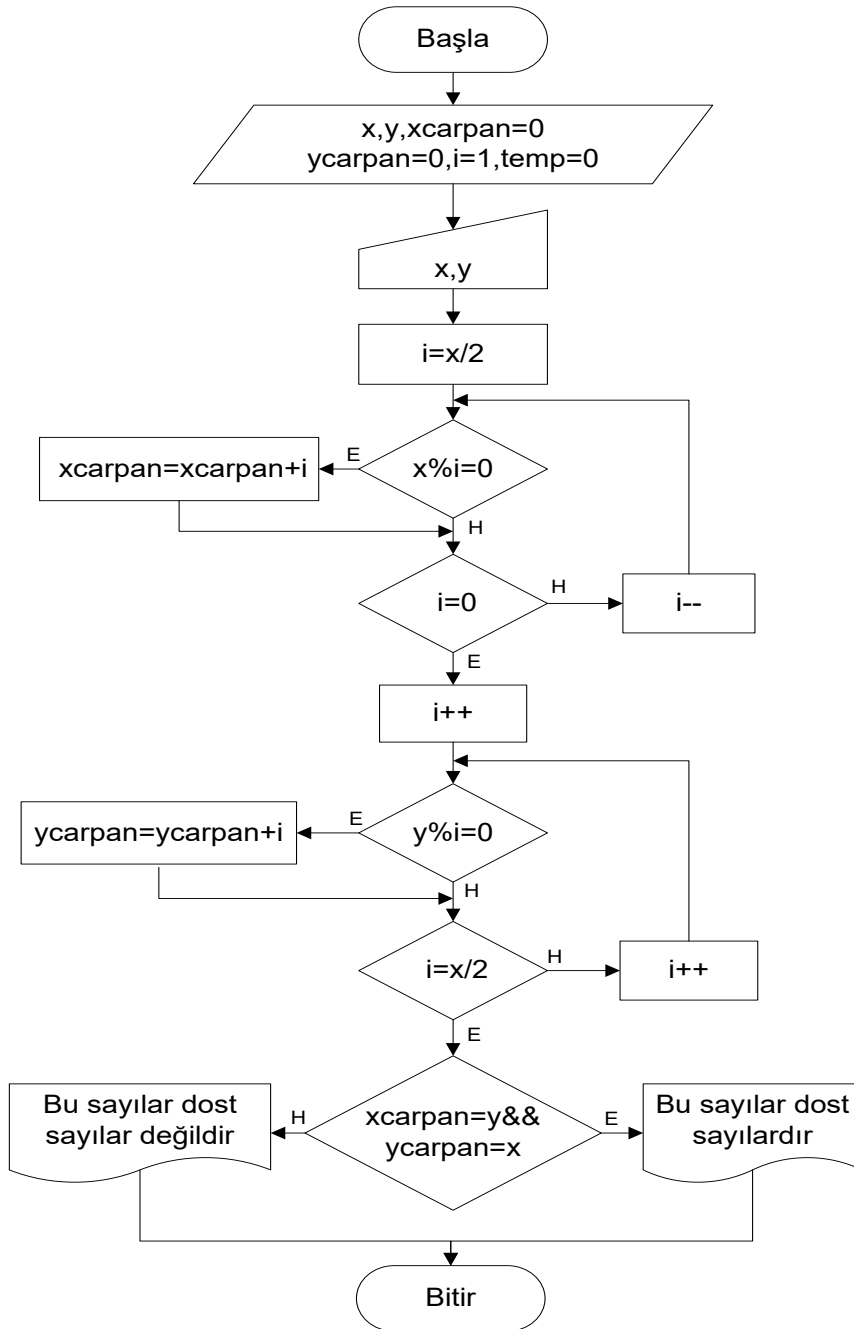
Örnek : 32 sayısını bölen, kendisinden hariç en büyük sayı 16'dır. Bunun için döngümüzü i=0 dan i=16 ya kadar götürmek hız açısından daha uygun olacaktır.

Bu soru da öncelikle X sayısının tam bölenlerini bir değişkene atıyor, sonra Y değişkeninin bölenlerini bir değişkene atıyor. Sonra bu iki değişkeni <eğer> ile karşılaştırıp eşitse ekrana "dost" ya da eşitsizliğe göre "dost değil" olarak yazdırıyor. Bu soru döngü ve karar mekanizmaları açısından bize idman yaptıran önemli bir sorudur.

Algoritma Testi:

Adım	X	Xcarpan(x'i tam bölen sayılar)	Y	ycarpan(y'i tam bölen sayılar)
Baş.	220	0	284	0
1	"	1	"	1
2	"	2	"	2
3	"	4	"	4
4	"	5	"	71
5	"	10	"	142
6	"	20	Toplam: 220	
7	"	22		
8	"	44		
9	"	55		
10	"	110		
11	Toplam : 284			

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main(void)
{
    int x,y,xcarpan=0,ycarpan=0,i;
    scanf("%d%d",&x,&y);
    i=(x-(x%2))/2;
    while(i>0)
    {
        if(x%i==0)
            xcarpan=xcarpan+i;
        i--;
    }
    i=1;
    while(i<y)
    {
        if(y%i==0)
            ycarpan=ycarpan+i;
        i++;
    }
    if(xcarpan==y && ycarpan==x)
        printf("%d ile %d arkadas sayilardir" ,x,y);
    else
        printf("%d ile %d arkadas degildir" ,x,y);
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int x, y, i;
            int xcarpan = 0, ycarpan = 0;
            Console.WriteLine("1.sayıyı giriniz = ");
            x = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("2.sayıyı giriniz = ");
            y = Convert.ToInt32(Console.ReadLine());
            i = (x - (x % 2)) / 2;
            while (i > 0)
            {
                if (x % i == 0)
                    xcarpan = xcarpan + i;
                i--;
            }
            i = 1;
            while (i < y)
            {
                if (y % i == 0)
                    ycarpan = ycarpan + i;
                i++;
            }
            if (xcarpan == y && ycarpan == x)
            {
                Console.WriteLine(x + " ile " + y + "dost sayilardir");
            }
            else
            {
                Console.WriteLine(x + " ile " + y + " dost degildir");
            }
            Console.ReadLine();
        }
    }
}
```

18. Fibonacci serisinin ilk 10 terimini ekrana basan algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $\text{sayi1}=1, \text{sayi2}=1, \text{sayi3}, \text{sayac}=2$
- 3- Yazdır $\text{sayi1}, \text{sayi2}$
- 4- $\text{sayi3}=\text{sayi1}+\text{sayi2}$
- 5- Yazdır sayi3
- 6- $\text{sayi1}=\text{sayi2}, \text{sayi2}=\text{sayi3}$
- 7- $\text{sayac}++$
- 8- Eğer $\text{sayac}=11$ ise devam et,
değilse 4'e git
- 9- Bitir

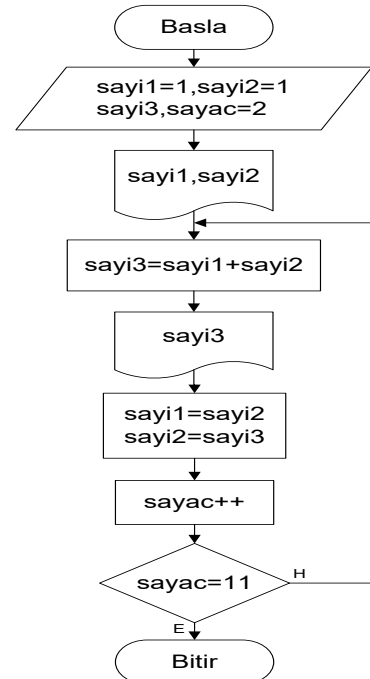
Açıklama:

Daha önce 6. yüzyılda Hintli matematikçiler tarafından bulunmuş olan bu sayı dizisi Liber Abaci kitabında tavşanların üremesiyle ilgili problemin hesaplanması sonucu Fibonacci tarafından 1202 yılında ortaya konmuştu. Dizinin ilk sayı değeri 0, ikincisi 1 ve her ardışık elemanı da önceki iki elemanın sayı değerinin toplamı alınarak bulunur ve bu halde 0, 1, 1(1+0), 2(1+1), 3(2+1), 5(3+2), 8(5+3), 13(8+5),... şeklinde artar.

Algoritma Testi:

	Sayı1	Sayı2	Sayı3	Sayac
1	1	1	2	2
2	1	2	3	3
3	2	3	5	4
4	3	5	8	5
5	5	8	13	6
6	8	13	21	7
7	13	21	34	8
Sonuç	21	34	55	9

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    int sayi1=1,sayi2=1,sayi3,sayac;

    printf("%d ",sayi1);

    printf("%d ",sayi2);

    for(sayac=2;sayac<=11;sayac++)
    {
        sayi3=sayi1+sayi2;

        printf("%d ",sayi3);

        sayi1=sayi2;

        sayi2=sayi3;

    }

    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi1 = 1;
            int sayi2 = 1;
            int sayi3;
            int sayac = 2;
            Console.Write(sayi1);
            Console.Write(sayi2);
            for(sayac=2;sayac<=10;sayac++)
            {
                sayi3 = sayi1 + sayi2;
                Console.Write(sayi3);
                sayi1 = sayi2;
                sayi2 = sayi3;
            }
            Console.ReadLine();
        }
    }
}
```

19. Klavyeden girilen bir sayının negatif, pozitif veya 0 olup olmadığını bulan programın algoritma ve akış diyagramını çizin

Algoritma:

- 1- Başla
- 2- sayı
- 3- sayı gir
- 4- Eğer $\text{sayı} > 0$ ise yazdır "pozitif",
değilse devam et
- 5- Eğer $\text{sayı} < 0$ ise yazdır "negatif",
değilse yazdır "sayı 0"
- 6- Bitir

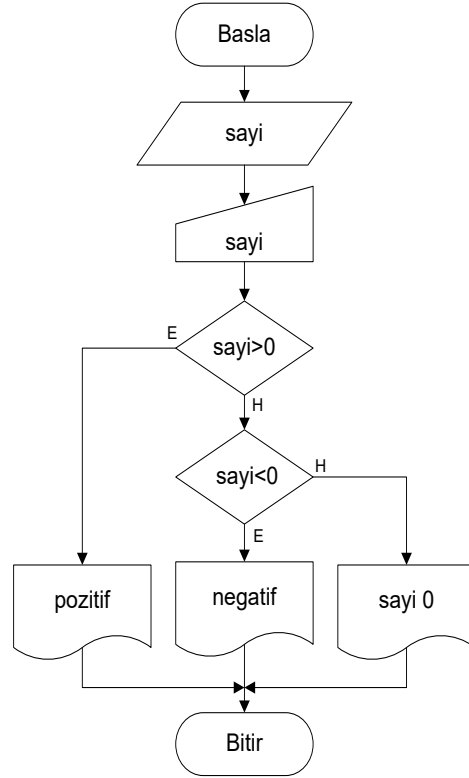
Açıklama:

Temel sorulardan biri sayının negatif mi pozitif mi olduğudur. Burada 3 sayı varsa ya da 3 işlem varsa $3-1=2$ adet eğer gerekir. Eğer (if) programlama dilinde önemli bir kavramdır. Bu soru bu gibi programlarda fazla eğer kullanmamızı önlemek amacı içerir, daha az eğer kullanarak pratik yoldan yapma mantığını gösteren güzel ve anlaşılır bir örnektir. Öğrencilerin kesinlikle öğrenmesi gereken bir kavramdır.

Algoritma Testi :

	Sayı	Sonuç
1	5	Pozitif
2	-1	Negatif
3	0	Sayı 0

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    int sayi;

    scanf("%d",&sayi);

    if(sayi>0)

        printf("sayi pozitifdir");

    else

        {

            if(sayi<0)

                printf("sayi negatifdir");

            else

                printf ("sayi 0 'a esittir");

        }

    getch();

}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi;
            Console.Write("Sayıyı giriniz = ");
            sayi = Convert.ToInt32(Console.ReadLine());
            if (sayi > 0)
            {
                Console.Write("Sayı pozitifdir");
            }
            else if (sayi < 0)
            {
                Console.Write("Sayı negatifdir");
            }
            else
            {
                Console.Write("Sayı 0'a eşittir");
            }
            Console.ReadLine();
        }
    }
}
```

20. Girilen sayının mükemmel sayı olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $sayi, sayac=2, top=1$
- 3- $sayi$ gir
- 4- Eğer $sayi \% sayac=0$ ise
 $top=top+sayac, sayac++$
değilse $sayac++$
- 5- Eğer $sayac > (sayi/2)$ ise devam et,
değilse 4'e git
- 6- Eğer $top=sayi$ ise yazdır "Mükemmel
sayı", değilse yazdır "Mükemmel sayı
değildir"
- 7- Bitir

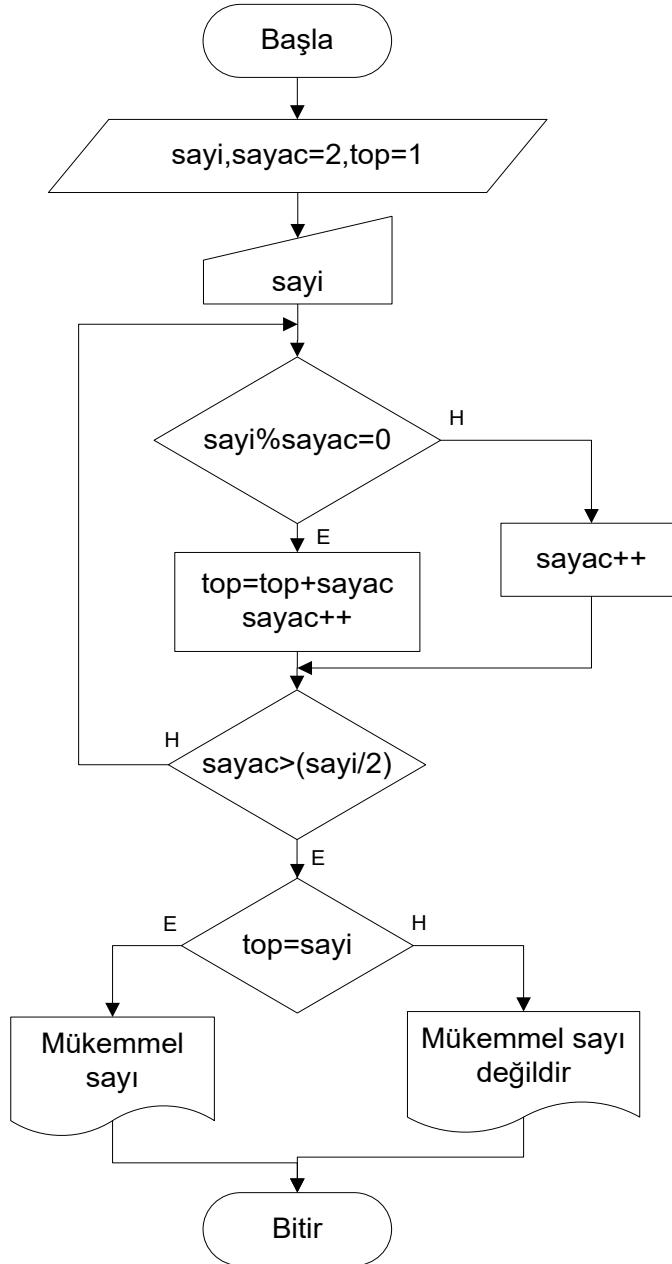
Algoritma Testi:

	Sayı	Sayac	Top
1	28	2	1
2	28	3	3
3	28	4	7
4	28	5	7
5	28	6	7
6	28	7	14
:	:	:	:
Sonuç	28	14	28

Açıklama:

Mükemmel sayı sorusu bizim en sevdiğimiz soru çeşitlerindendir. Döngü kavramı karşımıza burada daha açık bir şekilde çıkmaktadır. Mükemmel sayı, kendisini tam bölen sayıların toplamı, kendine eşit olan sayılardır. Örnek 28'dir. $1+2+4+7+14 = 28$

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int sayi,sayac=2,top=1;
    scanf("%d",&sayi);
    dnz:
    if(sayi%sayac==0)
    {
        top=top+sayac;
        sayac++;
    }
    else
    sayac++;
    if(sayac>sayi/2)
    {
        if(top==sayi)
            printf("mukemmeldir");
        else
            printf("mukemmel degildir");
    }
    else
        goto dnz;
    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi;
            int sayac = 2;
            int top = 1;
            Console.Write("Sayıyı giriniz = ");
            sayi= Convert.ToInt32(Console.ReadLine());
            dnz:
            if (sayi % sayac == 0)
            {
                top = top + sayac;
                sayac++;
            }
            else
            {
                sayac++;
            }
            if (sayac > sayi / 2)
            {
                if (top == sayi)
                    Console.WriteLine("Mükemmeldir");
                else
                    Console.WriteLine("Mükemmel degildir");
            }
            else
            {
                goto dnz;
            }
            Console.ReadLine();
        }
    }
}
```

21. 1-100 arasındaki çift sayıların toplamının mükemmel sayı olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başlat
- 2- $i=0, top=0, y=0$
- 3- $top=top+(2+i)$
- 4- Eğer $(i+2)=100$ ise devam et,
değise $i=i+2$ 3'e git
- 5- $i=i+1$
- 6- Eğer $top \% i = 0$ ise $y=y+i$,
değilse devam et
- 7- Eğer $top=i+1$ ise devam et,
değilse $i=i+1$,6'ya git
- 8- Eğer $top=y$ ise yazdır
"Mükemmeldir"
değilse "Mükemmel değil"
- 9- Bitir

Açıklama:

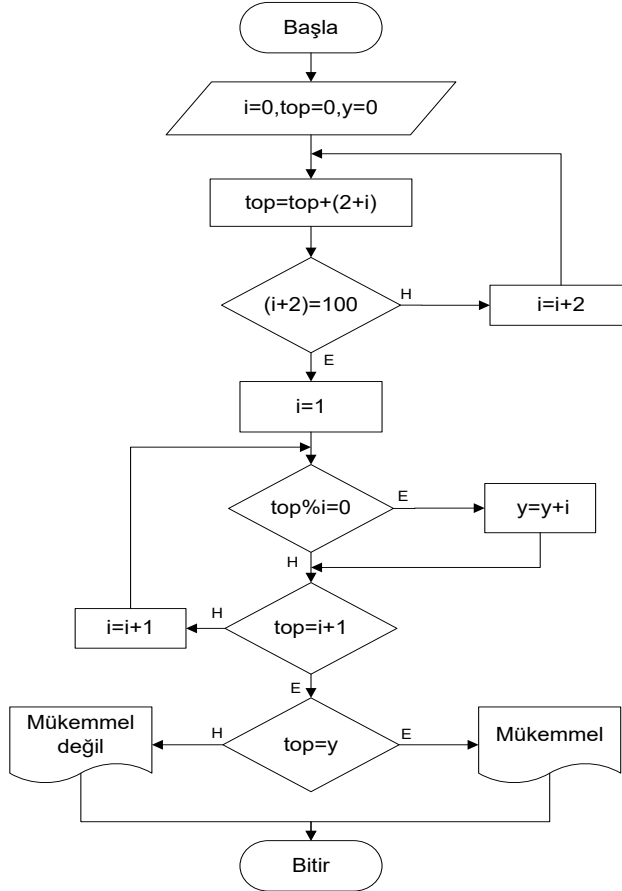
Bu soruda şunu bilmeliyiz ki 1 ile 100 arasında... tarzındaki sorularda 1'den 100'e kadar program kodumuz istediğimiz yerden döngüye girecektir. Buna göre i değerimiz yani bir sayacımız olacak, değer her defasında artacaktır. Eğer programda 100'den 1'e denilirse bu değer her defasında azaltılacaktır. C böyle sorularda her zaman For döngüsünü sever. Bir de burada fazladan bir $(sayi \% 2)$ eğerini koymak yerine, zaten çift sayıları bildiğimizden hemen **$top=top+2$** diyerek, çiftleri direkt döngüye katmamız uygun olacaktır. Sonra bu toplama (**top** değişkeninin içindeki), mükemmel sayı mı diye bakılır. Bu durum 20. Soruda açıklanmıştır. Şimdi de algoritma testine geçelim.

Algoritma Testi:

(i)	Top
2	2
4	6
6	12
...	...
....	...
100	2550

Sayı	Y
2550	1
"	(1+3) 4
"	(4+5) 9
"	...
"	...
2550	2232(Mükemmel değil çünkü toplamın 2550 çıkması gerekirdi)

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    int i,top=0,y=0;

    for(i=0;i<=98;i++)
    {
        top=top+(2+i);
    }

    for(i=1;i<=(top/2);i++)
    {
        if((top%i)==0)
            y=y+i;
    }

    if(top==y)
        printf("mukemmel");
    else
        printf("mukemmel degildir");
    getch();
}
```

C# Kod

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i;
            int top = 0;
            int y = 0;
            for (i = 0; i <= 98; i++)
            {
                top = top + (2 + i);
            }
            for (i = 1; i <= (top / 2); i++)
            {
                if ((top % i) == 0)
                {
                    y = y + i;
                }
            }
            if (top == y)
            {
                Console.WriteLine("Mükemmel");
            }
            else
            {
                Console.WriteLine("Mükemmel
değildir");
            }
            Console.ReadLine();
        }
    }
}
```

22. Herhangi bir sayının herhangi bir dereceden kuvvetini bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- sayi,us,top=1
- 3- sayi,us gir
- 4- $top=top*sayi$
- 5- $us--$
- 6- Eğer $us=0$ ise devam et, değilse 3'e git
- 7- Yazdır top
- 8- Bitir

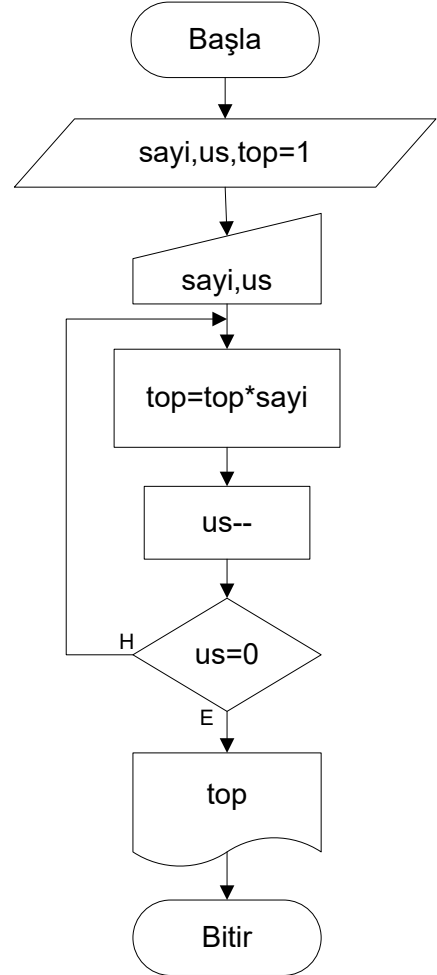
Açıklama:

Bu soruda program dili ile kullandığımız (^) işareti ya da hazır fonksiyonları kullanmadık. Girilen sayının us değerine göre aldığı sonucu hesapladık. Burada sayac olarak **us** değişkenini kullandık. Azalma için ise “--” ifadesini kullandık.

Algoritma Testi:

	Sayı	Us	top
1	3	2	1
2	3	1	3
Sonuç	3	0	9

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    int sayi,us,top=1;

    printf("Sayiyi giriniz");

    scanf("%d",&sayi);

    printf("Ussu giriniz");

    scanf("%d",&us);

    while(us>0)
    {
        top=top*sayi;

        us -- ;
    }

    printf("%d",top);

    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi;
            int us;
            int top = 1;
            Console.Write("Sayıyı giriniz = ");
            sayi =
            Convert.ToInt32(Console.ReadLine());
            Console.Write("Ussu giriniz = ");
            us =
            Convert.ToInt32(Console.ReadLine());
            while (us > 0)
            {
                top = top * sayi;
                us--;
            }
            Console.Write("Sonuç = "+top);
            Console.ReadLine();
        }
    }
}
```

23. Girilen sayının abundant (güçlü) sayı mı ya da Deficient (güçsüz) sayı mı olduğunu bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- sayı ,bolen=1,sayac=2
- 3- sayı gir
- 4- Eğer sayı%sayac=0 ise
bolen=bolen+sayac ,
değilse sayac++
- 5- Eğer sayac<sayı ise devam et,
değilse 3'e git
- 6- Eğer bolen>sayı ise yazdır
"Abundant sayı", değilse yazdır
"Deficient sayı"
- 7- Bitir

Açıklama:

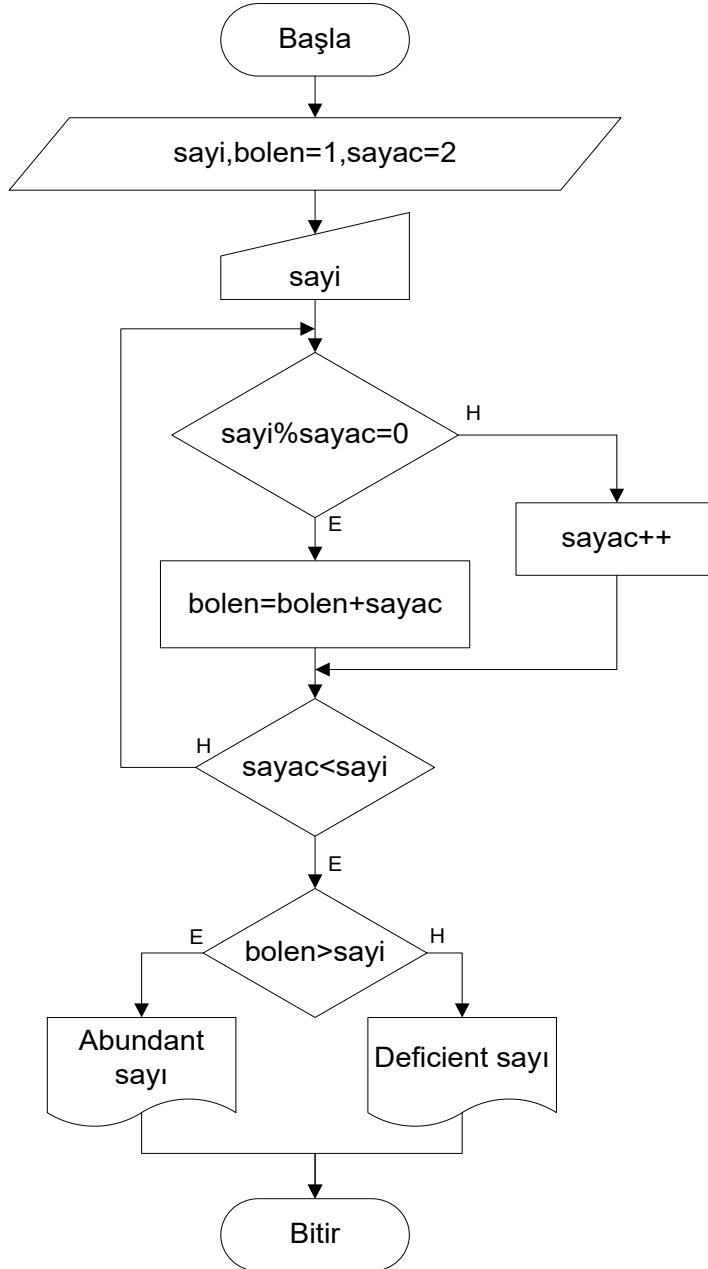
Bu soru aslında mükemmel sayı mantığına yakın bir sorudur. Döngü ve karar mekanizmaları bu soruda da bulunmaktadır. Önemli olan programlamada if ve döngü yapılarıdır. Bunlar çok iyi kavranmalıdır. Bu soruda bir sayı giriyoruz. Bu sayıyı tam bölen sayıları bulmak için , sayı/2' ye kadar sayımızı sayaca böldürüp kalan değer 0 ise bunu bölen adlı değişkenin içine toplayarak atıyoruz. Çıkan bölen değer ile sayıyı karşılaştırıp büyük

ise "güçlü" , küçük ise "güçsüz" diye ekrana bastırıyoruz. Şimdi Algoritma testine geçelim.

Algoritma Testi:

Adım	bolen	Sayac	Sayı
1	1	2	8
2	3	3	8
3	7	4	8
4	7	5	8
5	7	6	8
Son	7	7	8
Sonuç	7	deficient	8

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    int sayi ,bolen=1,sayac=2;

    scanf("%d",&sayi);

    while(sayac<sayi)
    {
        if(sayi%sayac==0)

            bolen=bolen+sayac;

            sayac++;

    }

    if(bolen>sayi)

        printf("abondant sayi");

    else

        printf("Deficient sayi");

    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi;
            int bolen = 1;
            int sayac = 2;
            Console.Write("Sayıyı giriniz = ");
            sayi =
            Convert.ToInt32(Console.ReadLine());
            while (sayac < sayi)
            {
                if (sayi % sayac == 0)
                {
                    bolen = bolen + sayac;
                }
                sayac++;
            }
            if (bolen > sayi)
            {
                Console.Write("Abondant sayi");
            }
            else
            {
                Console.Write("Deficient sayi");
            }
            Console.ReadLine();
        }
    }
}
```

24. 1'den 500'e kadar olan tamsayıların toplamını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $i=1, \text{toplam}=0$
- 3- $\text{toplam}=\text{toplam}+i$
- 4- Eğer $i=500$ ise 6.adıma git,
değilse devam et
- 5- $i=i+1$ ve 3. adıma git
- 6- yazdır toplam
- 7- Bitir

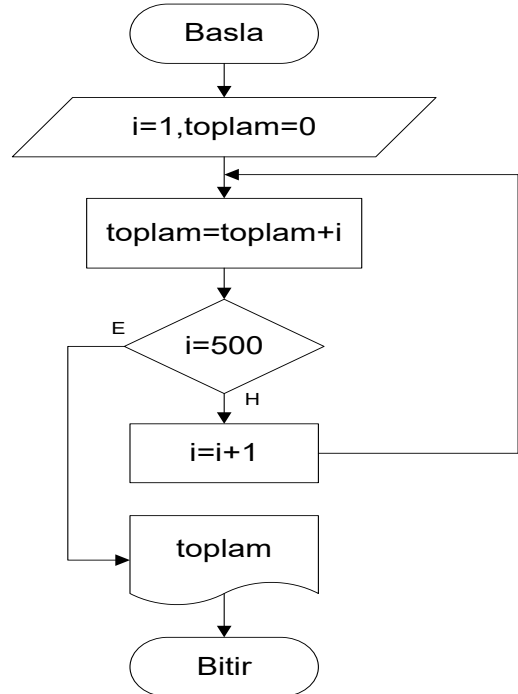
Açıklama:

Bu soru tipik döngü sorusu. Her zaman belirttiğimiz gibi bir sayı aralığı veriliyorsa (burada olduğu üzere 1 ile 500 gibi) hemen aklımıza i sayısı ve döngü yani i değerimiz 500'e eşit olana kadar programı tekrarlamamız gerektiği gelmelidir. Burada i değişkeni 500'e eşit olduğunda döngüden çıkılacak ve toplam değişkeni ekrana basılacaktır. ($\text{toplam}=\text{toplam}+i$)

Algoritma Testi:

i	Top
1	1
2	3
3	6
...	...
...	...
500	125250

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i,toplam=0;
    for(i=1;i<=500;i++)
        toplam=toplam+i;
    printf("%d",toplam);
    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i;
            int toplam = 0;
            for (i = 1; i <= 500; i++)
            {
                toplam = toplam + i;
            }
            Console.WriteLine("Toplam = " + toplam);
            Console.ReadLine();
        }
    }
}
```

25. Girilen a ve b sayısı 50'den büyük olduğunda $c=a+b$ işlemini yapan değilse bu sayılar uygun değil yazdıran programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- a, b
- 3- a, b gir
- 4- Eğer $a>50$ && $b<50$ ise devam et, değilse 7'ye git
- 5- $c=a+b$
- 6- Yazdır c, 8'e git
- 7- Yazdır "Bu sayılar uygun değildir"
- 8- Bitir

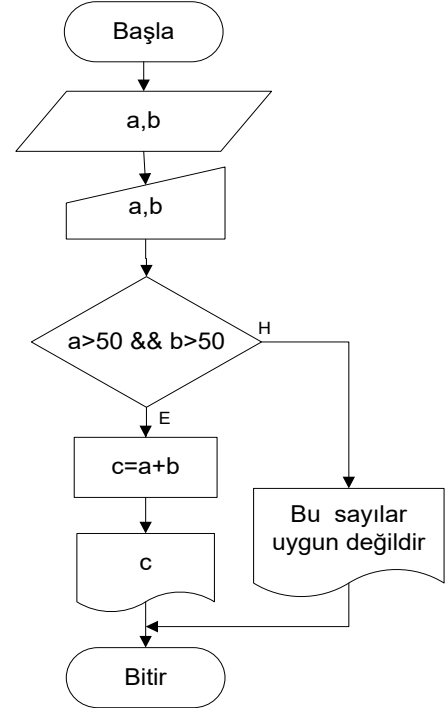
Açıklama:

Bu soru artık sizlere yavaş yavaş basit gelmeye başlayacaktır. Burada değişik olan && operatörüdür. Daha önce hiç ikili karşılaştırma yapmamıştık. Algoritma, burada 4. satırda $a > 50$ ve aynı zamanda $b < 50$ olması yani ikisinin de doğru olması durumunda $c=a+b$ işlemini yapacak yoksa "bu sayılar uygun değildir" yazacaktır. Bu şekilde programlama diline de geçince iki eğer kullanacağımıza, tek eğer içinde operatörler kullanarak işlemimiz tek hamlede çözülebilir hale gelecektir. Şimdi algoritma testine geçelim.

Algoritma Testi:

Adım	a	b	Sonuç
1	45	89	Bu sayılar uygun değildir
2	70	40	$C = 110$

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

main(void)
{
    int a,b,c;

    printf("a sayisini giriniz");

    scanf("%d",&a);

    printf("b sayisini giriniz");

    scanf("%d",&b);

    if(a>50 && b>50)
    {
        c=a+b;

        printf("%d",c);

    }

    else

    printf("bu sayilar uygun degildir");

    getch();

}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int a, b, c;
            Console.Write("A sayısını giriniz = ");
            a=Convert.ToInt32(Console.ReadLine());
            Console.Write("B sayısını giriniz = ");
            b = Convert.ToInt32(Console.ReadLine());
            if (a > 50 && b > 50)
            {
                c = a + b;
                Console.Write("Toplam = " + c);
            }
            else
            {
                Console.Write("Bu sayılar uygun değildir");
            }
            Console.ReadLine();
        }
    }
}
```

26. 1'den 63'e kadar olan sayılar arasında istenilen sayıyı maksimum 6 seferde bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $alt=1, ust=63, tutulan(rastgele), tahmin, sayac=0$
- 3- tahmin gir
- 4- $tahmin=(alt+ust)/2$
- 5- $sayac++$
- 6- Eğer $tahmin=tutulan$ ise
7. adıma git,
değilse devam et
- 7- Eğer $tahmin>tutulan$ ise
 $ust=tahmin$ 3.adıma git,
değilse $alt=tahmin$
3.adıma git
- 8- Yazdır $sayac, tahmin$
- 9- Bitir

ve en yüksek değerleri alırız, dolayısıyla bu soruda $alt=1$ ve $ust=63$ olur. Bir de bu seride arayacağımız sayımız olacaktır. Biz bu soruda bunu rastgele bilgisayara aldıracağız. Tutulan sayıyı bulana kadar seriyi her defasında tutulan sayının büyük ya da küçük olmasına göre böyle böyle küçülteceğiz. Tabii bu arada her tahminde sayacımızı 1 arttıracamız. Bu algoritma testinde daha iyi kavranacaktır. Burada algoritmanızın doğruluğunu test etmenin önemini bir kez daha vurgulayarak algoritma testimize geçiyoruz.

Açıklama:

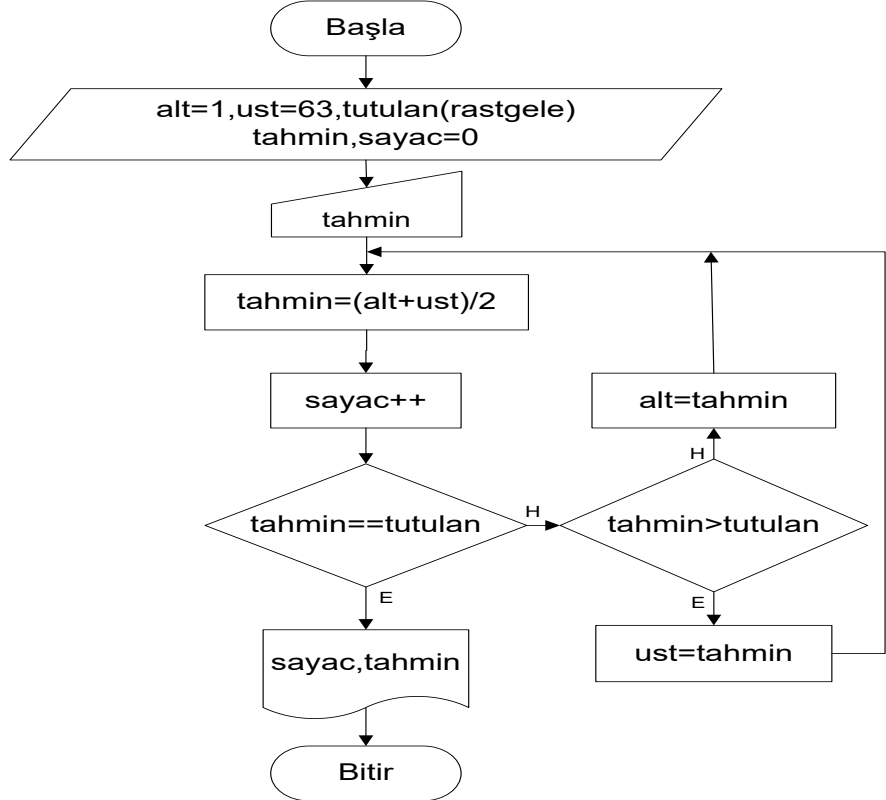
Bu soru, matematik (algoritma) olimpiyatlarında sorulmuş bir sorudur ve ileride dizi sorularında binary(ikili) arama konusunda anlatacağımız olan algoritmayı kullanan bir sorudur. Bu sorularda 1-63 arası... dendiğinde alt ve ust değer olarak en düşük

Algoritma test:

Sayac	Alt	Ust	Sayı(tutulan)	Tahmin	Değerlendirme
1	1	63	7	$32 > 7$	$(1+63)/2$
2	1	$32(>7)$	7	$16 > 7$	$(1+32)/2$
3	1	$16(>7)$	7	$8 > 7$	$(1+16)/2$
4	1	$8(>7)$	7	$4 < 7$	$(1+8)/2$
5	$4(<7)$	8	7	$6 < 7$	$(4+8)/2$
6	$6(<7)$	8	7	$7 = 7$	$(6+8)/2$

Akış

Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
main(void)
{
int alt=1,ust=63,tahmin,sayac=0;
randomize();
int tutulan=(rand() % 63) + 1;
dnz:
tahmin=(alt+ust)/2;
sayac++;
if(tahmin==tutulan)
printf("tutulan :%d sayac %d",
tutulan,sayac);
else
{
if(tahmin>tutulan)
ust=tahmin;
else
alt=tahmin;
goto dnz;
}
getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
class Program
{
static void Main(string[] args)
{
int alt = 1;
int ust = 63;
int tahmin;
int sayac = 0;
Random rnd = new Random();
int tutulan = rnd.Next(63) + 1;
dnz:
tahmin = (alt + ust) / 2;
sayac++;
if (tahmin == tutulan)
{
Console.Write("Tutulan = "+tutulan+
sayac = " + sayac);
}
else
{
if (tahmin > tutulan)
{
ust = tahmin;
}
else
{
alt = tahmin;
}
goto dnz;
}
Console.ReadLine();
}
}
}
```

27. Girilen decimal (onluk) bir sayının binary (ikilik) bir sayıya dönüştüren programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- sayi,i=0,top=0
- 3- sayi gir
- 4- sayi>=2 olana kadar 7. adıma kadar olan işlemleri yaptır
- 5- $top = top + ((sayi \% 2) * (10^i))$
- 6- $sayi = sayi / 2$
- 7- $i++$
- 8- $top = top + (sayi * (10^i))$
- 9- Yazdır top
- 10- Bitir

Açıklama:

Bu soru, kitapta çok kez çeşitlerini çözdüğümüz sorulardan biridir. Bildiğimiz üzere, bilgisayar 2'lik (binary) sayılarla işler. O zaman bir 10'luk tabandaki sayıyı çevirmesi gerekir. Bu durum matematik derslerinde taban değiştirme olarak da daha önce karşımıza çıkmıştır. Bu soruda döngü kullanarak algoritma ve akış diyagramımızı şekillendirdik. Diğer sorularda da döngü mantığını kullandık ama algoritmada veya akış diyagramında eğer mekanizması ile gösterdik ve ona göre dallandırdık. Burada girilen 10'luk tabandaki sayıyı 2'ye böldürüp kalanı 10'nun

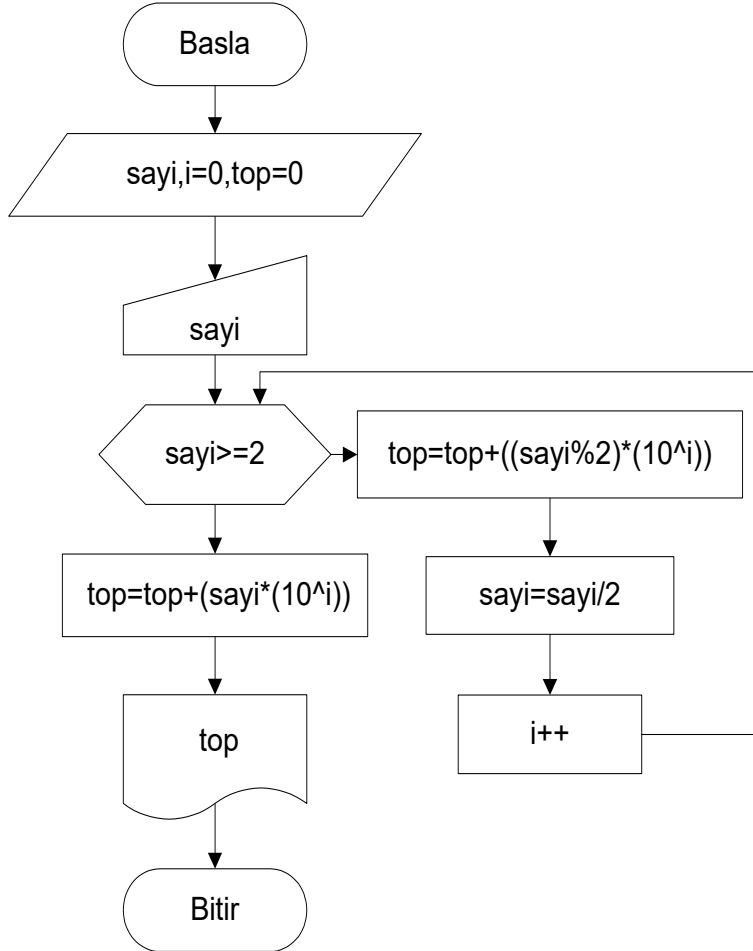
katlarına sırayla (0 dan itibaren) çarptırıp bir değişkende topladık. Aslında bu soruyu dizi mantığıyla yapmak daha kolaydır fakat burada bir kandırmaca yaparak soruyu çözüyoruz. Buna göre topladığımız sayılar yine 10'luk sistem olmasına rağmen ekrana basınca ikilik gibi duracaktır. Algoritma ve bilgisayar programcılığı böyle bir şeydir. ☺

Algoritma Testi:

sayi	i	(Sayi % 2) * 10 ⁱ	Top
5	0	1*1	1
2	1	0*10	1
1	2	1*100	101

Not(sayi 1 olunca işlem 1 kereye mahsus (sayi*10ⁱ) yapıcak ve biticek)

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

#include <math.h>

main(void)
{
    int sayi,i=0,top=0;

    printf(" Bir sayı giriniz");

    scanf("%d",&sayi);

    while(sayi>=2)
    {
        top=top+(sayi%2)*pow(10,i);

        sayi=sayi/2;

        i++;

    }

    top=top+sayi*pow(10,i);

    printf("binary values %d",top);

    getch();

}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi;
            int i = 0;
            double top = 0;
            Console.Write("Sayıyı giriniz = ");
            sayi = Convert.ToInt32(Console.ReadLine());
            while (sayi >= 2)
            {
                top = top + (sayi % 2) * Math.Pow(10, i);
                sayi = sayi / 2;
                i++;
            }
            top = top + sayi * Math.Pow(10, i);
            Console.Write("Binary values = " + top);
            Console.ReadLine();
        }
    }
}
```

28. Binary olarak girilen sayıyı decimal sayıya çeviren programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- sayi(binary),sayac=0 ,top=0,bas
- 3- sayi gir
- 4- Eğer sayi>9 ise devam et,
değilse 7'e git
- 5- Bas=sayi%10,sayi=sayi/10
- 6- Top=top + (bas^sayac),sayac++
4'e git
- 7- Top=top+(sayi^sayac)
- 8- Yazdır top
- 9- Bitir

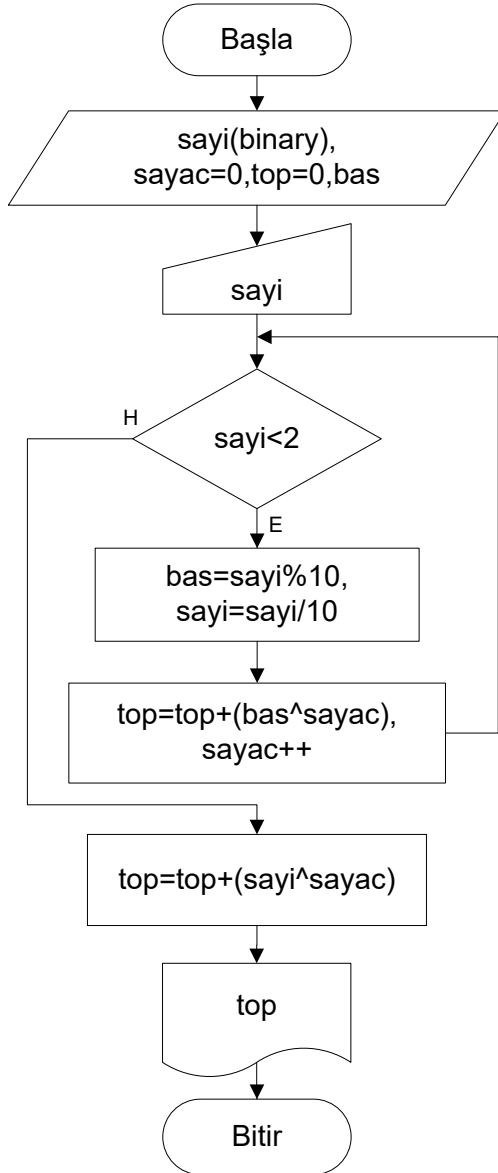
Algoritma Testi:

sayi	Sayac	Bas	(top+bas*2^sayac)	Top
101	0	1	0+1*1	1
10	1	0	1+0*2	1
1	2	1	1+1*4	5

Açıklama:

Bu soruda bir önceki sorunun tersini yapmamız gerekmektedir. Bu sefer de ikilik düzende verilen 1 ve 0 (bitlerden) dan oluşan bir ikilik sayıyı 10'luk tabana çevireceğiz. Bunun için 2'lik sayının birler basamağından başlayarak en büyük basamağına kadar sayıları 10'nun katları ile çarpıp (0 dan başlamak koşulu ile) bir değişkende toplayacağız. Sonra bunu ekrana basacağız. Bu soruyu çözerken Soru 10'daki basamaklara ayırma metodunu kullanacağız.

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

#include <math.h>

main(void)
{
    int sayi,sayac=0,top=0,bas;

    printf(" Bir sayı giriniz");

    scanf("%d",&sayi);

    while(sayi>9)

    {

        bas=sayi%10;

        sayi=sayi/10;

        top=top+bas*pow(2,sayac);

        sayac++;

    }

    top=top+sayi*pow(2,sayac);

    printf("decimal value %d",top);

    getch();

}
```

C#Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi;
            int sayac = 0;
            double top = 0;
            int bas;
            Console.Write("Sayıyı giriniz = ");
            sayi = Convert.ToInt32(Console.ReadLine());
            while (sayi > 9)
            {
                bas = sayi % 10;
                sayi = sayi / 10;
                top = top + bas * Math.Pow(2, sayac);
                sayac++;
            }
            top = top + sayi * Math.Pow(2, sayac);
            Console.WriteLine("Decimal value = " + top);
            Console.ReadLine();
        }
    }
}
```

29. Verilen yılın artık yıl olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- yıl
- 3- yıl gir
- 4- Eğer yıl %4=0 ise yazdır
"artık yıldır"
değilse yazdır "artık yıl değildir"
- 5- Bitir

Algoritma Testi:

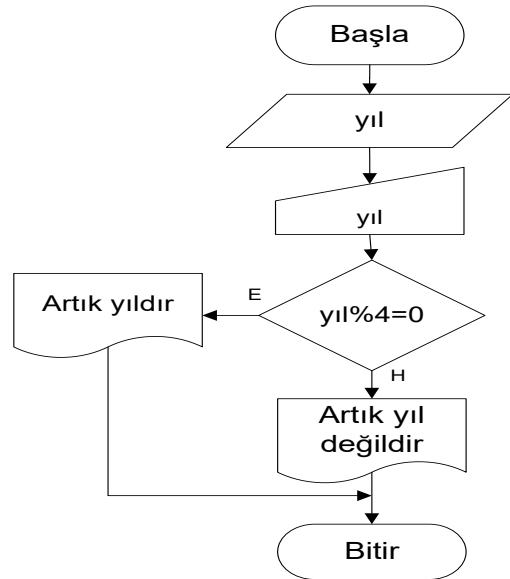
Yıl	İşlem	Sonuç
2000	$2000 \% 4 = 0$	Artık Yıldır
2001	$2001 \% 4 = 1$	Artık Yıl değildir.
2009	$2009 \% 4 = 1$	Artık Yıl değildir.

Açıklama:

Dünyamız Güneş çevresindeki dolanımını 365 gün 6 saatte tamamlar. Her yıl 365 günden artan 6 saatler 4 yılda bir 24 saat, yani 1 gün eder. Bu bir gün 4 yılda bir Şubat ayına eklenir. Böylelikle, 28 gün olan Şubat ayı 4 yılda bir 29 gün olur. Buna artık yıl denir. 4'le bölünebilen yıllar artık yıldır. İşte girilen sayının artık yılı olup olmadığını bulacağımız bu soru aslında basit bir sorudur. Sadece bir eğer ile işlemimizi tamamlayabiliriz. Bunun

için ekrana sonuca göre mesajımızı yazdırırız. Artık yıl, girilen yılı 4' e bölerek elde edilir. Sonuçta kalan 0 ise bu yıl artık yıldır, değilse artık yıl değildir, denilir ve program bitirilir.

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

#include <math.h>

main(void)
{
    int yil;

    printf("yili giriniz");

    scanf("%d",&yil);

    if(fmod(yil,4)==0)

        printf("artik yildir");

    else

        printf("artik yil degildir");

    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int yil;
            Console.Write("Yılı giriniz = ");
            yil = Convert.ToInt32(Console.ReadLine());
            if ((yil % 4) == 0)
            {
                Console.Write("Artik yildir");
            }
            else
            {
                Console.Write("Artik yıl değildir");
            }
            Console.ReadLine();
        }
    }
}
```

30. Boyu ile kilosunu girilen kişinin şişman mı, zayıf mı yoksa ideal kiloda mı olduğunu gösteren programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- Kilo ,boy
- 3- Kilo, boy gir
- 4- $boy = boy \% 100$
- 5- Eğer $boy - kilo \geq 11$ ise 7'ye git
değilse devam
- 6- Eğer $boy - kilo = 11$ ise 8'e git
değilse devam
- 7- Eğer $boy - kilo \leq 11$ ise 9'a git
değilse devam
- 8- Yazdır kilo alman gerek
- 9- Yazdır ideal kilo
- 10- Yazdır kilo vermen gerek
- 11- Bitir

"kilo vermelisin" mesajını verdiğinde, babamın yüzünü görmeliydiniz. ☺) Bu durumda 3 şıkkımız olduğu için 3-1 yani 2 eğer kullanmamız gerekir. Algoritma testinde bunu daha iyi kavrayacağız.

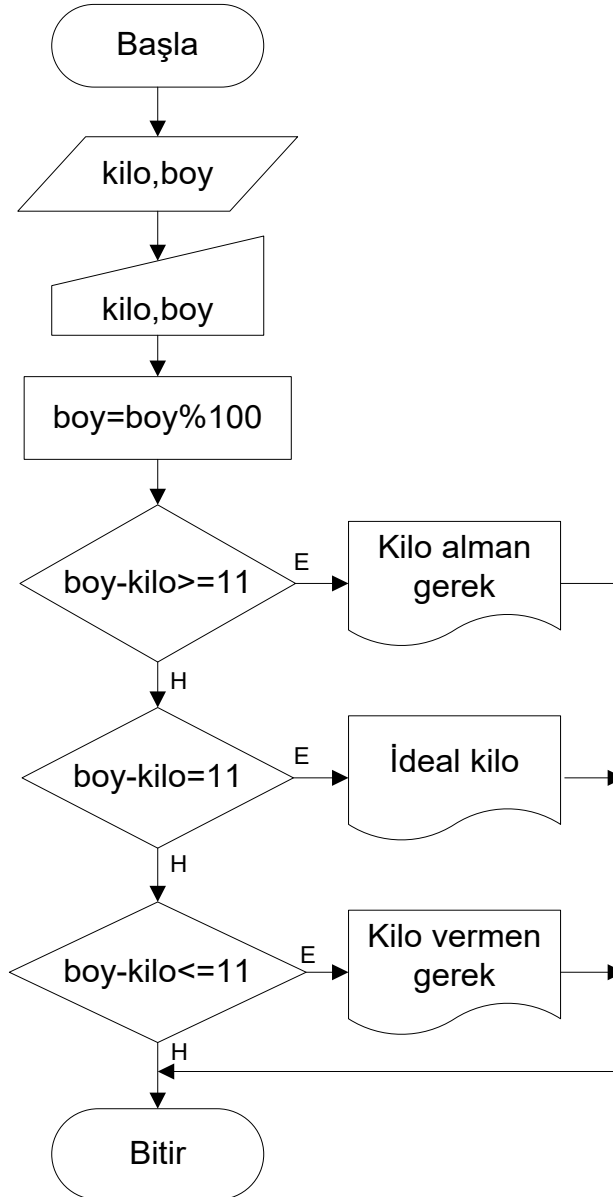
Algoritma Testi:

kilo	boy	İşlem	Sonuç
80	180	$(180 - 100 - 80) < 11$	Kilo vermelisiniz
71	182	$(182 - 100 - 71) = 11$	İdeal
60	190	$(190 - 100 - 60) > 11$	Kilo Almalısınız

Açıklama:

Bu soru da if (eğer)'in çok kullanıldığı bir sorudur. Excel'de çok kullandığımız bu uygulama eğer (karar) yapısını öğrenmek için önemlidir. Standartlara göre bir kişinin boyu ve kilosunu arası 11 ise kişi ideal konumdadır. 11'den büyük veya küçük olduğu durumlar için ekrana basılır. (İlk programlamayı öğrendiğimde pascalda bunu yazmıştım ve babama göstermiştim. Babamın bilgisayarına

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main(void)
{
    int kilo,boy;
    printf("kilonuzu giriniz");
    scanf("%d",&kilo);
    printf("boyunuzu giriniz");
    scanf("%d",&boy);
    boy=boy%100;
    if(boy-kilo>=11)
        printf("kilo almanız gerek");
    if(boy-kilo==11)
        printf("kilonuz ideal");
    else
        printf("kilo vermeniz gerek");
    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int kilo, boy;
            Console.Write("Kilonuzu giriniz = ");

            kilo=Convert.ToInt32(Console.ReadLine());
            Console.Write("Boyunuzu giriniz = ");

            boy=Convert.ToInt32(Console.ReadLine());
            boy = boy % 100;
            if (boy - kilo >= 11)
            {
                Console.Write("Kilo almanız gerek");
            }
            if (boy - kilo == 11)
            {
                Console.Write("Kilonuz ideal");
            }
            else
            {
                Console.Write("Kilo vermeniz gerek");
            }
            Console.ReadLine();
        }
    }
}
```

31. Dairenin alanını ve çevresini bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $\pi = 3.14$, alan, çevre
- 3- r gir
- 4- $\text{Alan} = \pi * r^2$
- 5- $\text{Çevre} = 2 * \pi * r$
- 6- Yazdır "çevresi:" çevre , "alanı:" alan
- 7- Bitir

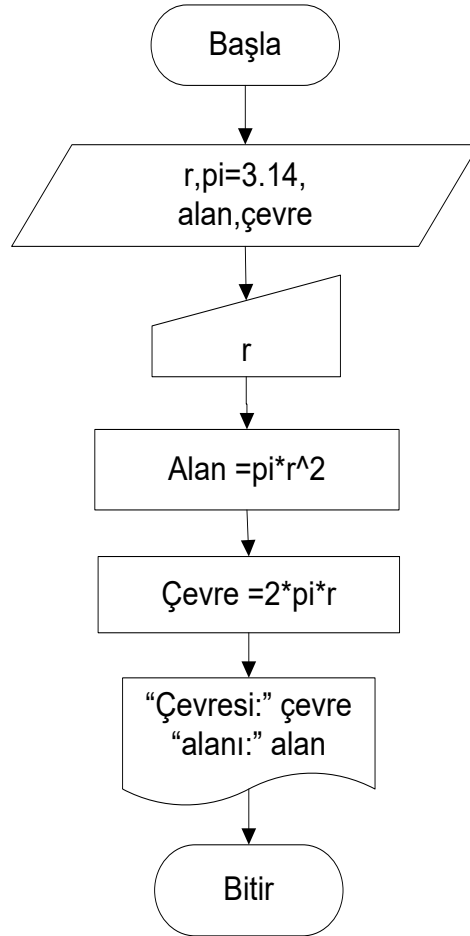
Açıklama:

Bu soruda sadece formülü bilmek yeterli olacaktır. Algoritma, klasik bir matematik problemi olarak karşımıza çıkmıştır. Dairenin alanında kullandığımız yarıçap ve pi değerini bilirsek , bunu formülde yerine koyup ekrana basmamız yeterli olacaktır.

Algoritma Testi:

yarıçap	Pi	$\pi * \text{Yarıçap}^2$
3	3,14	$3,14 * 9 = 28,26$
1	3,14	$3,14 * 1 = 3,14$

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

#include <math.h>

main(void)

{

int r;

float alan,cevre,pi=3.14;

printf("yaricapi giriniz");

scanf("%d",&r);

alan=pi*pow(r,2);

cevre=2*pi*r;

printf("Dairenin cevresi:%f\n alani:%f",cevre,alan);

getch();

}
```

C# Kod:

```
using System;

namespace dmg
{
    class Program
    {
        static void Main(string[] args)
        {
            int r;

            double alan, cevre, pi = 3.14;

            Console.Write("Yarıçapı giriniz = ");

            r = Convert.ToInt32(Console.ReadLine());

            alan = pi * Math.Pow(r, 2);

            cevre = 2 * pi * r;

            Console.Write("Dairenin cevresi = " + cevre +
                " Alanı = " + alan);

            Console.ReadLine();
        }
    }
}
```

32. Kenarları A,B,C,D olan bir dörtgenin kare olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- a,b,c,d
- 3- a,b,c,d gir
- 4- Eğer $a+c=b+d$ eşitse 5 e git
değilse devam et
- 5- Yazdır "kare değildir"
- 6- Yazdır "karedir"
- 7- Bitir

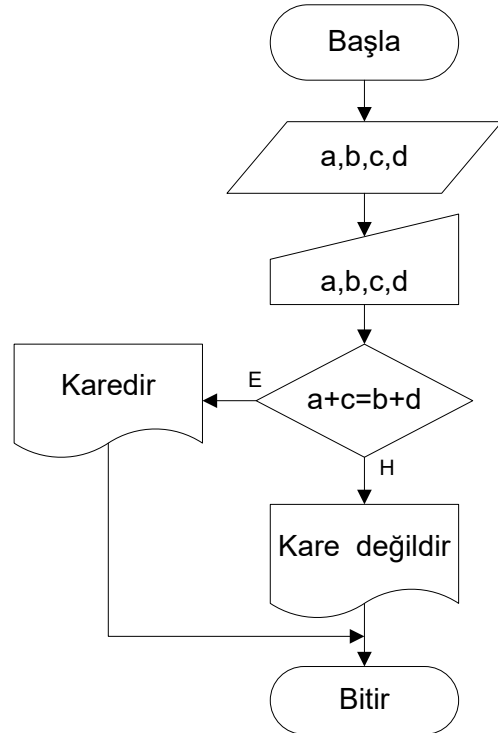
Açıklama:

Bu soru da if (eğer)'in kullanıldığı bir sorudur. Aslında bu soru çok çeşitli şekillerde çözülebilir, fakat bizim seçtiğimiz yol dikdörtgenin karşılıklı kenarları eşittire dayanmaktadır. Karşılıklı kenarları toplayıp birbirine eşitlediğimizde sonuç aynı ise bu dikdörtgen karedir. Unutulmamalıdır ki bir dikdörtgende karşı köşeler yani $a=c$ ve $b=d$ olmak zorundadır. Sonucun doğruluğuna göre de ekrana iki mesaj yazıp, programı sonlandırırız.

Algoritma Testi:

A	B	C	D	İşlem($A+C=B+D$)	Sonuç
3	6	3	6	$6=12$	Kare değildir
3	3	3	3	$16=6$	Karedir.

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

#include <math.h>

main(void)

{
    int a,b,c,d;
    printf("kenarlari giriniz");
    scanf("%d%d%d%d",&a,&b,&c,&d);
    if( a+c==b+d)
        printf("karedir");
    else
        printf("kare degildir");
    getch();
}
```

C# Kod:

```
using System;

namespace dmg
{
    class Program
    {
        static void Main(string[] args)
        {
            int a, b, c, d;
            Console.WriteLine("Kenarlari giriniz");
            a = Convert.ToInt32(Console.ReadLine());
            b = Convert.ToInt32(Console.ReadLine());
            c = Convert.ToInt32(Console.ReadLine());
            d = Convert.ToInt32(Console.ReadLine());
            if (a + c == b + d)
            {
                Console.Write("Karedir");
            }
            else
            {
                Console.Write("Kare degildir");
            }
            Console.ReadLine();
        }
    }
}
```

33. Bir uçak 15 dk boyunca düzgün hızlanarak hızı 480 km/s oluyor. Sonra 20 dk sabit hızla gidiyor ve 15 dk boyunca düzgün yavaşlayarak hızı sıfır oluyor. Herhangi bir t anında hızı veren algoritmayı ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- t ,v (t gir)
- 3- Eğer ($t < 15$) ise $v = 32 * t$ 5 e git,
değilse devam et
- 4- Eğer ($t > 35$) ise $v = 1600 - (32 * t)$,
değilse $v = 480$
- 5- Yazdır v
- 6- Bitir

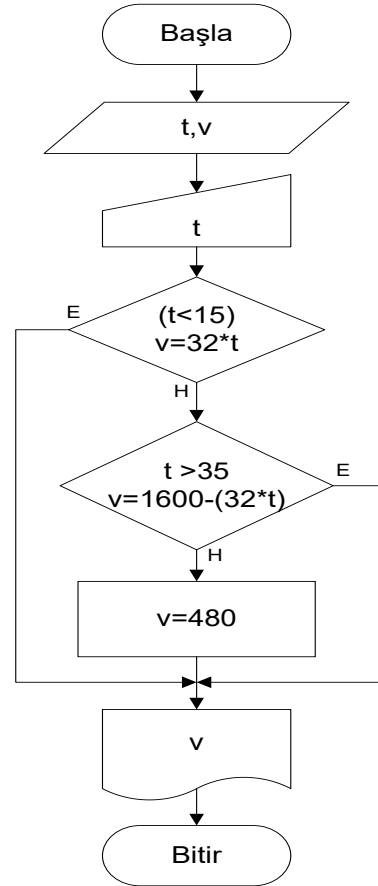
Açıklama:

Bu soru da dairenin alanı gibi değerleri formüle edip bir şarta göre değerlendirerek ekrana basan programın algoritmasıdır. Matematiksel ve formüle dayalı bir soru olmasına rağmen kendimizi bu konuda daha fazla geliştirebilmek açısından önemlidir.

Algoritma Testi:

t	v	İşlem
10	320	$V = t * 32$
20	480	Sabit
40	320	$V = 1600 - t * 32$

Akış diyagramı



C Kod:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
main(void)
{
    int t,v;
    printf("t degerini giriniz");
    scanf("%d",&t);
    if( t<15)
    {
        v=32*t;
        printf("%d",v);
    }
    else
    {
        if(t>35)
        v=1600-(32*t);
        else
        v=480;
        printf("%d",v);
    }
    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int t, v;
            Console.Write("t değeri giriniz = ");
            t = Convert.ToInt32(Console.ReadLine());
            if (t < 15)
            {
                v = 32 * t;
                Console.Write(v);
            }
            else
            {
                if (t > 35)
                {
                    v = 1600 - (32 * t);
                }
                else
                {
                    v = 480;
                }
                Console.Write(v);
            }
            Console.ReadLine();
        }
    }
}
```

34. Girilen dört basamaklı sayılardan ilk iki basamağı ile son iki basamağının toplamının karesi, sayının kendine eşit olan sayılara orijinal sayı denir. Girilen sayının orijinal olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- sayı , top=0,temp=0,x
- 3- sayı gir
- 4- x=sayı
- 5- temp=sayı%100, top=temp+top
sayı=sayı/100, top=sayı+top
top=top^2
- 6- Eğer x=top ise yazdır "Orjinaldir",
değilse "Orijinal değildir"
- 7- Bitir

Açıklama:

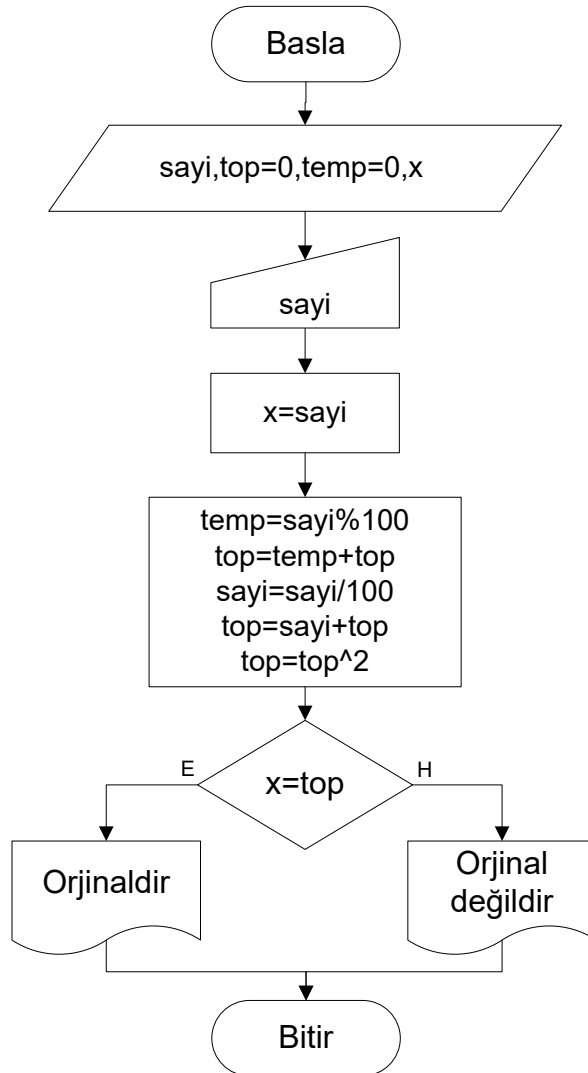
Kitabımızda sayılarla çok uğraşıyoruz. Fakat zaten algoritmanın ya da bilgisayarın temeli matematiğe dayanıyor. Bu soruda karşımıza orjinal sayı çıkıyor. İlk önce sayıyı alıp ilk 2 basamak değerini, sonra da son 2 basamağını buluyoruz. Yapılması gereken bu işlemi daha önceki sorularda da anlatmış ve örnek vermiştik. Burada 4 basamaklı bir sayının ilk 2 hanesini top değişkeninin içine atıyoruz sonra son iki basamağını bulup top

değişkeninin içine atarak, bu sayıyı karesini aldıktan sonra, X değişkenine attığımız ilk sayı değerine eşit mi diye bakıyoruz. Eşit ise "orijinal" diye ekrana bastırıyoruz. Hem döngü, hem de eğer (karar) mekanizmasını işletiyoruz. Bu soruları yapmamızdaki en önemli hedefimiz öğrenciyi artık döngü kurma ve eğer (if) kullanma konusunda geliştirmektir.

Algoritma Testi:

Sayı	X	Top	Top= x mi?
1212	1212	0+12	
121	1212	12+12	
12	1212	24^2	
Sonuç	1212	576	Orjinal Değildir

Akış Diyagramı



CKod :

```
#include <stdio.h>
#include <conio.h>
int sayi;
int top = 0;
int temp = 0;
int x;
main()
{
    printf("Sayi giriniz = ");
    scanf("%d",&sayi);
    x=sayi;
    temp = sayi % 100;
    top = temp + top;
    sayi = sayi / 100;
    top=top+sayi;
    top=top^2;
    if (x== top)
        printf("Orjinaldir");
    else
        printf("Orjinal değildir");
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi;
            int top = 0;
            int temp = 0;
            int x;
            Console.Write("Sayıyı giriniz = ");
            sayi = Convert.ToInt32(Console.ReadLine());
            x=sayi;
            temp = sayi % 100;
            top = temp + top;
            sayi = sayi / 100;
            top=top+sayi;
            top=top^2;
            if (x == top)
            {
                Console.Write("Orjinaldir");
            }
            else
            {
                Console.Write("Orjinal değildir");
            }
            Console.ReadLine();
        }
    }
}
```

35. 1 ile 500 arasındaki tam sayılardan tek sayıların toplamı ile çift sayıların toplamının farkı negatif mi, pozitif mi olduğunu bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- sayaç=1,tek=0,çift=0
- 3- tek=tek+sayaç
- 4- çift=çift+(sayaç+1)
- 5- eğer sayaç=499 ise 8 adıma git
- 6- sayaç=sayaç+2, 3. adıma git
- 7- Eğer tek-çift<0 ise yazdır “fark negatif” değilse “fark pozitif”
- 8- Bitir

sorulardır. Ancak bu soruda dikkati çeken algoritmanın birinde, (bu çözümde) eğer ile işlemin tamamlanmasıdır. (Algoritma olarak yalnız)

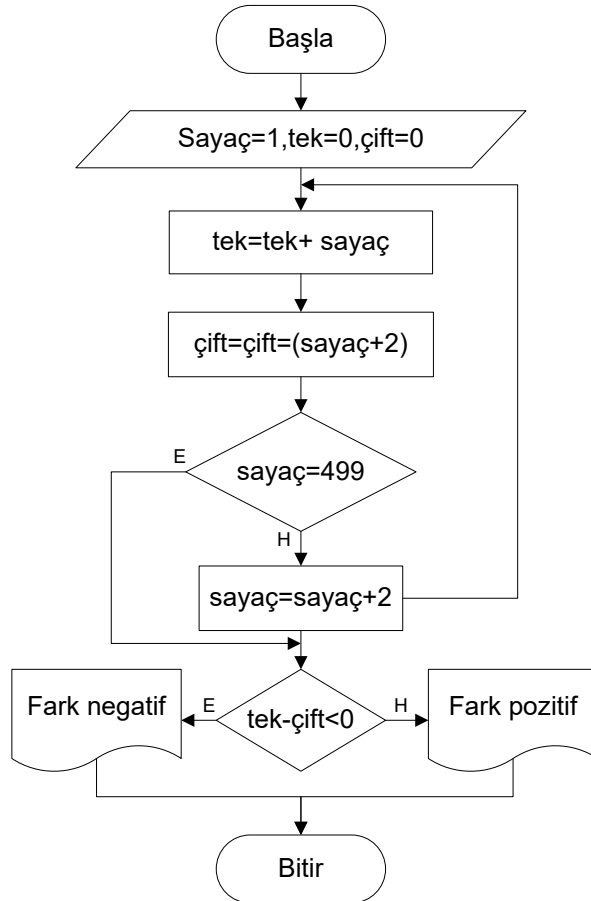
Algoritma Testi:

TekToplamı	ÇiftToplamı
1	2
3	4
...	...
499	500
Fark(cifttop-tektop)	Negatif

Açıklama:

Bu sorumuzda farkın her zaman negatif olacağı aslında bellidir ama daha önce de ifade ettiğimiz gibi 1 ile 500 arası olduğu için döngü ve çıkan farkın negatif mi yoksa pozitif mi olduğunu bulmak için eğer kullanmayı vurgulaya vurgulaya öğretmeyi amaçlayan bir sorudur. Bilmeliyiz ki en temel ve gerekli programlama işlemleri bunlardır. Bunların üstüne yeni eklenerek sanki bir binanın katlarını çıkıyor gibi kendimizi geliştirmemiz gerekmektedir. Arka arkaya verdiğimiz bu iki soru aynı doğrultuda

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    int sayac,tek=0,cift=0;
    for(sayac=1;sayac<500;sayac++)
    {
        tek=tek+sayac;
        cift=cift+(sayac+1);
    }
    if((tek-cift)<0)
        printf ("fark negatif");
    else
        printf("fark pozitif");
    getch();
}
```

C#Kod

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayac;
            int tek = 0;
            int cift = 0;
            for (sayac = 1; sayac < 500; sayac++)
            {
                tek = tek + sayac;
                cift = cift + (sayac + 1);
            }
            if ((tek - cift) < 0)
            {
                Console.WriteLine("Fark negatif");
            }
            else
            {
                Console.WriteLine("Fark pozitif");
            }
            Console.ReadLine();
        }
    }
}
```

36.1 ile 500 arasındaki tam sayılardan tek sayıların toplamı ile çift sayıların toplamının farkı negatif mi, pozitif mi olduğunu bulan programın algoritma ve akış diyagramını çiziniz.(Döngü Kullanarak)

Algoritma:

- 1- Başla
- 2- sayac=1,tek=0,cift=0
- 3- sayac=1'den sayac<=500 olana kadar 6.adıma kadar olan işlemleri yaptır
- 4- tek=tek+sayac
- 5- cift=cift+(sayac+1)
- 6- Eğer (tek-cift)<0) ise yazdır "Negatif "değilse yazdır "Pozitif"
- 7- Bitir

algoritmanın birinde, (bu çözümde) döngü kullanarak işlemin tamamlanmasıdır. (Algoritma olarak yalnız)

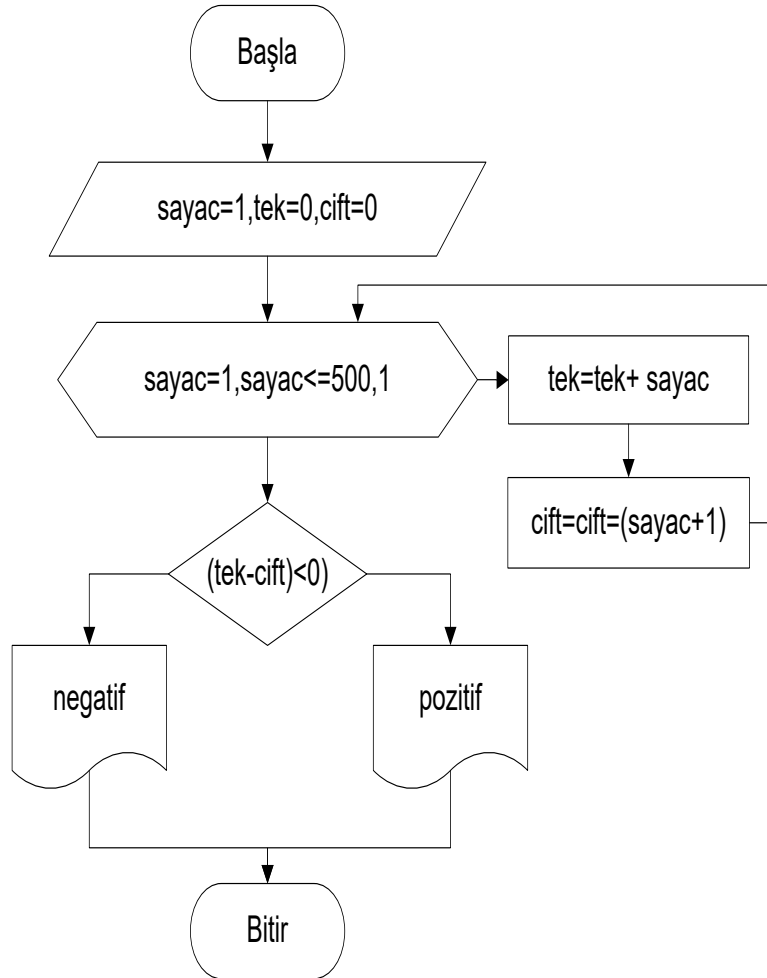
Algoritma Testi:

TekToplamı	ÇiftToplamı
1	2
3	4
...	...
499	500
Fark(cifttop-tektop)	Negatif

Açıklama:

Bu sorumuzda farkın her zaman negatif olacağı aslında bellidir ama daha önce de ifade ettiğimiz gibi 1 ile 500 arası olduğu için döngü ve çıkan farkın negatif mi yoksa pozitif mi olduğunu bulmak için eğer kullanmayı vurgulaya vurgulaya öğretmeyi amaçlayan bir sorudur. Bilmeliyiz ki en temel ve gerekli programlama işlemleri bunlardır. Bunların üstüne yeni eklenerek sanki bir binanın katlarını çıkıyor gibi kendimizi geliştirmemiz gerekmektedir. Arka arkaya verdiğimiz bu iki soru aynı doğrultuda sorulardır. Ancak bu soruda dikkati çeken

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    int sayac,tek=0,cift=0;
    for(sayac=1;sayac<500;sayac++)
    {
        tek=tek+sayac;
        cift=cift+(sayac+1);
    }
    if((tek-cift)<0)
        printf ("fark negatif");
    else
        printf("fark pozitif");
    getch();
}
```

C#Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayac;
            int tek = 0;
            int cift = 0;
            for (sayac = 1; sayac < 500; sayac++)
            {
                tek = tek + sayac;
                cift = cift + (sayac + 1);
            }
            if ((tek - cift) < 0)
            {
                Console.WriteLine("Fark negatif");
            }
            else
            {
                Console.WriteLine("Fark pozitif");
            }
            Console.ReadLine();
        }
    }
}
```

37.4 haneli bir sayının birler, onlar, yüzler ve binler hanesini bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- sayi,yuzler,onlar,birler,binler,temp
- 3- sayi gir
- 4- $temp=sayi\%1000$
- 5- $onlar=(sayi\%100)/10$
- 6- $yuzler=temp/100$
- 7- $binler=sayi/1000$
- 8- $birler=sayi-(binler*1000)-$
 $(yuzler*100)-(onlar*10)$
- 9- Yazdır binler,yuzler,onlar,birler
- 10- Bitir

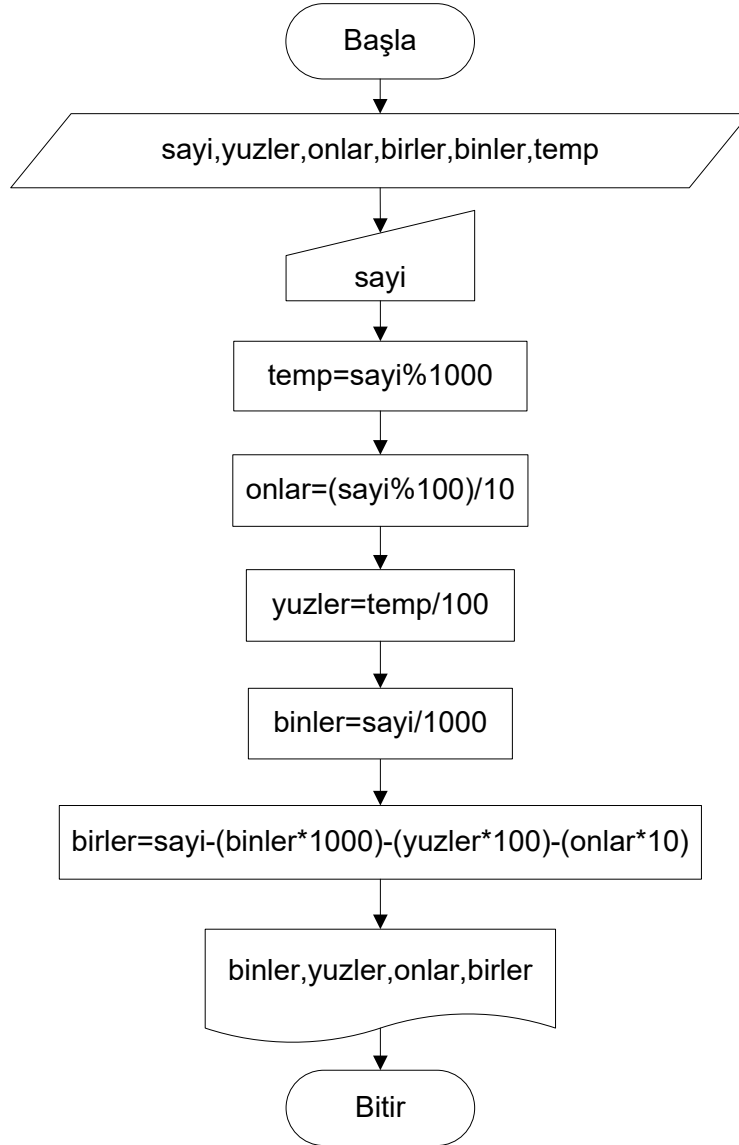
Algoritma Testi:

Sayi	binler	yüzler	onlar	Birler
1234	-	-	-	$1234\%1000= 4$
123	-	-	$123\%10= 3$	4
12	-	$12\%10=2$	3	4
1	1(sayi)	2	3	4

Açıklama:

Bu soruda sınırları verilmiş yani 4 basamaklı bir sayının basamaklarını ayıracağız. Sınırlar belli olduğu için hiç döngü kullanmamıza gerek yoktur. Statik sorular çok beğenilen sorular değildir. Çünkü bu soru sadece 4 basamak için işimizi görecektir. Bunun için unutmayalım iyi bir program dinamik olmalı yani girilen her sayı ya da değere göre işimize yarmalıdır. Buna her zaman dikkat etmenizi tavsiye ediyoruz. Bu soruda yeniden tekrarlayalım ki $\% \rightarrow$ kalanı verir . $/ \rightarrow$ bölüm (tam kısmı) verir.

Akış Diyagramı



C Kod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    int sayi,birler,onlar,yuzler, binler,temp;

    scanf("%d",&sayi);

    temp=sayi%1000;

    onlar=(sayi%100)/10;

    yuzler=temp/100;

    binler=sayi/1000;

    birler=sayi-(binler*1000)-(yuzler*100)-
    (onlar*10);

    printf("%d\n %d\n %d\n
    %d\n",binler,yuzler,onlar,birler);

    getch();

}
```

C#Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi, birler, onlar, yuzler, binler,
            temp;
            Console.Write("Sayıyı giriniz = ");
            sayi = Convert.ToInt32(Console.ReadLine());
            temp = sayi % 1000;
            onlar = (sayi % 100) / 10;
            yuzler = temp / 100;
            binler = sayi / 1000;
            birler = sayi - (binler * 1000) - (yuzler * 100) -
            (onlar * 10);
            Console.Write("Binler = " + binler + " Yüzler =
            " + yuzler + " Onlar = " + onlar + " Birler = " +
            birler);
            Console.ReadLine();
        }
    }
}
```

38. Rastgele girilen 50 sayıdan negatif olanların ve pozitif olanların sayısını bulan programın algoritma ve akış şemasını çiziniz.

Algoritma:

- 1- Başla
- 2- sayi, i=1,neg=0,poz=0
- 3- sayi gir
- 4- Eğer sayi=0 ise
7'ye git
- 5- Eğer sayi<0 ise neg=neg+1,
7'ye git
- 6- poz=poz+1
- 7- Eğer i=50 ise 10' a git
- 8- i=i+1 3'e git
- 9- Yazdır "Negatif:" neg , "Pozitif:" poz
- 10- Bitir

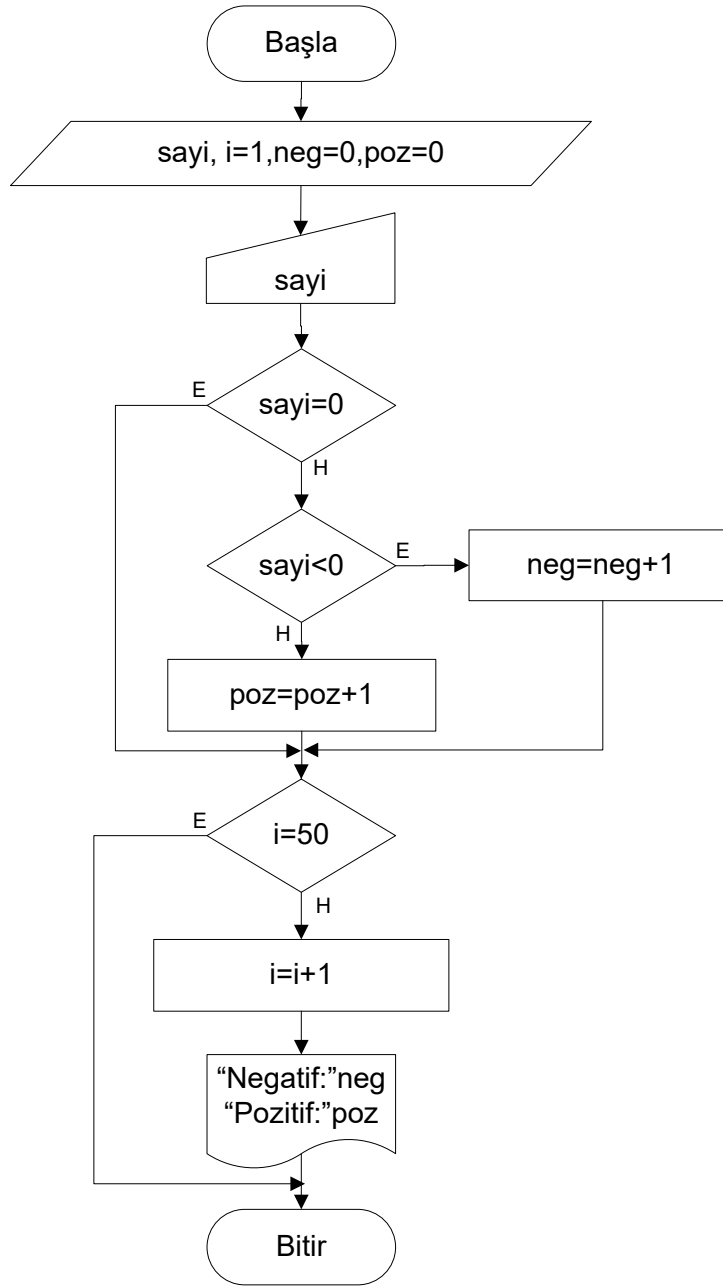
Açıklama:

Bu soruda sınırları verilmiş yani 4 basamaklı bir sayının basamaklarını ayıracağız. Sınırlar belli olduğu için hiç döngü kullanmamıza gerek yoktur. Statik sorular çok beğenilen sorular değildir. Çünkü bu soru sadece 4 basamak için işimizi görecektir. Bunun için unutmayalım iyi bir program dinamik olmalı yani girilen her sayı ya da değere göre işimize yaramalıdır. Buna her zaman dikkat etmenizi tavsiye ediyoruz. Bu soruda yeniden tekrarlayalım ki % → kalanı verir . / → bölüm (tam kısmı) verir.

Algoritma Testi :

Sayı	binler	yüzler	onlar	Birler
1234	-	-	-	$1234 \% 1000 = 4$
123	-	-	$123 \% 10 = 3$	4
12	-	$12 \% 10 = 2$	3	4
1	1(sayı)	2	3	4

Akış Diyagramı



CKod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    int sayi,i=1,neg=0,poz=0;

    for(i=1;i<=50;i++)
    {
        scanf("%d",&sayi);

        if(sayi<0)

            neg=neg+1;

        else

        {
            if(sayi>0)

                poz=poz+1;

        }

    }

    printf("negatif %d , pozitif %d ",neg,poz);

    getch();

}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi;
            int i = 1;
            int neg = 0;
            int poz = 0;
            for (i = 1; i <= 5; i++)
            {
                Console.Write(i+".Sayıyı giriniz = ");
                sayi = Convert.ToInt32(Console.ReadLine());
                if (sayi < 0)
                {
                    neg = neg + 1;
                }
                else
                {
                    if (sayi > 0)
                        poz = poz + 1;
                }
            }
            Console.Write("Negatif = " + neg + " Pozitif = "
                + poz);
            Console.ReadLine();
        }
    }
}
```

39.Sayı bulmaca oyunu programının algoritma ve akış diyagramı çiziniz.

Kural: 10 sefer hakkınız var

Kullanıcının tahminine göre tahmini yükselt yada tahmini azalt diyerek yönlendirme yapacak.

Bulduğu sefer sayısını ekrana basacak veya üzgünüz bir daha ki sefere diyecek.

Algoritma:

- 1- Başla
- 2- tutulan=(Random),sayac=1
- 3- tahmin gir
- 4- Eğer tahmin=tutulan ise 8'e git,
değilse devam et
- 5- Eğer tahmin<tutulan ise yazdır
"tahmini yükselt",
değilse "tahmini azalt"
- 6- Eğer sayac=10 ise devam et,
değilse sayac++ 3'e git
- 7- Yazdır "üzgünüz bi daha ki sefere"
- 8- Yazdır "tebrikler" sayac "seferde
buldunuz"
- 9- Bitir

ürettiriyoruz. Bu üretilen değeri tahmin etmek için 10 sefer dışarıdan sayı giriyoruz. Girilen sayı ile tutulan sayı aynı ise "tebrikler" yazıyor, fakat bir sayaç ile 10 seferi kontrol ediyoruz. Sayac 10 olduğunda hala sayıyı bulamıyorsak "üzgünüz" mesajı çıkıyor. Girdiğimiz sayıya göre de karar yapıları (eğer) ile "azalt" ya da "yükselt" diyerek yeni girecek sayımız için fikir sahibi oluyoruz.

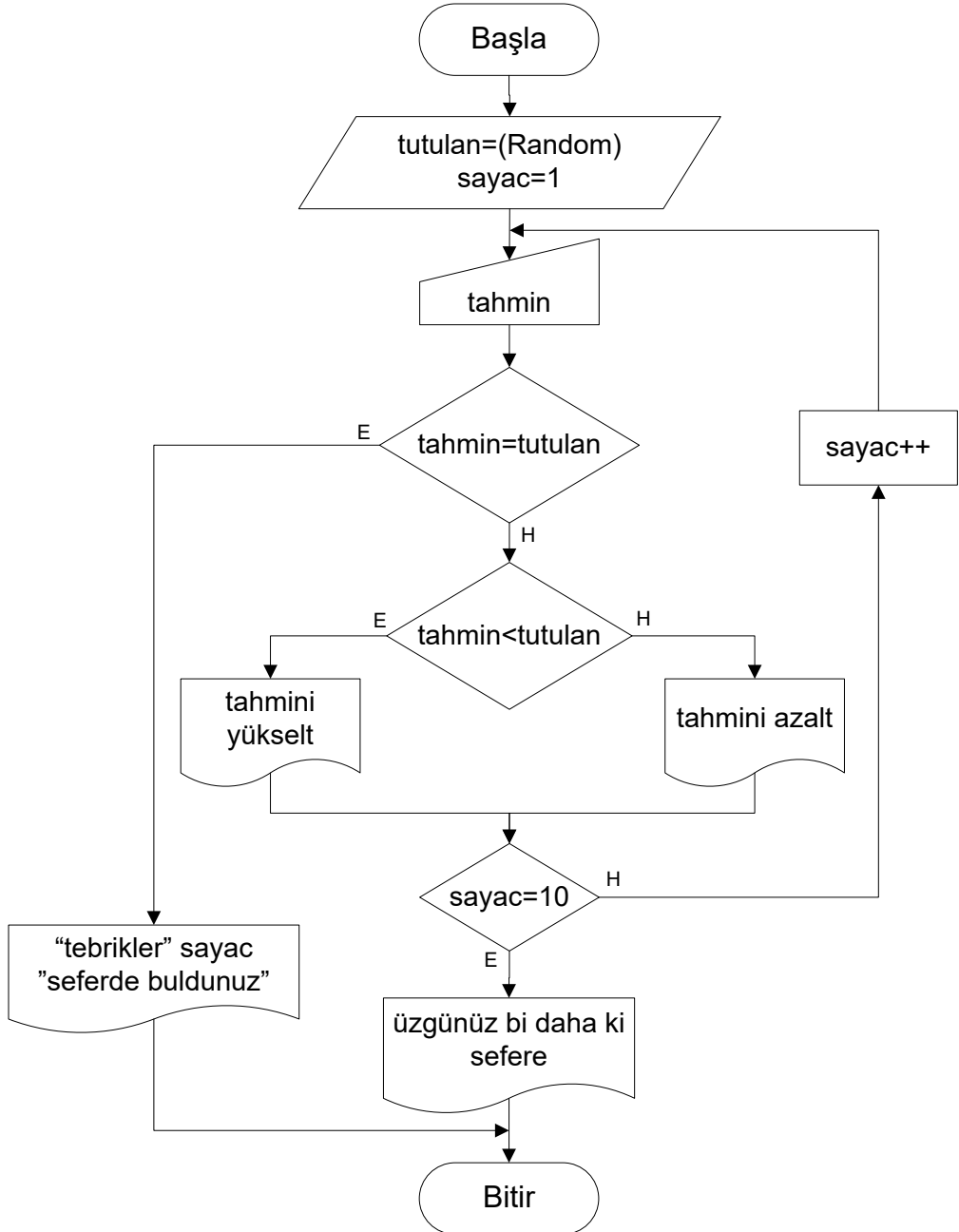
Algoritma Testi:

tutulan	Tahmin	Sayac	Mesaj
7	1	1	Yükselt
7	3	2	Yükselt
7	5	3	Yükselt
7	8	4	Azalt
7	7	5	5.sefer tebrikler

Açıklama:

Bu soru programlama ile ilk kez karşılaşan öğrenciler için eğlenceli ve basit bir sayı tutma oyunudur. Bunun için bilgisayara rastgele olarak 1 ile 10 arası bir değer

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>

main()
{
    int tahmin,tutulan,sayac;
    randomize();
    tutulan=((rand()% 10)+1);
    for(sayac=1;sayac<=10;sayac++)
    {
        scanf("%d",&tahmin);
        if(tahmin==tutulan)
            printf("tebrikler %d seferde
            buldunuz",sayac);
        else
        {
            if(tahmin<tutulan)
                printf("tahmini yükselt");
            else
                printf("tahmini azalt");
        }
        if(sayac==10)
            break;
    }
    printf("üzgünüz bi daha ki sefere");
    getch();
}
```

C#Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int tahmin, tutulan, sayac;
            Random rnd = new Random();
            tutulan = ((rnd.Next(99)) + 1);
            for (sayac = 1; sayac <= 10; sayac++)
            {
                Console.Write("Tahmininizi giriniz = ");
                tahmin =
                Convert.ToInt32(Console.ReadLine());
                if (tahmin == tutulan)
                {
                    Console.Write("Tuttuğum sayı " +
                    tutulan + " dı. Tebrikler " + sayac + " seferde
                    buldunuz");
                    break;
                }
                else if (tahmin < tutulan)
                {
                    Console.WriteLine("Tahmini yükselt");
                }
                else if (tahmin > tutulan)
                {
                    Console.WriteLine("Tahmini azalt");
                }
                else if (sayac == 10)
                {
                    Console.Write("Üzgünüz bi daha ki sefere");
                    break;
                }
            }
            Console.ReadLine();
        }
    }
}
```

40. 10 ile 200 arasındaki tam sayılardan 3 katının 2 fazlası 5 ile tam bölünebilen sayıları gösteren programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $i=10, k, a$
- 3- $k=i*3+2$
- 4- $a=(k/5)*5$
- 5- Eğer $k=a$ ise yazdır a
değilse devam et
- 6- Eğer $i = 200$ ise 8 a git
değilse devam et
- 7- $i=i+1$ 3. adıma git
- 8- Bitir

Açıklama:

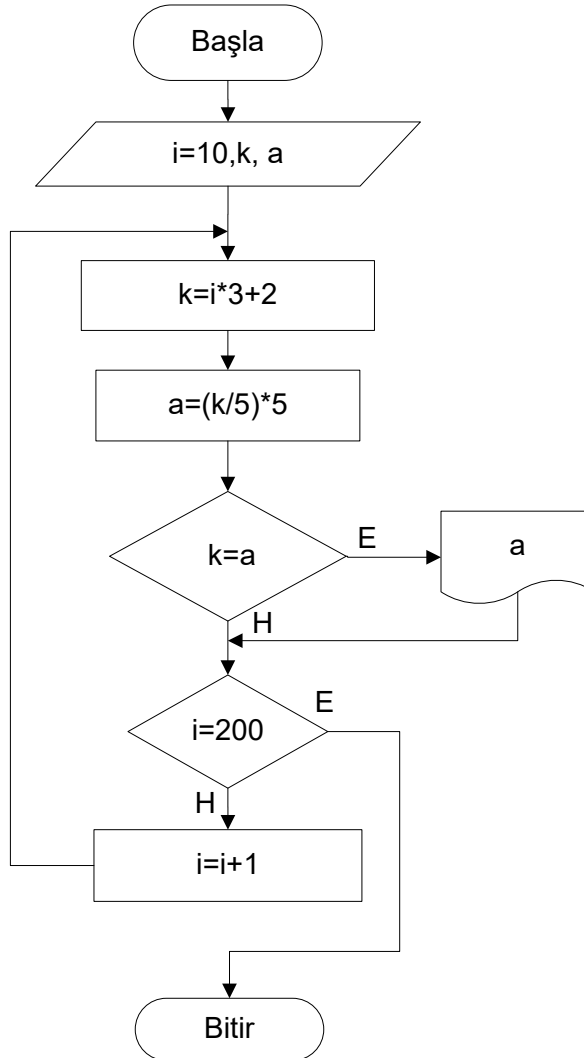
Bu soruda bir döngü olduğu hemen aklımıza gelmelidir. Çünkü bir sayı aralığı verilmiştir. (10 ile 200 arasında gibi.) Akış diyagramını çizerken döngü şeklini kullanmadık fakat ileri ki sorularda döngü şekli ile işlemimizi daha kolay yapabileceğiz. Bu, algoritma açısından da kolay olacaktır, sonuç da günlük dil kullanmaktayız. Bu tarz sorular aslında sizleri problem çözmeye alıştırmak için yapılmış basit sorulardır. Bu soruda da döngü vardır ve sayacımız yine (i)'dir. Bu i'yi unutmamak gerekir. Sayımız 10 için geldiğinde elimizde $k=10*3+2= 32$

olacaktır. k değerini 5 böldüğümüzde kalan 0 çıkmayacağı için ekrana bu i değeri de çıkmayacaktır. Fakat i yani sayaç değer 200 oluncaya kadar arttırılmalıdır.

Algoritma Testi:

i	K	a	Yazdır
10	32	30	-----
11	35	35	i=11
...	...	...	
200	602	600	-----

Akış Diyagramı:



C Kod:

```
#include <conio.h>

#include <stdio.h>

main()
{
    int i,k,a;
    for(i=10;i<=200;i++)
    {
        k=i*3+2;
        a=(k/5)*5;
        if(k==a)
            printf(a);
    }
    getch();
}
```

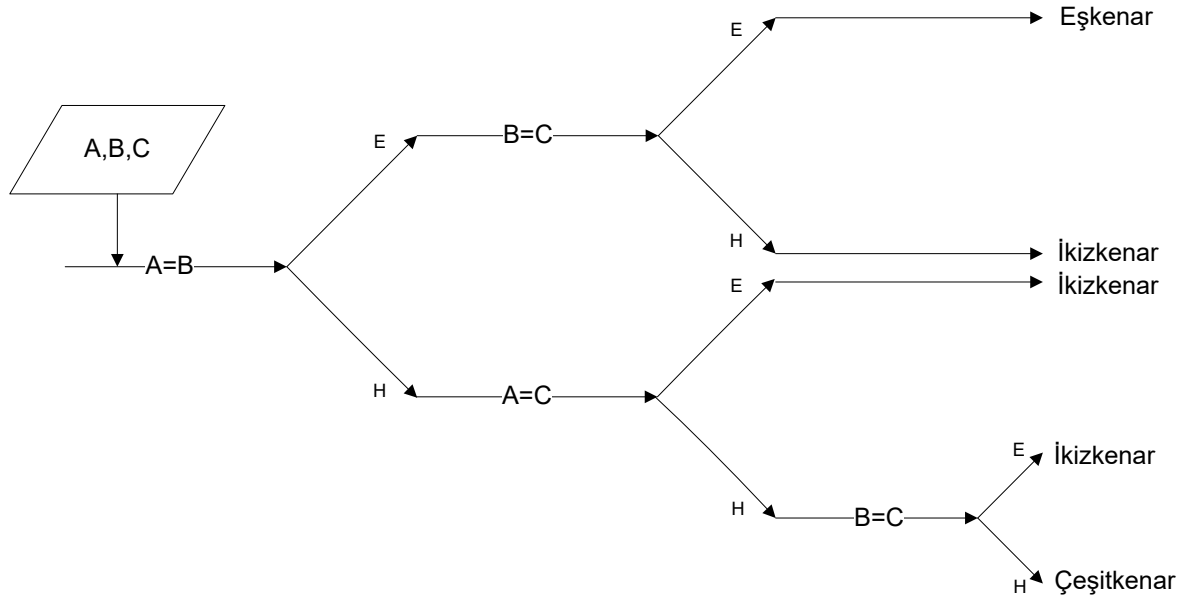
C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i, k, a;
            for (i = 10; i <= 200; i++)
            {
                k = (i * 3) + 2;
                a = (k / 5) * 5;
                if (k == a)
                {
                    Console.WriteLine(i);
                }
            }
            Console.ReadLine();
        }
    }
}
```

41. İç açıları verilen üçgenin karar ağacı, algoritma ve akış diyagramını çiziniz.

Karar Ağacı :



Algoritma:

- 1- Başla
- 2- a,b,c
- 3- a,b,c gir
- 4- Eğer a=b ise 5'e git,değilse 6'ya git
- 5- Eğer b=c ise yazdır "Üçgen eşkenardır" 8'ye git,değilse yazdır "Üçgen ikizkenardır" 8'ye git
- 6- Eğer a=c ise yazdır "Üçgen ikizkenardır" 8'ye git,değilse devam et
- 7- Eğer b=c ise yazdır "Üçgen ikizkenardır",değilse "Üçgen çeşitkenardır"
- 8- Bitir

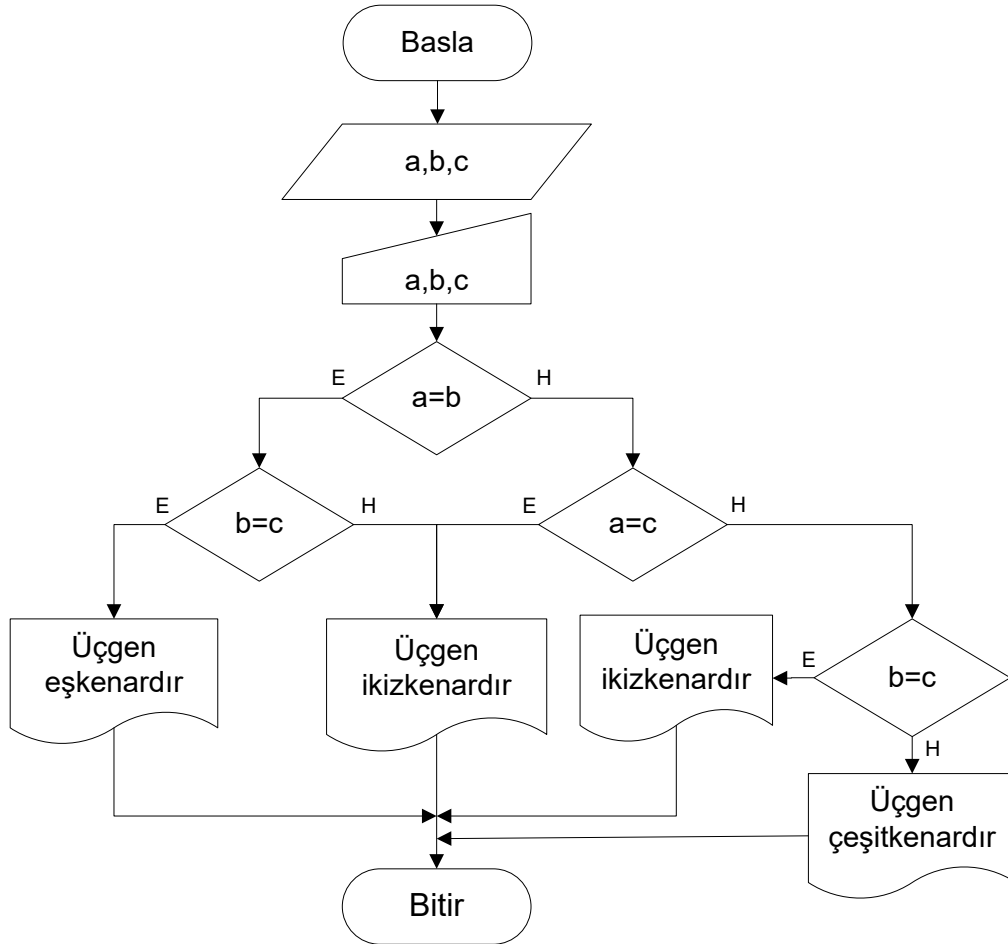
Açıklama:

Üçgenin karar ağacı yapısı ile ilgili bu soru, karar (eğer) yapılarını iyi öğreten bir soru tarzıdır. Bir üçgenin 3 kenarı vardır. Bu kenarların eşitliği veya eşitsizliği durumunda üçgen ya eşkenar ya ikizkenar ya da çeşitkenardır. Yukarıdaki şekilde de bunu açıkça görebiliriz. Soruda 3 kenarı dışarıdan giriyoruz. Eşit olma ya da olmama durumuna göre üçgenin tipini ekrana yazdırıyoruz.

Algoritma Testi:

a	b	c	işlem	Sonuç
3	3	1	$a=b$	İkizkenar
3	3	3	$a=b=c$	Eşkenar
3	1	1	$b=c$	İkizkenar
2	4	2	$a=c$	İkizkenar
3	4	5	Eşitsizlik	Çeşitkenar

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a,b,c;
    scanf("%d%d%d",&a,&b,&c);
    if(a==b)
    {
        if(b==c)
        {
            printf("ucgen eskenardir");
            goto dnz;
        }
        else
        {
            printf("Üçgen ikizkenardir");
            goto dnz;
        }
    }
    if(a==c)
    {
        printf("Üçgen ikizkenardir");
        goto dnz;
    }
    if(b==c)
    printf("uçgen ikizkenardir");
    else
    printf("Üçgen çeşitkenardir");
dnz:
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int a, b, c;
            a = Convert.ToInt32(Console.ReadLine());
            b = Convert.ToInt32(Console.ReadLine());
            c = Convert.ToInt32(Console.ReadLine());
            if (a == b)
            {
                if (b == c)
                {
                    Console.Write("Üçgen eskenardir");
                    goto dnz;
                }
                else
                {
                    Console.Write("Üçgen ikizkenardir");
                    goto dnz;
                }
            }
            if (a == c)
            {
                Console.Write("Üçgen ikizkenardir");
                goto dnz;
            }
            if (b == c)
            {
                Console.Write("uçgen ikizkenardir");
            }
            else
            {
                Console.Write("Üçgen
                çeşitkenardir");
            }
            dnz:
                Console.ReadLine();
        }
    }
}
```

42. Girilen sayının yaklaşık olarak karekökünü hesaplayan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- x,a,b,i
- 3- x gir
- 4- Eğer $(i*i) > a$ ise $b=i*i, a=(i-1)(i-1)$,
değilse $i++$ 4'e git
- 5- $x=(i-1)+((x-a)/(b-x))$
- 6- Yazdır x
- 7- Bitir

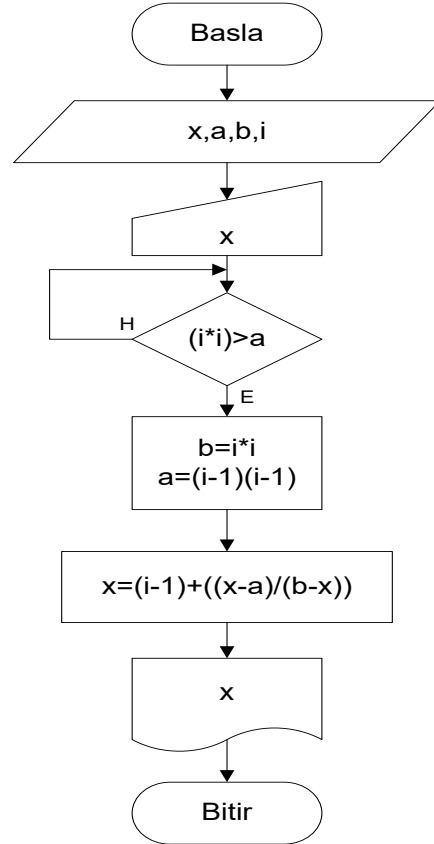
Açıklama:

Bu soru, ÖSYM'nin yaptığı ailesinin birinde bir yaklaşım sorusu olarak karşımıza çıkmıştı. Bu yaklaşımı kendimize göre ayarlayıp, soruyu da kitaba ekledik. Bu soruda, girilen tam kare olmayan sayının yaklaşık olarak değerini hesaplayan algoritma söz konusudur. Burada amaç $a < x < b$ mantığına göre değişir. x girilen sayımız, a da x ten küçük en yakın tam karesi olan sayı, b ise x'ten büyük en yakın tam karesi olan sayıdır. Bu değerleri algoritma bulacak ona göre formüle yerleştirip bize float yani ondalıklı sayı olarak sonuç verecektir.

Algoritma Testi:

X	a	B
17	16	25
sonuç	$X^{1/2} = 4.125$	

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    float a,b,i;

    float x;

    scanf("%f",&x);

    dnz:

    if((i*i)>x )
    {
        b=i*i;

        a=(i-1)*(i-1);

    }

    else

    {

        i++;

        goto dnz;

    }

    x=(i-1)+((x-a)/(b-x));

    printf("%f",x);

    getch();

}
```

C # Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            double a, b, i=0;
            double x;
            Console.Write("Sayıyı giriniz = ");
            x = Convert.ToDouble(Console.ReadLine());
            dnz:
            if (i * i > x)
            {
                b = i * i;
                a = (i - 1) * (i - 1);
            }
            else
            {
                i++;
                goto dnz;
            }
            x = (i - 1) + ((x - a) / (b - x));
            Console.Write(x);
            Console.ReadLine();
        }
    }
}
```

43. Obek ve Okek bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $s1, s2, x, y, okek$
- 3- $s1, s2$ gir
- 4- $x=s1, y=s2$
- 5- Eğer ($s1 < s2$) ise devam et, değilse 8'a git

- 6- Eğer $s1 > s2$ ise $s1=s1-s2$, değilse devam et
- 7- Eğer $s2 > s1$ ise $s2=s2-s1$, değilse 5'e git
- 8- Yazdır "obek:" $s1$
- 9- $okek=(x*y)/s1$
- 10- Yazdır "okek:" $okek$
- 11- Bitir

Açıklama:

Bu soru her sınavda karşımıza çıkan (ÖSS, ÖYS, DGS vs.) bir matematik sorusunun algoritmasıdır. Bu soruda iki sayı veriliyor. Ortak bölenlerin en büyüğü (obek) ile ortak katların en küçüğü (okek) bulunuyor. Bizim algoritmamıza göre $s1$ ve $s2$ iki sayı giriliyor.

İki sayı birbirinden farklı ise yandaki algoritma testinde de gösterdiğimiz gibi 2. adıma geçiliyor. Bu adım sayıların birbirine göre büyüklük küçüklük durumudur. Şart tuttuğunda $obek=s1$, $okek=(x*y)/s1$ olarak ekrana yazılıyor. OKEK ve OBEB matematik açıklamasını da aşağıda veriyoruz.

ORTAK BÖLENLERİN EN BÜYÜĞÜ (OBEB)

En az biri sıfırdan farklı iki ya da daha fazla tam sayının ortak bölenlerinin en büyüğüne bu sayıların ortak bölenlerinin en büyüğü denir ve OBEB biçiminde gösterilir. OBEB bulunurken verilen sayılar asal çarpanlarına ayrılır. Ortak olan asal çarpanlardan büyük

olmayan üslülerin çarpımı bu sayıların OBEB'ini verir.

ORTAK KATLARIN EN KÜÇÜĞÜ (OKEK)

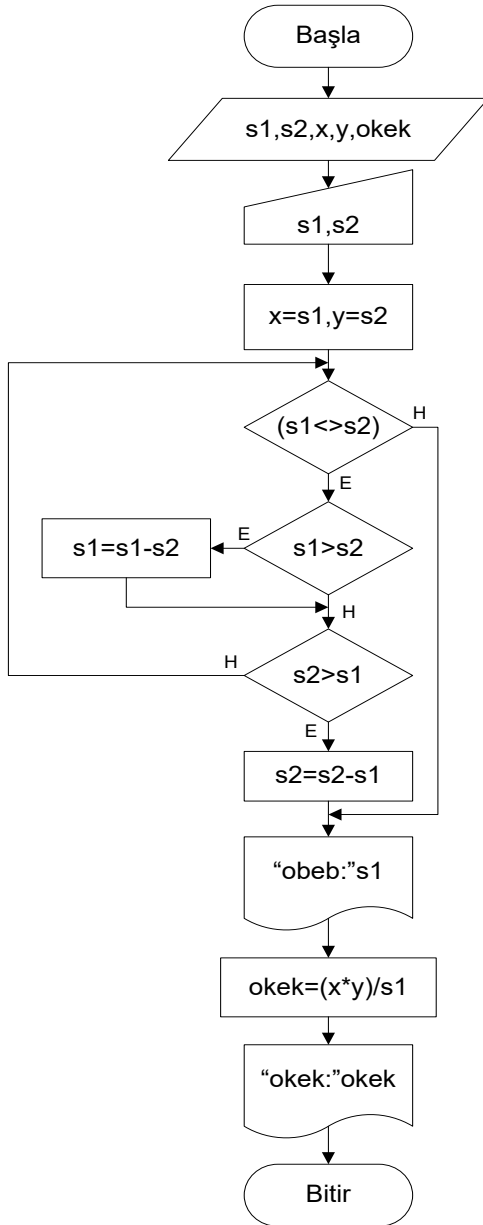
Hepsi sıfırdan farklı iki ya da daha fazla tam sayının pozitif ortak katlarının en küçüğüne bu sayıların ortak katlarının en küçüğü denir ve OKEK biçiminde gösterilir.

OKEK bulunurken verilen sayılar asal çarpanlarına ayrılır. Ortak olan asal çarpanlardan küçük olmayan üslülerin çarpımı bu sayıların OKEK ini verir.

Algoritma Testi :

S1	S2	obek	okek	İşlem
42	28	0	0	42-28
14	28	0	0	28-14
14	12	14	84	$(42*28)/14$

Akış Diyagramı:



C Kod:

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int S1,S2,x,y,okek;
    printf("iki Sayı Giriniz:");
    scanf("%d%d",&S1, &S2);
    x=S1;
    y=S2;
    while(S1!=S2)
    {
        if (S1>S2)
        {
            S1=S1-S2;
        }
        else if (S2>S1)
        {
            S2=S2-S1;
        }
    }
    printf("OBEB %d\n", S1);
    okek=(x*y)/S1;
    printf("OKEK %d\n", okek);
    getch();
}
```

C#Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi1, sayi2, obeb, okek, temp;
            Console.Write("1.Sayıyı giriniz = ");
            sayi1 = Convert.ToInt32(Console.ReadLine());
            Console.Write("2.Sayıyı giriniz = ");
            sayi2 = Convert.ToInt32(Console.ReadLine());
            dnz:
            if ((sayi1 / sayi2) == 0)
            {
                obeb = sayi2;
                if (obeb == 1)
                {
                    okek = sayi1 * sayi2;
                }
                else
                {
                    sayi1 = sayi1 * sayi2;
                    okek = sayi1 / obeb;
                    goto cikis;
                }
            }
            else
            {
                temp = sayi2;
                sayi2 = sayi1 / sayi2;
                sayi1 = temp;
                goto dnz;
            }
            cikis:
            Console.ReadLine();
        }
    }
}
```

44. 1 ile 25 arasındaki tam sayıların karelerinin çarpımını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $i=1, \text{çarpım}=1$
- 3- $\text{çarpım}=\text{çarpım}*(i^2)$
- 4- Eğer $i=25$ ise 7'ye git
- 5- $i=i+1$, 3'e git
- 6- Yazdır çarpım
- 7- Bitir

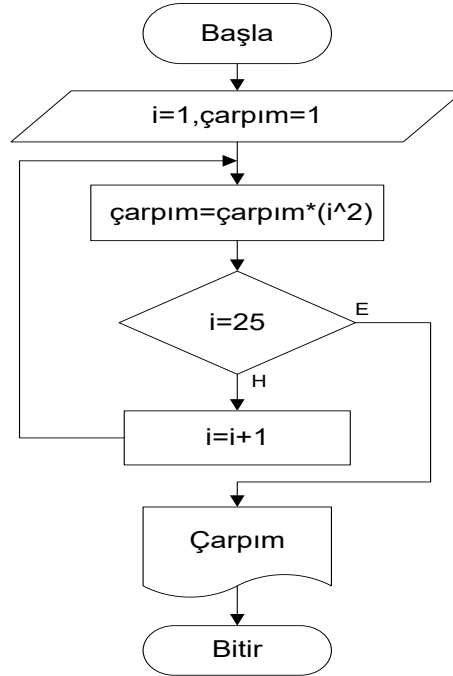
Algoritma Testi:

i	Çarpım
1	1
2	4
3	36
4	576
5	14400

Açıklama:

Bu soruda döngü olduğunu artık herkes anlamış olmalı çünkü 1 ile 25 arasında bir seri söz konusu. $\text{Çarpım}=1$ almak zorundayız. Çünkü toplama işleminde 0, çarpım işlemlerinde sayıları biriktireceğimiz değişkeni 1 almalıyız. Döngü kullanıyorsak %80 ihtimalle i diye bir değişkeni sayaç olarak kullanırız. Bu soruda öyle $i=1$ den başlayarak 25 olana kadar sayıların kareleri alınıp birbiriyle çarpılarak çarpım değişkeninin içine atılır ve sonunda çarpım ekrana basılır.

Akış Diyagramı :



C Kod:

```
#include <stdio.h>

#include <conio.h>

#include <math.h>

main()
{
    int i;

    long carpim=1;

    for(i=1;i<=25;i++)
    {
        carpim=carpim*pow(i,2);
    }

    printf("carpim = %ld",carpim);

    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i;
            long carpim = 1;
            for (i = 1; i <= 25; i++)
            {
                carpim = carpim *
                Convert.ToInt64(Math.Pow(i, 2));
            }
            Console.Write("Çarpım = " + carpim);
            Console.ReadLine();
        }
    }
}
```

45. $ax^2 + bx + c = 0$ tipindeki bir denklemin köklerini bulan programın algoritma ve akış diyagramı çizini

Algoritma:

- 1- Başla
- 2- a,b,c,x,y,delta=0
- 3- a,b,c gir
- 4- $\text{delta} = b^2 - (4 * a * c)$
- 5- Eğer $\text{delta} = 0$ ise devam et, değilse 7'ye git
- 6- $x = -b / 2 * a$ 10'a git
- 7- Eğer $\text{delta} < 0$ ise yazdır "reel kök yok" 11'e git, değilse devam et
- 8- $x = (-b - (\text{delta}^{1/2})) / (2 * a)$, $y = (-b + (\text{delta}^{1/2})) / (2 * a)$
- 9- Yazdır "iki kök var", x,y 12'ye git
- 10- Yazdır "tek kök var", x
- 11- Bitir

Bu işlemi sadece Delta'yı karşılaştırmak için kullanınız.

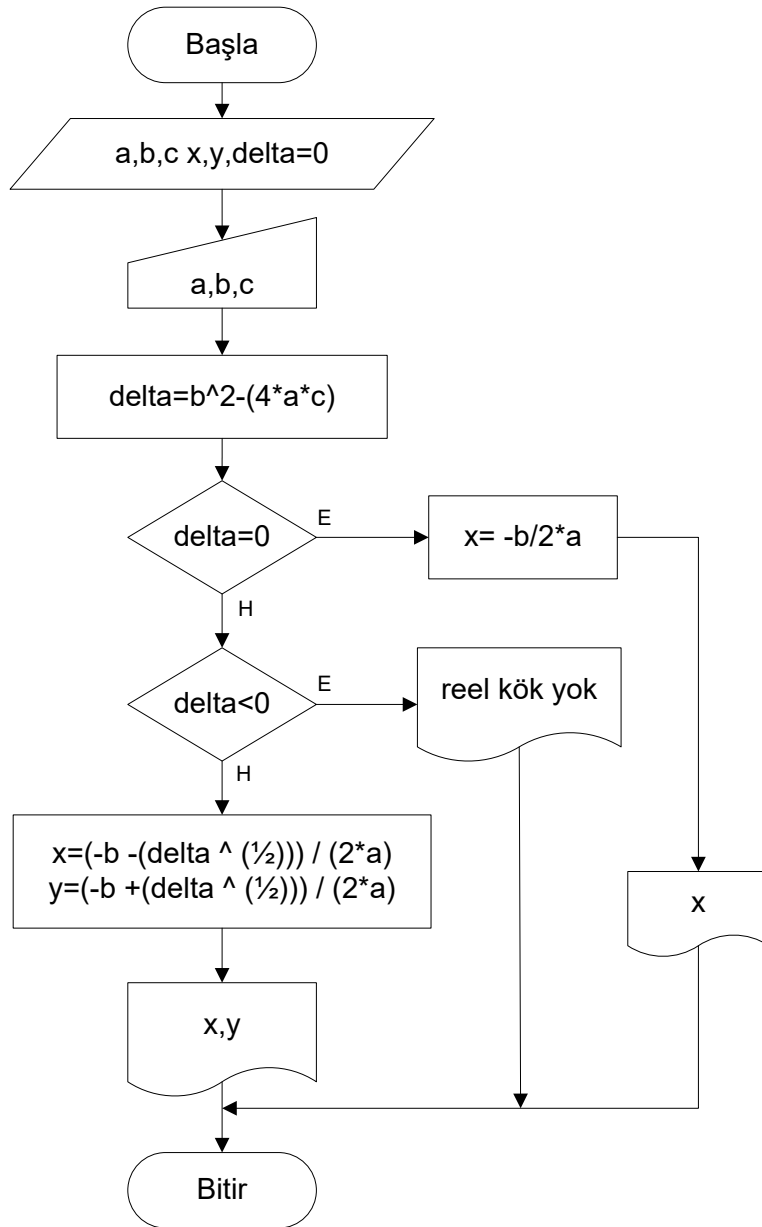
Algoritma Testi:

a	b	c	Delta	X1	X2
2	3	1	$1 > 0$	$(-3 - 1) / 2 * 2$	$(-3 + 1) / 2 * 2$
2	3	1	1	-1	-1/2

Açıklama:

$ax^2 + bx + c$ tipindeki bir denklem için a,b,c değerlerini dışardan girerek Delta değerimizi hesaplarız. Delta değerimizin 0 veya 0'dan küçük ya da büyük olma durumuna göre de denklemin köklerini buluruz. Değerleri alıp formüle uygularız. Direkt matematik ve iki bilinmeyenli bir denklem formülüne değerlerimizi uygularız.

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a,b,c,delta=0;
    float x,y;
    delta=(b*b)-(4*a*c)
    if(delta==0)
    {
        x=(-b)/(2*a)
        printf("Tek kök var = %f ",x);
    }
    else
    {
        if(delta<0)
            printf("reel kok yok");
        else
        {
            x=(-b-(sqrt(delta))/(2*a);
            y=(-b+(sqrt(delta))/(2*a);
            printf("iki kok var  x1=%f x2=%f",x1,x2);
        }
    }
    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int a=0;
            int b=0;
            int c=0;
            int delta = 0;
            double x, y;
            a = Convert.ToInt32(Console.ReadLine());
            b = Convert.ToInt32(Console.ReadLine());
            c = Convert.ToInt32(Console.ReadLine());
            delta = (b * b) - (4 * a * c);
            if (delta == 0)
            {
                x = (-b) / (2 * a);
                Console.Write("Tek kök var = " + x);
            }
            else
            {
                if (delta < 0)
                {
                    Console.Write("reel kok yok");
                }
                else
                {
                    x = (-b - (Math.Sqrt(delta)) / (2 * a));
                    y = (-b + (Math.Sqrt(delta)) / (2 * a));
                    Console.Write("iki kok var " + x + " , " + y);
                }
            }
            Console.ReadLine();
        }
    }
}
```

46. 1-100 arasında kaç asal sayı vardır gösteren programın algoritma akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $sayi=8, adet=4$
- 3- $sayi++$
- 4- Eğer $sayi < 100$ ise devam et, değilse 10'a git
- 5- Eğer $sayi \% 2 = 0$ 3. adıma git, değilse devam et
- 6- Eğer $sayi \% 3 = 0$ 3. adıma git, değilse devam et
- 7- Eğer $sayi \% 5 = 0$ 3. adıma git, değilse devam et
- 8- Eğer $sayi \% 7 = 0$ 3. adıma git, değilse devam et
- 9- $adet++$ 3'e git
- 10- Yazdır adet
- 11- Bitir

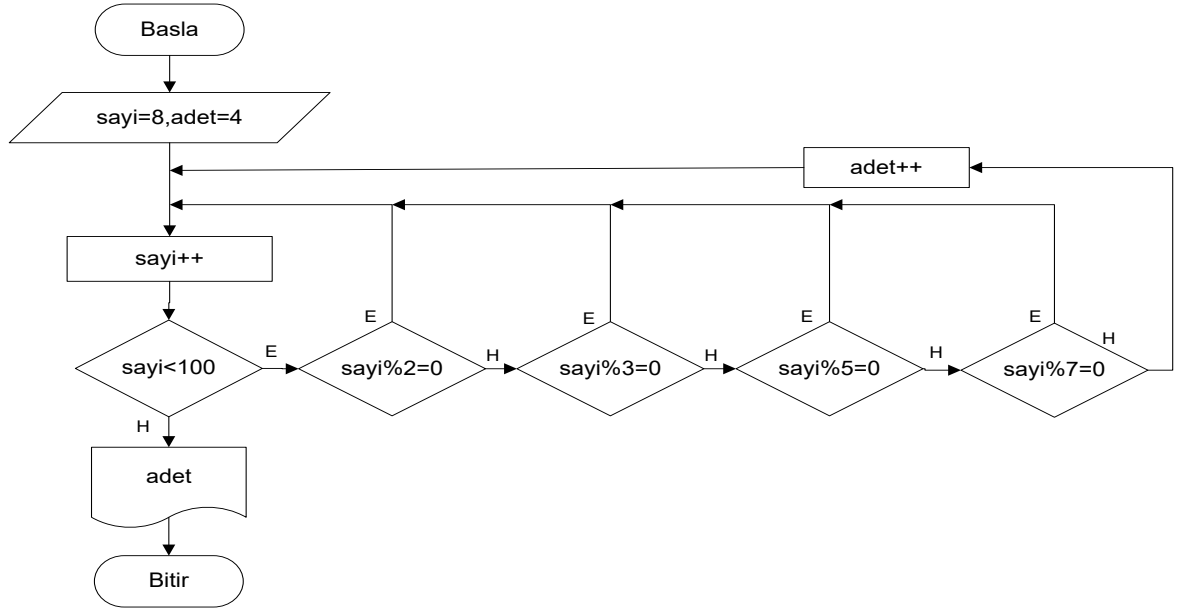
Açıklama:

Asal sayı soruları aslında algoritmada olmazsa olmazdır. Sınavda öğrencilerin if (eğer) mekanizmasını anladığının bir göstergesidir. 1 ile 100 arasındaki sayılar dediği için hemen aklımıza döngü gelecektir. Ancak burada *i* değişkeni 8 den başlamalıdır. Çünkü 2,3,5,7 değerleri asaldır. Buna göre program *i*=8 den başlamalıdır. Bundan sonra *i*=8 den başlayarak 100 e kadar devamlı 2,3,5,7 sayılarına böldürüp kalana bakarız. Kalan 4, değere bölünce de 0 çıkıyorsa adet diye kullandığımız başta 4 olan sayacımızı bir arttırmalıyız. *i*=100 değerini alınca 100’de baktırıp adet değişkenini ekrana bastırarak programı sonlandırmalıyız.

Algoritma Testi:

Adet	<i>i</i>	(<i>i</i> %2=0 && <i>i</i> %3=0&& <i>i</i> %5=0& <i>i</i> %7=0
4	8	False
4	9	False
4	10	False
5 (4+1)	11	True
...	...	
25	97	True
25	98	false
25	99	false
25	100	False(Program Bitti)

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    int sayi,adet=4;
    for(sayi=8;sayi<=100;sayi++)
    {
        if((sayi%2)!=0)
        {
            if((sayi%3)!=0)
            {
                if((sayi%5)!=0)
                {
                    if((sayi%7)!=0)
                    {
                        adet++;
                    }
                }
            }
        }
    }

    printf("adet= %d",adet);

    getch();
}
```

C #Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi, adet = 4;
            for (sayi = 8; sayi <= 100; sayi++)
            {
                if ((sayi % 2) != 0)
                {
                    if ((sayi % 3) != 0)
                    {
                        if ((sayi % 5) != 0)
                        {
                            if ((sayi % 7) != 0)
                            {
                                adet++;
                            }
                        }
                    }
                }
            }
            Console.WriteLine("Adet = " + adet);
            Console.ReadLine();
        }
    }
}
```

47. 10-100 arasındaki asal sayıları gösteren programın algoritma akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $sayi=10$
- 3- $sayi++$
- 4- Eğer $sayi<100$ ise devam et,değilse 10'a git
- 5- Eğer $sayi\%2=0$ 3. adıma git,değilse devam et
- 6- Eğer $sayi\%3=0$ 3. adıma git,değilse devam et
- 7- Eğer $sayi\%5=0$ 3. adıma git,değilse devam et
- 8- Eğer $sayi\%7=0$ 3. adıma git,değilse devam et
- 9- Yazdır $sayi$, 3'e git
- 10- Bitir.

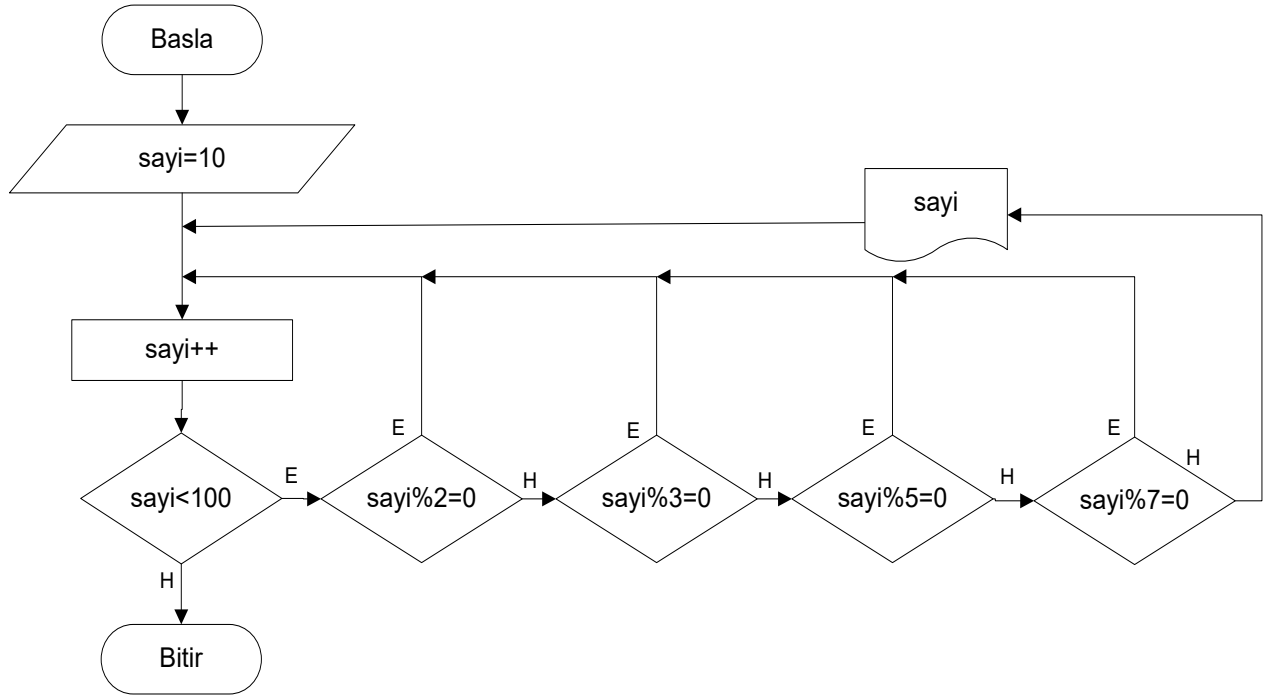
Açıklama:

10 ile 100 arasındaki sayılar dediği için hemen aklımıza döngü gelmelidir. Program $i=10$ dan başlayacak. Bundan sonra $i=10$ den başlayarak 100 e kadar devamlı 2,3,5,7 sayılarına böldürüp kalana bakarız. Kalan 4, değere bölünce de 0 çıkıyorsa sayı asaldır. Bu sayıları $i=100$ değerini alıncaya kadar şartımıza uyan sayıları ekrana basıp programı sonlandırırız.

Algoritma Testi:

Sayı	$(i\%2=0 \&\&i\%3=0\&\&i\%5=0\&\&i\%7=0)$
10	False
11(ilk)	True
12	False
13	True
...	...
97(son)	True
...	...
100	False(program Bitti)

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int sayi;
    for(sayi=10;sayi<=100;sayi++)
    {
        if((sayi%2)!=0)
        {
            if((sayi%3)!=0)
            {
                if((sayi%5)!=0)
                {
                    if((sayi%7)!=0)
                    printf("%4d",sayi);
                }
            }
        }
    }
    getch();
}
```

C # Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi;
            for (sayi = 10; sayi <= 100; sayi++)
            {
                if ((sayi % 2) != 0)
                {
                    if ((sayi % 3) != 0)
                    {
                        if ((sayi % 5) != 0)
                        {
                            if ((sayi % 7) != 0)
                                Console.WriteLine(sayi);
                        }
                    }
                }
            }
            Console.ReadLine();
        }
    }
}
```

48. Girilen sayının smith sayısı olup olmadığını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- sayı,x,top=0,top1=0,i=2
- 3- sayı gir
- 4- x=sayı
- 5- top=top+(sayı%10),sayı=sayı/10
- 6- Eğer sayı<10 ise top=top+sayı, değilse 5'e git
- 7- Eğer x%i=0 ise devam et, değilse i++ 7'ye git
- 8- Eğer (i=2 || i=3 || i=5 || i=7) ise top1=top1+i, x=x/i, değilse devam et
- 9- top1=top1+(i%10),i=i/10
- 10- Eğer i<10 ise devam et, değilse 9'a git
- 11- x=x/i
- 12- Eğer x=1 ise devam et, değilse 7'ye git
- 13- Eğer top1=top ise yazdır "smith", değilse yazdır "smith değil"
- 14- Bitir

Açıklama:

1 den büyük asal olmayan bir tam sayının rakamlarının toplamı, sayı, asal çarpanlarına ayrılarak yazıldığında bu yazılışta bulunan tüm asal çarpanların rakamlarının toplamına eşit oluyorsa bu tür

sayılara **Smith sayısı** denir.

$$\begin{aligned} 121 &= 11 * 11 \\ 1 + 2 + 1 &= 1 + 1 + 1 + 1 \\ 4 &= 4 \quad (121 \text{ bir Smith sayısıdır.}) \end{aligned}$$

$$\begin{aligned} 166 &= 2 * 83 \\ 1 + 6 + 6 &= 2 + 8 + 3 \\ 13 &= 13 \quad (166 \text{ bir Smith sayısıdır.}) \end{aligned}$$

Bu sayının nasıl ortaya çıktığını merak ediyor musunuz?

1982 yılında matematikçi Albert Wilansky, kardeşi Smith'i ararken onun telefon numarasının (4937775) bu ilginç özelliğini fark etmiş. Bundan dolayı da bu sayılara Smith sayıları adını vermiş.

Bu sayıyı da inceleyelim;

$$4937775 = 3 * 5 * 5 * 65837$$

$$\begin{aligned} 4 + 9 + 3 + 7 + 7 + 7 + 5 \\ = \\ 3 + 5 + 5 + 6 + 5 + 8 + 3 + 7 \end{aligned}$$

$$42 = 42 \quad (4937775 \text{ bir Smith sayısıdır.})$$

Algoritma Testi:

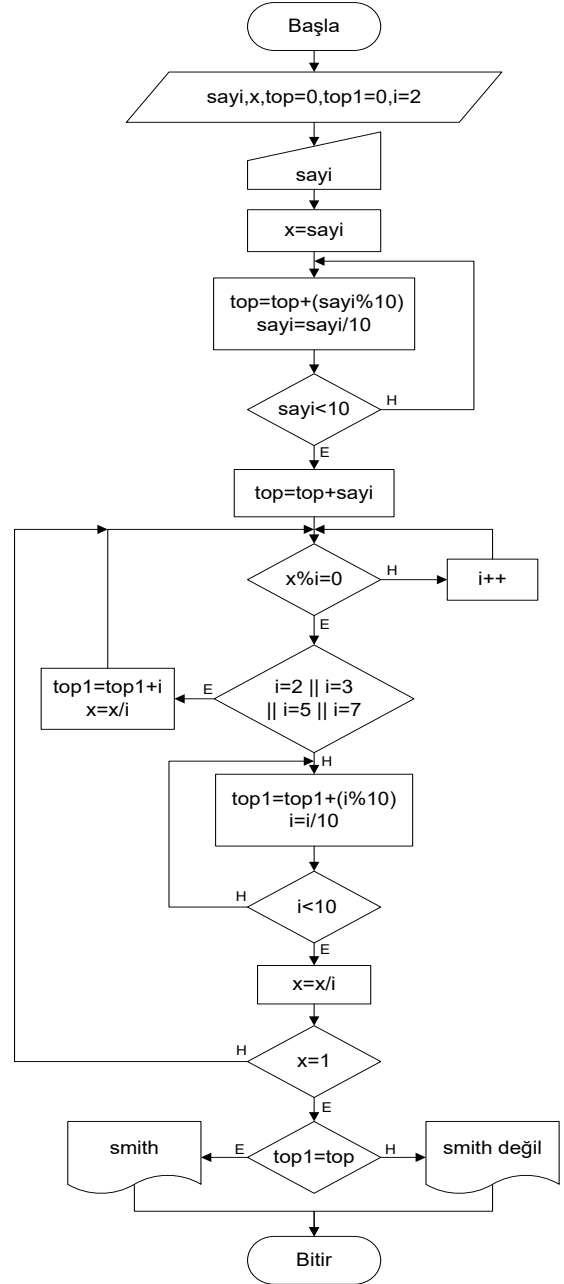
sayi	Top	İşlem
121	0	---
12	1	Top=top+1
1	3	Top=top+2
--	4	Top=top+1

X	i	Top1
121	2	0
121	3	0
121	...	0
11	11	Top1=top1+1 Top1=top1+1
11	11	Top1=top1+1 Top1=top1+1
1	11	4

Asal bölenler $11 * 11 = 1+1+1+1$

Top=top1 ise Sayımız Smith.

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
int sayi,x,i=2,top=0,top1=0,y;
main()
{
clrscr();
scanf("%d",&sayi);
x=sayi;
while(sayi>10)
{
top=top+sayi%10;
sayi=sayi/10;
}
top=top+sayi;
while(x>1)
{
if((x%i)==0)
{
if(i==2 || i==3 || i==5 || i==7)
{
top1=top1+i;
x=x/i;
}
}
else
{

```

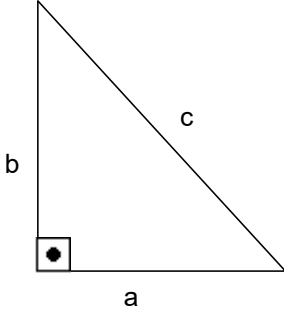
```
y=i;
while(y>10)
{
top1=top1+y%10;
y=y/10;
}
top1=top1+y;
x=x/i;
}
}
else
i++;
}
if(top==top1)
printf("Smith");
else
printf("smith değil");
getch();
}
```

C #Kod :

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayı, x, i = 2, top = 0, top1 = 0, y;
            Console.Write("Sayıyı Giriniz = ");
            sayı = Convert.ToInt32(Console.ReadLine());
            x = sayı;
            while (sayı > 10)
            {
                top = top + sayı % 10;
                sayı = sayı / 10;
            }
            top = top + sayı;
            while (x > 1)
            {
                if ((x % i) == 0)
                {
                    if (i == 2 || i == 3 || i == 5 || i == 7)
                    {
                        top1 = top1 + i;
                        x = x / i;
                    }
                    else
                    {
                        y = i;
                        while (y > 10)
                        {
                            top1 = top1 + y % 10;
                            y = y / 10;
                        }
                        top1 = top1 + y;
                        x = x / i;
                    }
                }
                i++;
            }
        }
    }
}
```

```
if (top == top1)
{
    Console.Write("Smith");
}
else
{
    Console.Write("smith değil");
}
Console.ReadLine();
}
```

49. Dışarıdan iki dik kenarı girilen üçgenin hipotenüsünü hesaplayan programın algoritma ve akış diyagramını çiziniz.



$$c = \sqrt{a^2 + b^2}$$

Algoritma Testi :

a	b	C
2	3	$\sqrt{13}$
1	1	$\sqrt{2}$

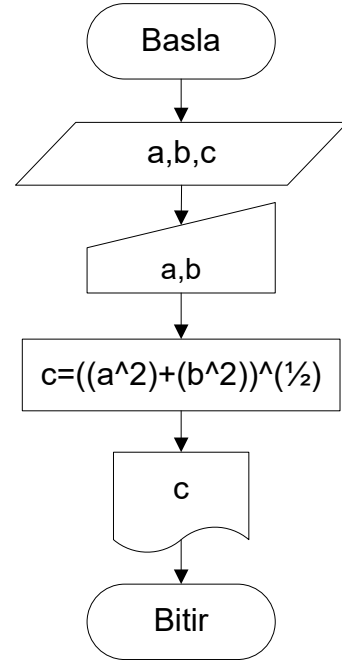
Akış Diyagramı:

Algoritma:

- 1- Başla
- 2- a,b,c
- 3- a,b gir
- 4- $c = ((a^2) + (b^2))^{1/2}$
- 5- Yazdır c
- 6- Bitir

Açıklama:

Bu soruda dairenin alanı gibi bir dik üçgen formülümüz bulunmaktadır. Buna göre kullanıcıdan bu değerleri alıp, formülde yerine koyduktan sonra sonucu ekrana basarız. Bu soru, konuyla ilgili temel aksettiren sorulardandır. Şimdi algoritma testine bakalım.



C Kod:

```
#include <stdio.h>

#include <conio.h>

#include <math.h>

main()

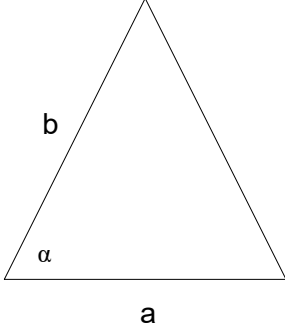
{
    int a,b;
    float c;
    scanf("%d",&a);
    scanf("%d",&b);
    c=sqrt(pow(a,2)+ pow(b,2));
    printf("%f",c);
    getch();
}
```

C#Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int a, b;
            double c;
            Console.Write("A Gir = ");
            a = Convert.ToInt32(Console.ReadLine());
            Console.Write("B Gir = ");
            b = Convert.ToInt32(Console.ReadLine());
            c = Convert.ToDouble(Math.Sqrt((Math.Pow(a, 2) +
            Math.Pow(b, 2))));
            Console.Write(c);
            Console.ReadLine();
        }
    }
}
```

50. Dışarıdan iki kenarı ve aradaki açısı girilen üçgenin alanını hesaplayan programın algoritma ve akış diyagramını çiziniz.



$$Alan = \frac{1}{2} \cdot a \cdot b \cdot \sin(\alpha)$$

Algoritma Testi:

a	b	aci	Alan
3	5	Sin(30)=0.5	7,5
2	4	Sin(60)=0.88	7,04

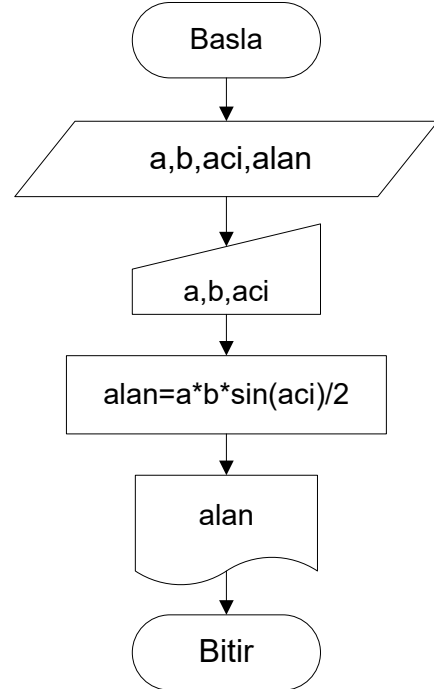
Akış Diyagramı :

Algoritma:

- 1- Başla
- 2- a,b,aci,alan
- 3- a,b,aci gir
- 4- alan=a*b*sin(aci)/2
- 5- yazdır alan
- 6- Bitir

Açıklama:

Geometri problemi gibi görülsede başlangıç için tekrarlanan sorulardan biri de bu sorudur. Burada da girilen değerleri formülde kullanıp sonucu ekrana yazdırıyoruz. Ancak bu örnekteki en önemli unsur sadece C dilinde bunu kodlarken sin fonksiyonu kullanmamız gerektiğidir. Bunun için de math.h kütüphanesini programa include ile dahil etmeliyiz.



C Kod:

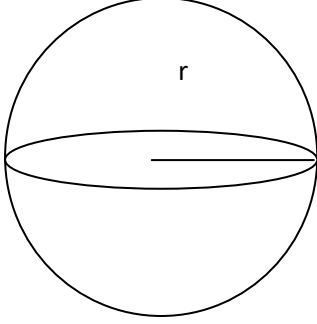
```
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{
    int a,b,aci;
    float alan;
    scanf("%d%d",&a,&b);
    scanf("%d",&aci);
    alan=a*b*sin(aci)/2;
    printf("%f",alan);
    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int a, b, alan,aci;
            Console.Write("A Gir = ");
            a = Convert.ToInt32(Console.ReadLine());
            Console.Write("B Gir = ");
            b = Convert.ToInt32(Console.ReadLine());
            Console.Write("Açı Gir = ");
            aci = Convert.ToInt32(Console.ReadLine());
            alan = Convert.ToInt32(a * b * Math.Sin(aci) / 2);
            Console.Write(alan);
            Console.ReadLine();
        }
    }
}
```

51. Dışarıdan yarıçapı girilen kürenin alanını ve hacmini hesaplayan programın algoritma akış diyagramını çiziniz.



$$Alan = 4 \cdot \pi \cdot r^2$$

$$Hacim = \frac{4}{3} \pi \cdot r^3$$

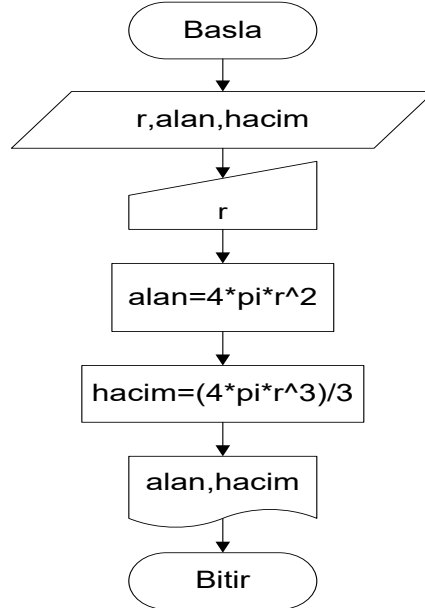
Algoritma Testi:

Yarıcap(r)	Alan	Hacim
3	$(4) \cdot (3,14) \cdot (3^2)$ =113,04	$(4/3) \cdot (3,14) \cdot (3^3)$ =113,04
2	$(4) \cdot (3,14) \cdot (2^2)$ =50,24	$(4/3) \cdot (3,14) \cdot (2^3)$ =33,49

Algoritma:

- 1- Başla
- 2- r,alan,hacim
- 3- r gir
- 4- $alan=4 \cdot \pi \cdot r^2$
- 5- $hacim=(4 \cdot \pi \cdot r^3)/3$
- 6- Yazdır alan,hacim
- 7- Bitir

Akış Diyagramı:



Açıklama:

Bir önceki soruda olduğu gibi girilen değerleri formülde kullanıp sonucu ekrana yazdırıyoruz. Yarıçap değeri dışarıdan girilecektir, zaten pi sayının değeri sabit (const) 3,14 olduğundan bu değerler ile kürenin alan ve hacmini buluyoruz.

C Kod:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{
    int r ;

    float alan,hacim;

    float pi=3.14;

    scanf("%d",&r);

    alan=4*pi*pow(r,2);

    hacim=(4*pi*pow(r,3))/3;

    printf("%f",alan,hacim);

    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int r, alan, hacim;

            Console.Write("R Gir = ");

            r = Convert.ToInt32(Console.ReadLine());

            alan = Convert.ToInt32(4 * Math.PI *
Math.Pow(r,2));

            hacim = Convert.ToInt32((4 * Math.PI *
Math.Pow(r,3))/3);

            Console.WriteLine("Alan = " + alan);

            Console.Write("Hacim = " + hacim);

            Console.ReadLine();

        }
    }
}
```

52. Girilen bir tam sayının hanelerindeki en büyük sayıyı bulan algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- a ,enb,b
- 3- a gir
- 4- enb=0
- 5- $b=a-a/10*10$
- 6- Eğer $enb < b$ ise $enb=b$
- 7- Eğer $a=0$ ise 9 adıma git
- 8- $a=(a/10)$,4.adıma geri dön
- 9- Yaz enb
- 10- Bitir

Açıklama:

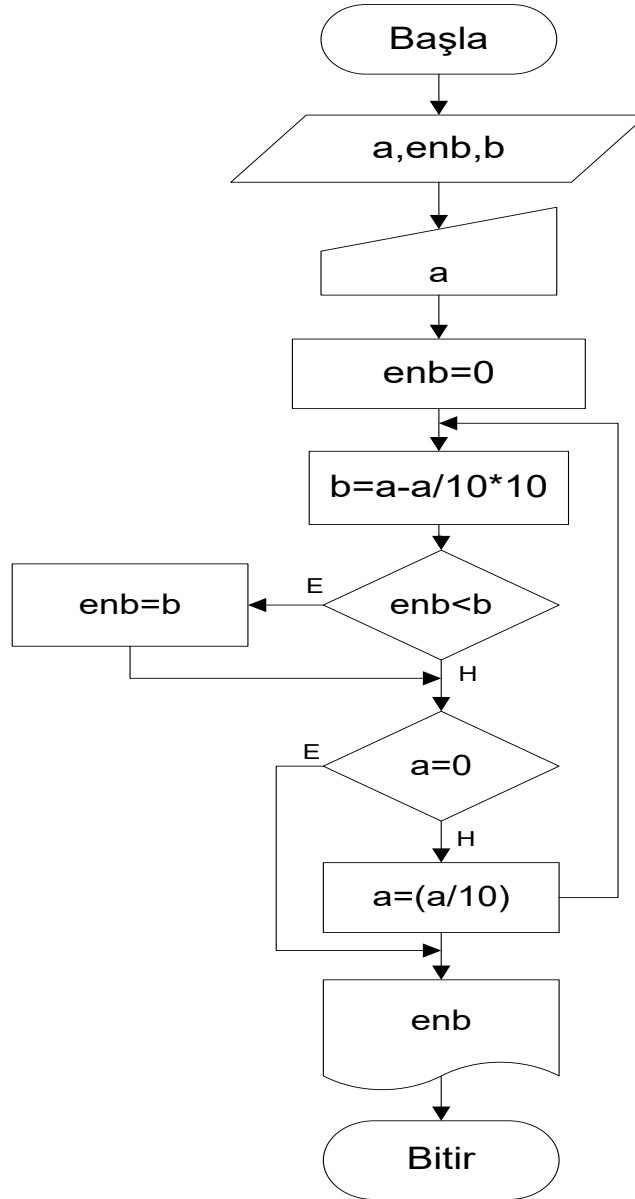
Bu soruda bir tam sayı giriyor, bunun basamaklarını bulup birbiriyle karşılaştırıyoruz. Dinamik bir soru kaç basamaklı bir sayı olduğu bilinmiyor ve kullanıcıdan isteniliyor. Bunun için biz olsak sayıyı devamlı 10'a bölerek bunu tanımladığımız enb değişkene atarak karşılaştırma yaparız. Fakat bu soruda farklı bir çözüm örneği görüyoruz. Bu soruyu birden çok çözüm yöntemi olabileceğini de göz önüne alarak özellikle kitabımıza aldık. Fakat burada önemli olanın en kısa kodda, ya da en hızlı çalışan, az değişkenli bir algoritma diğerine göre daha güçlü olacağını unutmamaktır. A, enb ve b 3 adet değişken

tanımladık. A burda dışarıdan girilen sayımızdır. Enb değişkeni ise takas alanıdır. En büyük basamak değeri her adımda şart uyarınca enb içine atılır. Yani her defasında enb içeriği değişebilir. Enb=0 başlangıçta, sonra b değişkenine birler basamağını atıyoruz. Daha sonra bu b değerini enb ile karşılatırıyoruz. Küçükse $enb=b$ 'yi atıyoruz. Bu işlemi $a=0$ olana kadar devam ettiriyoruz. Son olarak enb değerini ekrana basıyoruz.

Algoritma Testi:

a	b	enb
334	0	0
334	4	4
33	3	4
3	3	4
0	Program Bitti	

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a,enb,b;
    scanf("%d",&a);
    enb=0;
    while(a!=0)
    {
        b=a-a/10*10;
        if(enb<b)
            enb=b;
        a =(a/10);
    }
    printf("enbuyuk %d",enb);
    getch();
}
```

C#Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int a, enb, b;
            Console.Write("A = ");
            a = Convert.ToInt32(Console.ReadLine());
            enb = 0;
            while (a != 0)
            {
                b = a - a / 10 * 10;
                if (enb < b)
                    enb = b;
                a = (a / 10);
            }
            Console.Write("En büyük = "+ enb);
            Console.ReadLine();
        }
    }
}
```

53. Ekrandan girilen bir sayı eğer 5-10 arasında ise girilen sayının karesini alıp gösteren, eğer 5'ten küçük ise faktöriyelini alan, 10'dan büyük ise sayıyı ikiye bölüp bir eksiğini yazan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- sayi,fak=1,i
- 3- sayi gir
- 4- Eğer sayi $\geq$ 5 && sayi $\leq$ 10 ise
yazdır sayi²,9'e git,
değilse devam et
- 5- Eğer sayi $\leq$ 5 ise devam et,
değilse 8'e git
- 6- fak=fak*i
- 7- Eğer i=sayi ise yazdır fak,
değilse i++ 6'e git
- 8- Eğer sayi $\geq$ 10 ise yazdır ((sayi/2)-1),
değilse devam et
- 9- Bitir

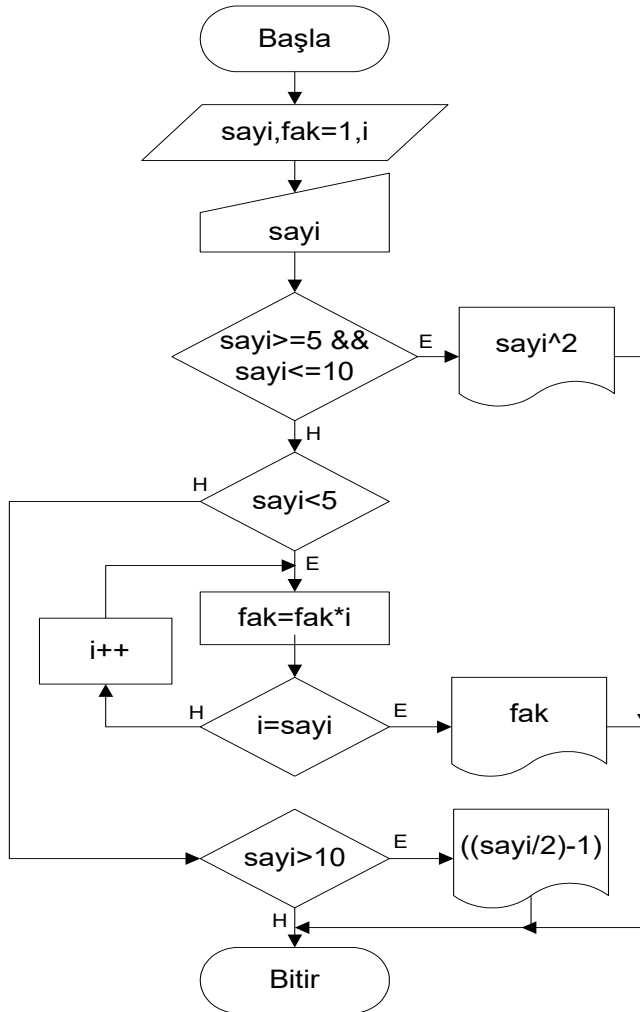
Açıklama:

Bu soruda if (eğer) önemli rol taşımaktadır. Çünkü 3 adet seçeneğimiz bulunmaktadır. Bunlardan birini seçmek için eğer kullanmalıyız. Faktöriyel, döngü ister ama diğer kare alma (^) ve sayıyı ikiye bölüp bir eksiğini alma işlemi, direkt işlemi yapıp ekrana basmakla sona erer. Seçimlik sorular ileride işimize yarayabilir. Mesela şu işlem için 1'e bas , şu işlem için 2'ye bas gibi program menu girişlerinde if yada case mekanizmalarını kullanırız. Bunun için önemlidir.

Algoritma Testi:

sayi	İşlem1 (sayi ²)	İşlem2 (sayi!)	İşlem3 ((sayi/2)-1)
100	False	False	49
6	36	False	False
3	False	1*2*3=6	False

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{
    int sayi,fak=1,i;
    scanf("%d",&sayi);
    if(sayi>=5 && sayi<=10)
        printf("karesi : %f",pow(sayi,2));
    if(sayi<5)
    {
        for(i=1;i<=sayi;i++)
            fak=fak*i;
        printf("faktoriyel :%d",fak);
    }
    if(sayi>10)
        printf("sonuc :%d",((sayi/2)-1));
    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int sayi;
            int fak = 1;
            int i = 1;
            Console.Write("Sayıyı Giriniz = ");
            sayi = Convert.ToInt32(Console.ReadLine());
            if (sayi >= 5 && sayi <= 10)
            {
                Console.Write(Math.Sqrt(Convert.ToDouble(sayi)));
            }
            else if (sayi < 5)
            {
                while (i <= sayi)
                {
                    fak = fak * i;
                    i++;
                }
                Console.Write(fak);
            }
            else if (sayi > 10)
            {
                Console.Write(((sayi / 2) - 1));
            }
            Console.ReadLine();
        }
    }
}
```

54. Dışarıdan 'Derece' cinsinden girilen açıyı ; 'Radyan' ve 'Grad' cinsine çeviren programın algoritma ve akış diyagramını çiziniz.

$$\frac{D}{180} \equiv \frac{R}{\pi} \equiv \frac{G}{200}$$

Algoritma:

- 1- Başla
- 2- d,r,g,pi=3.14
- 3- d (açı) gir
- 4- $r=d*\pi/180$
- 5- $g=d*200/180$
- 6- yaz r,g
- 7- Bitir

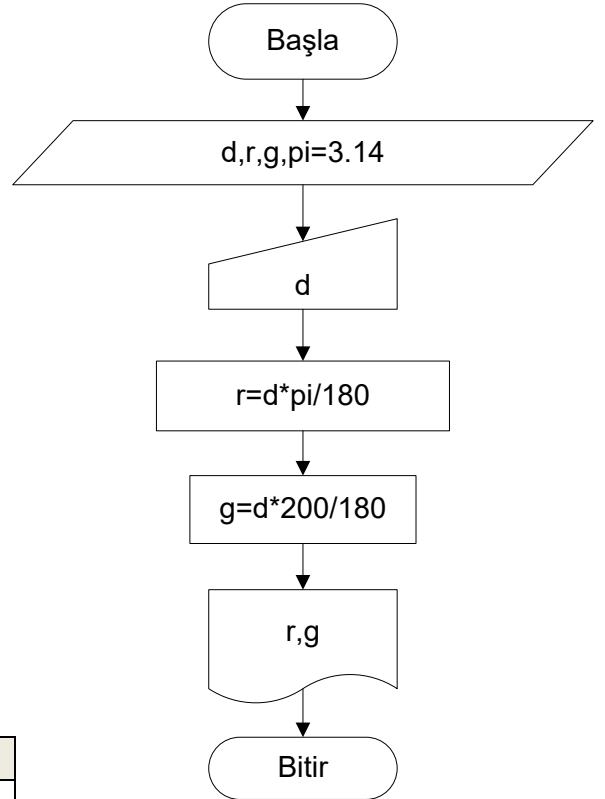
Açıklama:

Klasik işlem sorusu dışarıdan alınan değerleri , dönüşüm formülüne uygulayınca sonucu ekrana basan programın algoritmasıdır.

Algoritma Testi:

d(açı)	R(radyan)	Grad(g)
30	$30*3,14/180=0,52$	$30*200/180=33,3$
45	$45*3,14/180=0,785$	$45*200/180= 50$

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    float aci,r,g,pi=3.14;
    printf("Derece cinsinden aci giriniz");
    scanf("%f",&aci);
    r=(aci*pi)/180;
    g=(aci*200)/180;
    printf("radyan= %f , grad = %f",r,g);
    getch();
}
```

C# Kod:

```
using System;
namespace dmj
{
    class Program
    {
        static void Main(string[] args)
        {
            int d, r, g;
            double pi = 3.14;
            Console.Write("Açıyı Giriniz = ");
            d = Convert.ToInt32(Console.ReadLine());
            r = Convert.ToInt32(d * pi / 180);
            g = d * 200 / 180;
            Console.WriteLine("Radyan = " + r);
            Console.Write("Grad = " + g);
            Console.ReadLine();
        }
    }
}
```

55. Arka arkaya girilen rastgele 10 tam sayının ortalaması ile bu sayılardan en büyük ve en küçük olanın ortalamasını bularak elde edilen bu iki ortalamanın farkını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- a, enk, enb, temp, i, ort, fark, b
- 3- a gir
- 4- enk=a, enb=a, temp=a
- 5- i=2
- 6- a'yı gir
- 7- Eğer enk>a ise enk=a
- 8- Eğer enb<a ise enb=a
- 9- temp=temp+a
- 10- Eğer i=10 ise 13. adıma git
- 11- i=i+1 7. adıma git
- 12- $b = (enb + enk) / 2$, ort=temp/10 ,
fark=ort-b
- 13- Yazdır fark
- 14- Bitir

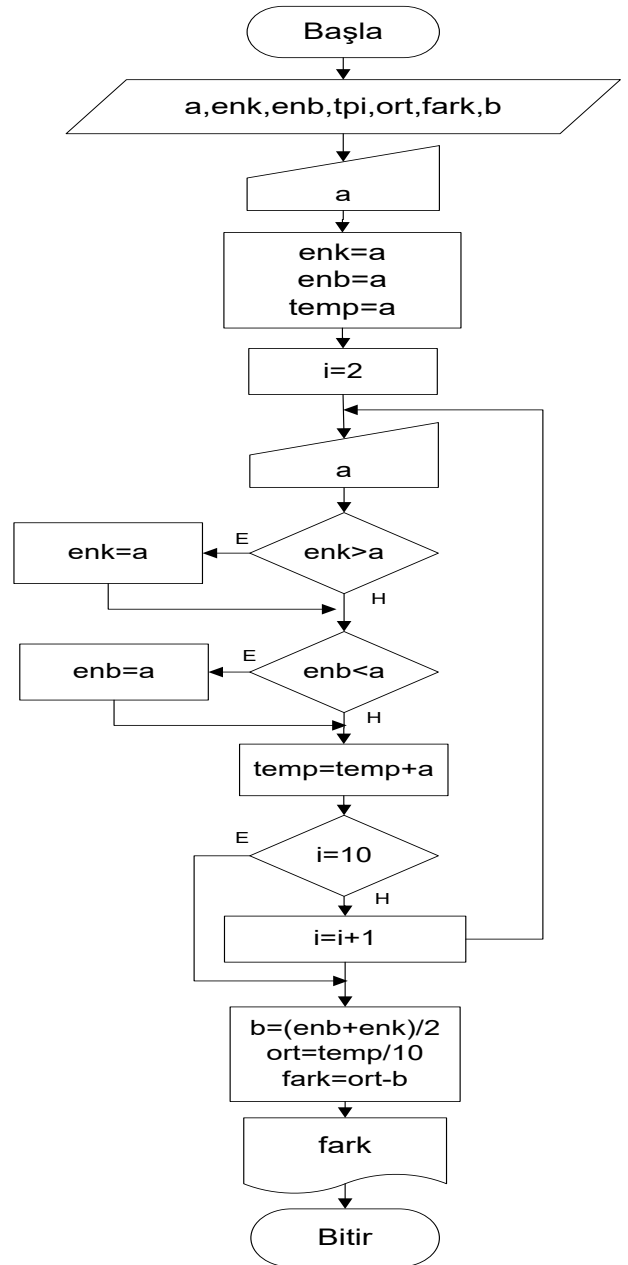
Açıklama:

Soruda arka arkaya girilen 10 sayı dediğine göre bir sayaç değerimiz olmalıdır. Bunun için de her zaman belirttiğimiz gibi l kullanırız . l=1 den 10 oluncaya kadar bir döngü oluşturur. Bunları top denilen bir değişkene her defasında bir önceki ile toplaya atarız ve 10 bölüp ortalamayı buluruz. Döngüyü kullandık . Şimdi bir de takas algoritmamızı kullanarak soruyu çözelim. En büyük ve en küçük gibi sorularda unutulmamalıdır ki enb veya enk diye bir değişken alırız, ilk sayıyı bunun içine atarız ve diğer kaç sayı gelecekse enb ile karşılaştırırız. Bu değerden büyük enb içindeki değer değişir. Enk için de bu geçerlidir. Bunu da yaptıktan sonra enb ve enk toplayıp 2'ye bölüp ortalamayı buluruz. İlk ortalama ve son ortalama değeri çıkarıp sonucu ekrana yazdırırız. Burada fark da eğer 'e sokulsaydı tüm mekanizmaları barından bir soru elde edilebilirdi. Fakat sonuçta eğerin (karar yapılarının) öğrenildiğini ümit ediyoruz.

Algoritma Testi:

Sayı	Enk	Enb	i	Ort	B	Fark
10	10	10	1	6	6	0
5	5	10	2			
3	3	10	3			
3	3	10	4			
6	3	10	5			
8	3	10	6			
9	3	10	7			
11	3	11	8			
1	1	11	9			
4	1	11	10			

Akış Diyagramı:



C Kod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    int a,enk,enb,temp,i;
    float b,ort,fark;
    scanf("%d",&a);
    enk=a;
    enb=a;
    temp=a;
    for(i=2;i<=10;i++)
    {
        scanf("%d",&a);
        if(enk>a)
            enk=a;
        if(enb<a)
            enb=a;
        temp=temp+a;
    }
    b=(enb+enk)/2 ;
    ort=temp/10 ;
    fark=ort-b;
    printf("%f",fark);
    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int enk, enb, ort, fark, b;
            int temp;
            int i;
            int a;
            Console.Write("Sayıyı Giriniz = ");
            a = Convert.ToInt32(Console.ReadLine());
            enk = a; enb = a; temp = a;
            i = 2;
            for (i = 1; i < 10; i++)
            {
                Console.Write("Sayıyı Giriniz = ");
                a = Convert.ToInt32(Console.ReadLine());
                if (enk > a)
                {
                    enk = a;
                }
                if (enb < a)
                {
                    enb = a;
                }
                temp = temp + a;
            }
            b = (enb + enk) / 2;
            ort = temp / 10;
            fark = ort - b;
            Console.Write(fark);
            Console.ReadLine();
        }
    }
}
```

56.1 k sayısı tek ise 3 ile çarpılıp 1 ekleniyor çift ise 2 ile bölünüyor işlem k sayısı 1 olana kadar devam ediyor bu işlemin kaç adım sürdüğünü,işlem sırasında k sayısının aldığı max değeri k sayısının hangi sayıdan sonra hep çift olarak 1'e ulaştığını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başlat
- 2- k,sayac=0,max=0,bolunen=0
- 3- k gir
- 4- Eğer $k \neq 1$ ise sayac++ devam et,
değilse 8'e git
- 5- Eğer $k \% 2 = 1$ ise $k = k(k*3)+1$,sayac++
bolunen=0 , değilse devam et
- 6- Eğer (bolunen<k) ise bolunen=k,
değilse devam et
- 7- $k = k/2$
- 8- Eğer (max<k) ise max=k 4'egit,
değilse devam et
- 9- Yazdır max,sayac,bolunen
- 10- Bitir

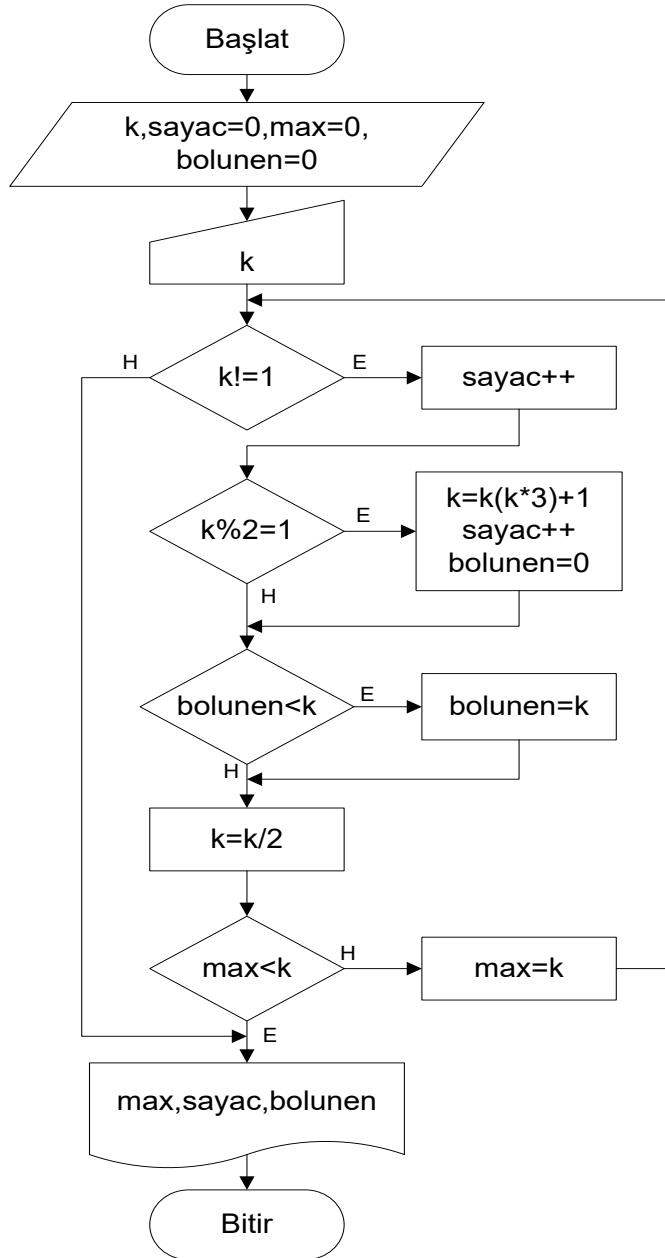
Açıklama:

Bu soru ilk sorduğumda öğrencilerimin kafasını çok karıştırmıştı ve ancak birkaçı cevap verebilmişti. Aslında sorudaki işlemler gayet basittir ama çözüm yeteneği gelişmiş kişilerin yapabileceği türden bir sorudur. Çünkü algoritma ya da problem çözme biraz da çalışma azmi ve yetenek ister kanaatindeyiz. Bu soru sayılarla oyun yapılan ilgi çekici bir sorudur. Bir K sayısı giriliyor ve bu k sayısı 1 olana kadar bazı işlemler uygulanıyor. Bu sorunun kodunu yaparken While kullanmak bizce en uygunu olacaktır. Direkt algoritma testine geçiyoruz. Bu algoritma testleri de sizlere programın nasıl çalıştığını gösterir. Adımlarla karşılaştırarak algoritmayı daha iyi anlayabiliriz.

Algoritma Testi:

K	Tek İşlem	Çift İşlem	Sayac	Max	Bolunen
10	False	$10/2=5$	1	10	10
5	$5*3+1=16$	False	2	16	16
16	False	$16/2=8$	3	16	16
8	False	$8/2=4$	4	16	16
4	False	$4/2=2$	5	16	16
2	False	$2/2=1$	6	16	16
1	False	False	6	16	16

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int k,sayac=0,max=0,bolunen=0;
    scanf("%d",&k);
    while(k!=1)
    {
        sayac++;
        if(k%2==1)
        {
            k=(k*3)+1;
            sayac++;
            bolunen=0;
        }
        else
        {
            if(bolunen<k)
            bolunen=k;
            k=k/2;
        }
        if(max<k)
        {
            max=k;
        }
    }
    printf("maks:%d ,islem :%d ,soncift
:%d",max,sayac,bolunen);
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int k, sayac = 0, max = 0, bolunen = 0;
            Console.Write("K = ");
            k = Convert.ToInt32(Console.ReadLine());
            while (k != 1)
            {
                sayac++;
                if (k % 2 == 1)
                {
                    k = (k * 3) + 1; sayac++;
                    bolunen = 0;
                }
                else
                {
                    if (bolunen < k)
                        bolunen = k;
                    k = k / 2;
                }
                if (max < k)
                {
                    max = k;
                }
            }
            Console.WriteLine("Maks: " + max);
            Console.WriteLine("İslem: " + sayac);
            Console.WriteLine("Soncift: " + bolunen);
            Console.ReadLine();
        }
    }
}
```

Bölüm 5

Seri Algoritma Soru Çözümleri

57. e^x fonksiyonunun seriye açılımı aşağıdadır. Buna göre; dışarıdan girilen x ve N değerine göre; e^x 'i hesaplayan programın algoritma ve akış diyagramını çiziniz.

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots + \frac{x^n}{N!} = \sum_{k=0}^N \frac{x^k}{k!}$$

Algoritma:

- 1- Başla
- 2- x,n ,t=1,f=1,i=1
- 3- x, n gir
- 4- Eğer n+1=i ise devam et,
değilse f=f*i , t=t+(x^i)/f , i++
3'e git
- 5- Yazdır t
- 6- Bitir

Açıklama:

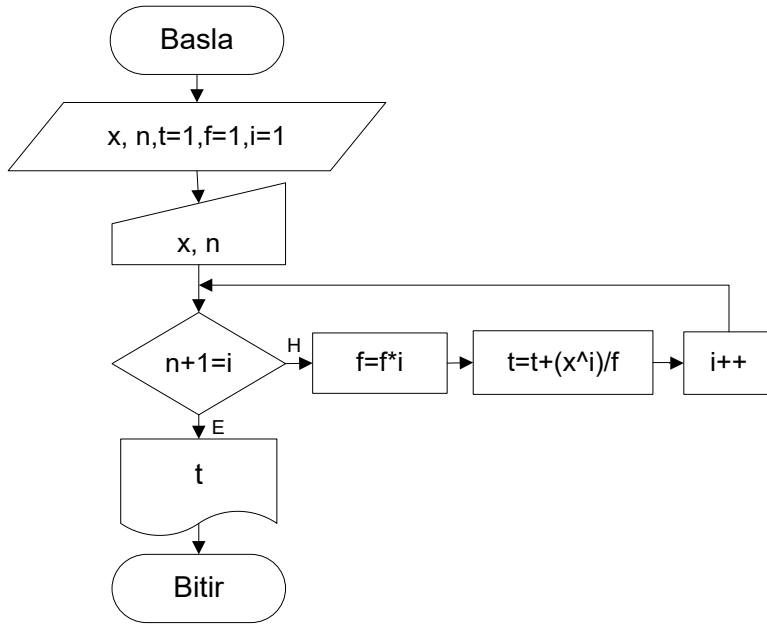
Bu soru seri örneklerimizden, içinde faktoriyel işlemini de barındırmaktadır. Çünkü faktoriyel yine algoritma sorularında döngü mekanizmasını anlamak için önemlidir. Burada fak=fak* gibi kuralları her zaman akılda tutmak gerektiğinin önemini bir kez daha vurguluyoruz. Bu soru da verilen N ve X değerine göre e^x işleminin sonucunu veren bir algoritmadır. Bunu iki şekilde çözdük. Çünkü önceki sorularda algoritma yazarken döngü kullanmamıştık ama konu üzerinde kendimizi geliştirdikçe sorularımızda döngü kullanmamaya çalışmalıyız. Zaten unutulmamalıdırki algoritma çözüme giden adımları oluşturur. Bu soruda iki tane

değişken tanımladık: Bunlardan birini faktoriyel için birisini de işlemin sonucunu tutan T değişkeni olarak ayarladık ve 1 değerinden başlattık. İşlem N kadar dönecek, her N değerindeki sonucu T değişkeninin içine atacak, sonra T yi ekran basacak ve işlem tamamlanacaktır.

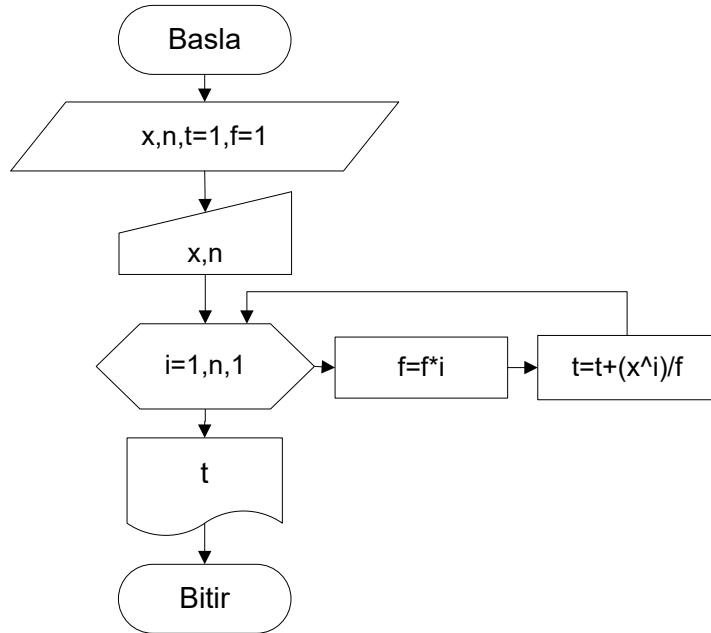
Algoritma Testi:

x	n	t	f	i
3	3	1	1	1
3	3	4	1	1
3	3	8,5	2	2
3	3	13	6	3

Akış Diyagramı 1



Akış Diyagramı 2:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int x,n,t=1,fak=1,i;
    scanf("%d",&x);
    for(i=1;i<=n+1;i++)
    {
        fak=fak*i;
        t=t+(x^i)/f;
    }
    printf("toplama : %d",t);
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int n, x;
            int t = 1;
            int f = 1;
            int i = 1;
            Console.Write("X = ");
            x = Convert.ToInt32(Console.ReadLine());
            Console.Write("N = ");
            n = Convert.ToInt32(Console.ReadLine());
            do
            {
                f = f * i;
                t = ((x^i) / f);
                i++;
            } while (n + 1 == i);
            Console.Write(t);
            Console.ReadLine();
        }
    }
}
```

58. $x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \dots + \frac{x^n}{n!}$ x ve n değişkenine göre yandaki işlemi yapan programın algoritmayı ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $n, x, fak=1, top=0, i=1, j=1$
- 3- n, x gir
- 4- $top=top+(x^i)/fak$
- 5- $i++$
- 6- Eğer $i < j$ ise 9'a,
değilse devam et
- 7- $fak=fak*j$
- 8- $j++, 6'e$ git
- 9- Eğer $i=n$ ise yazdır top ,
değilse $j=1$ 3'e git
- 10- Bitir

Açıklama:

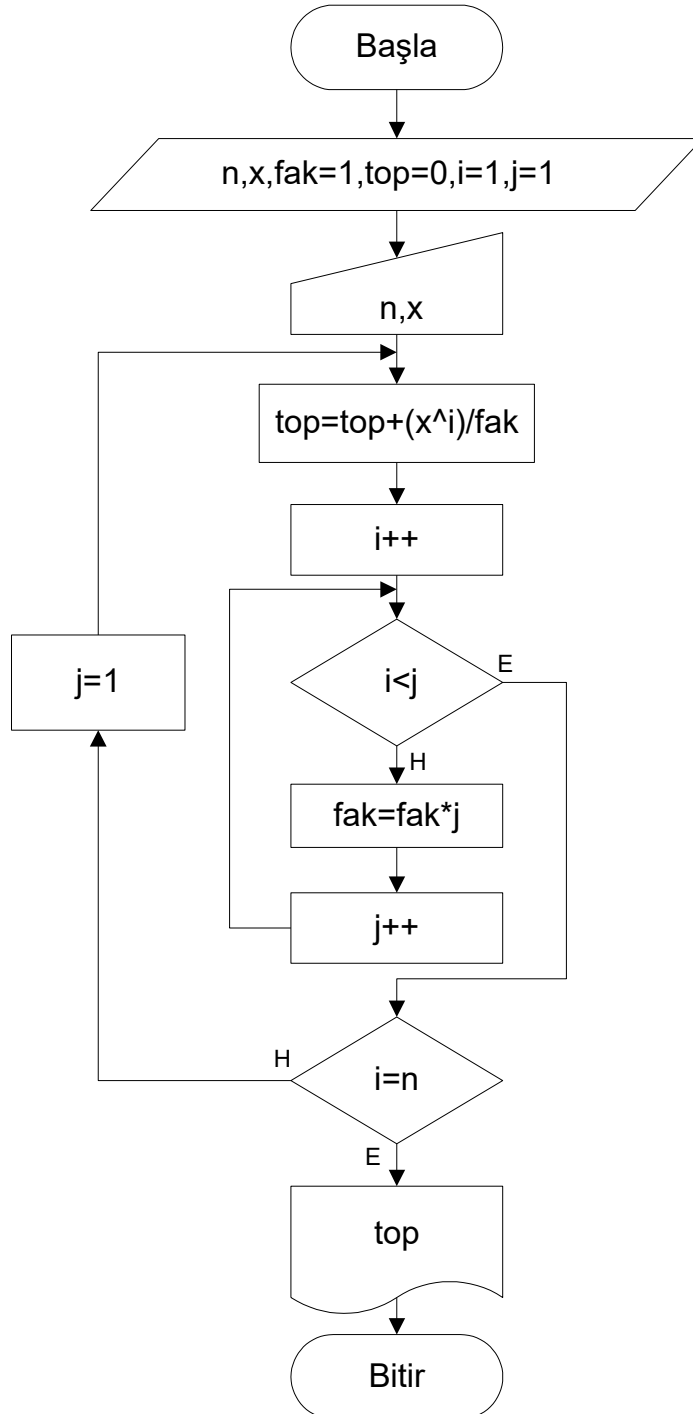
Bu soru 52 .soruya benzer bir sorudur. Girilen N ve X değerine göre sonucu ekrana basan algoritma ve akış diyagramını yaptık. Burada yine faktoriyel için başta $fak=1$ tanımladık, fakat $top=0$ yaptık. Çünkü serimiz X ile başlıyor, önceki soruda ise 1 ile başlıyordu. Buna dikkat etmemiz gerekmektedir. Arka arkaya bu soruları yazmamızın sebebi konunun pekişmesini sağlamaktır. 52. soruda açıklama yaptığımız

için burada hemen algoritma testine geçiyoruz. Faktoriyel olduğuna göre hemen aklımıza döngü gelmelidir.

Algoritma Testi:

x	n	t	F	i
3	3	0	1	1
3	3	3	1	1
3	3	7,5	2	2
3	3	12	6	3

Akış Diyagramı



C Kod:

```
#include <stdio.h>

#include <conio.h>

main()
{
    int n,x,fak=1,top=0,i=1,j=1

    scanf("%d%d",&n,&x);

    top=top+(pow(x,i)/fak);

    while(i!=n)
    {
        for(j=1;j<=i;j++)

        fak=fak*j;

        top=top+(pow(x,i)/fak);
    }

    printf("top :",top);

    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int n, x, fak = 1, top = 0, i = 1, j = 1;
            Console.Write("N = ");
            n = Convert.ToInt32(Console.ReadLine());
            Console.Write("X = ");
            x = Convert.ToInt32(Console.ReadLine());
            while (i != n)
            {
                top = Convert.ToInt32(top + (Math.Pow(x, i) / fak));
                for (j = 1; j <= i; j++)
                {
                    fak = fak * j;
                }
                i++;
            }
            Console.Write(top);
            Console.ReadLine();
        }
    }
}
```

59. $1-1/3+1/5-1/7+1/9-1/11+...$ serisinin n tane terim için toplamını hesaplayan programın algoritma ve akış diyagramını çiziniz.

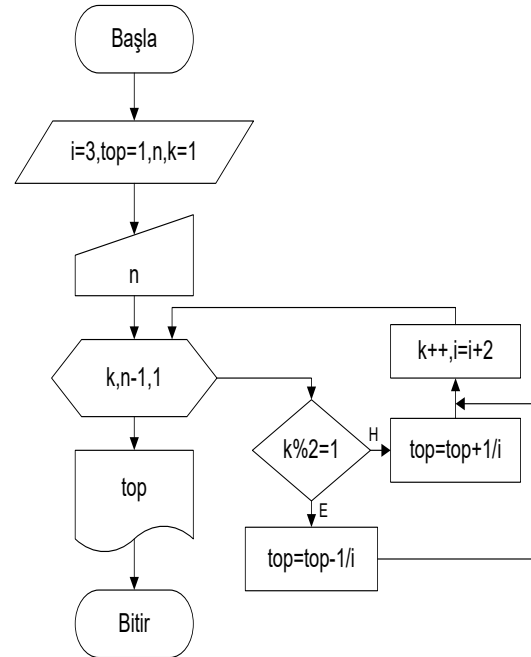
Algoritma Testi:

Algoritma:

- 1- Başla
- 2- $i=3, \text{top}=1, n, k=1$
- 3- n gir
- 4- $k=n-1$ olana kadar 7. adıma kadar olan işlemleri yap
- 5- Eğer $k\%2=1$ ise $\text{top}=\text{top}-1/i$, değilse $\text{top}=\text{top}+1/i$
- 6- $k++, i=i+2$
- 7- Yazdır top
- 8- Bitir

n	i	k	top
3	3	1	1
3	3	2	$1-1/3$
3	3	3	$1-1/3+1/5$
Sonuç = 0,8667			

Akış Diyagramı :



Açıklama:

Seri sorularımızdan bir tanesi de bu sorudur. Bu sorular sizin döngü kurma kabiliyetinizi arttıracaktır. Elimizde sayıların toplanacağı bir değişken bulunmaktadır. Bu değişkenin başlangıç değeri 1'dir. Çünkü $N=3$ girildiğinde örnek olarak, 3 terimi toplayıp ekrana basmamız gerekir. Fakat biz burada her zaman ilk terimi baştan top değişkeninin içine atıyoruz. O da zaten 1'dir. Burada şaşırlacak tek nokta serinin bir - bir + şeklinde ilerlemesidir. Bunu şöyle çözdük. 2.terim $k=1$ olarak geçiyor. O zaman tek olanlar için -, çift olanlar için + kullandık. Bu çift ve tek olan durumu K değişkenine aittir.

C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
float i=3,top=1;
int n,k;
scanf("%d",&n);
for(k=1;k<n;k++)
{
if(k%2==1)
top=top-1/i;
else
top=top+1/i;
i=i+2;
}
printf("%f",top);
getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
class Program
{
static void Main(string[] args)
{
double i = 3, top = 1;
int n, k;
Console.Write("N = ");
n = Convert.ToInt32(Console.ReadLine());
for (k = 1; k < n; k++)
{
if (k % 2 == 1)
{
top = top - 1 / i;
}
else
{
top = top + 1 / i;
}
i = i + 2;
}

Console.Write(top);
Console.ReadLine();
}
}
```

60. Cos(x) fonksiyonu seriye aşağıdaki gibi açılmaktadır. Buna göre dışarıdan girilen x değerinin cosinüs'ünü hesaplayan programın algoritmasını ve akış diyagramını çiziniz

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \dots = \sum_{k=0}^{\infty} \frac{(-1)^k}{(2k)!}$$

Algoritma:

- 1- Başla
- 2- t=1,i,J,fak,x,N
- 3- x,N
- 4- Eğer i=1,n-1,1 ise fak=1
değilse yazdır t
- 5- Eğer j=1,2*i,1 ise fak=fak+j
değilse t=t+((-1)^i)*(x^(2*i))/fak
- 6- Eğer j=1,2*i,1 ise fak=fak+j
değilse t=t+((-1)^i)*(x^(2*i))/fak
- 7- Yazdır t
- 8- Bitir

once de böyle sorular yapmıştık. Bu soruları sıkça ele almamızın amacı alıştırmaları bol bol yaparak programlama konusunda giriş kısmından orta seviyelere geçilebilmesidir. Bu sorunun algoritmasında olmasa bile akış diyagramında döngümüzü kullanabiliriz. Zaten program kodlarına bakıldığında hep döngü komutları görülecektir.

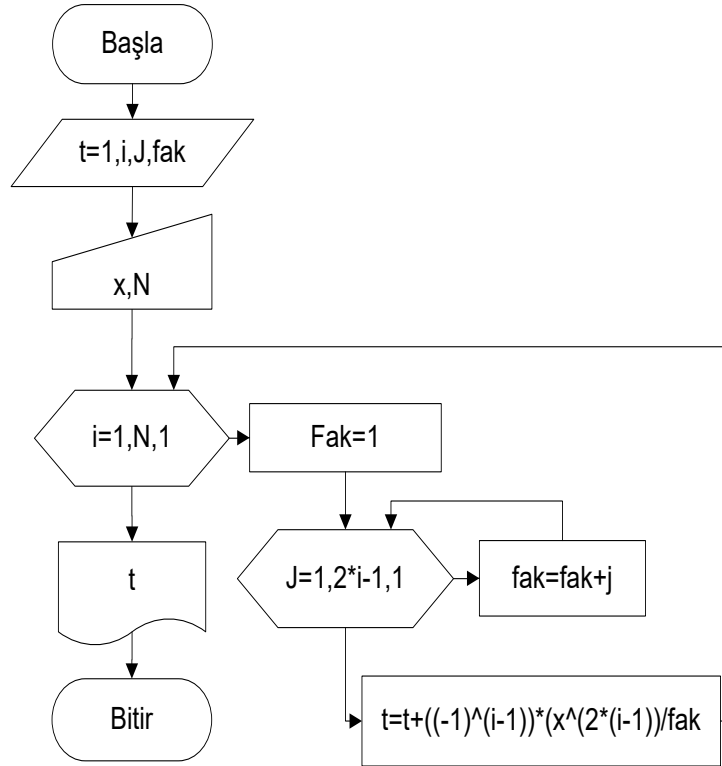
Algoritma Testi:

x	n	fak	t
2	3	1	1
2	3	2!=2	1-4/2
2	3	4!=24	-1+16/24
			Sonuç = - 0,3334

Açıklama:

Bu soruları yaparken bazen öğrencilerimin "Hocam biz matematikçi mi olacağız?" dediklerini duyar gibiyim. Onlara her zaman cevabım şu olmuştur: "Bilgisayarın temeli matematiktir fakat formülü verilmiş bir işlemi programlamak basittir. Çünkü dışarıdan N ve X değerini alacağız ve bu seriyi bu iki değere göre hesaplayacağız." Daha

Akış Diyagramı :



C Kod:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
main()
{
    int x,n,i,j,fak=1;
    float t;
    scanf("%d%d",&x,&n);
    t=1;
    for(i=1;n-1;i++)
    {
        fak=1;
        for(j=1;2*i;i++)
            fak=fak*j;
        t=t+pow(-1,i)*pow(x,(2*i))/fak;
    }
    printf("%f",t);
    getch();
}
```

C#Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int x, n, i, j, fak = 1;
            double t;
            Console.Write("X = ");
            x = Convert.ToInt32(Console.ReadLine());
            Console.Write("N = ");
            n = Convert.ToInt32(Console.ReadLine());
            t = 1;
            for (i = 1; i <= n - 1; i++)
            {
                fak = 1;
                for (j = 1; j <= 2 * i; j++)
                    fak = fak * j;
                t = Convert.ToDouble(t + Math.Pow(-1, i) *
                    Math.Pow(x, (2 * i)) / fak);
            }
            Console.Write(t);
            Console.ReadLine();
        }
    }
}
```

61. Aşağıda verilen işlemin sonucunu girilen değer için hesaplayan programın algoritma ve akış diyagramını çiziniz.

$$\sum_{i=1}^N \left(\frac{1}{i!} + \frac{i}{(n-i)!} \right)$$

Algoritma:

- 1- Başla
- 2- n , t=0 , i=1 , f1=1 , f2=1
- 3- n gir
- 4- Eğer i=n ise 13'e git , değilse devam et
- 5- Eğer j=i ise 8'e git, değilse devam et
- 6- f1 = f1*j
- 7- j++ , 5'e git.
- 8- Eğer j=n-i ise 12'ye git, değilse devam et
- 9- f2 = f2*j
- 10- j++ , 8'e git
- 11- t = t+(1/f1 + i/f2)
- 12- i++ , 4'e git
- 13- Yazdır t
- 14- Bitir

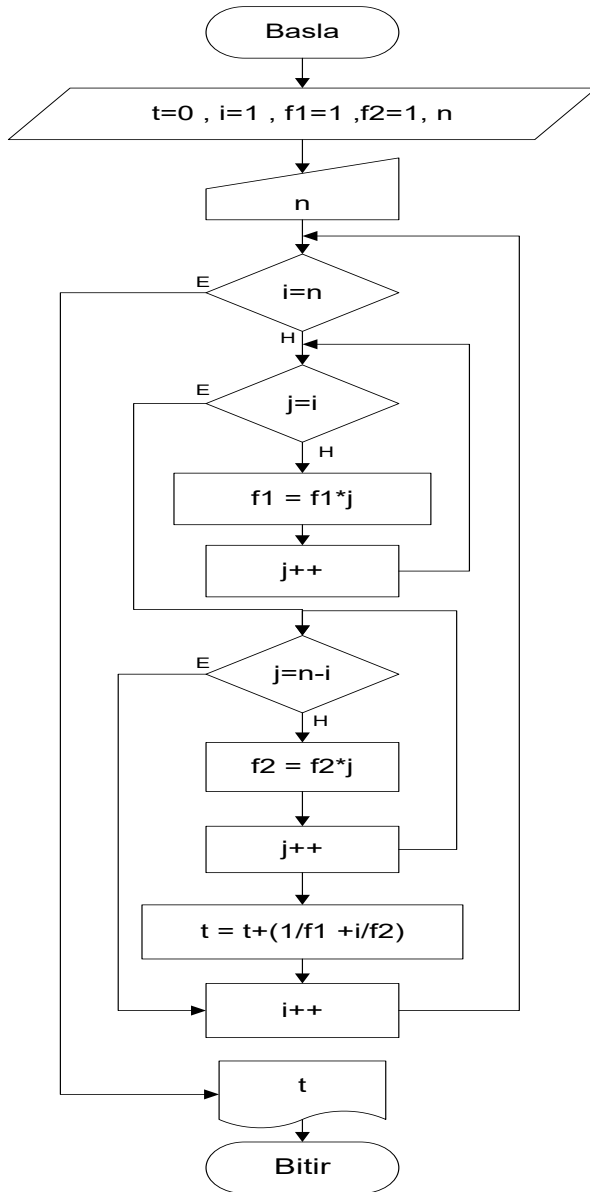
Açıklama:

Seri sorularımızdan belki en karışığı gibi görünen soru budur. Fakat problem çözme yeteneğiniz arttıkça, döngü ve karar yapıları iyice aklınıza yerleştikçe bu sorular sizlere kolay gelecektir. Bu seri sorularında her zaman bir değişken vardır. Çünkü her terim o değişkenin içine toplanarak atılacaktır. Faktoriyel varsa bunu da unutmamalıyız: fak=fak*i'dir. Girilen N değerine göre kaç terim hesaplayacağımızı öğreniyor, i = N oluncaya kadar formülü işletiyoruz.

Algoritma Testi:

i	n	fak	t
1	3	1	0
1	3	1! Ve 2!	1/1+1/2
2	3	2! Ve 1!	3/2+1/2+2
3	3	3! Ve 0!	4+1/6+3
			Sonuç = 7,1666

Akış Diyagramı:



CKod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int n,i,f1=1,f2=1,j,t=0;
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=i;j++)
            f1 = f1*j;
        for(j=1;j<=(n-i);j++)
            f2 = f2*j;
        t=t+((1/f1) + (i/f2));
    }
    printf("toplamlam =%d" ,t);
    getch();
}
```

C# Kod:

```
using System;
namespace dm
{
    class Program
    {
        static void Main(string[] args)
        {
            int n, i, f1 = 1, f2 = 1, j, t = 0;
            Console.WriteLine("N = ");
            n = Convert.ToInt32(Console.ReadLine());
            for (i = 1; i <= n; i++)
            {
                for (j = 1; j <= i; j++)
                {
                    f1 = f1 * j;
                }
                for (j = 1; j <= (n - i); j++)
                {
                    f2 = f2 * j;
                }
                t = t + ((1 / f1) + (i / f2));
            }
            Console.WriteLine("Toplam = " + t);
            Console.ReadLine();
        }
    }
}
```

Bölüm 6

Dizi Algoritma Soru Çözümleri

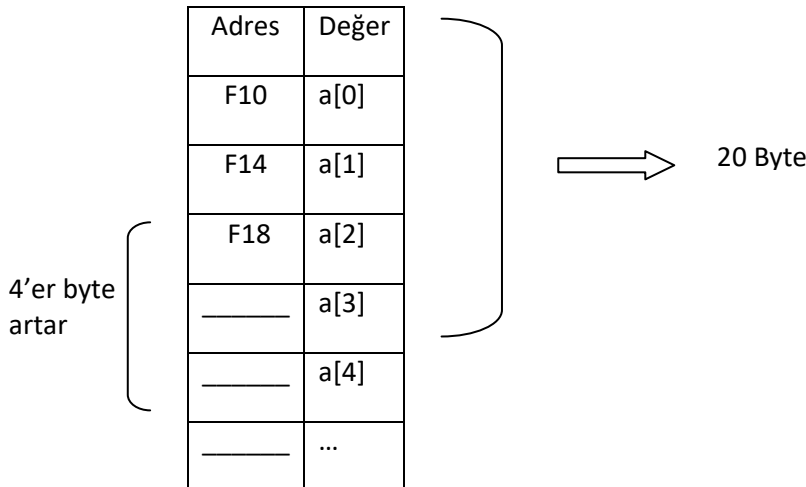
Dizi Kullanımı ve Mantığı

Hangi programlama dilini kullanıyor olursanız olun değişken tanımlamanın mantığı aynıdır. Saklanacak bilginin tipine göre bir tip seçilir, buna bir isim verilir ve hafızada bu değişken için bellek ayrılır. Her değişken için durum aynıdır. C programlama dilinde biz bunları integer sayı, char a gibi tanımlıyoruz. Peki aynı özelliklere sahip birden fazla değişkene ihtiyaç duyarsak ne yapılmalıdır? Bunu bir örnekle açıklamak öğrencilerin daha kolay kavraması açısından uygun olacaktır.

Örneğin bir fabrikada çalışan işçilerin bilgileri tutulmak isteniyor. (Ad, soyad, maaş bilgileri vb.) 100 adet işçimiz olduğunda $100 \times 3 = 300$ tane değişkene ihtiyacımız olacak. Bunun için hafızadan 300 tane farklı adreslerde yer almamız yani değişken tanımlamamız gerekir. Bu da bizim işimize uygun düşmez. Çünkü hafızada aldığımız değişkenler farklı adreslerde bulunur. Erişmek istediğimizde farklı adreslerden 1 işçinin bilgilerini çekmek lazım olur. 2.si de hafızadan çok yer almış oluruz. Bunu yapmanın daha kolay bir yolu bulunmaktadır: Dizileri kullanmak. Dizilerin en önemli özelliği aynı tipteki ve birbiri ile alakalı verileri toplu olarak tanımlamaya yaramasıdır. Bu durumda örneğimizdeki işçi sayısı kadar elemanı olan bir tek dizi değişkeni tanımlar ve bilgileri bir döngü içerisinde indis kullanarak dizinin uygun yerine yerleştirir, ayrıca bir dizinin ilk elemanının adresine ulaştığınızda, dizinin o adresine ulaşmış olursunuz.

Bir örnekle açıklayalım.

İnt a[a]; Gibi bir dizi tanımlaması yapıldığını düşünelim. Windows işletim sisteminde çalışıyorsa derleyici a dizisi için bellekte $4 \times 5 = 20$ byte yer ayırır. Bellekte bu dizinin yerleşimi aşağıdaki gibi olur:



Programcı açısından dizi kullanımının getirdiği en büyük avantaj döngü deyimleri kullanarak tüm dizi elemanlarının kolay bir şekilde erişilebilmesidir.

C programlama dilinde Dizi tanımlama

int dizi[10] → 10 elemanlı bir sayı dizisi

Char dizi[10] → 10 elemalı bir karakter dizisi

Bilinmelidir ki dizinin ilk elemanı d[0], son elemanı d[9]'dur. Yani elaman sayısı N ise 0-9 indisli 10 eleman kullanabiliriz.

D[0]	D[1]	D[2]		...					D[9]
------	------	------	------	-----	------	-------	-------	------	------

Tek boyutlu diziler olabileceği gibi çok boyutlu diziler de olabilir. Bu durum özellikle matris soruları olmak üzere sorularımızda da bulunmaktadır.

int d[satır][sütun] ;

int d[2][5] ;

12345	Deniz	Gezgin	300 ytl	İzmir
12346	Pınar	Gezgin	200 ytl	İzmir

Artık dizi kullanmanın avantajlarını biliyoruz. Bu avantajla aynı türden ve birbiri ile ilişkili bilgilere kolay erişebiliyor ve hafızadan daha iyi yararlanabiliyoruz. Bu nedenle devamlı tutulmasını istediğimiz veriler için dizi tanımlamalıyız.

62. 10 elemanlı bir sayı dizisini girişini yapan algoritma ve akış diyagramını çiziniz.

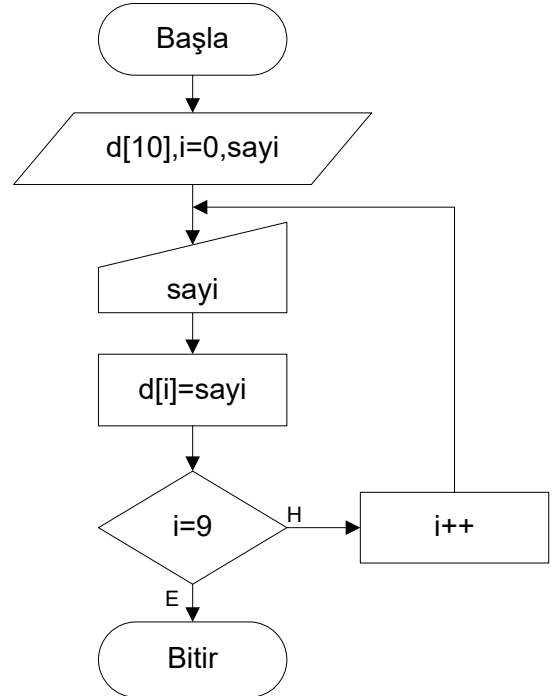
Algoritma:

- 1- Başla
- 2- $d[10]$, $i=0$,sayı
- 3- sayı gir
- 4- $d[i]=sayı$
- 5- Eğer $i=9$ ise devam et
değilse $i++$ 3'e git
- 6- Bitir

Algoritma Testi:

i (eleman Sayısı)	Sayı	D[i]
0	5	D[0]=5
1	6	D[1]=6
....		
10	145	D[9]=145

Akış Diyagramı :



Açıklama:

Dizi sorularının ilki olan bu soruda temel işlem olan diziyi doldurmayı göreceğiz. 10 elemanlı bir dizi denildiğinden indis için i değişkenini alırırız. Dizinin ilk elemanı olması açısından $i=0$ olmalıdır. $i=9$ olana kadar dışarıdan sayı girilir. (Unutulmamalıdır ki dizinin ilk elemanı $d[0]$ 'dir.)

C Kod:

```
#include<stdio.h>
#include<conio.h>
int d[10],i;
main()
{
clrscr();
printf("Dizinin elemanlarini giriniz");
for(i=0;i<10;i++)
{
printf("\n\n");
printf("d[%d]=",i);
scanf("%d",&d[i]);
}
getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
class Program
{
static void Main(string[] args)
{
int[] dizi = new int[10];
int i;
Console.WriteLine("Dizinin elemanlarini
giriniz");
for (i = 0; i < 10; i++)
{
Console.Write(i + ".Elemanı Giriniz = ");
dizi[i] = Convert.ToInt32(Console.ReadLine());
}
Console.ReadLine();
}
}
}
```

63. Fibonacci serisinin ilk 10 terimini dizi kullanarak bulan programın algoritma ve akış diyagramı çiziniz.

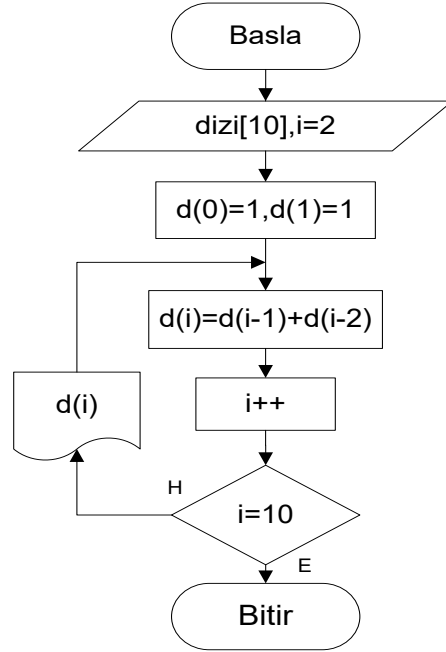
Algoritma:

- 1- Başla
- 2- dizi[10],i=2
- 3- $d(0)=1, d(1)=1$
- 4- $d(i)=d(i-1)+d(i-2)$
- 5- $i++$
- 6- Eğer $i=10$ ise devam,
değilse yazdır $d(i)$, 4.adıma git
- 7- Bitir

Algoritma Testi:

d[0]	d[1]	d[2]	d[3]	d[4]
1	1	2	3	5
d[5]	d[6]	d[7]	d[8]	d[9]
8	13	21	34	55

Akış Diyagramı:



Açıklama:

Fibonacci Serisinin ilk 10 terimi sorusunu dizi kullanmadan da daha önceki sorularda çözmüştük. Fakat orada ekrana basıp program bitiyordu. Yani 6. terime tekrar dönmek istense orada dönmek mümkün olmayacaktır. Çünkü bir yerde tutulmamaktadır. Dizi kullanarak istediğimiz an istediğimiz terime ulaşabiliriz. Bunu bir kütüphanenin (dizi) raflarındaki kitaplara (terim-değer) benzetebiliriz. $i=2$ den başlatmamızın sebebi de ilk iki terim sabit 1 olduğu için , sonra programın döngüye giriyor olmasıdır.

C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int dizi[10] ,i;
    dizi[0]=1,dizi[1]=1;
    printf("%d %d ",dizi[0],dizi[1]);
    for(i=2;i<10;i++)
    {
        dizi[i]=dizi[i-1]+dizi[i-2];
        printf("%d ",dizi[i]);
    }
    getch();
}
```

C#Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] dizi = new int[11];
            int i;
            dizi[0] = 1;
            dizi[1] = 1;
            Console.Write(dizi[0]);
            Console.Write(dizi[1]);
            for (i = 2; i <= 10; i++)
            {
                dizi[i] = dizi[i - 1] + dizi[i - 2];
                Console.Write(dizi[i]);
            }
            Console.ReadLine();
        }
    }
}
```

64. Girilen bir kelimenin uzunluğunu bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- dizi[100],i=0,kelime
- 3- kelime gir
- 4- Eğer d[i]=null ise yazdır i+1, 5'e git değilse devam et
- 5- i++ ve 4'e git
- 6- Bitir

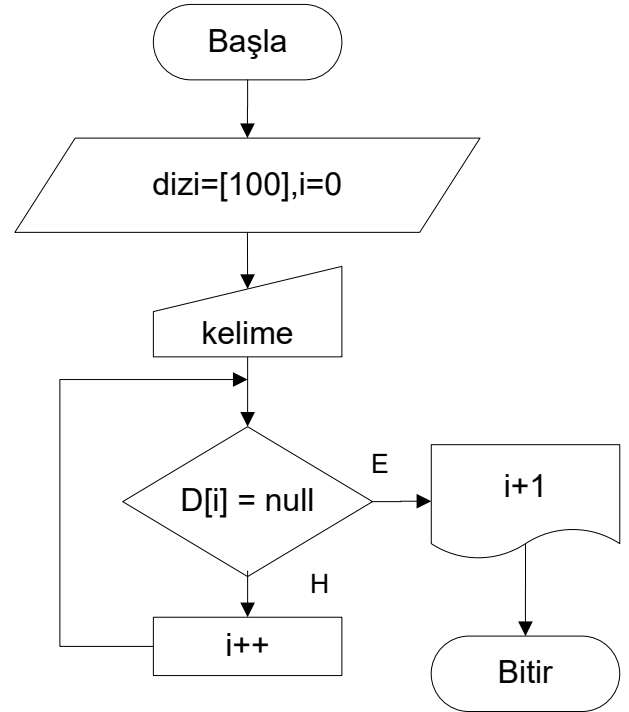
Açıklama:

Bu soruda bir karakter dizisi tanımlıyoruz. Kelimeyi de okutmak için gets() fonksiyonunu kullanabiliriz (c 'de) veya kullanıcının boş bir değer girmesi yerine karakterler okutulup diziyeye atılabilir. 100 eleman tanımlamamızın sebebi dizi tanımlanırken, dinamik dizi tanımlamadığımızda her zaman kaç elemanlı olduğunu derleyiciye belirtmektir. Biz burada onun için 100 elemanlı demeyi tercih ettik. Sonuçta NULL girince döngü sonlanacaktır. Gets() ile sisteme kelime okutursak, son karakteri girdikten sonra dizinin sonu, kendine has bir işaret koyar('\0' gibi). Bu da bize dizinin sonuna geldiğimizi gösterir.

Algoritma Testi:

d[0]	d[1]	d[2]	d[3]	d[4]	i=(4+1)=5
'D'	'E'	'N'	'İ'	'Z'	Null(Boş)

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    char d[100];
    int i=0;
    gets(d);
    for(i=0;i<100;i++)
    {
        if(d[i]=='\0')
        {
            printf("%d",i);
            break;
        }
    }
    getch();
}
```

C#Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i;
            string[] dizi = new string[100];
            string deger = "null";
            for (i = 0; i <= 100; i++)
            {
                Console.Write(i + ".Karakteri Giriniz = ");
                dizi[i] = Convert.ToString(Console.ReadLine());
                if (dizi[i] == deger)
                {
                    goto dmğ;
                }
            }
            dmğ:
            Console.Write(i);
            Console.ReadLine();
        }
    }
}
```

65. Bir sayı dizisinin en büyük elemanını bulan programın algoritma ve akış diyagramı çiziniz.

Algoritma:

- 1- Başla
- 2- dizi,es,enb,i=0
- 3- enb=dizi[i]
- 4- i++
- 5- Eğer enb<dizi[i] ise enb=dizi[i],
değilse devam et
- 6- Eğer i=es-1 ise enb=dizi[i],
değilse devam et
- 7- Yazdır enb
- 8- Bitir

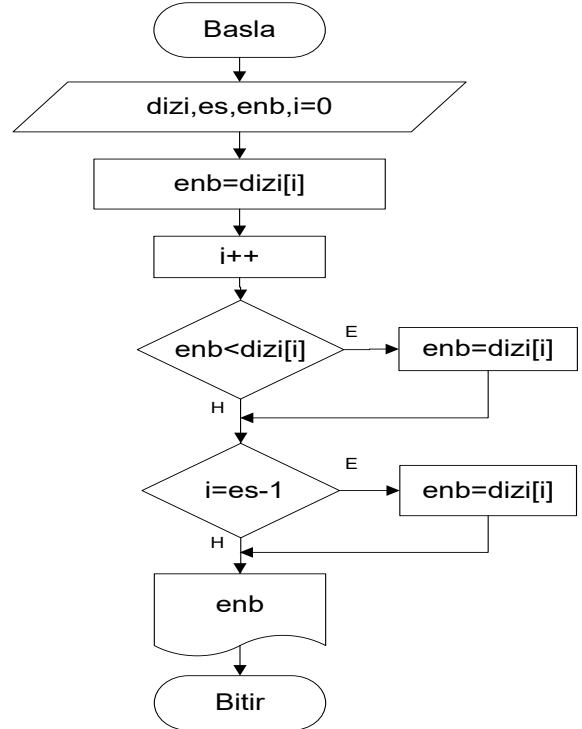
Açıklama:

Burada dizimize 100 tane sayı girilmiş gibi kabul ediyoruz. Enb değişkenimize ilk önce dizinin ilk elemanını atıyoruz. i=99 olana kadar dizinin diğer elemanları ile enb içindeki sayıyı karşılaştırıyoruz. Her zaman karşılaştırmada büyük olanı enb içine atıyoruz. Son olarak enb değişkeni ekrana basıyoruz. Daha önce enb, enk sorularını nasıl çözeceğimizi anlatmıştık. Burada sadece dizi içinden en büyüğü buluyoruz. Örnek olması açısından algoritma testini 5 eleman için yapıyoruz.

Algoritma Testi :

d[i]	enb	i
5	5	0
6	6	1
3	6	2
8	8	3
4	8	4

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
#define n 10
main()
{
    int dizi[n];
    int i,enb=0,k;
    printf("dizinin elemanlarini gir\n");
    for(i=0;i<n;i++)
    {
        scanf("%d",&dizi[i]);
        if(enb < dizi[i])
            enb=dizi[i];
    }
    printf("%d",enb);
    getch();
}
```

C#Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] dizi = new int[11];
            int i = 0;
            int enb;
            enb = dizi[i];
            for (i = 1; i <= 10; i++)
            {
                Console.Write(i + ".Sayıyı giriniz = ");
                dizi[i] = Convert.ToInt32(Console.ReadLine());
                if (enb < dizi[i])
                {
                    enb = dizi[i];
                }
            }
            Console.Write("Dizinin en büyük elemanı = "+enb);
            Console.ReadLine();
        }
    }
}
```

66. Girilen kelimeyi tersten yazdıran programın algoritmasını ve akış diyagramını çiziniz.

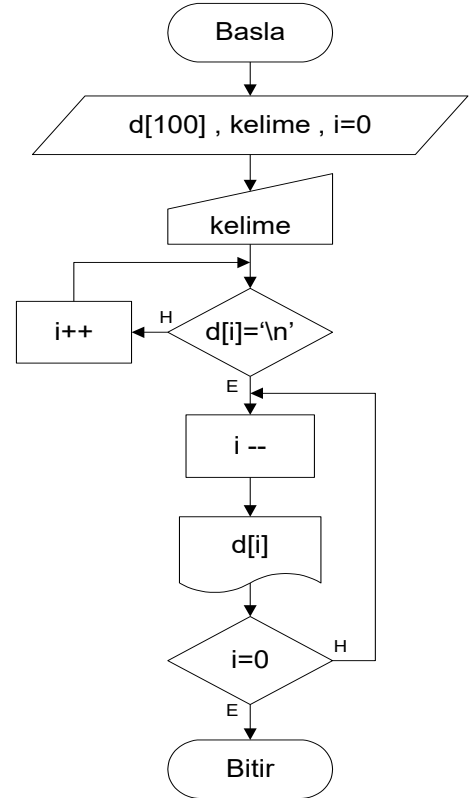
Algoritma Testi:

Algoritma:

- 1- Başla
- 2- d[100] , kelime , i=0
- 3- kelime gir
- 4- Eğer d[i]='\n' ise 5'e git ,
değilse i++ 4'e git
- 5- i --
- 6- Yazdır d[i]
- 7- Eğer i=0 ise devam et,
değilse 5'e git
- 8- Bitir

i(7-1)	d[i]
6	N
5	A
4	K
3	T
2	R
1	E
0	M

Akış Diyagramı:



Açıklama:

Kullanıcıdan bir kelime girmesi istenir. Kelimenin harfleri tanımladığımız karakter dizisine girilir. Kelimenin son karakteri girilip enter tuşuna basıldığında dizi sonu olarak bir işaret konulur ('\o' gibi). Dizi sonu gösteren karakterin bulunduğu dizinin indis değeri 1 eksilterek, i=0 olana kadar her dizi elemanı yazdırılır. Program, dizinin ilk elemanını da yazdırınca biter.

Mertkan → 7 eleman

C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    char d[100];
    int i=0;
    gets(d);
    for(i=0;i<100;i++)
    {
        if(d[i]=='\0')
            break;
    }
    while (i>-1)
    {
        printf("%c",d[i]);
        i--;
    }
    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i;
            int n;
            Console.Write("Kaç karakterden oluşan
kelime girmek istiyorsunuz = ");
            n = Convert.ToInt32(Console.ReadLine());
            char[] dizi = new char[n+1];
            for (i = 1; i <= n; i++)
            {
                Console.Write(i+".Karakteri Giriniz = ");
                dizi[i] = Convert.ToChar(Console.ReadLine());
            }
            Console.Write("Girmiş olduğunuz kelime = ");
            for (i = 1; i <= n; i++)
            {
                Console.Write(dizi[i]);
            }
            Console.WriteLine();
            Console.Write("Kelimenin Tersten yazılışı = ");
            for (i = n; i >= 1; i--)
            {
                Console.Write(dizi[i]);
            }
            Console.ReadLine();
        }
    }
}
```

67. Bir decimal sayıyı binary (10'luk-2'lik) sayıya çeviren programın algoritmasını ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- d[10] , sayı , i=0
- 3- sayı gir
- 4- d[i]=sayı%2, sayı=sayı/2
- 5- Eğer sayı<2 ise d[i+1]=sayı 6'ya git
,
değilse i++ 4'e git
- 6- Yazdır d[i+1]
- 7- Eğer i=(-1) ise 8 'e git
değilse i-- 6'ya git
- 8- Bitir

Bu soruyu dizi kullanmadan da yapmıştık. Fakat her zaman belirttiğimiz gibi her hanedeki sayılara tek tek ulaşmak

istiyorsak bunları diziye girmeliyiz. Sorunun mantığı

d[0]	d[1]	d[2]	d[3]
1	0	1	0

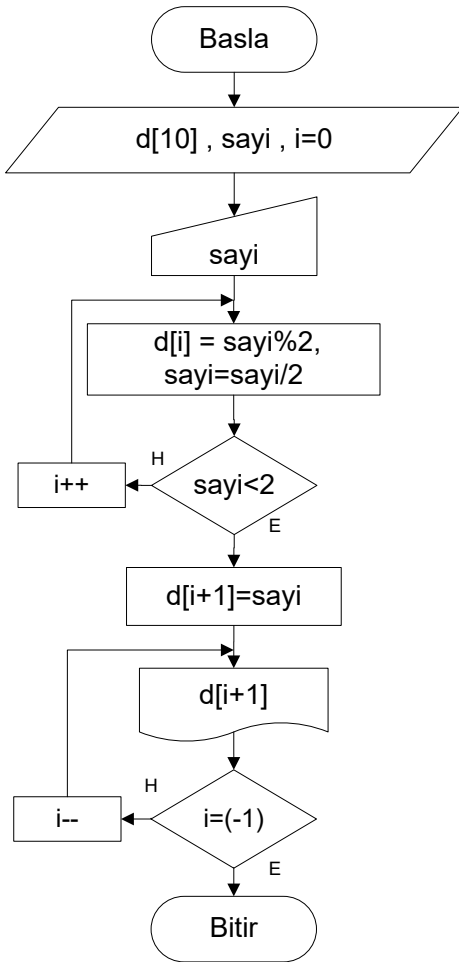
daha önceki soru ile aynıdır.

Algoritma Testi :

Akış Diyagramı:

Açıklama:

i	sayı	d[i]
3	10	0
2	5	1
1	2	0
0	1	1



C Kod:

```

#include <stdio.h>
#include <conio.h>
main()
{
    int d[8],sayi,i;
    scanf("%d",&sayi);
    for(i=0;i<8;i++)
    {
        d[i]=sayi%2;
        sayi=sayi/2;
        if(sayi<2)
        {
            i=i+1;
            d[i]=sayi;
            break;
        }
    }
    while(i>-1)
    {
        printf("%d",d[i]);
        i--;
    }
    getch();
}
    
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] dizi = new int[8];
            int sayı, i;
            Console.Write("Sayıyı Giriniz = ");
            sayı = Convert.ToInt32(Console.ReadLine());
            for (i = 0; i < 8; i++)
            {
                dizi[i] = sayı % 2;
                sayı = sayı / 2;
                if (sayı < 2)
                {
                    i = i + 1;
                    dizi[i] = sayı;
                    break;
                }
            }
            while (i > -1)
            {
                Console.Write(dizi[i]);
                i--;
            }
            Console.ReadLine();
        }
    }
}
```

68. 10 elemanlı bir sayı dizisinde en küçük elemanın bu dizinin kaçınıcı elemanı olduğunu bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- dizi[10],i,x=1
- 3- i=9 olana kadar 6. adıma kadar olan işlemleri yap
- 4- sayı gir
- 5- dizi[i]=sayı
- 6- i=1
- 7- enk=d[0]
- 8- Eğer $enk < d[i]$ ise devam et, değilse $enk = d[i]$, $x = i + 1$
- 9- Eğer $i = 9$ ise devam et, değilse $i++$ 8'e git
- 10- Yazdır x
- 11- Bitir

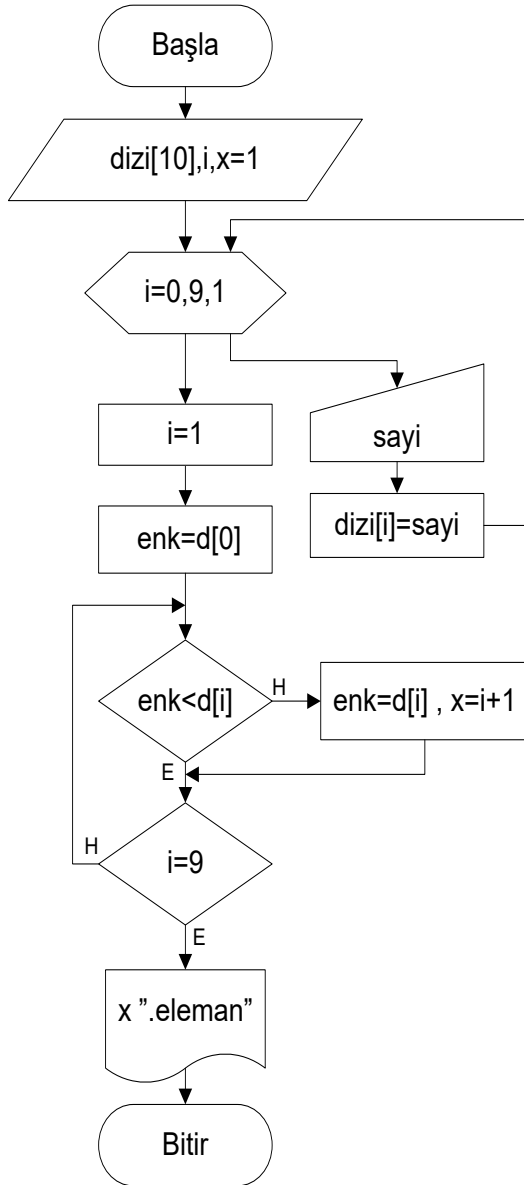
Algoritma Testi :

Enk	i	d[i]
5	0	5
5	1	10
2	2	2
2	3	8
1	4	1

Açıklama:

Bu soru bize dizinin eleman sayısı ile dizi indisi arasındaki farkı gösteren bir örnektir. En küçük eleman örnek olarak d[4] 'te bulunduyorsa bu dizinin $4 + 1 = 5$. elemanıdır. Bunu unutmamalıyız. Enk değişkeni her sayı ile karşılaştırıldığında takas alanı gibi en son değerini alana kadar değişir. Bunu 5 sayı ile test edelim ama program kodlarımız tam algoritmanın aynısıdır.

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
int d[10],i,x=0,enk;
main()
{
    scanf("%d",&d[0]);
    enk=d[0];
    for(i=1;i<10;i++)
    {
        scanf("%d",&d[i]);
        if( enk>d[i])
        {
            enk=d[i];
            x=i+1;
        }
    }
    printf("En küçük eleman %d.sirada",x);
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] dizi = new int[10];
            int i = 0, x = 0, enk;

            Console.Write(i + ".Elemanı Giriniz = ");
            dizi[0] = Convert.ToInt32(Console.ReadLine());
            enk = dizi[0];

            for (i = 1; i < 10; i++)
            {
                Console.Write(i + ".Elemanı Giriniz = ");
                dizi[i] = Convert.ToInt32(Console.ReadLine());
                if (enk > dizi[i])
                {
                    enk = dizi[i];
                    x = i + 1;
                }
            }

            Console.Write("En küçük eleman " + x + " sırada");

            Console.ReadLine();
        }
    }
}
```

69.N elemanlı bir dizi bulunmaktadır. Klavyeden girilen sayılar diziye, bir tane baştan bir tane sondan olmak üzere yerleştirilmektedir. Örneğin ilk sayı birinci elemana, ikinci sayı N'inci elemana, üçüncü sayı ikinci elemana, dördüncü sayı N-1'inci elemana... şeklinde yerleştirilmektedir. N tane sayıyı klavyeden okuyup diziye yerleştiren ve diziyi ekranda görüntüleyen programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $n, \text{sayi}, d[n], i=0$
- 3- n gir
- 4- $i=n-1$ olana kadar 10. adıma kadar olan işlemleri yap
- 5- sayı gir
- 6- $d[i]=\text{sayi}$
- 7- sayı gir
- 8- $d[n-1]=\text{sayi}$
- 9- $n--$
- 10- $i=0$
- 11- $i=n-1$ olana kadar yazdır $d[i]$
- 12- Bitir

dizinin ortasına gelince sayılar yerleşmiş hem de program bitmiş olur. $N--$ yapmak dizinin sonundan başa doğru gelmemizi sağlar.

Algoritma Testi:

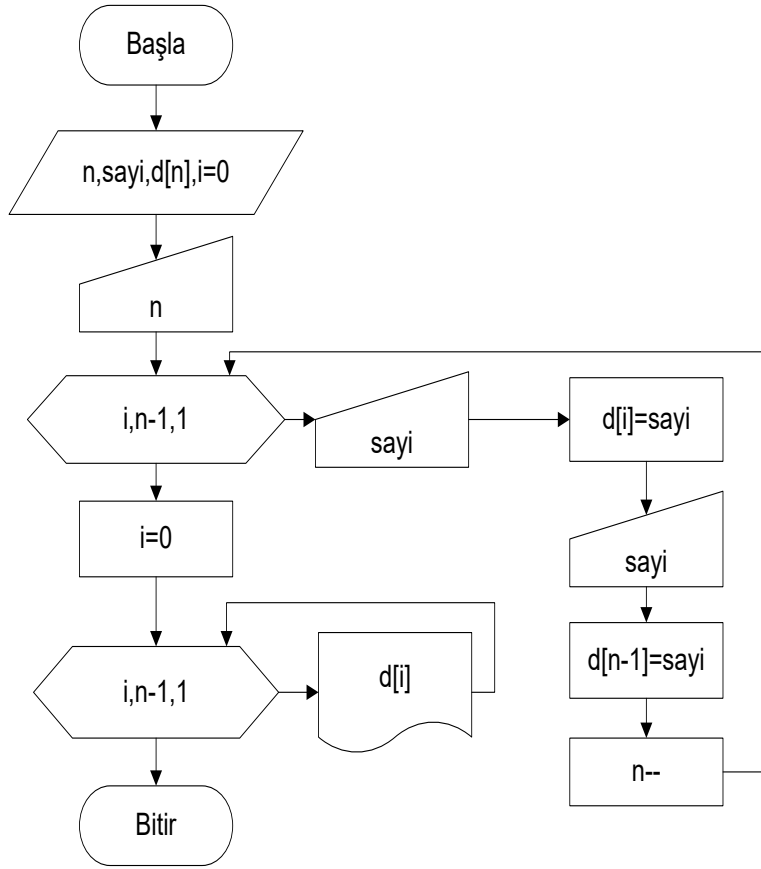
N=5 için

i	D[i]	D[n-1]	N
0	D[0]	D[4]	5
1	D[1]	D[3]	4
2	D[2](ilk)	D[2](işlemez)	3

Açıklama:

Bu soru dizi ve diziyle döngünün kullanımı ile alakalı önemli bir sorudur. N elemanlı bir dizinin ilk elemanı $d[0]$, son elemanı ise $d[N-1]$ 'dir. i her zamanki gibi sayacımızdır. Sayacımız $i = n-1$ olana kadar bir tane baştan, bir tane sondan değer girilir. Burada işlemi tamamlayan, döngü içerisinde N değerini azaltmaktır. Böylece hem

Akış Diyagramı :



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int n=5;
    int d[5];
    int i;
    for(i=0;i<n;i++)
    {
        scanf("%d",&d[i]);
        if((n-1)!=i)
            scanf("%d",&d[n-1]);
        n=n-1;
    }
    n=5;
    for(i=0;i<n;i++)
    {
        printf("%d",d[i]);
    }
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] dizi = new int[5];
            int n = 5;
            int i;
            for (i = 0; i < n; i++)
            {
                Console.Write(i + ".Sayıyı Giriniz = ");
                dizi[i] = Convert.ToInt32(Console.ReadLine());
                if ((n - 1) != i)
                {
                    Console.Write(n + ".Sayıyı Giriniz = ");
                    dizi[n - 1] = Convert.ToInt32(Console.ReadLine());
                    n = n - 1;
                }
            }
            n = 5;
            for (i = 0; i < n; i++)
            {
                Console.WriteLine(dizi[i]);
            }
            Console.ReadLine();
        }
    }
}
```

70. Aşağıdaki Çıktıyı Veren Programın Algoritma ve Akış Diyagramını Çiziniz.

Bilgisayar
ilgisayarb
lgisayarbi
gisayarbil
:
:
Bilgisayar

Algoritma:

- 1- Başla
- 2- d [10]='BİLGİSAYAR',i=0,j=0,k=0
- 3- Yazdır d[i]
- 4- Eğer i=9 ise devam et ,
değilse i++ 3'e git
- 5- j++,i=j
- 6- Eğer k<j ise devam et ,
değilse 8'e git
- 7- yazdır d [k], k++ 6 'ya git
- 8- k=0
- 9- Eğer i<10 ise 3'e git , değilse
devam et
- 10- Bitir

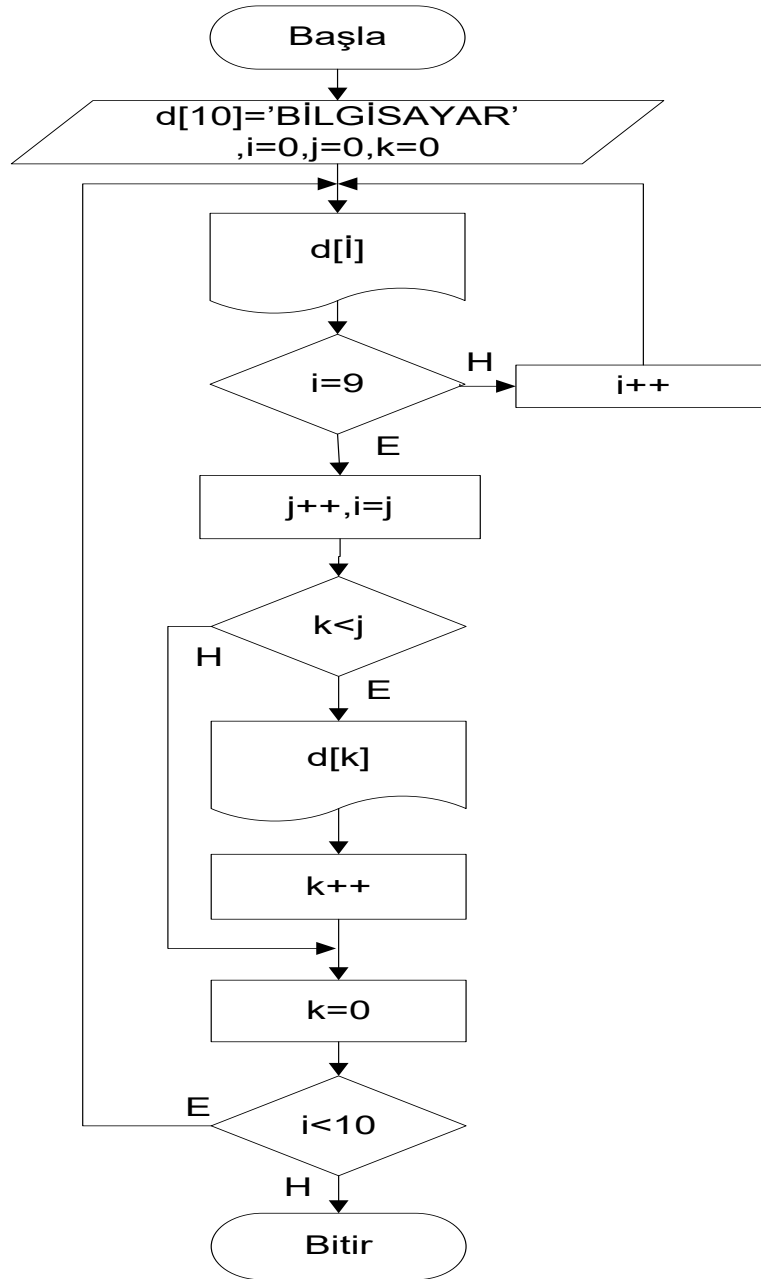
Algoritma Testi:

i	k	J	Çıktı
0...9	1	0	D[i]
1...9	0	1	D[i] , d[k]
2....9	0..1	2	D[i] , d[k]
3....9	0...2	3	D[i] , d[k]
4....9	0..3	4	D[i] , d[k]
5....9	0...4	5	D[i] , d[k]
6....9	0...5	6	D[i] , d[k]
7....9	0...6	7	D[i] , d[k]
8....9	0...7	8	D[i] , d[k]
9...9	0...8	9	D[i] , d[k]
0..9	1	0	D[i]

Açıklama:

Bu soru da dizi ve döngünün birlikte kullanıldığı önemli sorulardandır. Yukarıda bulunan tablodaki şekli ekrana basmak için ilk tanımlama esnasında BİLGİSAYAR kelimesini diziye direkt giriyoruz. Bu örneğin algoritma testi de şöyledir:

Akış Diyagramı :



C Kod:

```
#include <conio.h>
#include <stdio.h>
main()
{
    clrscr();
    int i,j=0,k=1;
    char d[10]={'B','i','l','g','i','s','a','y','a','r'};
    for(i=0;i<10;i++)
    {
        printf(" %c ",d[i]);
        j++;
        i=j;
        while(i<10)
        {
            if(k<j)
            {
                printf(" %c ",d[k]);
                k++;
            }
            else
                k=0;
        }
    }
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            char[] dizi = new char[10];
            dizi[0] = 'B'; dizi[1] = 'i';
            dizi[2] = 'L'; dizi[3] = 'G';
            dizi[4] = 'i'; dizi[5] = 'S';
            dizi[6] = 'A'; dizi[7] = 'Y';
            dizi[8] = 'A'; dizi[9] = 'R';
            int i = 0; int j = 0;
            int k = 0;
            dmğ:
            for (i = j; i <= 9; i++)
            {
                Console.Write(dizi[i]);
            }
            i = j;
            while (k<j)
            {
                Console.Write(dizi[k]);
                k++;
            }
            j++;
            k = 0;
            if (i < 10)
            {
                Console.WriteLine();
                goto dmğ;
            }
            Console.ReadLine();
        }
    }
}
```

71. Bir sınıftaki 50 öğrencinin bir dersten aldıkları yıl sonu notları veriliyor. Başarı notu 50 olduğuna göre kaç öğrencinin başarılı olduğunu bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $n, bas=0, i=0, ogr[n], sayi$
- 3- n gir
- 4- Sayı gir
- 5- $ogr[i]=sayi$
- 6- Eğer $i=n-1$ ise 8'e git,
değilse devam et
- 7- Eğer $ogr[i] \geq 50$ ise $bas=bas+1, i++$
4'e git,
değilse $i++$ 4'e git
- 8- Yazdır bas
- 9- Bitir

büyük ve küçüklüğüne göre sayacı arttıracak,
daha sonra sayacı ekrana basacaktır.

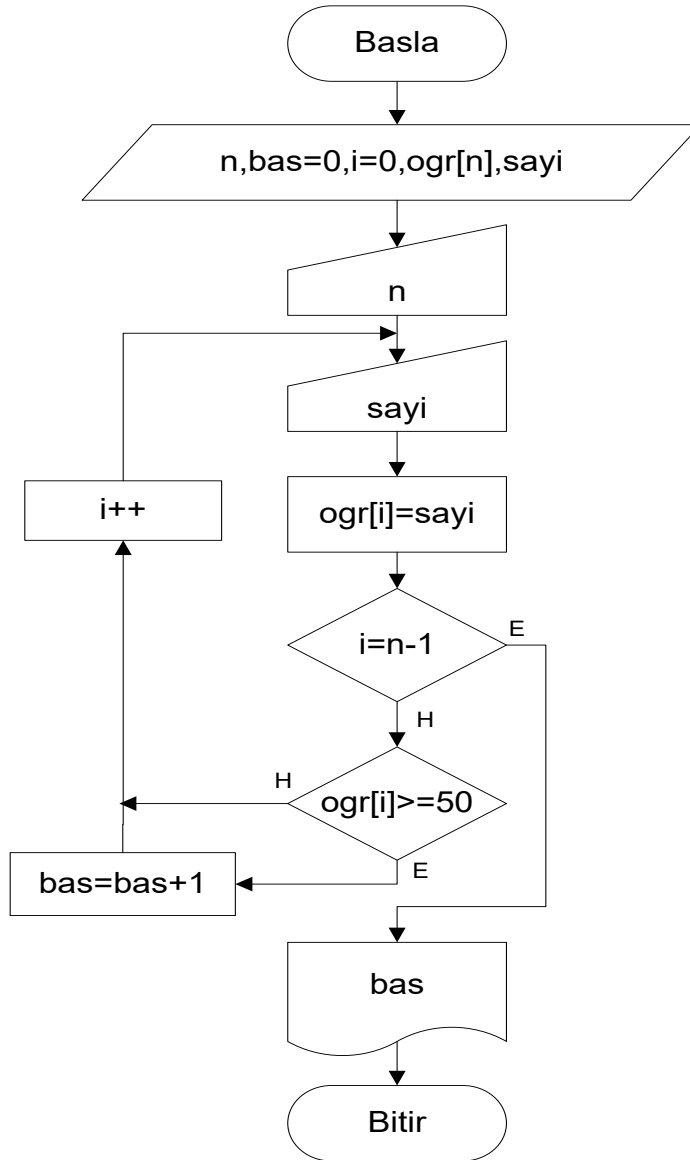
Algoritma Testi:

i	n	ogr[n]	bas
0	50	50	1
1	50	35	1
2	50	60	2
3	50	55	3
....			---
49	50	89	15(örnek)

Açıklama:

Biz öğretmenler, yıl sonunda geçenlere ve ortalamaya bakarız. Bu nedenle notları girdiğimizde kaç öğrencinin geçtiğini hesaplayan bir program koyduk. Aynı zamanda diziler için güzel bir örnek teşkil etmesi açısından 50 öğrenci için 50 ayrı değişken tanımlamak gerekemekteydi. Bu da gösterilmiş oldu. Algoritma, girilen notları diziye atacak ve 50 den

Akış Diyagramı:



C Kod:

```
#define n 50
#include <conio.h>
#include <stdio.h>
main()
{
    int bas=0,i=0,ogr[n],sayi;

    while(i<50)
    {
        scanf("%d",&sayi);
        ogr[i]=sayi;
        if(ogr[i]>=50)
            bas=bas+1;
        i++;
    }
    printf("%d",bas);
    getch();
}
```

C#Kod :

```
using System;

namespace dmrg
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] ogr = new int[51];
            int bas = 0;
            int i;
            for (i = 1; i <= 50; i++)
            {
                Console.Write(i + ".Öğrencinin Notunu Giriniz = ");
                ogr[i] = Convert.ToInt32(Console.ReadLine());
                if (ogr[i] >= 50)
                {
                    bas = bas + 1;
                }
            }
            Console.Write("Başarılı öğrenci sayısı = " +
                bas);
            Console.ReadLine();
        }
    }
}
```

72. 10 elemanlı bir sayı dizisinin ortalaması tam sayı ise bu sayıdan dizide kaç tane olduğunu veren programın algoritma ve akış diyagramını çiziniz.

Algoritma:

soru, dizi ile oynanan sorulardan bir tanesidir.
Artık döngü sizin için çok kolay olmalı ☺

- 1- Başla
- 2- $d[10], i=0, \text{sayı}, \text{top}, \text{ort}, \text{sayac}=0$
- 3- $i=9$ olana kadar 6. adıma kadar olan işlemleri yap
- 4- sayı gir
- 5- $d[i]=\text{sayı}, \text{top}=\text{top}+\text{sayı}$
- 6- Eğer $\text{top}\%10=0$ ise devam et, değilse 10'a git
- 7- $\text{ort}=\text{top}/10$
- 8- Eğer $\text{ort}=d[i]$ ise $\text{sayac}++$, değilse devam et
- 9- Eğer $i=0$ ise yazdır sayac, değilse $i--$ 8'e git
- 10- Bitir

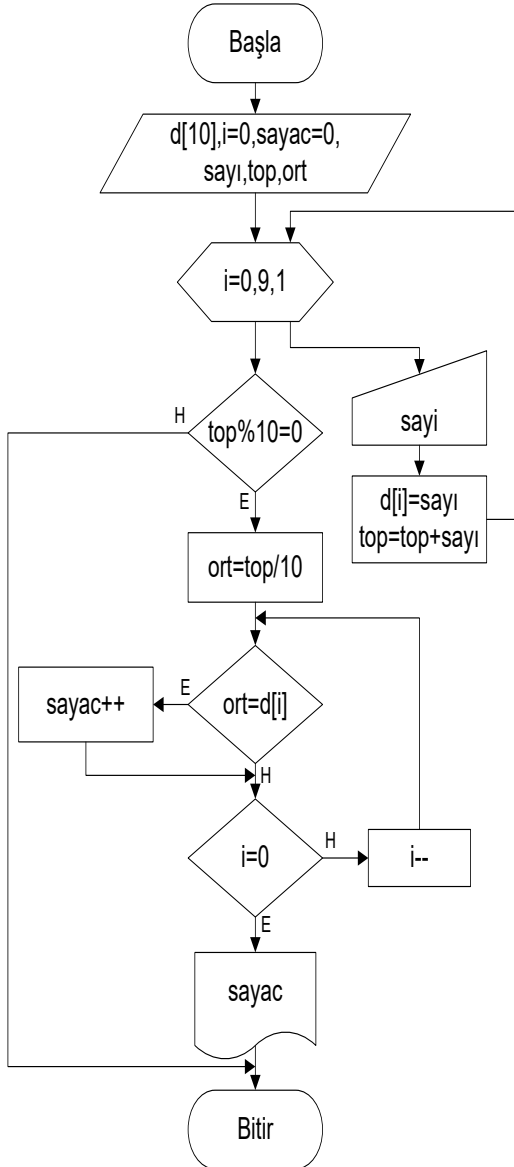
Algoritma Test : (5 elemanlık)

S1	S2	S3	S4	S5	Sayac	ort
1	3	4	2	5	1(3)	15/5=3
1	4	4	2	5	0	16/5=3,2

Açıklama:

10 elemanlı bir sayı dizisinin tüm elemanlarını toplayıp 10'a böldüğümüzde çıkan sayı tam sayı ise bunu dizinin elemanları ile tek tek karşılaştırmalı ve buna göre sayacımızı arttırmalıyız. Sonuçta da dizinin sonu gelince sayacı ekrana basmalıyız. Ancak çıkan ortalama değeri tam çıkmazsa program sonlanacaktır. (Bu

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
int d[10],top=0,i,ort,sayac=0;
main()
{
    clrscr();
    for(i=0;i<10;i++)
    {
        scanf("%d",&d[i]);
        top=top+d[i];
    }
    if(top%10==0)
    {
        ort=top/10;
        for(i=0;i<10;i++)
        {
            if(d[i]==ort)
                sayac++;
        }
        printf("%d tane var",sayac);
    }
    else
        printf("ortalama tam sayi degil");
    getch();
}
```

C#Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] dizi = new int[10];
            int top=0,i,ort,sayac=0;
            for (i = 0; i < 10; i++)
            {
                Console.Write(i + ".Sayıyı Giriniz = ");
                dizi[i] = Convert.ToInt32(Console.ReadLine());
                top = top + dizi[i];
            }
            if (top % 10 == 0)
            {
                ort = top / 10;
                for (i = 0; i < 10; i++)
                {
                    if (dizi[i] == ort)
                    {
                        sayac++;
                    }
                }
                Console.Write(sayac + " tane var");
            }
            else
            {
                Console.Write("ortalama tam sayi degil");
            }
            Console.ReadLine();
        }
    }
}
```

73. İstenildiği kadar elemandan oluşan bir sayı dizisinde negatif ve pozitif elemanların sayısını bulan algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $i=1, n=0, p=0, c$
- 3- $d[i]$ gir
- 4- Eğer $d[i]=0$ ise 7'e git
- 5- Eğer $d[i]<0$ ise $n=n+1$ ve 7'e git
- 6- $p=p+1$
- 7- c gir
- 8- Eğer $c='H'$ ise 10'a git
- 9- $i++$, 3'e git
- 10- Yazdır p, n
- 11- Bitir.

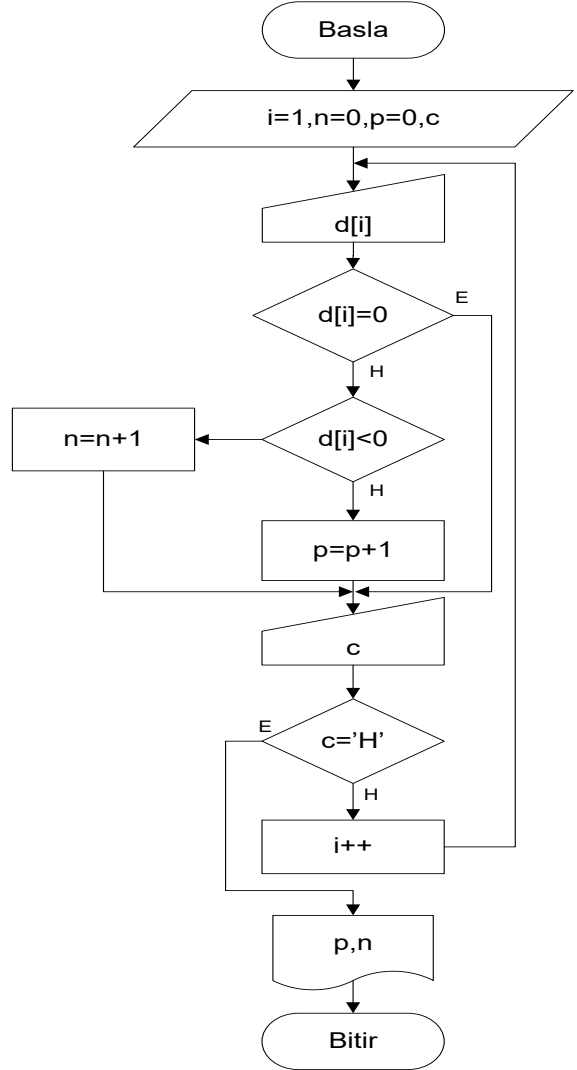
Açıklama:

Girilen eleman sayısı kadar oluşturulan dizinin elemanlarını doldurduktan sonra, dizi elemanlarını baştan yani 0. indisten başlayarak, dizi sonuna kadar $0 < \text{sayı}$ ya da $0 > \text{sayı}$ şartına göre negatifleri n sayacına, pozitifleri p sayacına atan ve bunları en son ekrana basan programın algoritmasıdır.

Algoritma Testi: (5 elemanlık)

S1	S2	S3	S4	S5	P	N
-1	3	-4	2	5	3	2

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i;
    int n = 0;
    int p = 0;
    int dizi[5];
    for (i = 0; i < 5; i++)
    {
        scanf("%d",&dizi[i]);
        if (dizi[i] > 0)
            p = p + 1;
        else if (dizi[i] < 0)
            n = n + 1;
    }
    printf("Pozitif sayisi = %d ",p);
    printf("\n\n");
    printf("Negatif sayisi = %d ",n);
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] dizi = new int[11];
            int i;
            int n = 0;
            int p = 0;
            int topn=0;
            int topp=0;
            int ortn=0;
            int ortp=0;
            for (i = 1; i <= 10; i++)
            {
                Console.Write(i + ".Sayıyı Giriniz = ");
                dizi[i] = Convert.ToInt32(Console.ReadLine());
                if (dizi[i] > 0)
                {
                    topp = topp + dizi[i];
                    p = p + 1;
                }
                else if (dizi[i] < 0)
                {
                    topn = topn + dizi[i];
                    n = n + 1;
                }
            }
        }
    }
}
```

```
    }  
    }  
    if (p == 0 && n != 0)  
    {  
        ortn = topn / n;  
    }  
    else if (p!=0 && n==0)  
    {  
        ortp = topp / p;  
    }  
    else if (p != 0 && n != 0)  
    {  
        ortp = topp / p;  
        ortn = topn / n;  
    }  
    Console.WriteLine("Pozitif ortalaması = " + ortp);  
    Console.WriteLine("Negatif ortalaması = " + ortn);  
    Console.ReadLine();  
    }  
    }  
}
```

74. 10 elemanlı bir dizinin elemanlarından hem 4'e hemde 5'e bölünen sayıları bulan programın algoritma ve akış diyagramı çiziniz.

Algoritma:

- 1- Başla
- 2- $d[10], i=0$
- 3- Eğer $d[i]\%20=0$ ise Yazdır $d[i]$,
değilse devam et
- 4- $i=i+1$
- 5- Eğer $i<10$ ise 3'e git,
değilse devam et
- 6- Bitir

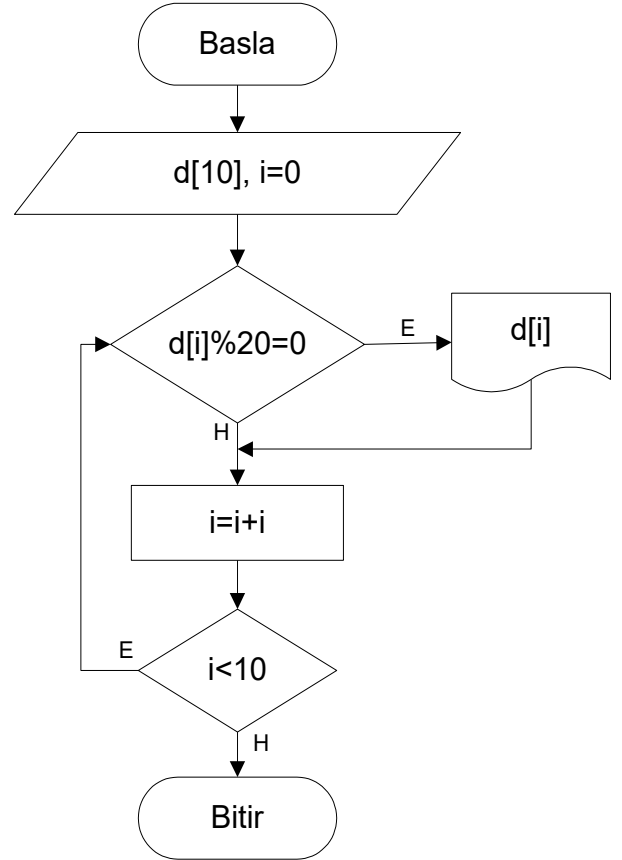
Açıklama:

Bu soruda dizideki her elemanı 20 ye böldüğümüzde kalan 0 ise bu sayı hem 4 'e hem 5'e bölünebilir. Algoritma, buna uyan dizi elamanlarını da ekrana basacaktır.

Algoritma Testi: (5 elemanlık)

S1	S2	S3	S4	S5	Sonuç (Çıktı)
20	30	40	10	60	20 , 40 , 60

Akış Diyagramı:



C Kod:

```
#include <conio.h>
#include <stdio.h>
main()
{
    clrscr();
    int d[10],a[10],i,j=0;
    for(i=0;i<10;i++)
    {
        scanf("%d",&d[i]);
        if(d[i]%20==0)
            a[j]=d[i];
        j++;
    }
    getch();
}
```

C#Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] dizi = new int[11];
            int i;
            Random rnd = new Random();
            for (i = 0; i <= 10; i++)
            {
                dizi[i] = ((rnd.Next(99)) + 1);
            }
            Console.WriteLine("4'e ve 5'e tam
            bölünebilenler");
            for (i = 0; i <= 10; i++)
            {
                if (dizi[i] % 20 == 0)
                {
                    Console.WriteLine(dizi[i]);
                }
            }
            Console.ReadLine();
        }
    }
}
```

75. Girilecek 10 adet sayıyı bir diziye aktararak bu dizideki sayıların ortalamasını bulduran programın algoritmasını ve akış diyagramını çiziniz.

Algoritma Testi : (5 elemanlık)

S1	S2	S3	S4	S5	Ort
-1	3	-4	2	5	5/5=(1)

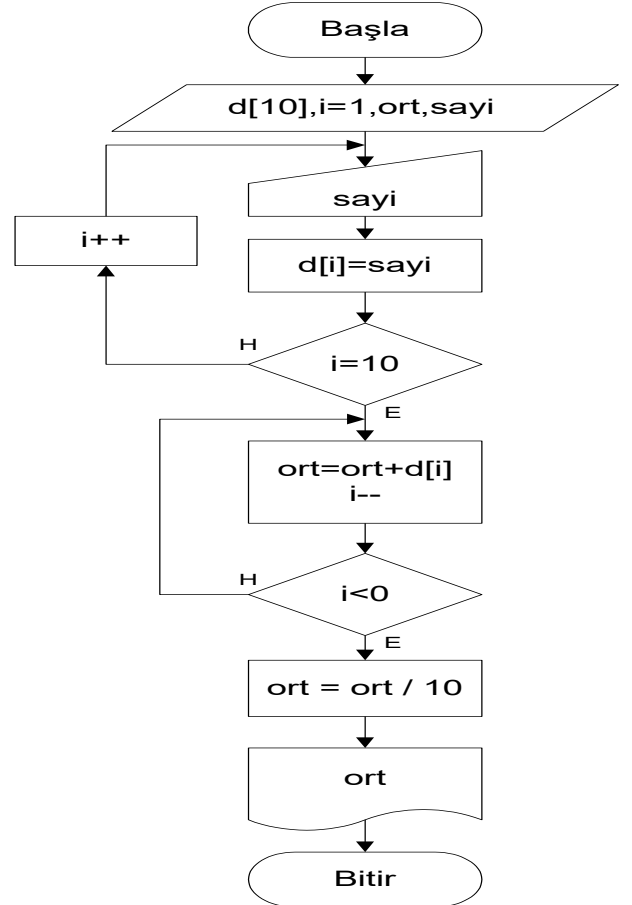
Algoritma:

- 1- Başla
- 2- d[10],i=1,ort,sayi
- 3- sayi gir
- 4- d[i]=sayi
- 5- Eğer i=10 ise devam et,
değilse i++ 3'e git
- 6- ort=ort+d[i],i--
- 7- Eğer i<0 ise devam et,
değilse 6'ya git
- 8- ort = ort / 10
- 9- Yazdır ort
- 10- Bitir

Açıklama:

Bu soruya benzer soruları daha önce de çözmüştük. Diziye elemanlarımızı girip bunun ortalamasını bularak ekrana basacağız. Sorunun basit olarak görülmesine rağmen uzmanlaşmanın basit sorularla başladığını da burada hatırlatmak isteriz.

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int d[10],i;
    float ort=0;
    for(i=0;i<10;i++)
    {
        scanf("%d",&d[i]);
        ort=ort+d[i];
    }
    ort = ort / 10;
    printf("%f",ort);
    getch();
}
```

C#Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] dizi = new int[10];
            int i;
            float ort = 0;
            for (i = 0; i < 10; i++)
            {
                Console.Write(i + ".Sayıyı Giriniz = ");
                dizi[i] = Convert.ToInt32(Console.ReadLine());
                ort = ort + dizi[i];
            }
            ort = ort / 10;
            Console.Write("Ortalama = " + ort);
            Console.ReadLine();
        }
    }
}
```

76. Girilen cümlede, girilen karakterden kaç tane olduğunu bulan programın algoritması ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $d[n], n, i=0, ch, a$
- 3- n, a gir
- 4- $d[i]=a$
- 5- Eğer $i=n+1$ ise devam et,
değilse $i++$ 4'e git
- 6- ch gir
- 7- Eğer $d[i]=ch$ ise sayac++,
değilse devam et
- 8- $i--$
- 9- Eğer $i<0$ ise yazdır sayac,
değilse 7'e git
- 10- Bitir

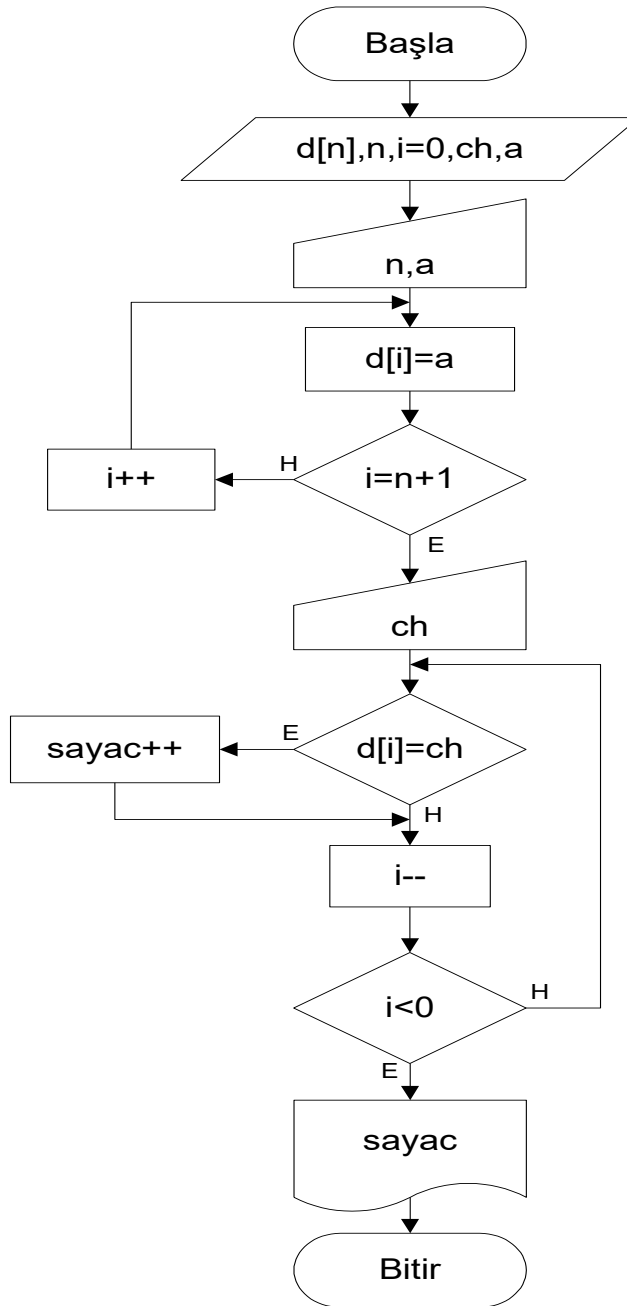
Algoritma Testi:

a1	a2	a3	a4	CH(Ara)	Sayac
E	M	E	L	'E'	2

Açıklama:

Dizi sorularında devamlı sayı dizisi tanımlıyordu. Bu sefer karakter dizisi tanımladık. Girilen cümlelerin içinde sonradan aranacak karakterimizi ch ile kıyaslıyoruz. Bu sorunun benzerini sayı dizisi için yapmıştık. Şimdi de karakter dizisi için tekrarlıyoruz. Sorumuzun mantığı sayı dizisinin mantığı ile aynıdır.

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i;
    int n=50;
    int sayac = 0;
    char ch;
    char d[50];
    gets(d);
    printf("Karakter Giriniz = ");
    scanf("%c",&ch);
    for (i = 0; i < n; i++)
    {
        if (d[i]!='\0')
            break;
        if (d[i] == ch)
        {
            sayac++;
        }
    }
    printf("%d tane var",sayac );
    getch();
}
```

C#Kod :

```
using System;

namespace dmng
{
    class Program
    {
        static void Main(string[] args)
        {
            int i=0;
            int n;
            int sayac = 0;
            char ch;
            Console.Write("N Giriniz = ");
            n = Convert.ToInt32(Console.ReadLine());
            char[] dizi = new char[n + 1];
            for (i = 0; i < n; i++)
            {
                Console.Write(i + ".Karakter Giriniz = ");
                dizi[i] = Convert.ToChar(Console.ReadLine());
            }
            Console.Write("Karakter Giriniz = ");
            ch = Convert.ToChar(Console.ReadLine());
            for (i = 0; i < n; i++)
            {
                if (dizi[i] == ch)
                {
                    sayac++;
                }
            }
            Console.WriteLine(sayac + " tane var");
            Console.ReadLine();
        }
    }
}
```

77. Tersinden ve düzünden okunuşu aynı olan yazılara polindrom denir. Bir yazının polindrom olup olmadığını bulan programın algoritmayı ve akış diyagramını çiziniz.

Örnek: kek , kütük

Algoritma:

- 1- Başla
- 2- d[20],i=0,j=0, kelime
- 3- kelime gir
- 4- Eğer d[i]='/'n' ise devam et, değilse i++ ve 4'e git
- 5- Eğer d[j]=d[i] ise devam et, değilse 8'e git
- 6- i++,j--
- 7- Eğer i=j ise yazdır "polindromdur", değilse 5'e git
- 8- Yazdır "polindrom değildir"
- 9- Bitir

Açıklama:

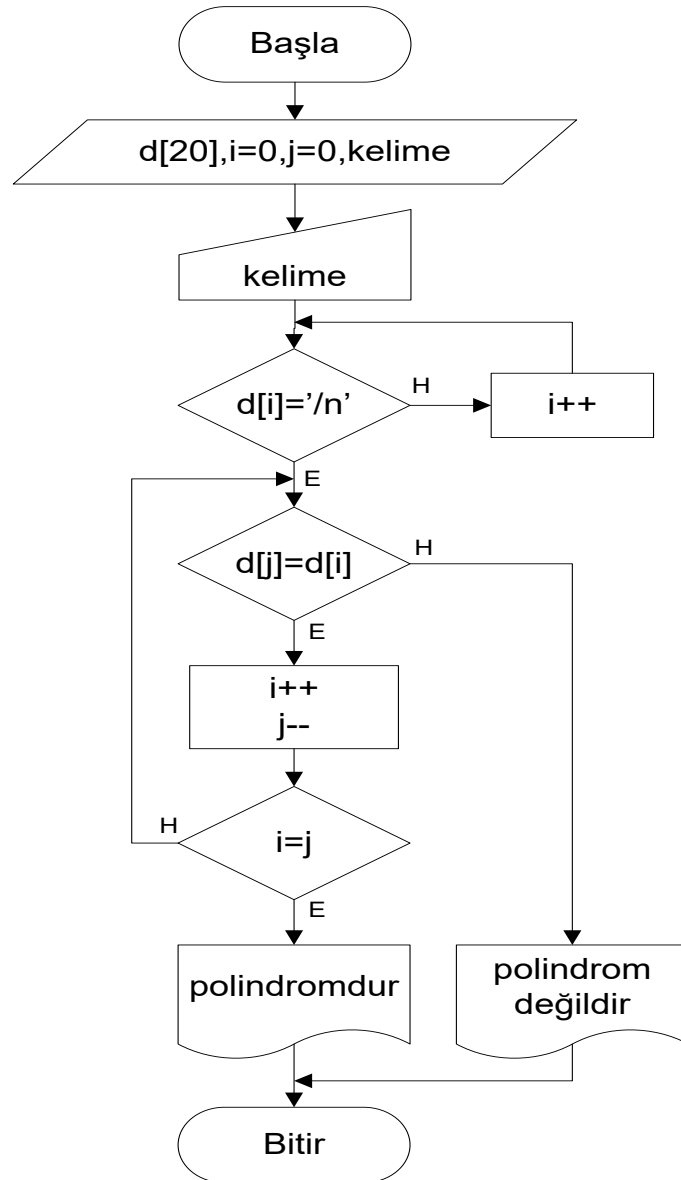
Bu soruda karşımıza ilginç bir algoritma çıkmaktadır. Örneğin 'Kütük' veya 'kek' kelimesi polindrom algoritmasına uyan kelimelerdir. Kelimeyi girdikten sonra girilen karakterler karakter dizisinin elamanları olarak yerlerini alır. Dizin kaç elemanlı olduğunu buluruz. Daha sonra son dizi son işaretine kadar elemanları saydırırız veya her giriş için sayacımızı 1 arttırırız. Sonra dizinin

başından ve sonundan gelerek elemanları karşılaştırırız. Aynılarsa dizinin ortasına geldiğimizde programı sonlandırırız. Bunun için de biri i biri j olmak üzere iki sayaç tutarız. Bu soru mantık olarak bu şekilde işlemektedir.

Algoritma Testi :

A0	A1	A2	A3	A4	i	J
K	ü	t	ü	K	4	0
K		=		k	4	0
	ü	=	ü		3	1
		t			2	2
Polindormdur						

Akış Diyagramı:



C Kod:

```
#include <conio.h>
#include <stdio.h>

main()
{
    int i,j;
    char d[20];
    gets(d);
    for(i=0;i<20;i++)
    {
        if(d[i]=='\0')
        {
            i=i-1;
            break;
        }
    }
    for(j=0;j<=i;j++)
    {
        if(d[j]!=d[i])
        {
            printf("polindrom değildir");
            goto dnz;
        }
        i--;
    }
    printf("polindromdur");
    dnz:
    getch();
}
```

C# Kod:

```
using System;

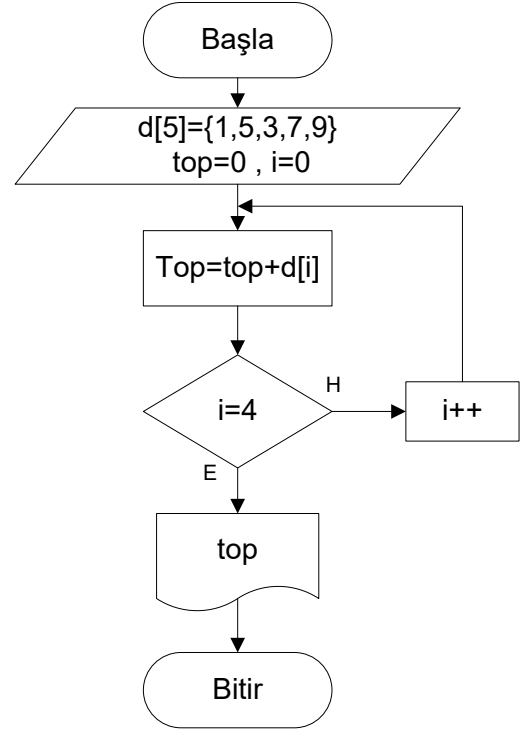
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int n, i,j;
            Console.Write("N Giriniz = ");
            n = Convert.ToInt32(Console.ReadLine());
            char[] dizi = new char[n + 1];
            for (i = 0; i < n; i++)
            {
                Console.Write(i + ".Karakteri Giriniz = ");
                dizi[i] = Convert.ToChar(Console.ReadLine());
            }
            n--;
            for (j = 0; j <= n; j++)
            {
                if (dizi[j] != dizi[n])
                {
                    Console.Write("polindrom değildir");
                    goto dnz;
                }
                n--;
            }
            Console.Write("polindromdur");
            dnz:
            Console.ReadLine();
        }
    }
}
```

78.5 elemanlı dizinin elemanlarının toplamlarını bulan algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $d[5]=\{1,5,3,7,9\}$, $top=0$, $i=0$
- 3- $top= top+d[i]$
- 4- Eğer ($i=4$) ise devam et ,
değilse $i++$ 3'e git
- 5- Yazdır top
- 6- Bitir

Akış Diyagramı:



Açıklama:

Konuya temel teşkil eden sorulardan bir tanesi de bu sorudur. Önce diziye adet sayı girilir. Sonra bunları top dediğimiz, başlangıçta değeri 0 olan bir değişkene toplayarak atarız. top değişkenini son olarak ekrana basarız. 5 eleman kolay gibidir ancak 1000 eleman girildiğinde durum zorlaşmaktadır.

Algoritma Testi:

A0	A1	A2	A3	A4	top
1					1
	2				3
		3			6
			4		10
				5	15

C Kod:

```
#include <stdio.h>
#include <conio.h>
int d[5]={1,5,3,7,9};
int top=0,i;
main()
{
    clrscr();
    for(i=0;i<5;i++)
        top=top+d[i];
    printf("%d",top);
    getch();
}
```

C#Kod :

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] dizi = new int[5] { 1, 5, 3, 7, 9 };
            int top = 0, i;
            for (i = 0; i < 5; i++)
            {
                top = top + dizi[i];
            }
            Console.WriteLine("Toplam = " + top);
            Console.ReadLine();
        }
    }
}
```

79. Bir dizide dışarıdan girilen bir sayının, dizinin elemanlarından kaç tanesinden küçük olduğunu bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $d[n], n, i=0, \text{sayi1}, \text{sayi2}, \text{say}$
- 3- $n, \text{sayi1}$ gir
- 4- $d[i]=\text{sayi1}$
- 5- Eğer $i=n$ ise devam et,
değilse $i++$ 4'e git
- 6- sayi2 gir
- 7- Eğer $\text{sayi2} < d[i]$ ise $\text{say}++$,
değilse devam et
- 8- $i--$
- 9- Eğer $i < 0$ ise devam et,
değilse 7'e git
- 10- Yazdır say
- 11- Bitir

amacımız sizlere bol bol soru çözdürerek
algoritma konusunda yeteneğinizi
arttırmaktır.

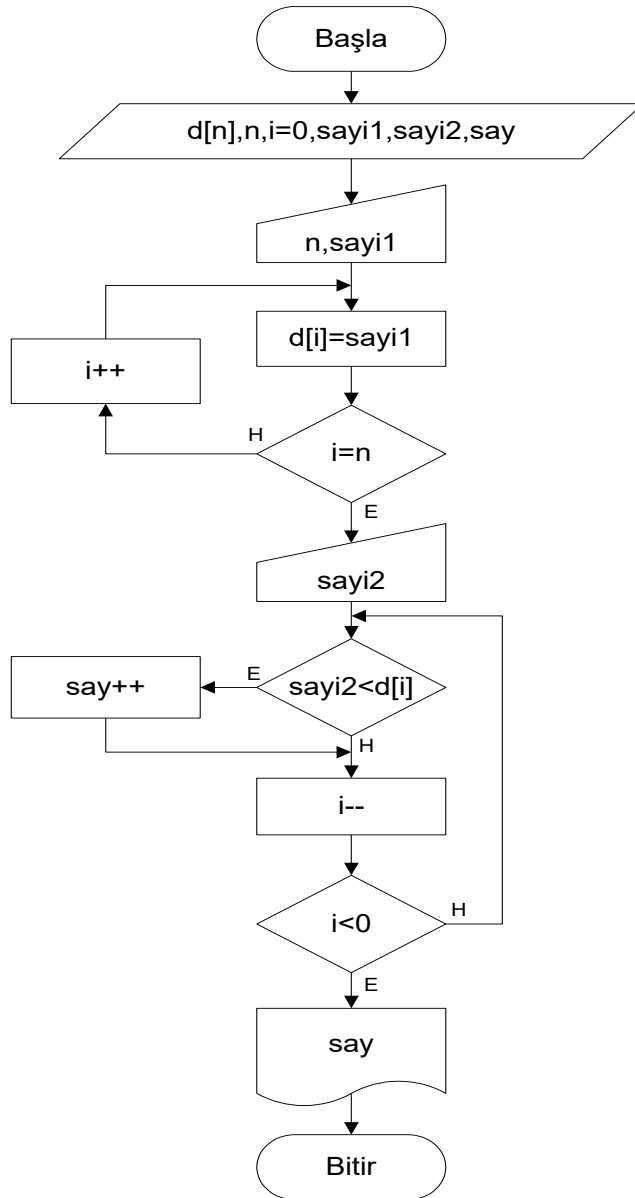
Algoritma Testi:

A0	A1	A2	A3	A4	Sayı2	Sayaç
1<	2<	3<	=4	< 8	4	1

Açıklama:

Bir diziye dışarıdan istenildiği kadar sayı giriyoruz. Sonra dışarıdan bir sayı girip dizinin her elemanı ile karşılaştırıp sayımızı küçük ise sayacımızı 1 arttırıyoruz. Kitabımızda daha önce buna benzer 2 soru yapmıştık. Zaten önceden de belirttiğimiz gibi

Akış Diyagramı:



C Kod:

```
#define N 20
#include <stdio.h>
#include <conio.h>
main()
{
    int i;
    int sayi;
    int say=0;
    int dizi[N];
    for (i=0;i<N;i++)
    {
        scanf("%d",&dizi[i]);
    }
    scanf("%d",&sayi);
    for (i=0;i<N;i++)
    {
        if (sayi<dizi[i])
        {
            say++;
        }
    }
    printf("%d",say);
    getch();
}
```

C#Kod :

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i=0;
            int sayi;
            int say=0;
            int n;
            Console.Write("N giriniz = ");
            n = Convert.ToInt32(Console.ReadLine());
            int[] dizi = new int[n+1];
            for (i = 1; i <= n; i++)
            {
                Console.Write(i+".Sayıyı Giriniz = ");
                dizi[i] = Convert.ToInt32(Console.ReadLine());
            }
            Console.Write("Karşılaştırılacak Sayıyı Giriniz = ");
            sayi = Convert.ToInt32(Console.ReadLine());
            foreach (int karsilastir in dizi)
            {
                if (sayi < karsilastir)
                {
                    say++;
                }
            }
            Console.Write("Dizi elemanlarından " + say + "
            tanesinden küçüktür");
            Console.ReadLine();
        }
    }
}
```

80. Bir dizide dizi elemanlarının sondan başa gelecek şekilde düzenlenmesini sağlayan algoritma ve akış diyagramının çiziniz.

Açıklama:

Bu soruda diziye elemanlarımızı giriyoruz. Sonra bir temp değişken tutuyoruz. Çünkü burada örnek olarak ilk d[0] ile d[n-1] yer değiştirecektir. Bunun için dizi eleman sayısının yarısına kadar sondaki elemanla ile baştaki elemanları değiştiriyoruz. Birebir değişim yapılamayacağı için temp diye bir takas alanı almak zorundayız. Dizinin orta elemanı geldiğinde takas işlemi sona erer ve diziyi tekrar i=0 dan başlamak üzere n-1 e kadar ekrana basarız.

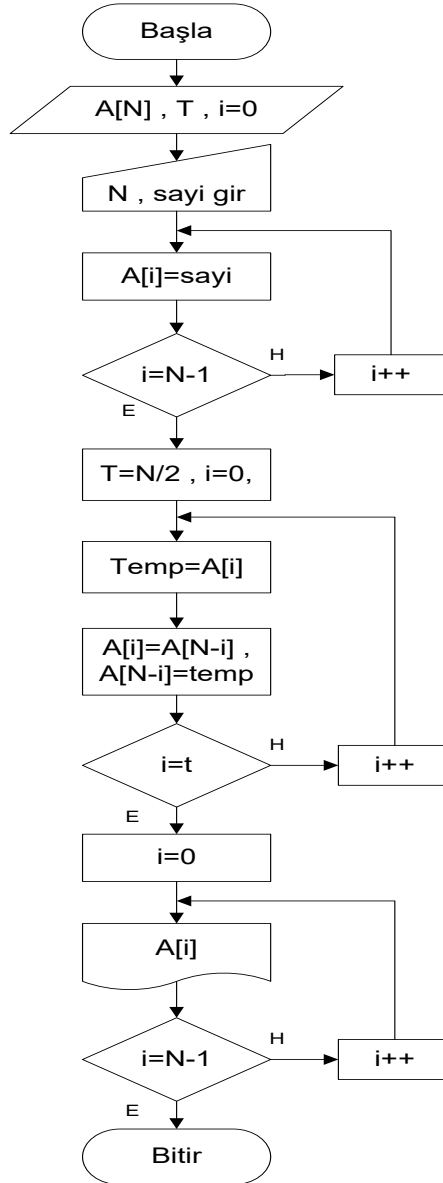
Algoritma:

- 1- Başla
- 2- N , A[N] , T , i=0
- 3- N , sayı gir
- 4- A[i]=sayı
- 5- Eğer (i=N-1) ise devam et ,
değilse i++ 4'e git
- 6- T= N/2 , i=0 ,
- 7- temp=A[i]
- 8- A[i]=A[N-i] , A[N-i]=temp
- 9- Eğer (i=t) ise devam et ,
değilse i++ 7'ye git
- 10- i=0
- 11- Yazdır A[i]
- 12- Eğer (i=N-1) ise devam et ,
değilse ,i++ 11 e git
- 13- Bitir

Algoritma Testi:

A0	A1	A2	A3	A4	i	T
1	2	3	4	5	0	2
5				1	1	2
5	4		2	1	2 =	2
5	4	3	2	1	Bitti.	

Akış Diyagramı:



C Kod:

```
#include<stdio.h>
#include<conio.h>
main()
{
int A[N],T,i,temp;
for(i=0;i<N;i++)
{
scanf("%d",&A[i]);
}
T= N/2;
for(i=0;i<=T;i++)
{
temp=A[i];
A[i]=A[N-1-i];
A[N-1-i]=temp;
}
for(i=0;i<N;i++)
{
printf("%d",A[i]);
}
getch();
}
```

C#Kod :

```
using System;
namespace dmğ
{
class Program
{
static void Main(string[] args)
{
int t, i, temp, n;
Console.WriteLine("N = ");
n = Convert.ToInt32(Console.ReadLine());
int[] dizi = new int[n];
for (i = 0; i < n; i++)
{
Console.WriteLine(i + ".Sayıyı Giriniz = ");
dizi[i] =
Convert.ToInt32(Console.ReadLine());
}
t = (n / 2) - 1;
for (i = 0; i <= t; i++)
{
temp = dizi[i];
dizi[i] = dizi[n - 1 - i];
dizi[n - 1 - i] = temp;
}
for (i = 0; i < n; i++)
{
Console.WriteLine(dizi[i]);
}

Console.ReadLine();
}
}
}
```

81. İkilik bir düzende aynı basamaklı iki sayı verildiğinde bunların toplamını yapan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $n, a[n], b[n], c[n+1], i, elde=0, sayi, top=0$
- 3- sayi gir
- 4- $a[i]=sayi$
- 5- Eğer $i=n-1$ ise devam et, değilse $i++$ 4'e git
- 6- $i=0$
- 7- sayi gir
- 8- $b[i]=sayi$
- 9- Eğer $i=n-1$ ise devam et, değilse $i++$ 8'e git
- 10- $top=a[i]+b[i]+elde$
- 11- $elde=top/2$
- 12- $c[i+1]=top\%2$
- 13- Eğer $i=0$ ise devam et, değilse $i--$ 10'a git
- 14- Yazdır $c[i]$
- 15- Eğer $i=n$ ise devam et, değilse $i++$ 14'a git
- 16- Bitir.

1 olabilir. $1+0$ topladığımızda 1 değerini alırken, $1+1$ topladığımızda ikilik düzende taban 0 olarak kalır. Elde 1 değeri kalır. Bunun için elde diye bir değişken tutuyoruz. Dizinin son elemanlarından toplayarak dizinin başına kadar ilerliyoruz. Bunu algoritma testi ile daha iyi anlayacağız.

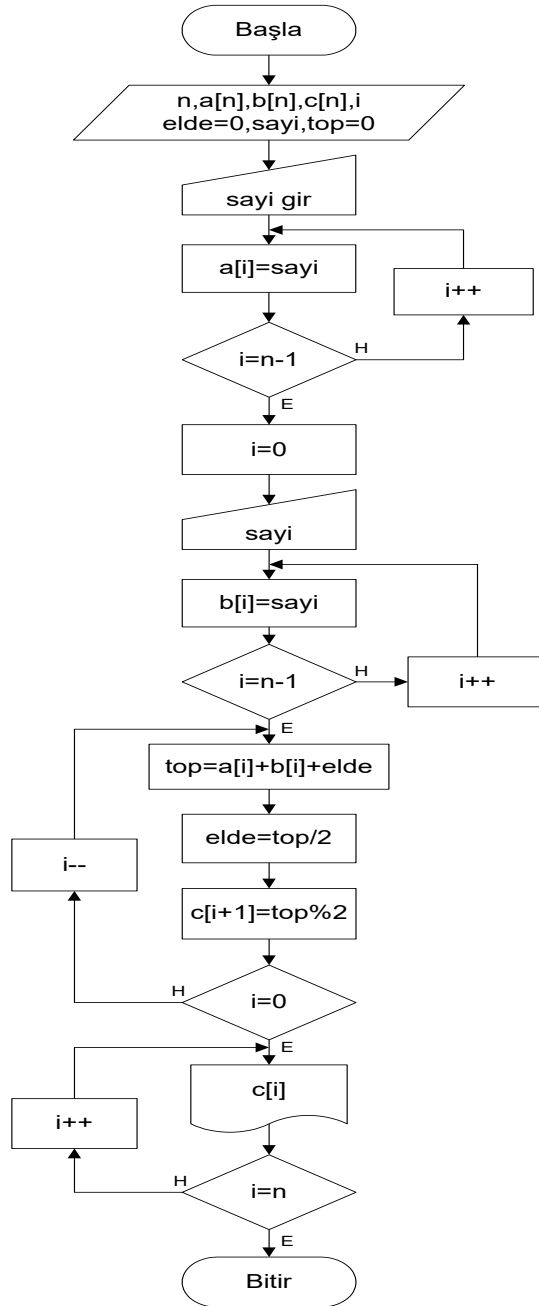
Algoritma Testi:

		0	1	2	3	4	5	6	7
A		1	1	0	1	1	1	1	0
B		1	1	0	0	0	0	0	0
	0	1	2	3	4	5	6	7	8
C	1	0	0	0	1	1	1	1	0

Açıklama:

3 adet dizimiz var a ve b 8 elemanlı, c 9 elemanlıdır. Çünkü 8 bitlik iki sayıyı topladığımızda toplam 9 değerli çıkabilir. Bunun için c dizimizi bir eleman fazla tanıyoruz. İkilik düzende iki sayı topladığımızda değerler 0 ya da

Akış Diyagramı:



C Kod:

```
#define n 8
#include<stdio.h>
#include<conio.h>
main()
{
int a[n],b[n],c[n+2],i,elde=0,top;
for(i=0;i<n;i++)
{
printf("a[%d]=" ,i);
scanf("%d",&a[i]);
}
for(i=0;i<n;i++)
{
printf("b[%d]=" ,i);
scanf("%d",&b[i]);
}
for(i=n-1;i>=0;i--)
{
top=a[i]+b[i]+elde;
elde=top/2;
c[i+1]=top%2;
}
i=0;
c[i]=elde;
for(i=0;i<=n;i++)
{
printf("%d",c[i]);
}
getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
class Program
{
static void Main(string[] args)
{
int i, elde = 0, top;
int n = 8;
int[] a = new int[n];
int[] b = new int[n];
int[] c = new int[n + 2];
for (i = 0; i < n; i++)
{
Console.Write("a[" + i + "] = ");
a[i] = Convert.ToInt32(Console.ReadLine());
}
for (i = 0; i < n; i++)
{
Console.Write("b[" + i + "] = ");
b[i] = Convert.ToInt32(Console.ReadLine());
}
for (i = n - 1; i >= 0; i--)
{
top = a[i] + b[i] + elde;
elde = top / 2;
c[i + 1] = top % 2;
}
i = 0;
c[i] = elde;
for (i = 0; i <= n; i++)
{
Console.Write(c[i]);
}
Console.ReadLine();
}
}
}
```

82.10 elemanlı bir sayı dizisinin en büyük ve en küçük elemanlarını ve yerini bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- enb=0,enk=0,enkyer,enbyer,d[10],i=0
- 3- enb=d[i],enk=d[i],enkyer=1,enbyer=1
- 4- i++
- 5- Eğer d[i]>enb ise
enb=d[i],enbyer=i+1,
değilse devam et
- 6- Eğer d[i]<enk ise enk=d[i],enkyer=i+1,
değilse devam et
- 7- Eğer i=9 ise devam et,
değilse 4'e git
- 8- Yazdır enb,enbyer,enk,enkyer
- 9- Bitir

Açıklama:

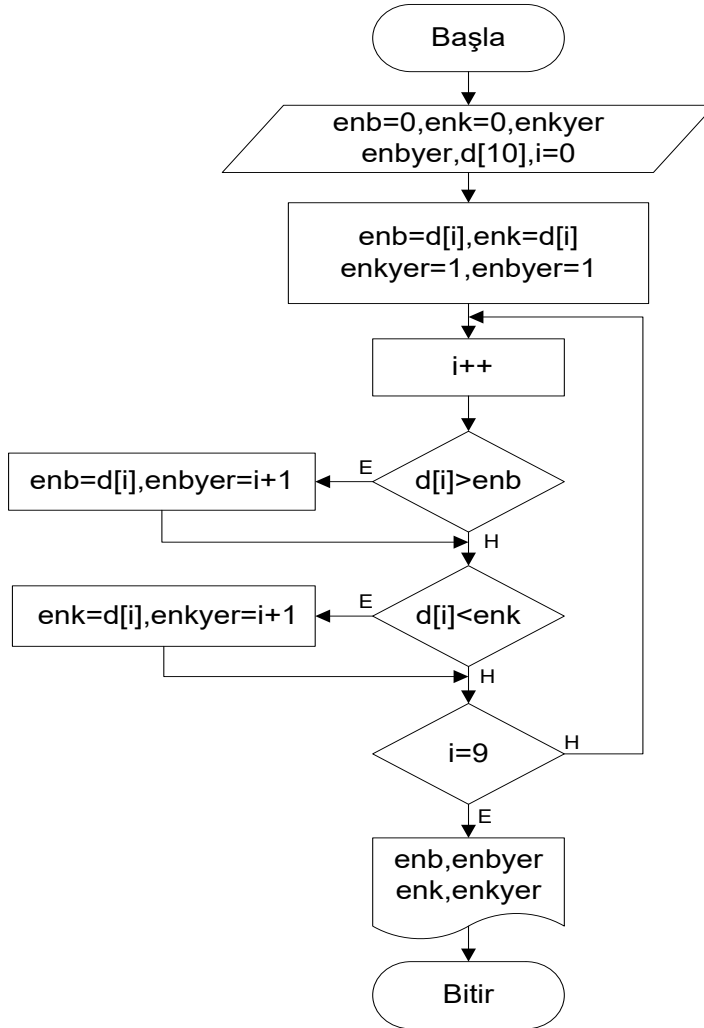
10 elemanlı bir sayı dizisinde en büyük ve en küçük değerlerini ve yerini bulan bu program enb ve enk şeklinde iki değişken almamız gerektiğini unutmamalıyız. Sonra ilk dizinin elemanını enb ve enk değişkenine atıp

diğer dizilerin elemanları ile karşılaştırmalıyız. Duruma göre enb ve enk değişkenlerinin değerleri değiştirilebilir. Daha sonra dizi sonu mu diye bakılıp enb ve enk değeri ile bunların yeri ekrana basılmalıdır.

Algoritma Testi: (5 elemanlık)

enb	enk	a0	a1	a2	a3	a4
5	5	5	2	4	6	1
5	2					
5	2	enkyer		enbyer		
6	2	1→5.eleman		6→4.eleman		
6	1	Program Bitti.				

Akış Diyagramı:



C Kod:

```
#include<stdio.h>
#include<conio.h>
main()
{
int enb,enk,enkyer=1,enbyer=1,d[10],i=0;
scanf("%d",&d[i]);
enb=d[i]; enk=d[i];
for(i=1;i<10;i++)
{
scanf("%d",&d[i]);
if(d[i]>enb)
{
enb=d[i];
enbyer=i+1;
}
if(d[i]<enk)
{
enk=d[i];
enkyer=i+1;
}
}
printf("\n");
printf("%d . eleman en büyük değeri :
%d",enbyer,enb);
printf("\n");
printf("%d . eleman en küçük değeri :
%d",enkyer,enk);
getch();
}
```

C#Kod :

```
using System;
namespace dmj
{
class Program
{
static void Main(string[] args)
{
int enb, enk, enkyer = 1, enbyer = 1, i = 0;
int[] dizi = new int[10];
Console.Write(i + ".Sayıyı Giriniz = ");
dizi[i] = Convert.ToInt32(Console.ReadLine());
enb = dizi[i];
enk = dizi[i];
for (i = 1; i < 10; i++)
{
Console.Write(i + ".Sayıyı Giriniz = ");
dizi[i] = Convert.ToInt32(Console.ReadLine());
if (dizi[i] > enb)
{
enb = dizi[i];
enbyer = i + 1;
}
if (dizi[i] < enk)
{
enk = dizi[i];
enkyer = i + 1;
}
}
Console.WriteLine();
Console.WriteLine(enbyer + ".eleman en büyük
değeri = " + enb);
Console.WriteLine();
Console.WriteLine(enkyer + ".eleman en
küçük değeri = " + enk);
Console.ReadLine();
}}}
```

83. Bir dizinin ikinci büyük elemanını bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $n, a[n], by1, by2, sayi, i=0$
- 3- n gir
- 4- $sayi$ gir
- 5- $a[i]=sayi$
- 6- Eğer $i=n-1$ ise devam et,
değilse $i++$ 4'e git
- 7- Eğer $a[0]>a[1]$ ise
 $by1=a[0], by2=a[1]$,
değilse $by1=a[1], by2=a[0]$
- 8- $i=2$
- 9- Eğer $a[i]>by2$ ise devam et,
değilse 11'e git
- 10- Eğer $a[i]>by1$ ise
 $by2=by1, by1=a[i]$,
değilse $by2=a[i]$
- 11- Eğer $i=n-1$ ise devam et,
değilse $i++$ 9'a git
- 12- Yazdır $by2$
- 13- Bitir

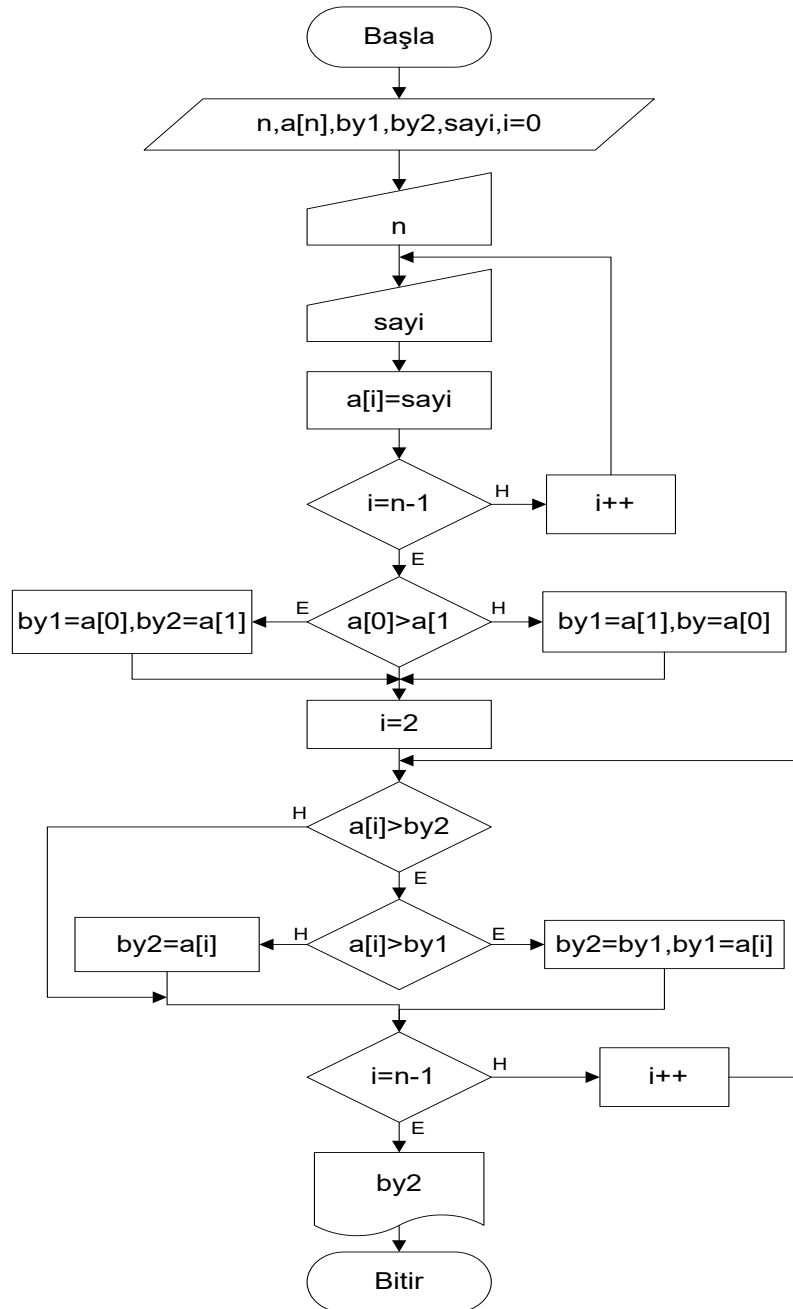
Açıklama:

Biz örnekte 5 elemanlık yaptık. Aslında malloc dediğimiz bir konu ile dinamik bellek yönetimi yaparak 100 elemanlı bir dizinin 5 elemanını kullanıyorsak geri kalan boş alanları hafızadan temizleyebiliriz. Fakat biz burada devamlı belirli bir sınırdaki diziyi tanımlıyoruz. İleriki baskılarda bunları da ekleyeceğiz. $by1$, $by2$ diye 2 adet değişken tanımlıyoruz. İlk eleman ile ikinci elemanı karşılaştırıp büyük olanı $by1$ 'e, küçük olanı $by2$ 'ye atıyoruz. Sonra 3. elemandan itibaren dizi sonuna kadar diğer elemanlar ile karşılaştırıyoruz. İlk önce $by2$ ile karşılaştırıyoruz. Eleman $by2$ 'den büyükse $by1$ 'le karşılaştırıp takas işlemine giriyoruz. Şimdi testini yapalım.

Algoritma Testi: (5 elemanlık)

By1	By2	a0	a1	a2	a3	a4
0	0	5	2	4	6	1
5	2	5	2	4	6	1
5	4	5	2	4	6	1
6	5	5	2	4	6	1
6	5	5	2	4	6	1

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
int n,d[15],by1,by2,i=0;
main()
{
    for(i=0;i<15;i++)
        scanf("%d",&d[i]);
    if(d[0]>d[1])
    {
        by1=d[0]; by2=d[1];
    }
    else
    {
        by1=d[1]; by2=d[0];
    }
    for(i=2;i<15;i++)
    {
        if(d[i]>by2)
        {
            if(d[i]>by1)
            {
                by2=by1; by1=d[i];
            }
            else
                by2=d[i];
        }
    }
    printf("%d",by2);

    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int by1, by2, i = 0;
            int[] dizi = new int[15];
            for (i = 0; i < 15; i++)
            {
                Console.Write(i + ".Sayıyı Giriniz = ");
                dizi[i] = Convert.ToInt32(Console.ReadLine());
            }
            if (dizi[0] > dizi[1])
            {
                by1 = dizi[0];
                by2 = dizi[1];
            }
            else
            {
                by1 = dizi[1];
                by2 = dizi[0];
            }
            for (i = 2; i < 15; i++)
            {
                if (dizi[i] > by2)
                {
                    if (dizi[i] > by1)
                    {
                        by2 = by1;
                        by1 = dizi[i];
                    }
                    else
                    {
                        by2 = dizi[i];
                    }
                }
            }
            Console.Write("İkinci en büyük eleman = " + by2);
            Console.ReadLine(); } }
```

84. Eleman değerleri verilmiş 7 elemanlı bir sayı dizisinde tekrarlanan sayıların ilk yazılanı dışında kalanları kaldırarak başa doğru öteleyen programın algoritma ve akış diyagramını çiziniz.

bitirdik. Bir döngü ile $i=0$ dan $n-1$ e kadar dizinin tekrarlı elemanlardan arınmış halini ekrana bastık.

Algoritma:

- 1- Başla
- 2- $d[7]=\{3,1,4,3,4,7,8\}, n=7, i, j, k$
- 3- $i=0, j=1$
- 4- Eğer $d[i]=d[j]$ ise $k=j$ devam et,
değilse $j++$ 7'e git
- 5- $k=n-2$ olana kadar $d[k]=d[k+1]$
- 6- $n--$
- 7- Eğer $j>(n-1)$ ise devam et,
değilse 4'e git
- 8- Eğer $i=n-1$ ise devam et,
değilse $i++, j=i+1$ 4'e git
- 9- $i=0$ 'dan $n-1$ olana kadar yazdır $d[i]$
- 10- Bitir

Algoritma Testi:

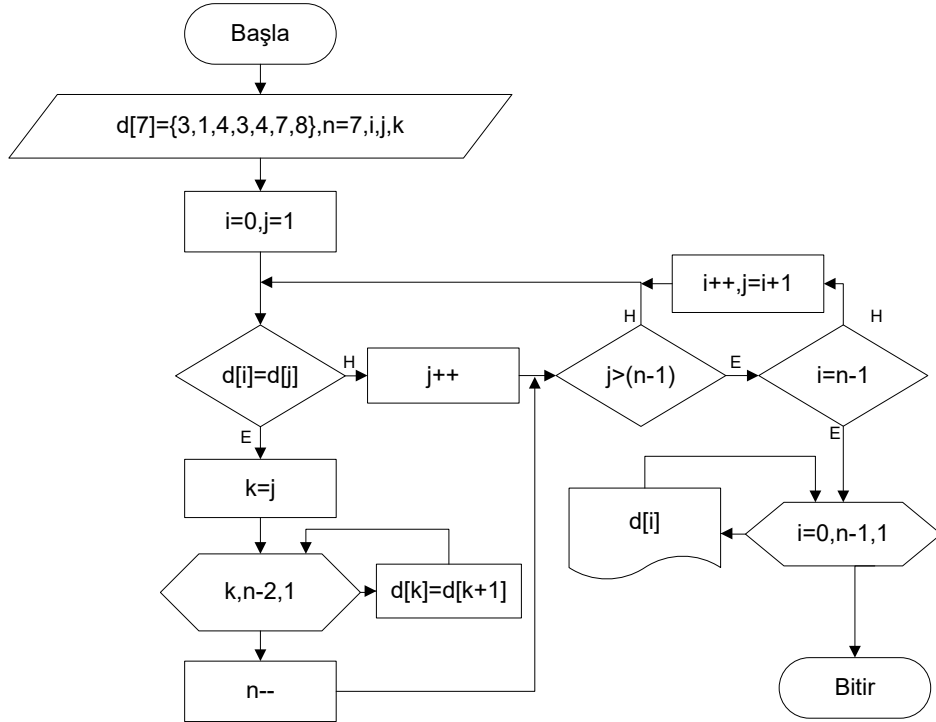
d-i	0	1	2	3	4	5	6
Baş.	3	1	4	3	4	7	8
0	3	1					
1	3	1	4	3	4	7	8
2	3	1	4	3	4	7	8
takas	3	1	4	4	7	8	-
3	3	1	4	4	7	8	
4	3	1	4	4	7	8	

Bu işlemin sadece 1 adımı $d[0]$ bakıldı Program dizi sonuna kadar her eleman için bakıcaktır.

Açıklama:

7 elemanlı bir diziyi baştan değerlerini vererek tanımladık ve dizinin 0. İndisinden itibaren $i=0$ 'dan , $j=1$ den $n-1$ olana kadar sayılarla karşılaştırdık. Aynı sayılar çıkarsa dizimizi öne doğru iteledik. $D[k]=d[k+1]$ ifadesini bunun için kullandık. N eleman sayımız ise ve eşit sayılar çıkarsa diziyi ötelediğimiz için 0. İndise doğru 1 eksilttik. Son elemana geldiğimizde artık karşılaştırma işlemini

Akış Diyagramı:



C Kod:

```
#include <conio.h>
#include <stdio.h>
void main()
{
    int d[7]={3,1,4,3,4,7,8};
    int n=7,i=0,j=1,k;
    while((n-1)!=i)
    {
        while(j<=(n-1))
        {
            if(d[i]==d[j])
            {
                for(k=j;k<n-1;k++)
                {
                    d[k]=d[k+1];
                }
                n--;
            }
            else
                j++;
        }

        i++;
        j=i+1;
    }
    for(i=0;i<n;i++);
    {
        printf("%d",d[i]);
    }
    getch();
}
```

C# Kod:

```
using System;
namespace dmj
{
    class Program
    {
        static void Main(string[] args)
        {
            int n = 7, i = 0, j = 1, k;
            int[] dizi = new int[7] { 3, 1, 4, 3, 4, 7, 8 };
            while ((n - 1) != i)
            {
                while (j <= (n - 1))
                {
                    if (dizi[i] == dizi[j])
                    {
                        for (k = j; k < n - 1; k++)
                        {
                            dizi[k] = dizi[k + 1];
                        }
                        n--;
                    }
                    else
                    {
                        j++;
                    }
                }
                i++;
                j = i + 1;
            }
            for (i = 0; i < n; i++)
            {
                Console.WriteLine(dizi[i]);
            }
            Console.ReadLine();
        }
    }
}
```

85. Klavyeden girilen maksimum 20 karakterli kelimedeki sesli harflerin kelimenin toplam karakter sayısına göre yüzde oranını hesaplayan programın algoritma ve akış diyagramını çiziniz.

Algoritma :

- 1- Başlat
- 2- d[20],ch,i=0,sesli,yuzde,j=0,k=0,
str[8]={‘a’,‘e’,‘ı’,‘i’,‘o’,‘ö’,‘u’,‘ü’}
- 3- ch gir
- 4- d[i]=ch
- 5- Eğer d[i]!='\0'(dizi sonu) ise i++ 3'e git,
değilse devam et
- 6- j=i olana kadar 9.adıma kadar olan
işlemleri yap
- 7- k=7 olana kadar 9.adıma kadar olan
işlemleri yap
- 8- Eğer d[j]==str[k] ise sesli++ 6'ya git,
değilse 7'e git
- 9- yuzde=(sesli*100)/i
- 10- Yazdır “Yüzde” yuzde ” seslidir”
- 11- Bitir

yani sayacımızı 1 arttırıyoruz. Dizi sonuna geldiğimizde yuzde=(sesli*100)/i (el.say.) formülüne

göre sesli harflerin yüzdesini hesaplayıp ekrana basıyoruz.

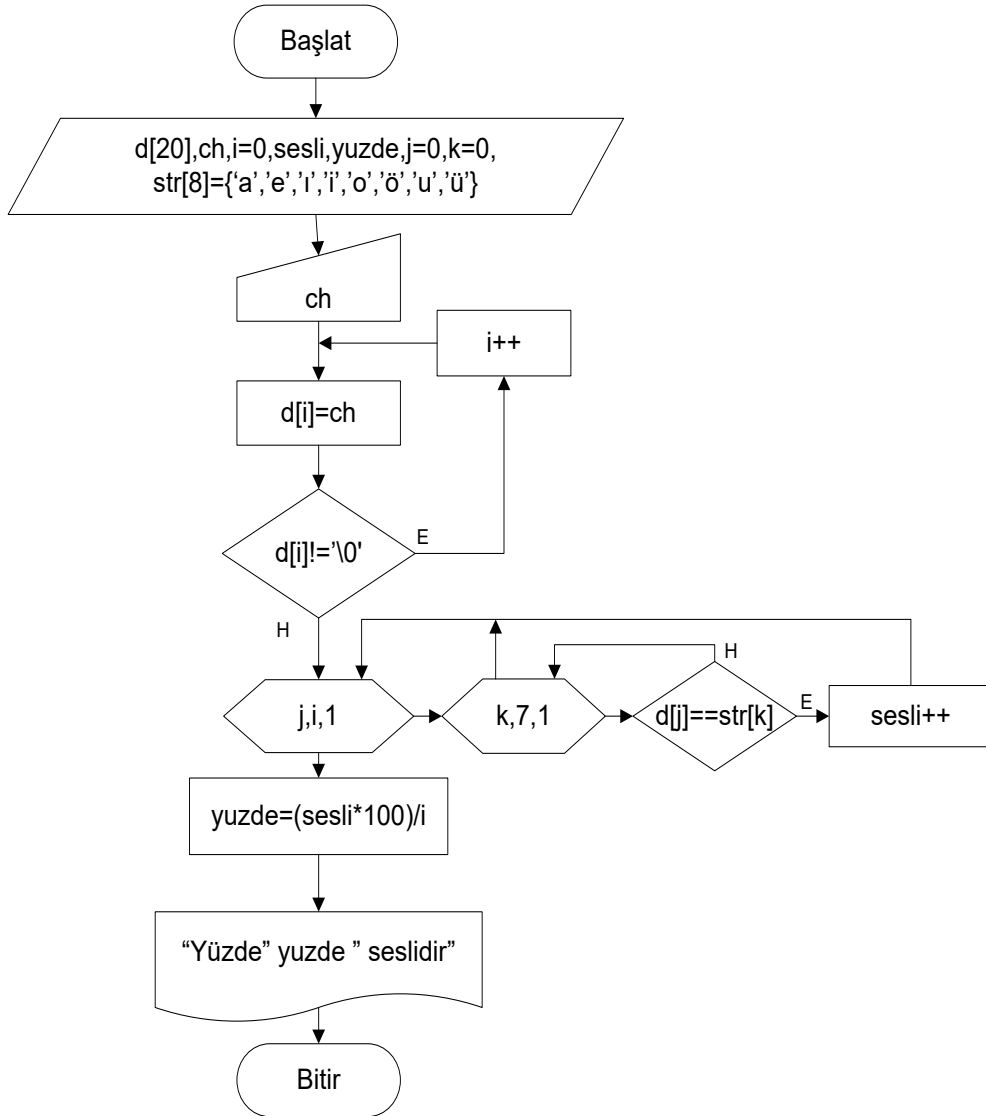
Algoritma Testi:

yuzde	sesli	i	a1
-	0	0	Y
-	1	1	U
-	1	2	S
-	2	3	U
(2*100)/5	2	4	F
Seslilerin oranı → %40			

Açıklama :

Bu soruda maksimum 20 kelimelik bir kelime giriyoruz. Her harfi önceden değerlerini sesli harflerle dolu str değişkeninin her biri ile karşılaştırıyoruz. Karşılaştırma sonucu harf sesli ise sesli değişkenimizi

Akış Diyagramı :



C Kod:

```
#include <stdio.h>
#include <conio.h>
char d[20],str[8]={'a','e','i','o','ö','l','u','ü'};
int i=0,yuzde,sesli,j=0,k=0;
main()
{
    gets(d);
    for(j=0;j<=i;j++)
    {
        for(k=0;k<8;k++)
        {
            if(d[j]==str[k])
            {
                sesli++;
                break;
            }
        }
    }
    yuzde=(sesli*100)/i;
    printf("%d 'si seslidir",yuzde);
    getch();
}
```

C# Kod :

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0, yuzde, sesli = 0, j = 0, k = 0, n;
            char[] dizi = new char[20];
            char[] str = new char[8] { 'a', 'e', 'i', 'o', 'ö', 'l', 'u', 'ü' };
            Console.Write("Harf Sayısı = ");
            n = Convert.ToInt32(Console.ReadLine());
            for (i = 0; i < n; i++)
            {
                Console.Write(i + ".Harf = ");
                dizi[i] = Convert.ToChar(Console.ReadLine());
            }
            for (j = 0; j <= i; j++)
            {
                for (k = 0; k < 8; k++)
                {
                    if (dizi[j] == str[k])
                    {
                        sesli++;
                        break;
                    }
                }
            }
            yuzde = (sesli * 100) / i;
            Console.Write("Yüzde " + yuzde + " seslidir");
            Console.ReadLine();
        }
    }
}
```

86. Tam sayılardan oluşan bir dizi veriliyor, bu dizi elemanlarından kaç tanesinin bir basamaklı, kaç tanesinin iki basamaklı, kaç tanesinin de üç basamaklı olduğunu bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $k1=0, k2=0, k3=0, \text{dizi}[n], i=0, \text{sayi}, n$
- 3- sayi gir
- 4- $\text{dizi}[i]=\text{sayi}$
- 5- Eğer $i=n-1$ ise devam et,
değilse $i++$ 3'e git
- 6- Eğer $a[i] \geq 10$ ise devam et,
değilse $k1=k1+1$ 9'a git
- 7- Eğer $a[i] \geq 100$ ise devam et,
değilse $k2=k2+1$ 9'a git
- 8- Eğer $a[i] < 1000$ ise $k3=k3+1$,
değilse devam et
- 9- Eğer $i=n-1$ ise devam et,
değilse $i--$ 6'a git
- 10- Yazdır $k1, k2, k3$
- 11- Bitir

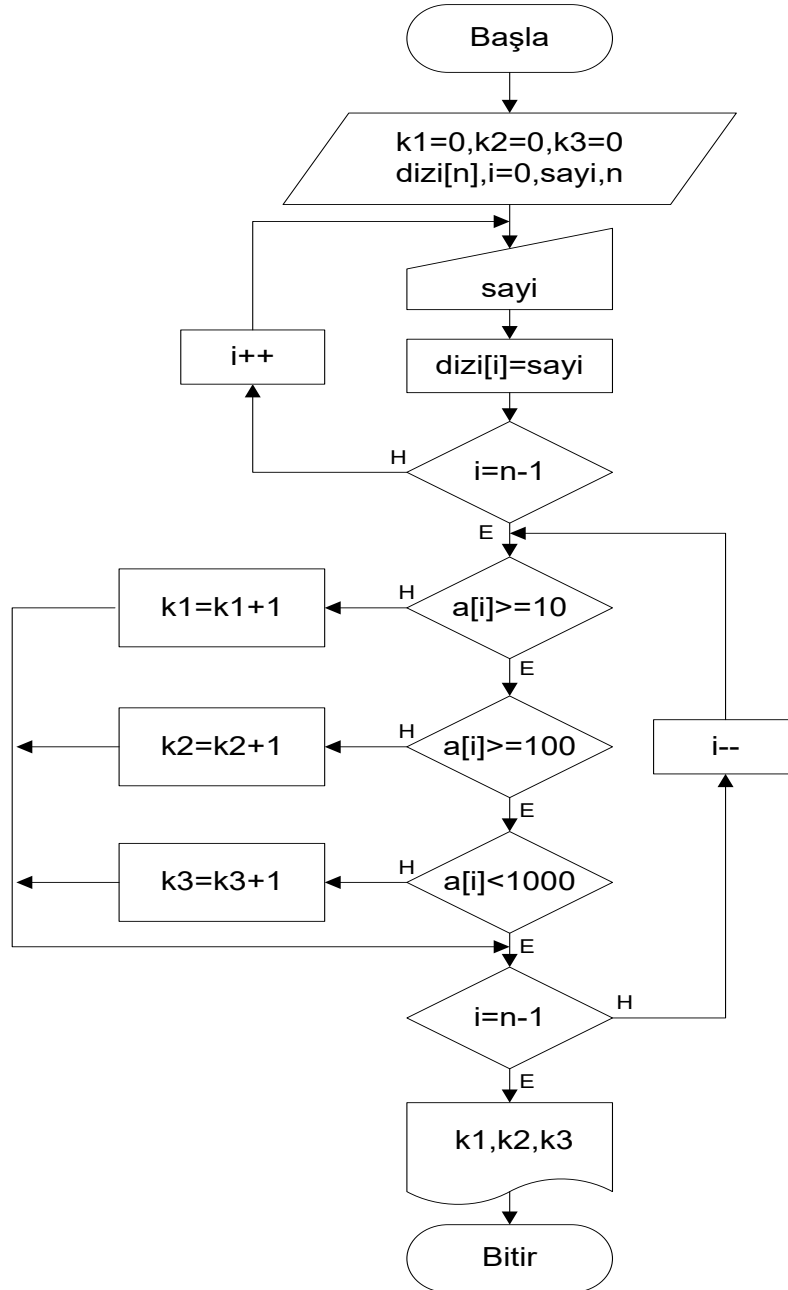
Açıklama:

Bir dizi tanımlarız ve örnek olarak 5 sayı gireriz. Bu sayıları tek tek dizinin başından $i=0$ 'dan elemanları $x < 10$, $x < 100$, $x < 1000$ şartına göre sınırlarız. Hangisine uyuyorsa o sayının basamak sayacını 1 arttırırız. Dizinin sonu gelince bu sayaç değerlerini ekrana basarız.

Algoritma Testi:

K1	K2	K3	i	D[i]	N
0	1	0	0	23	5
0	1	1	1	103	5
0	1	2	2	133	5
0	2	2	3	56	5
1	2	2	4	3	5

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
int d[15],k1=0,k2=0,k3=0,i=0;
main()
{
for(i=0;i<15;i++)
scanf("%d",&d[i]);
for(i=0;i<15;i++)
{
if(d[i]<10)
k1++;
else
{
if(d[i]<100)
k2++;
else
k3++;
}
}
printf("%2d%2d%2d",k1,k2,k3);
getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
class Program
{
static void Main(string[] args)
{
int k1 = 0;
int k2 = 0;
int k3 = 0;
int i = 0;
int sayi;
int n;
Console.Write("N giriniz = ");
n = Convert.ToInt32(Console.ReadLine());
int[] dizi = new int[n];
for (i = 0; i <= n - 1; i++)
{
Console.Write(i + ".Sayıyı Giriniz = ");
dizi[i] = Convert.ToInt32(Console.ReadLine());
}
for (i = 0; i <= n - 1; i++)
{
if (dizi[i] >= 10)
{
if (dizi[i] >= 100)
{
if (dizi[i] < 1000)
```

```
{
    k3 = k3 + 1;
}
}
else
{
    k2 = k2 + 1;
}
}
else
{
    k1 = k1 + 1;
}
}
Console.WriteLine("Bir basamaklı = " + k1);
Console.WriteLine("İki basamaklı = " + k2);
Console.WriteLine("Üç basamaklı = " + k3);
Console.ReadLine();
}
}
}
```

87. 50 elemanlı bir dizinin 1. ve 2. elemanları toplamının yarısı başka bir dizinin ilk elemanı, 3. ve 4. elemanları toplamının yarısı 2. elemanı şeklindedir. Bu işlemi yapan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- a[50], c[50], say=0, i=0, j=0
- 3- Eğer i<50 ise devam et,
değilse 7'e git
- 4- $c[j] = (a[i] + a[i+1]) / 2$
- 5- j=j+1 , i=i+2
- 6- say=say+1 3'e git
- 7- i=0
- 8- Yazdır c[i]
- 9- Eğer i=say ise devam et,
değilse i++ 8'a git
- 10- Bitir

dizinin 1. Elemanı olarak aktarıyoruz. Formülümüz $c[j] = (a[i] + a[i+1]) / 2$ 'dir. Burada j=0 'dan 1 artırım ile j=24 olana kadar dönerken i=0'dan 2 artırım ile i=49'a kadar dönüyor. Sonra c dizisini ekrana basıyoruz.

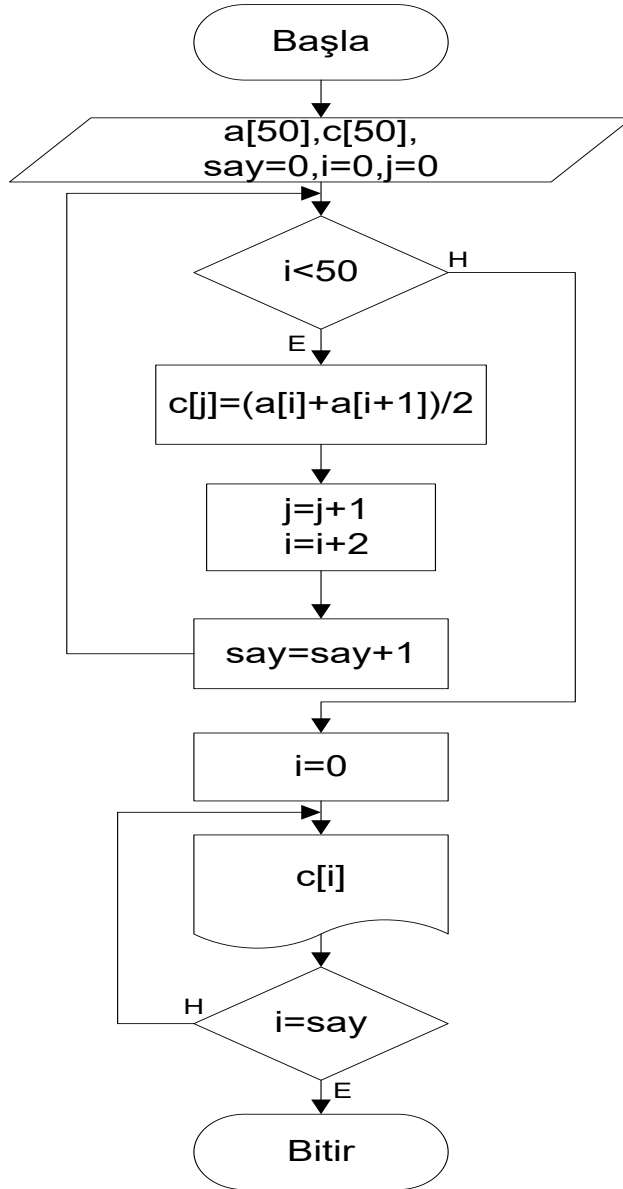
Algoritma Testi: (6 elemanlık)

A	0	1	2	3	4	5
Değ.	2	4	6	6	4	6
İşlem	2+4	6+6	4+6			
C	3	6	5			

Açıklama:

Bu soruda aktarma işlemi ön plandadır. 1 tane 50, 1 tane de 25 elemanlı dizi tanımladık. Bunlar a ve c dizileri , a dizisine indis 0 dan başlamak kaydıyla kullanıcıdan 50 tane değer alıyoruz. Bu işlemler tabii ki döngü içinde oluyor. Bundan sonra a dizisinin 1 . ve 2. Elemanın yarısını c

Akış Diyagramı :



C Kod :

```
#include <stdio.h>
#include <conio.h>
int a[10],c[10],say=0,i=0,j=0;
main()
{
for(i=0;i<10;i++)
scanf("%d",&a[i]);
for(i=0;i<10;i++)
{
c[j]=(a[i]+a[i+1])/2;
j=j+1;
i=i+2;
say++;
}
for(i=0;i<say;i++)
printf("%2d",c[i]);

getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
class Program
{
static void Main(string[] args)
{
int i = 0;
int j = 0;
int n = 0;
int say = 0;
double[] dizi = new double[50];
double[] c = new double[50];
for (n = 0; n < 50; n++)
{
dizi[n] = Convert.ToInt32(Console.ReadLine());
}
while (i < 50)
{
c[j] = (dizi[i] + dizi[i + 1]) / 2;
j = j + 1;
i = i + 2;
say = say + 1;
}
for (i = 0; i < say; i++)
{
Console.WriteLine(c[i]);
}
Console.ReadLine();
}
}
}
```

88. İki boyutlu olarak oluşturulan matrise $matris[i,j]$ dışarıdan değer girilen programın algoritma ve akış diyagramı çiziniz.

Algoritma:

- 1- Başla
- 2- $i=0, j=0, d[2][3], sayi$
- 3- Eğer $i < 2$ ise devam et,
değilse 9'a git
- 4- Eğer $j < 3$ ise devam et,
değilse 8'e git
- 5- sayi gir
- 6- $d[i][j]=sayi$
- 7- $j++$, 4'e git
- 8- $j=0, i++$, 3'e git
- 9- Bitir

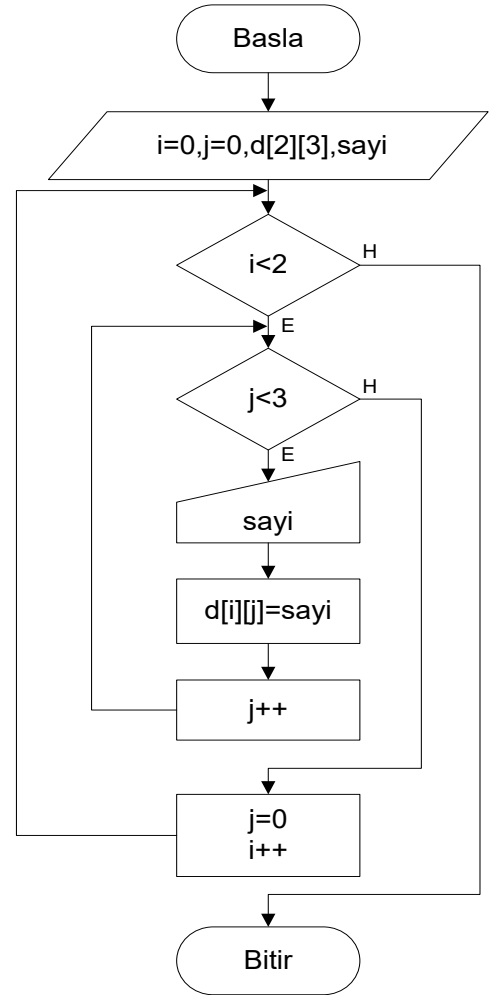
Açıklama:

Çok boyutlu dizilerde matris soruları önemlidir. Burada aslında dikkati çekmesi gereken iç içe for döngülerini kullanmaktır. Yani i döngüsü 1 kere döndüğünde içteki j döngüsü kendi kadar dönecektir. Onun için bu örneği vermeyi uygun bulduk. İç içe döngüleri kullanmak önemlidir. Burda 2 satır 3 sütunluk bir matrise kullanıcının değer girişini göstereceğiz.

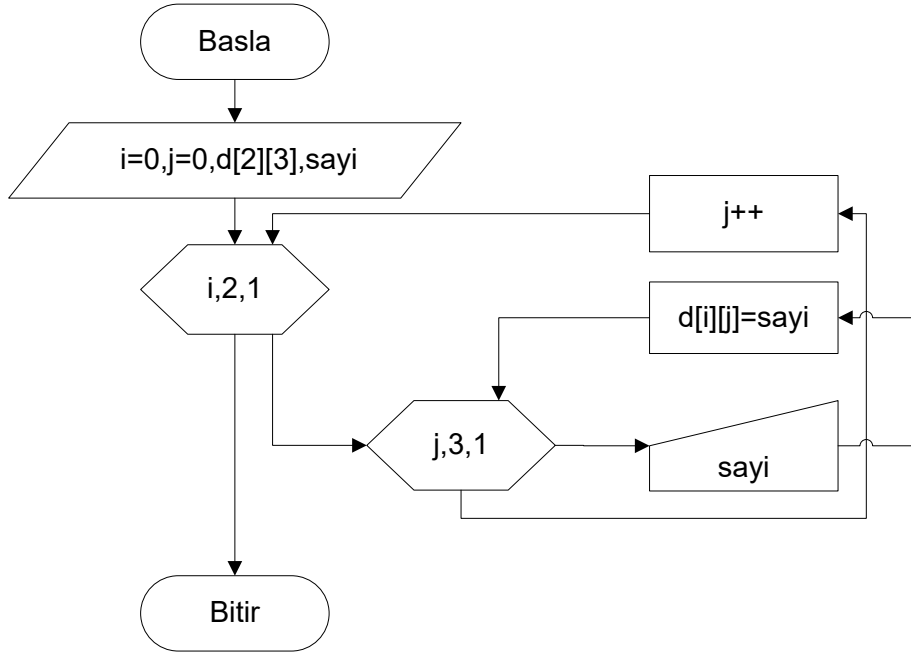
Algoritma Testi:

D	0	1	2
0	34	56	78
1	89	90	23

Akış Diyagramı:



Döngü Kullanarak Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{

    int i;
    int j;
    int dizi[2][3];
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            printf("d[%d][%d]=", i, j);
            scanf("%d", &
                dizi[i][j]);
        }
    }
    getch();
}
```

C# Kod :

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i;
            int j;
            int[,] dizi = new int[2,3];
            for (i = 0; i < 2; i++)
            {
                for (j = 0; j < 3; j++)
                {
                    Console.Write "[" + i + "]" + "[" + j + "]" + " "
Giriniz = ");
                    dizi[i,j] =
Convert.ToInt32(Console.ReadLine());
                }
            }
            Console.ReadLine();
        }
    }
}
```

89. [2x2] tipindeki bir kare matrisin transpozmesini veren algoritma ve akış diyagramını çiziniz.

Algoritma :

- 1- Başla
- 2- dizi [2][2] , i=0 , j=0 , tpoze[2][2]
- 3- Eğer (i<2) ise devam et,
değilse 8'e git
- 4- Eğer (j<2) ise devam et,
değilse 7'ye git
- 5- sayı gir
- 6- dizi[i][j]=sayı , j++, 4'e git
- 7- j=0 , j++, 3'e git
- 8- i=0, j=0
- 9- Eğer (i<2) ise devam et,
değilse 13'e git
- 10- Eğer (j<2) ise devam et ,
değilse 12'ye git
- 11- Tpoze [i][j]=dizi[j][i] , i++ 10'a git
- 12- j=0 , i++ , 9'a git
- 13- Bitir

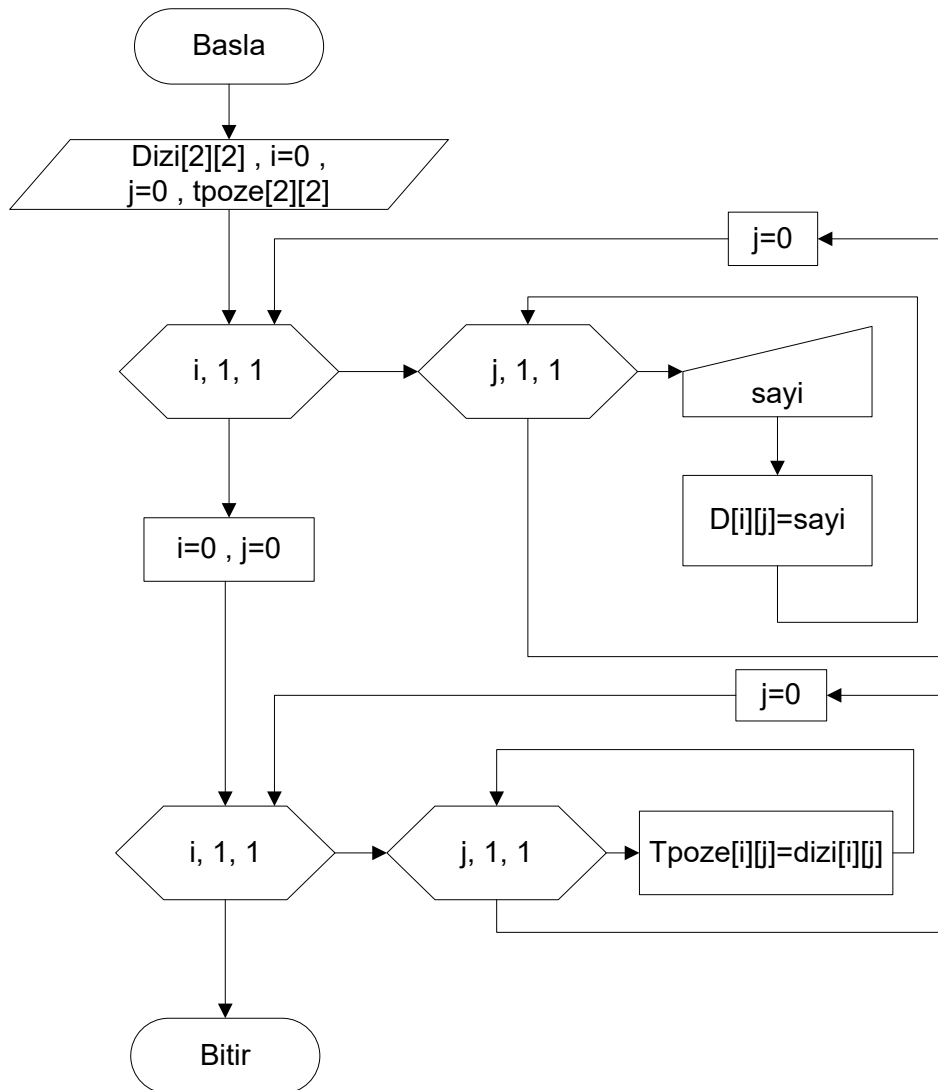
bu sorumuzda 2 X2 tipindeki matrisi, elemanları ile doldurduktan sonra bir döngü ile satır ve sütunların yerlerini değiştireceğiz. (i ve j yine karşımızda 😊)

Algoritma Testi:

a[i][j]	0	1
0	2	3
1	4	5
T[j][i]= T[a]	0	1
0	2	3
1	4	5

Açıklama:

Transpoze sorusu da çok boyutlu dizilere örnek olan matrislerde bir kavramdır. Matrisin satır ve sütunlarını değiştirerek elde edilen matrise o matrisin transpozesi denilir ve [A]^T ile gösterilir. Biz de



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int dizi [2][2],i,j,tpoze[2][2];
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
            scanf("%d",&dizi[i][j]);
    }
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
        {
            tpoze[i][j]=dizi[j][i];
            printf("%d",tpoze[i][j]);
        }
    }
    getch();
}
```

C# Kod:

```
using System;

namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0;
            int j = 0;
            int[,] dizi = new int[2, 2];
            int[,] tpoze = new int[2, 2];
            for (i = 0; i < 2; i++)
            {
                for (j = 0; j < 2; j++)
                {
                    Console.Write "[" + i + "]" + "[" + j + "]" + " Giriniz = ";
                    dizi[i, j] = Convert.ToInt32(Console.ReadLine());
                }
            }
            i = 0;
            j = 0;
            for (i = 0; i < 2; i++)
            {
                for (j = 0; j < 2; j++)
                {
                    tpoze[i, j] = dizi[j, i];
                    Console.WriteLine(tpoze[i, j]);
                }
            }
            Console.ReadLine();
        }
    }
}
```

90. İki boyutlu bir diziyi , tek boyutluya çeviren programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- d[Satır][sutun] , a[N], satir , sutun
 , k=0 , i=0 , j=0
- 3- satir gir , sutun gir
- 4- N= satir*sutun
- 5- Eğer i<satir ise devam et,
 değilse 10'a git
- 6- Eğer j<sutun ise devam et,
 değilse 9' a git
- 7- a[k]=d[i][j] , k++
- 8- j++ 6'ya git
- 9- j=0, i++ , 5'e git
- 10- Bitir

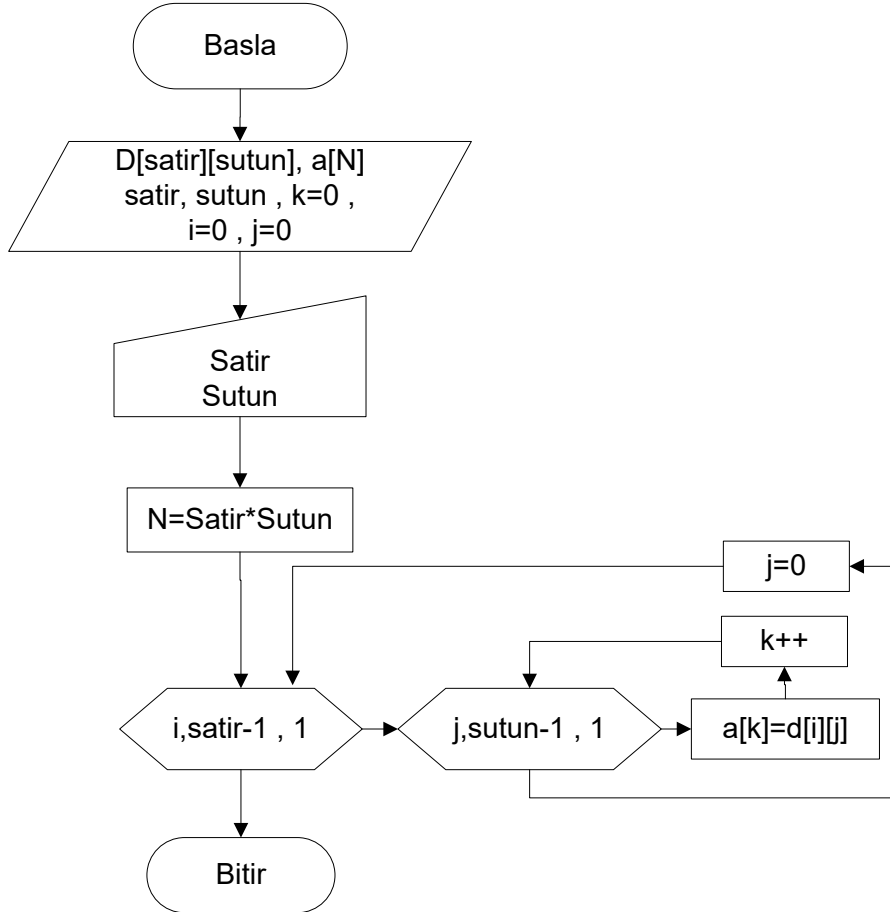
Algoritma Testi: (2X2 tipinde)

A[2][2]	0	1		
0	20	10		
1	15	12		
D[4]	20	10	15	12

Açıklama:

İki boyutlu bir diziyi, tek boyutlu hale getirmek gerekmektedir. Bunun için d[satır][sütun] olduğundan tek boyutlu dizinin eleman sayısı N=satır*sütun olmalıdır. i ve j iki boyutlu dizi için indis değerleri, k ise tek boyutlu dizi için indis değeridir. İç içe döngü kullanarak ve iki boyutlu dizinin [0,0] indisli elemanından başlayarak dizi sonuna kadar dizi elemanlarını d[k] dizisine k=0 dan olmak kaidesi ile atıyoruz. Formülümüz $d[k]=a[i][j]$ 'dir.

Akış Diyagramı:



C Kod:

```
#define satir 2
#define sutun 2
#define n satir*sutun
#include <conio.h>
#include <stdio.h>
void main()
{
    int i;
    int j;
    int k=0;
    int a[n];
    int d[satir][sutun] ;
    for (i = 0; i < satir; i++)
    {
        for (j = 0; j < sutun; j++)
        {
            scanf("%d",&d[i][j]);
        }
    }
    for (i = 0; i < satir; i++)
    {
        for (j = 0; j < sutun; j++)
        {
            a[k] = d[i][j];
            k++;
        }
    }
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i;
            int j;
            int k = 0;
            int satir;
            int sutun;
            int n;
            Console.Write("Satır Gir = ");
            satir = Convert.ToInt32(Console.ReadLine());
            Console.Write("Sutun Gir = ");
            sutun = Convert.ToInt32(Console.ReadLine());
            int[,] dizi = new int[satir,sutun];
            for (i = 0; i < satir; i++)
            {
                for (j = 0; j < sutun; j++)
                {
                    Console.Write("[ " + i + " ]" + " [ " + j + " ]" + " " +
                    Giriniz = ");
                    dizi[i,j] =
                    Convert.ToInt32(Console.ReadLine());
                }
            }
            n = satir * sutun;
            int[] a = new int[n];
            for (i = 0; i < satir; i++)
            {
                for (j = 0; j < sutun; j++)
                {
                    a[k] = dizi[i,j];
                    k++;
                }
            }
            Console.ReadLine();
        }
    }
}
```

91. İki kare [3x3] matrisin toplamını yapan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $a[3][3]=\{0,1,2,3,6,7,8,9,6\}$,
 $b[3][3]=\{2,5,7,8,3,6,7,9,10\}$,
 $c[3][3], i=0, j=0$
- 3- Eğer $i < 3$ ise devam et, değilse 8'e git
- 4- Eğer $j < 3$ ise devam et, değilse 7'e git
- 5- $c[i][j]=a[i][j]+b[i][j]$
- 6- $j++$ 4'e git
- 7- $j=0, i++$ 4'e git
- 8- $j=0, i=0$ 3'e git
- 9- Eğer $i < 3$ ise devam et, değilse 13'e git
- 10- Eğer $j < 3$ ise devam et, değilse 12'e git
- 11- Yazdır $c[i][j]$, $j++$ 10'a git
- 12- $i++$ 9'a git
- 13- Bitir

Açıklama:

Bu soru çok boyutlu dizilere örnek teşkil edecek klasik bir matris toplamıdır. Kare matris, satır ve sütun değerleri aynı olan matristir. Biz burada 3X3 tipinde bir çok boyutlu 3 adet dizi tanımladık. A ve B dizilerinin değerlerini baştan aldık. Bunlar için döngü kullanmak gerekmektedir. Hele hele matris denilirse iç içe döngü kesinlikle olmalıdır. İ, j aklımızdan asla çıkmamalıdır. C de de for döngüsü sanki bu sorular için çıkarılmıştır. Bunun için iki dizinin aynı indisli olanlarını toplayıp yeni diziye yani C dizisine atıyoruz. Ardından C dizisinin tüm elemanlarını ekrana basıyoruz.

Algoritma Testi:

A	0	1	2
0	0	1	2
1	3	6	7
2	8	9	6

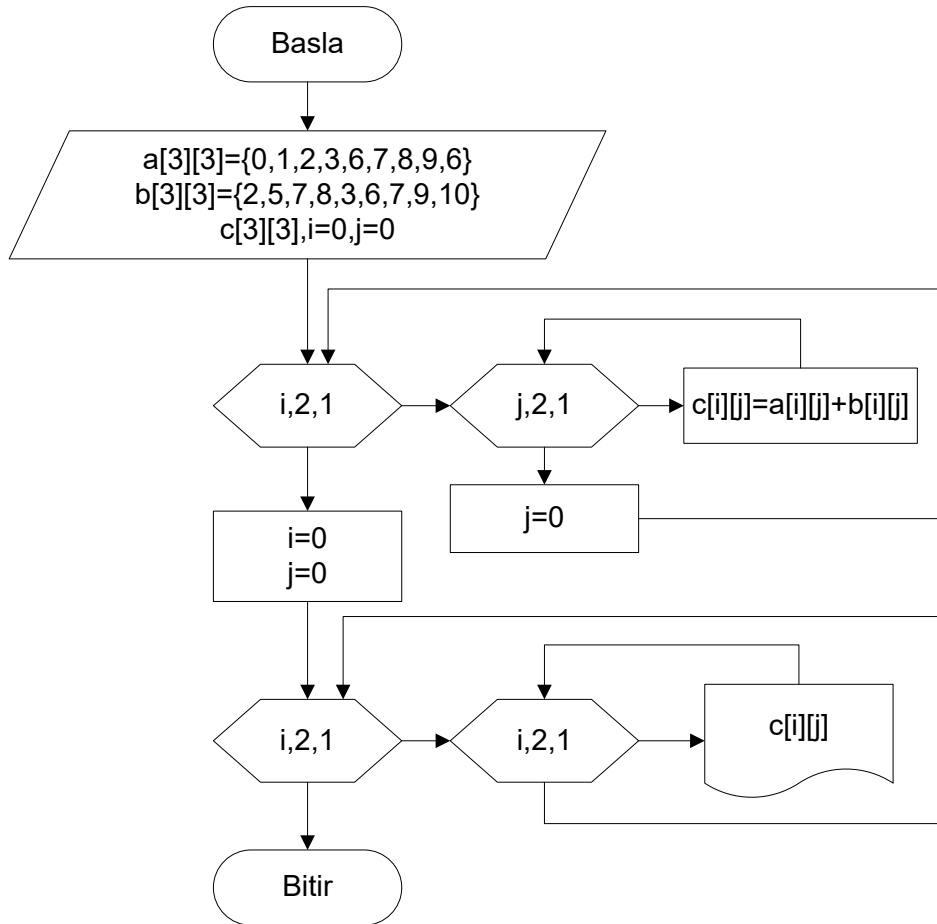
+

B	0	1	2
0	2	5	7
1	8	3	6
2	7	9	10

=

C	0	1	2
0	2	6	9
1	11	9	13
2	15	18	16

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i;
    int j;
    int a[3][3] = { 0, 1, 2, 3, 6, 7, 8, 9, 6 };
    int b[3][3] = { 2, 5, 7, 8, 3, 6, 7, 9, 10 };
    int c[3][3];
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
        {
            c[i][j] = a[i][j] + b[i][j];
        }
    }
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < 3; j++)
        {
            printf("%d", c[i][j]);
        }
    }
    getch();
    return 0;
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i;
            int j;
            int[,] a = { { 0, 1, 2 }, { 3, 6, 7 }, { 8, 9, 6 } };
            int[,] b = { { 2, 5, 7 }, { 8, 3, 6 }, { 7, 9, 10 } };
            int[,] c = new int[3, 3];
            for (i = 0; i < 3; i++)
            {
                for (j = 0; j < 3; j++)
                {
                    c[i, j] = a[i, j] + b[i, j];
                }
            }
            for (i = 0; i < 3; i++)
            {
                for (j = 0; j < 3; j++)
                {
                    Console.WriteLine(c[i, j]);
                }
            }
            Console.ReadLine();
        }
    }
}
```

92.5 adet öğrencinin numarası, vize, final notunu tutup matriste diğer sütuna atan ve listeleyen programın algoritma ve akış diyagramını çiziniz.

Açıklama:

Bu soru da iki boyutlu dizi sorusudur.

Burada 5 satır 4 sütunluk bir dizi tanımlanmıştır. Öğrencilerin numaraları, vize ve final notları dışarıdan diziye girilmiştir. Geçme notları da vize vefinala göre hesaplanıp diziye dahil edilmiştir. Son olarak da bunlar listelenmiştir.

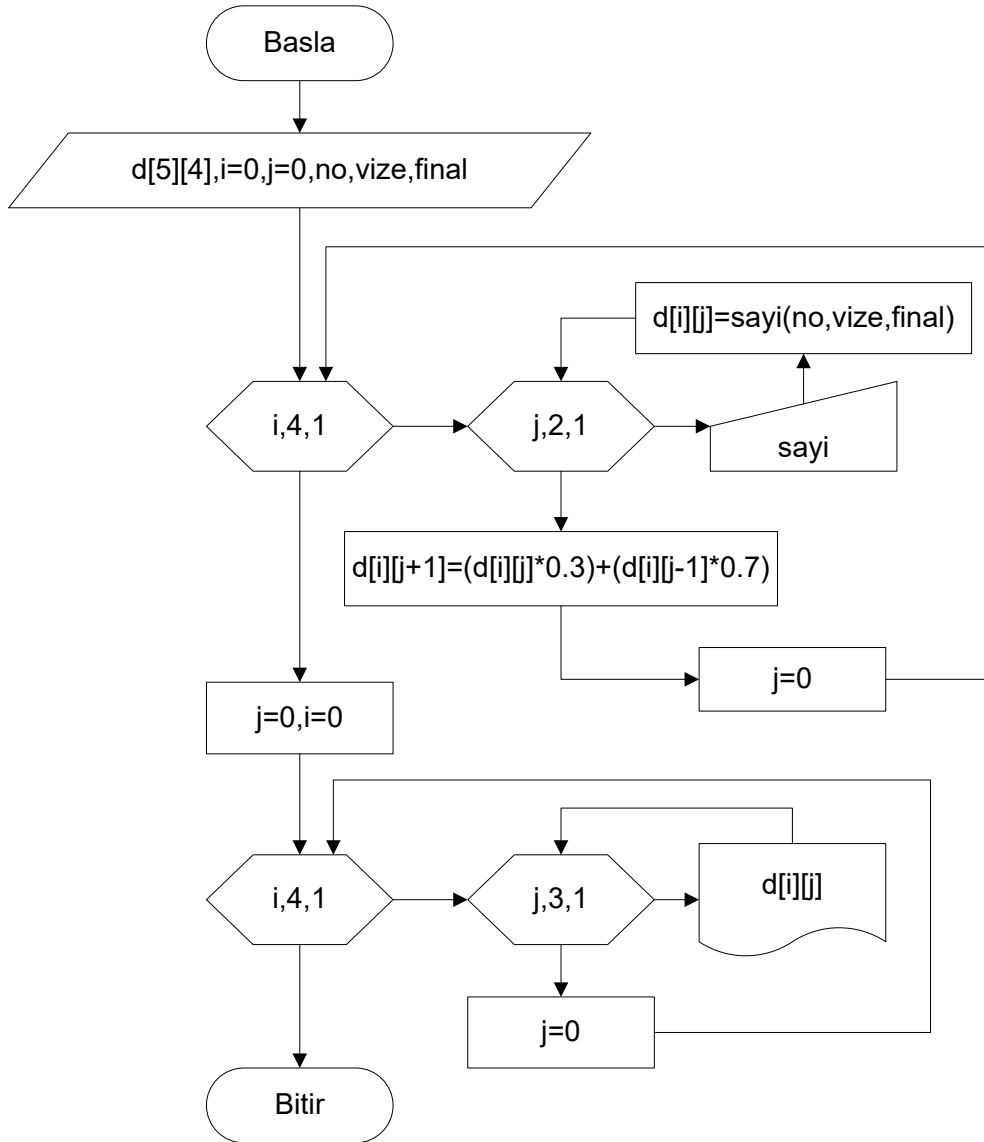
Algoritma Testi:

D	0	1	2	3
0	1468	70	90	84
1	7532	78	90	86
2	979	78	78	78
3	4254	45	100	83
4	7778	56	78	71

Algoritma:

- 1- Başla
- 2- $d[5][4], i=0, j=0, no, vize, final$
- 3- Eğer $i < 5$ ise devam et, değilse 9'a git
- 4- Eğer $j < 3$ ise devam et, değilse 7'e git
- 5- $d[i][j] = sayi(no, vize, final)$
- 6- $j++$, 4'e git
- 7- $d[i][j+1] = (d[i][j] * 0.3) + (d[i][j-1] * 0.7)$
- 8- $j=0, i++$ 3'e git
- 9- $j=0, i=0$
- 10- Eğer $i < 5$ ise devam et, değilse 14'e git
- 11- Eğer $j < 4$ ise devam et, değilse 13'e git
- 12- Yazdır $d[i][j]$
- 13- $j++$ 10'a git
- 14- $j=0, i++$ 9'a git
- 15- Bitir

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i;
    int j;
    int dizi [5][4];
    for (i = 0; i<5; i++)
    {
        printf("No = ");
        scanf("%d",&dizi[i][0]);
        printf("Vize = ");
        scanf("%d",&dizi[i][1]);
        printf("Final = ");
        scanf("%d",&dizi[i][2]);
        dizi[i][3]=((dizi[i][1]*0.3) + (dizi[i][2] * 0.7));
    }
    for (i = 0; i < 5; i++)
    {
        for (j = 0; j<4;j++)
        {
            printf("%d",dizi[i][j]);
        }
        printf("\n");
    }
    getch();
    return 0;
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i;
            int j;
            int[,] dizi = new int[5, 4];
            for (i = 0; i < 5; i++)
            {
                Console.Write("No = ");
                dizi[i, 0] =
                Convert.ToInt32(Console.ReadLine());
                Console.Write("Vize = ");
                dizi[i, 1] =
                Convert.ToInt32(Console.ReadLine());
                Console.Write("Final = ");
                dizi[i, 2] =
                Convert.ToInt32(Console.ReadLine());
                dizi[i, 3] = Convert.ToInt32((dizi[i, 1] * 0.3) +
                (dizi[i, 2] * 0.7));
            }
            for (i = 0; i < 5; i++)
            {
                for (j = 0; j < 4; j++)
                {
                    Console.WriteLine(dizi[i, j]);
                }
            }
            Console.ReadLine();
        }
    }
}
```

Bölüm 6

Arama ve Sıralama Algoritmaları Soru ve Çözümleri

93. Sıralı arama (sequential search) algoritması ile girilen bir sayıyı dizideki yerini bulan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $d[5] = \{11, 43, 4, 7, 32\}, N=5, i=0, \text{ara}$
- 3- ara gir
- 4- Eğer $i \leq n-1$ ise devam et,
değilse yaz "bulunamadı" 7'e git
- 5- Eğer $d[i] = \text{ara}$ ise yaz "yer:"i 7'e
git, değilse devam et
- 6- $i++$, 4'e git
- 7- Bitir

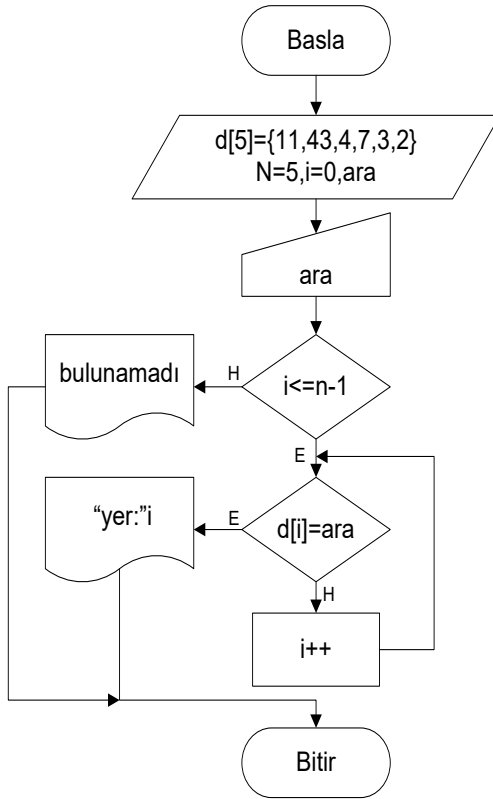
Algoritma Testi:

Aranan	D[5]	0	1	2	3	4
6	i=0	3	7	9	6	90
6	i=1	3	7	9	6	90
6	i=2	3	7	9	6	90
6	i=3	3	7		6(Bingo)	90
Aranan =6 , Yeri =d[3] , eleman sırası=(i+1)= 4						

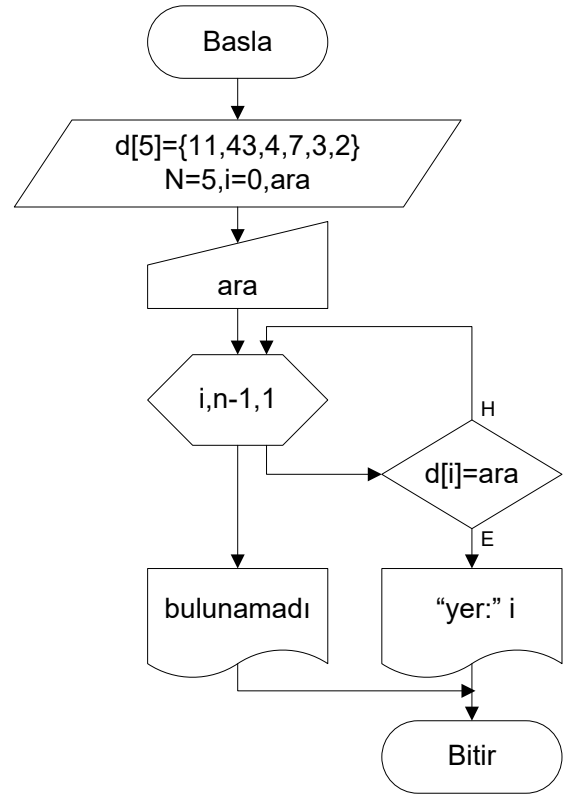
Açıklama:

Dizinin başından başlayarak istenilen elemana kadar sırayla arama yapan algoritmadır. Çok elemanlı dizilerde sonlardaki elemanları aramak zaman alır. Tıpkı bir kütüphanede 200.000 kitap olması ve sizin aradığınız kitabın da sonuncu olması gibi bu işlem de oldukça çok zaman alacaktır.

Akış Diyagramı:



Döngü Kullanarak Akış Diyagramı:



C Kod:

```
#define N 5
#include <stdio.h>
#include <conio.h>
int d[N]={11,43,4,7,32},i,ara;
main()
{
    clrscr();
    scanf("%d",&ara);
    for(i=0;i<=N-1;i++)
    {
        if(d[i]==ara)
        {
            printf("%d . eleman  ",i+1);
            break;
        }
    }
    printf("Bu eleman dizide yok");
    getch();
}
```

C#Kod :

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i=0;
            int n = 5;
            int ara;
            int[] dizi = { 11, 43, 4, 7, 32 };
            Console.Write("Ara = ");
            ara = Convert.ToInt32(Console.ReadLine());
            do
            {
                if (dizi[i] == ara)
                {
                    Console.Write("Yer = " + i);
                    goto cikis;
                }
                i++;
            } while (i < n );
            Console.Write("Bulunamadı");
            cikis:
            Console.ReadLine();
        }
    }
}
```

94. N elemanlı bir dizide ikili (binary) arama algoritması ile girilen bir sayıyı arayan algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $N=5, alt=0, ust=N, ara, i, d[N]=\{1,3,5,8,10\}$
- 3- ara gir(dizi elemanlarından biri girilmeli)
- 4- $i=(alt+ust)/2$
- 5- Eğer $d[i]=ara$ ise yazdır $i+1$ ".eleman" 7'e git, değilse devam et
- 6- Eğer $d[i]>ara$ ise $ust=i$ 4'e git, değilse $alt=i$ 4'e git
- 7- Bitir

Açıklama:

Bu algoritma için önemli olan, dizi sıralı değilse sıralama algoritması ile sıralayıp sonra arama yapmasıdır. İkili arama, sıralı dizilerde yapılır. Mantık olarak dizinin ortasındaki (Örnek 1 ile 63 arsında 32 ortadaki sayıdır) sayıya bakılır. Aradığımız sayı, bu sayıdan büyükse ortadaki ve sondaki arasına bakılıp yeni diziymiş gibi düşünülerek sayımız bulana kadar yeniden bu dizinin ortasındaki sayıya bakılır.

Örnekle açıklayalım:

0-1-2-3-4-5-6-7-8-9 sayılarımız 10 elemanlı sıralanmış bir dizi.

Aranan sayı 8 olsun.

Ortadaki sayı $(10 / 2 = 5)$.

$5 < 8$ olduğu için yeni aralığımız

5-6-7-8-9 Ortadaki sayı 7

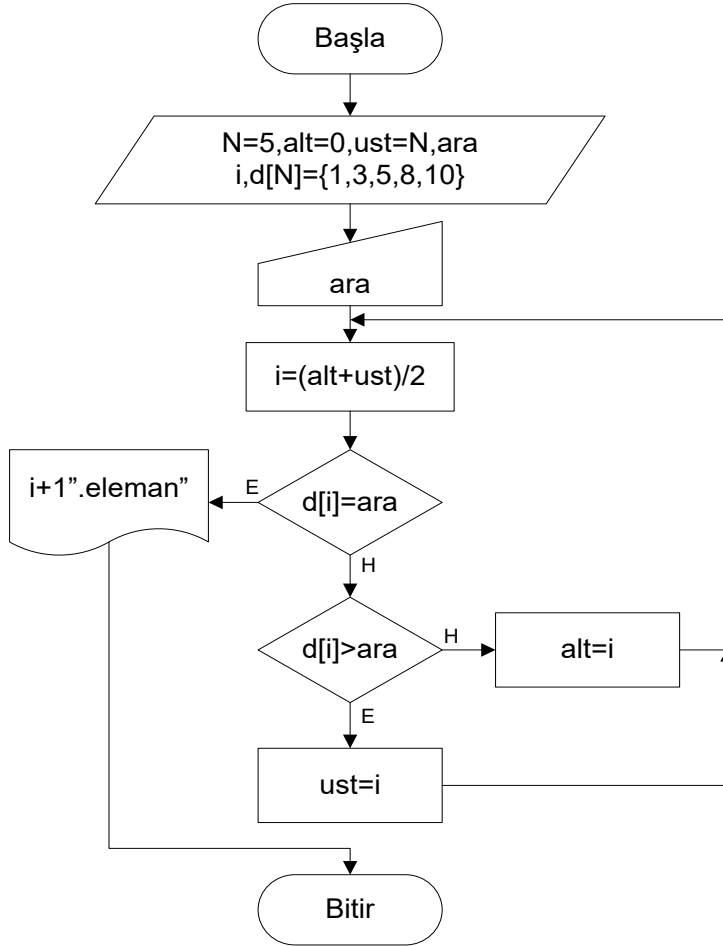
$7 < 8$ olduğu için yeri aralığımız

Yine aynı işlemi yapıyoruz.

8-9 Ortadaki veri 8 (bingo)

Verimizi 3 seferde bulduk. Ama sıralı arama ile bunu aramış olsaydık anca 8 adımda bulabilecektik.

Akış Diyagramı:



Algoritma Testi:

İ	Ara	alt	ust	A[0]	A[1]	A[2]	A[3]	A[4]
2	8	0	4	1	3	5	8	10
3	8	2	4	1	3	5	8(bingo)	10
(İ+1). Elaman = (3+1).eleman=8 , işlem 2								

C Kod:

```
#include <stdio.h>

#include <conio.h>

int alt=0,ust=5,ara,i=0,d[5]={1,3,5,8,10};

int main()
{
    clrscr();
    scanf("%d",&ara);
    if( ara<d[0] || ara>d[4])
    {
        printf("bu sayi dizide yok");
        goto dnz;
    }
    while((ust-alt)!=1)
    {
        i=(alt+ust)/2;
        if (d[i]==ara)
        {
            printf("%d .eleman",i+1);
            break;
        }
        if (d[i]>ara)
        {
            ust=i;
        }
        else
        {
            alt=i;
        }
    }
    dnz:
    getch(); }
```

C#Kod :

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int i = 0;
            int n = 5;
            int alt = 0;
            int ust = n;
            int ara;
            int[] dizi = { 1, 3, 5, 8, 10 };
            Console.Write("Ara = ");
            ara = Convert.ToInt32(Console.ReadLine());
            dmğ:
            i = (alt + ust) / 2;
            if (dizi[i] == ara)
            {
                Console.WriteLine((i + 1) + ".eleman");
            }
            else if (dizi[i] > ara)
            {
                ust = i;
                goto dmğ;
            }
            else
            {
                alt = i;
                goto dmğ;
            }
            Console.ReadLine();
        }
    }
}
```

95. Bubble (kabarcık) sıralama algoritması ile bir dizinin sıralanması programının algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- d[10],i=0,j=0, sayi,temp
- 3- i<10 olana kadar 6. adıma kadar yap
- 4- d[i]=sayi gir
- 5- i++
- 6- i=0
- 7- i<9 olana kadar 13. adıma kadar yap
- 8- j<9 olana kadar 13. adıma kadar yap
- 9- Eğer d[j]>d[j+1] ise devam et, değilse j++ 8'e git
- 10- temp=d[j]
- 11- d[j]=d[j+1]
- 12- d[j+1]=temp, j++
- 13- i=0
- 14- i<10 olana kadar 15.adımı yap
- 15- Yazdır d[i]
- 16- Bitir

Açıklama:

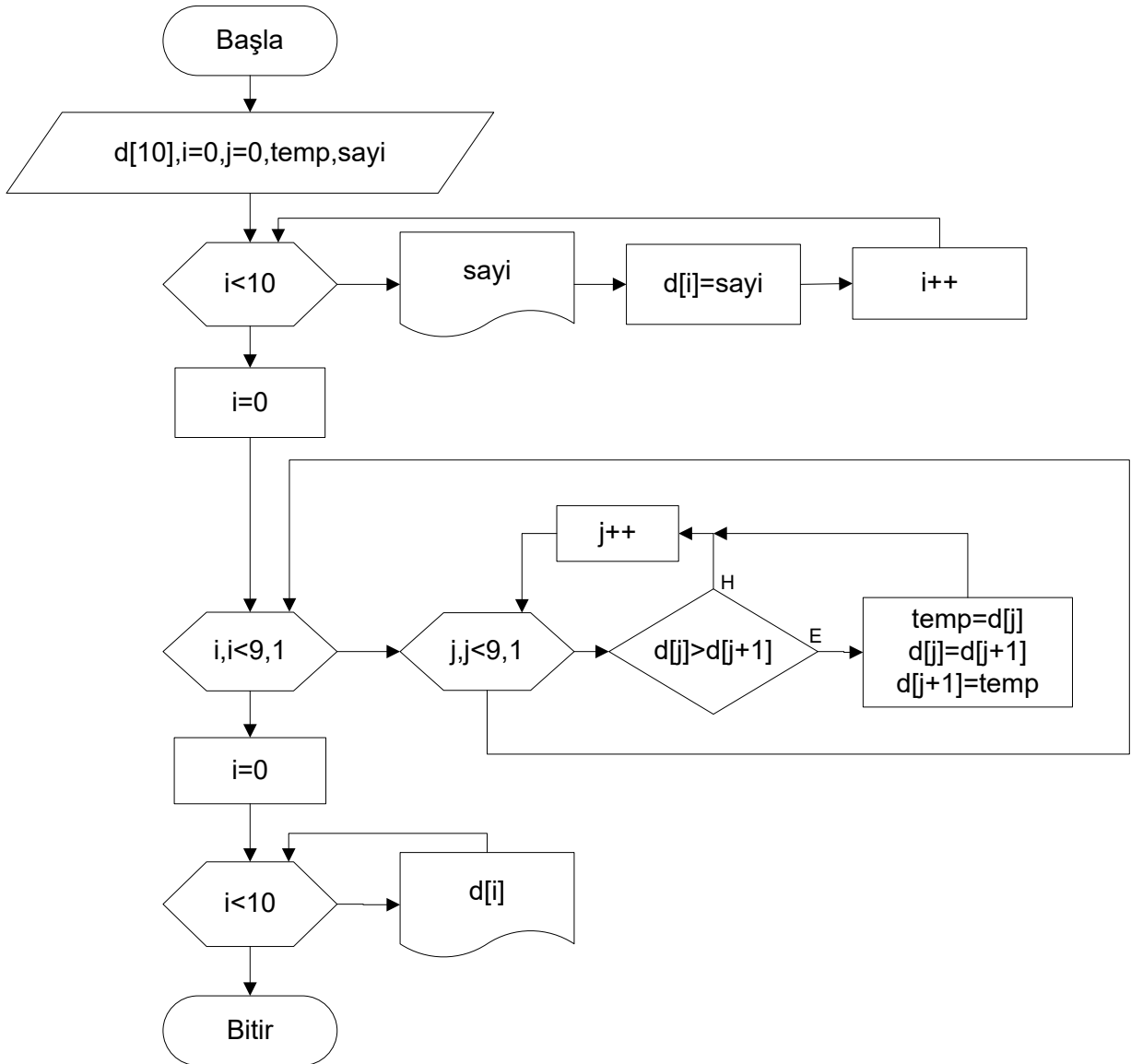
Bubble (kabarcık) sıralaması size algoritma ve akış diyagramını çizdiğimiz ilk sorudur. Bubble kullanılmayan, daha çok eğitim amaçlı anlatılan bir sıralama algoritmasıdır. Bu algoritmanın mantığı ilk iki terimin karşılaştırılıp küçük olanın sola büyük olanın sağa geçirilmesine dayanır,. Algoritma sonra sağdaki ile devam eder ve sağdaki sayı da bir sonraki elemanla karşılaştırılır. Büyük ve küçük olmasına göre sayılar yine küçük sola, büyük sağa gelmek koşuluyla dizi sonuna kadar devam eder. Bu dizi doğru sıralamaya ulaşana kadar devam etmelidir. Algoritma testinde sayılarla bu sıralama mantığını daha iyi kavrayabilirsiniz.

$O(n^2)$ çalışma karmaşıklığıdır.

Algoritma Testi:

i	j	d[i]	d[j]	d[i]	d[i]	d[i]	işlem	
0	0	3	5	7	2	4	3<5	
0	1	3	5	7	2	4	5<7	
0	2	3	5	7	2	4	7>2	Takas
0	3	3	5	2	7	4	7>4	Takas
1	0	3	5	2	4	7		
İşlem i=n-2 oluncaya kadar devam eder ve dizi sıralanır								

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
int i,j,d[10],temp;
main()
{
for(i=0;i<10;i++)
{
scanf("%d",&d[i]);
}
for(i=0;i<9;i++)
{
for(j=0;j<9;j++)
{
if(d[j]>d[j+1])
{
temp=d[j];
d[j]=d[j+1];
d[j+1]=temp;
}
}
}
for(i=0;i<10;i++)
{
printf("%2d",d[i]);
}
getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
class Program
{
static void Main(string[] args)
{
int i;
int j=0;
int temp;
int sayi;
int[] dizi = new int[10];
for (i = 0; i < 10; i++)
{
Console.Write(i+" Sayiyi giriniz = ");

dizi[i]=Convert.ToInt32(Console.ReadLine());
}
for (i = 0; i < 9; i++)
{
for (j = 0; j < 9; j++)
{
if(d[j]>d[j+1])
{
temp=d[j];
d[j]=d[j+1];
d[j+1]=temp;
}
}
j = 0;
}
for (i = 0; i < 10; i++)
{
Console.WriteLine(dizi[i]);
}
Console.ReadLine();
}
}
```

96. Selection sort (seçme sıralama) algoritması ile bir dizinin sıralanması programının algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $d[6]=\{10,1,9,2,8,3\}$, $i=0$, $j=0$, $\min$, temp , $n=6$, indis
- 3- $i < n-1$ olana kadar 14. adıma kadar yap
- 4- $\min = d[i]$
- 5- $\min = i$
- 6- $j = i+1$
- 7- $j < n$ olana kadar 11. adıma kadar yap
- 8- Eğer $d[j] < \min$ ise devam et, değilse $j++$ 7'e git
- 9- $\min = d[j]$
- 10- $\text{indis} = j$
- 11- $\text{temp} = d[i]$
- 12- $d[i] = \min$
- 13- $d[\text{indis}] = \text{temp}$
- 14- $i = 0$
- 15- $i < n$ olana kadar 16'yı yap
- 16- Yazdır $d[i]$
- 17- Bitir

Açıklama:

Bubble, sort'a göre daha iyidir ama bu algoritmadan daha iyi sıralama algoritmaları da vardır. Bu sıralama algoritmasının da çalışma karmaşıklığı $Q(n^2)$ 'dir. Mantık olarak dizinin ilk elemanını minumum alır, bütün dizinin elemanlarını bu minumum ile karşılaştırır, küçük bir değer bulursa dizi sonu gelince bulduğu en küçük değeri dizinin başına atar ve dizinin ikinci elemanı da tekrar bu işleme devam eder. Dizi sonu gelince dizi sıralanmış olur. Her zaman dizinin başına minumumu koyarak ve sağa doğru öteleyerek sıralama yapar. Algoritma testinde bu durum açıkça gösterilecektir.

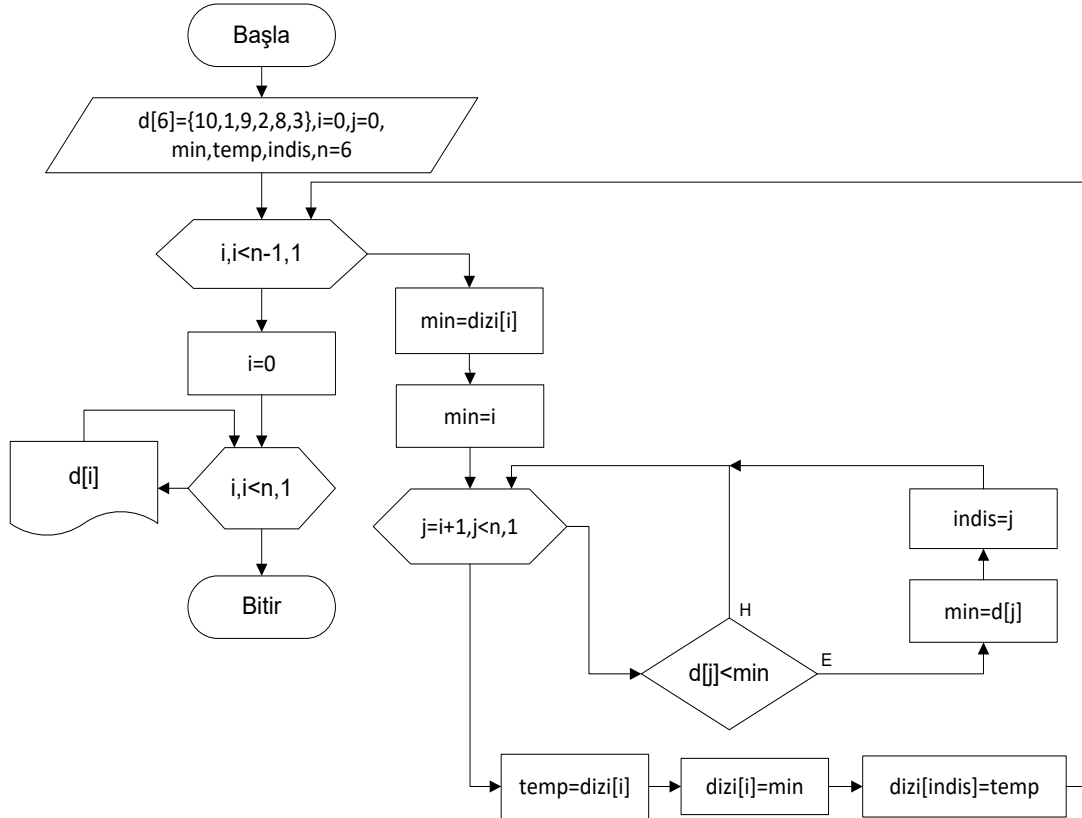
Örnek

min	1	2
9	1	3
0	min	2
9	1	3
0	min	2
9	1	3
Takas		
1	min	2
1	9	3
İkinci için tekrar bakılır		

Algoritma Testi:

n	temp	indis	min	i	j	d[0]	d[1]	d[2]	d[3]
4	0	0	10	0	1	10	1	9	2
min>d[j]									
	0	1	1	0	2	10	1	9	2
min<d[j]						10	1	9	2
min>d[j]						10	1	9	2
4	10	1	1	0	3	10	1	9	2
4	10	1	1	0	3	1	10	9	2
min ile dizinin ilk elemanı seçildi bu işlem i=2 adımında işledikten sonra program sonlanır									

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int dizi[6]={10,1,9,2,8,3};
    int n=6;
    int i, j, indis,temp;
    int min;
    for (i=0;i<n-1;i++)
    {
        min=dizi[i];
        indis=i;
        for (j=i+1;j<n;j++)
        {
            if (dizi[j]<min)
            {
                min=dizi[j];
                indis=j;
            }
        }
        temp=dizi[i];
        dizi[i]=min;
        dizi[indis]=temp;
    }
    printf("sıralanmis dizi\n");
    for(i=0;i<n;i++)
    printf("%d",dizi[i]);
    printf("\n");
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] dizi = new int[6] { 10, 1, 9, 2, 8, 3 };
            int n = 6;
            int i, j, indis, temp;
            int min;
            for (i = 0; i < n - 1; i++)
            {
                min = dizi[i];
                indis = i;
                for (j = i + 1; j < n; j++)
                {
                    if (dizi[j] < min)
                    {
                        min = dizi[j];
                        indis = j;
                    }
                }
                temp = dizi[i];
                dizi[i] = min;
                dizi[indis] = temp;
            }
            Console.WriteLine("Sıralanmis dizi");
            for (i = 0; i < n; i++)
            {
                Console.WriteLine(dizi[i]);
            }
            Console.ReadLine();
        }
    }
}
```

97. Insertion Sort (ekleme sıralaması) algoritmasını kullanarak sıralama yapan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- dizi[6]={0,1,9,2,8,3},n=6,i,j,index
- 3- i<n olana kadar 10. adıma kadar yap
- 4- index=dizi[i]
- 5- j=i
- 6- Eğer (j>0 && dizi[j-1])>index ise devam et,değilse 9'a git
- 7- dizi[j]=dizi[j-1]
- 8- j=j-1 , 6 'ya git
- 9- dizi[j]=index,i++
- 10- i=0
- 11- i<n olana kadar 12'yi yap
- 12- Yazdır d[i]
- 13- Bitir

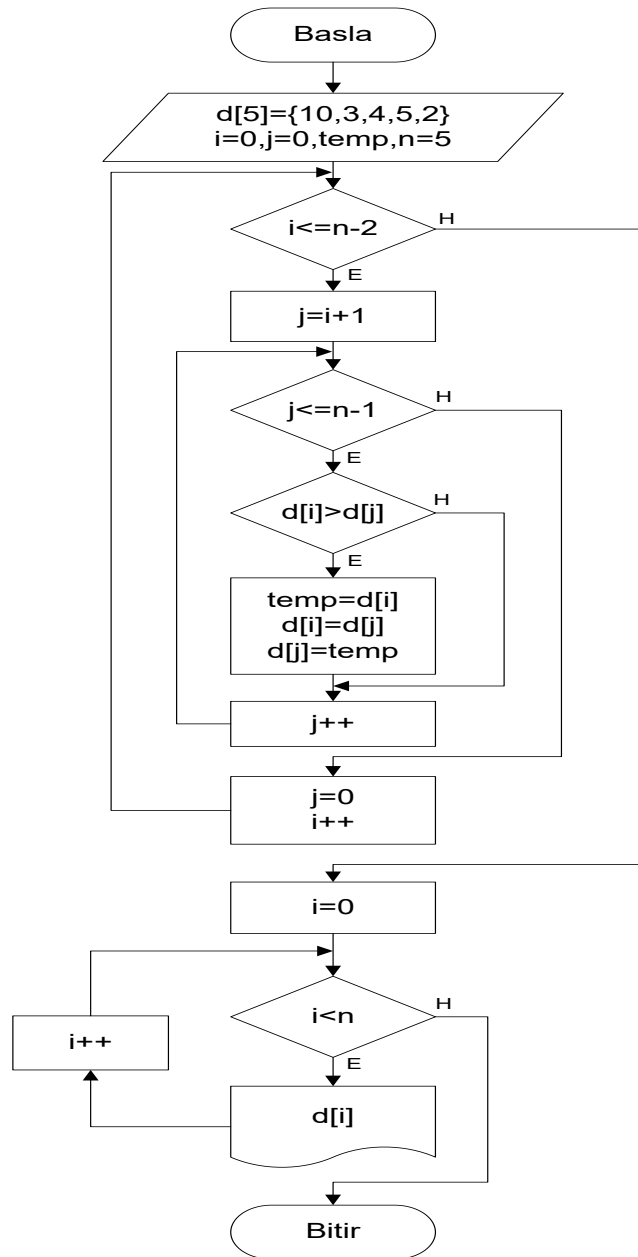
Algoritma Testi:

10	1	9	2	8	3	İlk rastgele dizimiz
10	1	9	2	8	3	1,10 un önüne eklendi
1	10	9	2	8	3	
1	9	10	2	8	3	
1	2	9	10	8	3	
1	2	8	9	10	3	
1	2	3	8	9	10	Sıralı dizi (son)

Açıklama:

Insertion sort 'da karmaşıklığı $Q(N^2)$ olan sort algoritmalarındandır. Insertion sort mantığında bir eleman kendinden önce gelen elemanla kontrol edilerek kendinden küçük elemana rastlanılana kadar o eleman için bir indis değeri yürütülür. Insertion sort, bubble sorttan neredeyse iki kat daha hızlıdır. Selection sort'a göre ise yaklaşık %40 daha efektif. Bu algoritmayı yerden sırayla çaktığımız iskambil kartlarını sıralamamıza benzetebiliriz.

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int dizi[6] = { 10,1,9,2,8,3 };
    int n=6;
    int i, j, index;
    for (i=0; i < n; i++)
    {
        index = dizi[i];
        j = i;
        while ((j > 0) && (dizi[j-1] > index))
        {
            dizi[j] = dizi[j-1];
            j = j - 1;
        }
        dizi[j] = index;
    }
    printf("sıralanmis dizi\n");
    for( i=0;i<n;i++ )
        printf("%d ",dizi[i] );
    printf("\n");
    getch();
}
```

C#Kod :

```
using System;
namespace dmj
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] dizi = new int[6] { 10, 1, 9, 2, 8, 3 };
            int n = 6;
            int i, j, index;
            for (i = 0; i < n; i++)
            {
                index = dizi[i];
                j = i;
                while ((j > 0) && (dizi[j - 1] > index))
                {
                    dizi[j] = dizi[j - 1];
                    j = j - 1;
                }
                dizi[j] = index;
            }
            Console.WriteLine("Sıralanmis dizi");
            for (i = 0; i < n; i++)
            {
                Console.WriteLine(dizi[i]);
            }
            Console.ReadLine();
        }
    }
}
```

98. Shell sort (kabuk sıralaması) algoritmasını yapan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $d[6]=\{1,15,12,9,11,2\}$,
 $n=6, i, j, x, ara$
- 3- $ara=n\backslash 2$
- 4- $ara>0$ olduğu sürece devam et,
değilse 13'e git.
- 5- $i=ara$
- 6- $i<n$ olduğu süre devam et
- 7- $x=d[i]$
- 8- $j=i-ara$
- 9- Eğer $x<d[j]$ && $j>=0$ ise devam et;
değilse 11'e git
- 10- $d[i+ara]=d[j]$ 8'e git
- 11- $d[j+ara]=x$, $i++$ 6'ya git
- 12- $ara=ara/2$ 4'e git
- 13- $i=0$
- 14- $i<n$ olduğu süre 15. adımı yap
- 15- Yazdır $d[i]$
- 16- Bitir

elamanlı bir dizi varsa, $x=6/2=3$ ise ilk adımda (0,3),(1,4),(2,5) karşılaştırılır ve yer değiştirilmesi gerekiyorsa yer değiştirilir. $x=3/2=1$, ise (0,1), (2,3), (4,5) karşılaştırılır. $x=0$ olduğunda program sonlanır.

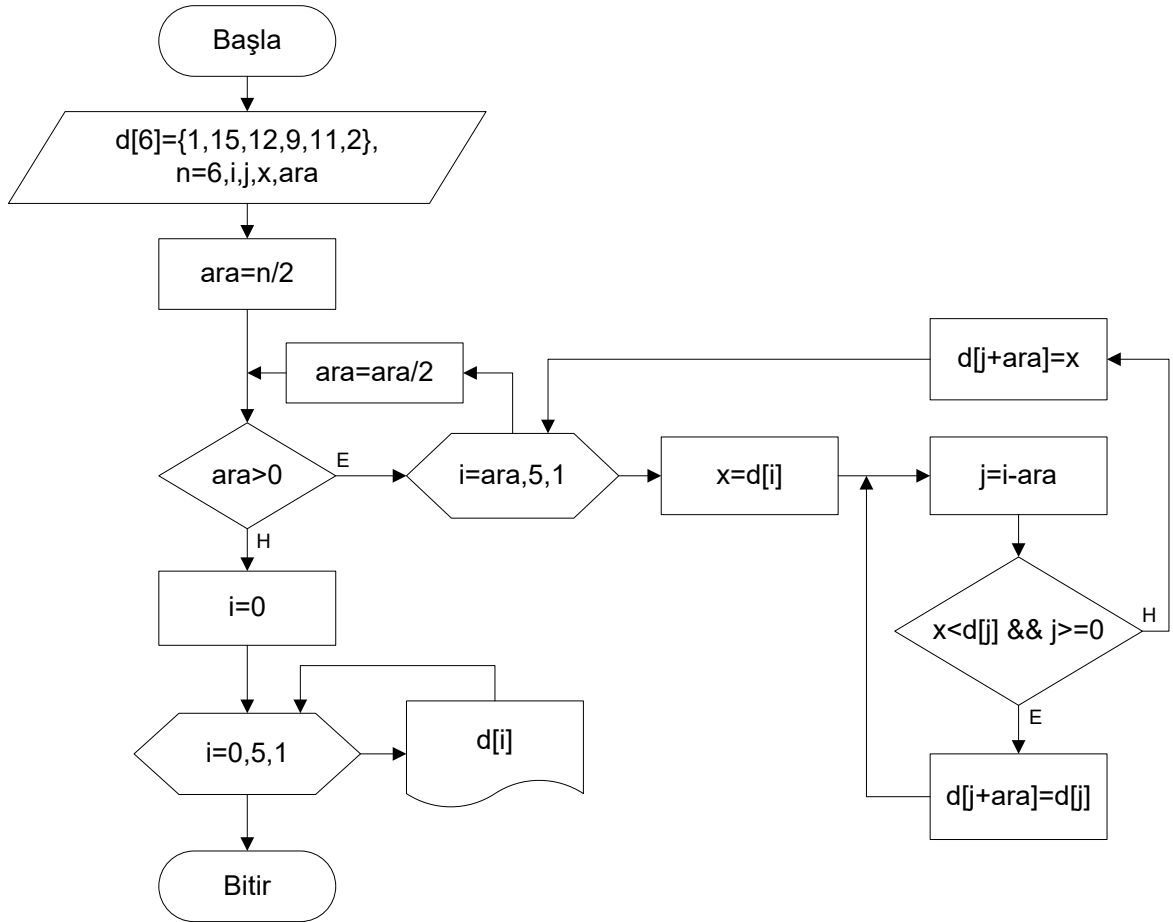
Algoritma Testi:

n	i	j	ara	d[0]	d[1]	d[2]	d[3]	d[4]	d[5]
6	3	0	$6/2=3$	1	15	12	9	11	2
6	4	1	$6/2=3$	1	15	12	9	11	2
6	5	2	$6/2=3$	1	15	12	9	11	2
6	1	0	1	1	11	2	9	15	12

Açıklama:

$O(n^2)$ karmaşıklığına sahip bir sıralama algoritmasıdır. Bu algoritmalar arasında en hızlı olanıdır. Shell algoritması, insertion algoritmasına benzer ancak bu algoritma daha hızlıdır. Çünkü sıralama işlemini bütün dizi üzerinde değil, diziyi gruplara bölerek yapar. Mantık olarak dizinin eleman sayısının yarısını bularak **indis=0** iken hangi elemanla karşılaşacağını belirleriz. 6

Akış Diyagramı:



C Kod:

```
#include <stdio.h>
#include <conio.h>
int main()
{
    int d[6]={1,15,12,9,11,2};
    int n=6;
    int i, j;
    int x;
    int orta;
    orta=n/2;
    while(orta>0)
    {
        for(i=orta; i < n; ++i)
        {
            x = d[i];
            for(j=i-orta; (x < d[j]) && (j >= 0); j=j-orta)
            {
                d[j+orta] = d[j];
            }
            d[j+orta] = x;
        }
        orta=orta/2;
    }
    printf("\n");
    for(i=0;i<6;++i)
    {
        printf("%4d", d[i]);
    }
    getch();
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] d = new int[6] { 1, 15, 12, 9, 11, 2 };
            int n = 6;
            int i, j;
            int x;
            int orta;
            orta = n / 2;
            while (orta > 0)
            {
                for (i = orta; i < n; ++i)
                {
                    x = d[i];
                    for (j = i - orta; (x < d[j]) && (j >= 0); j = j -
                    orta)
                    {
                        d[j + orta] = d[j];
                    }
                    d[j + orta] = x;
                }
                orta = orta / 2;
            }
            Console.WriteLine();
            for (i = 0; i < 6; ++i)
            {
                Console.Write(d[i]);
            }
            Console.ReadLine();
        }
    }
}
```

99. Quick Sort (hızlı sıralama) algoritmasını yapan programın algoritma ve akış diyagramını çiziniz.

Algoritma:

- 1- Başla
- 2- $d[6]=\{ 10,4,1,6,11,2\},k$
- 3- Fonksiyon(Başla)
- 4- $a[6]=\{ 10,4,1,6,11,2\},x=0,y=5,i,j,$
temp,pivot,orta
- 5- $i=x, j=y$
- 6- $orta=(i+j)/2$
- 7- $pivot=a[orta]$
- 8- $i<=j$ olduğu sürece devam et,
değilse 14'e git
- 9- Eğer $(a[i]<pivot)$ ise $i++$ 9'a git
- 10- Eğer $(a[j]>pivot)$ ise $j--$ 10'a git
- 11- Eğer $(i<=j)$ ise devam et,
değilse 14'e git
- 12- $temp=a[i], a[i]=a[j]$
- 13- $a[j]=temp, i++, j--$ 8'e git
- 14- eğer $(j>x)$ ise $y=j$ 5'e git
- 15- eğer $(i<y)$ ise $x=i$ 5'e git
- 16- $k<6$ olduğu sürece 17. Adımı
yap
- 17- yazdır $d[k]$
- 18- Bitir

herhangi bir sayı pivot alınır. Biz yine programımızda ortadaki sayıyı yani elaman sayısı/2'yi pivot alacağız. Pivottan küçükler sola, büyükler sağa (yani arkasına) geçecek şekilde dizi bölümlendirilir. Pivotun sağ ve solu içinde tekrar aynı algoritma kullanılarak sıralanır ve son olarak içinde sayı olmayan ($e.s=0$) bir alt dizi ulaştığında bir alt dizi sıralı hale gelmiştir. Bu soruda C kodunu da yazarken void bir fonksiyon kullandık. Bu, fonksiyonel yapı programlarında önemlidir. C programlamanın ileriki konularında int, void, char... fonksiyon tipleri ile karşılaşacağız. Void fonksiyonu dışarıdan değer alan ya da almayan fakat çağrıldığı yere değer döndürmeyen fonksiyontipidir.

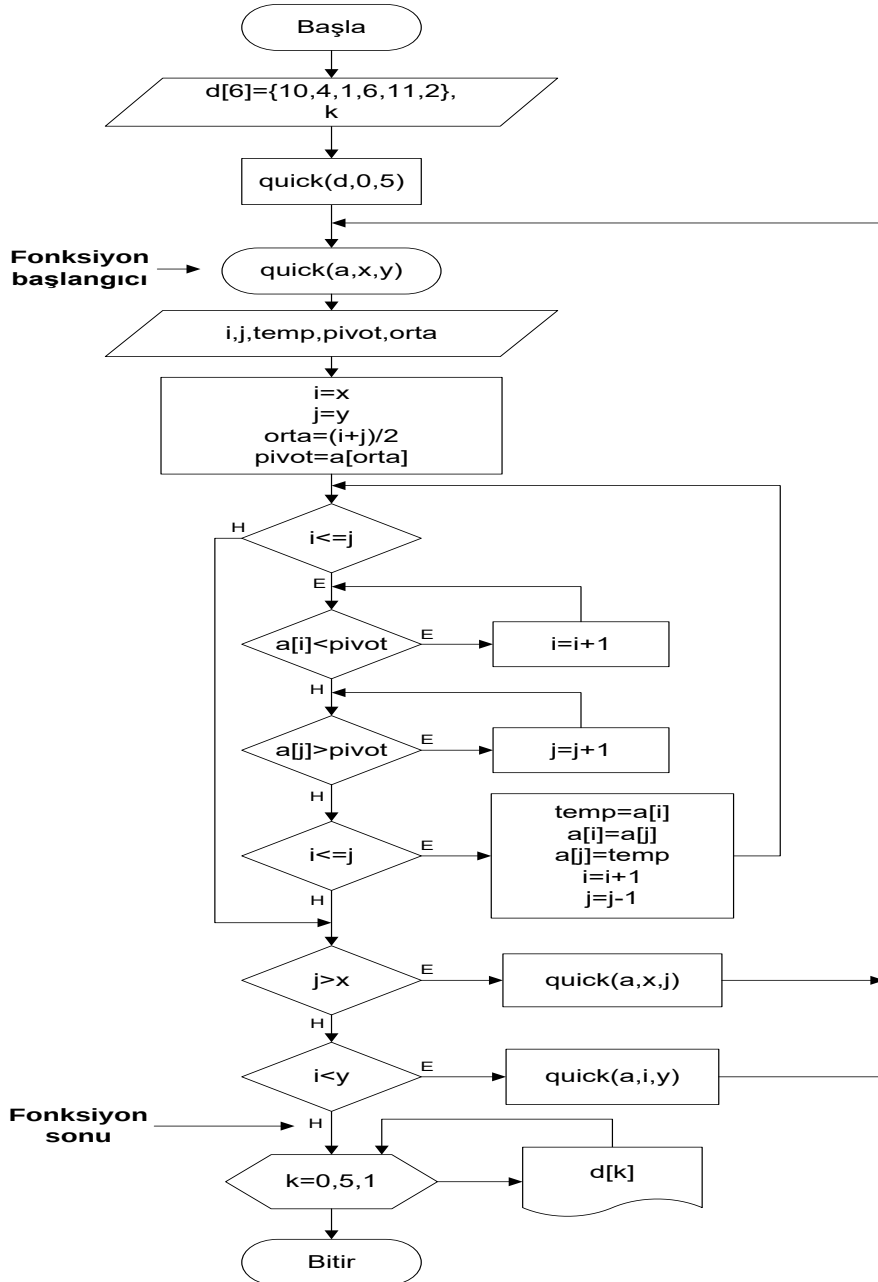
Açıklama:

$Q(n \log n)$ karmaşıklığına sahip bir algoritmadır. N adet sayıyı en kötü ihtimalle $Q(n^2)$ karmaşıklığında sıralar. Kullanılan bir algoritmadır. Mantık olarak sayı dizisinden

Algoritma Test:

	D(0)	D[1]	D[2]	D[3]	d[4]	D[5]
baş	1	4	10p.	6	11	2
1.adım	1	4	2	6	11	10
2.adım	1	2	4	6	11	10
3.adım	1	2	4	6	10	11
Sıralı	1	2	4	6	10	11

Akış Diyagramı:



C Kod:

```
#include <conio.h>
#include <stdio.h>
void quick(int a[],int x,int y)
{
    int i,j,temp,pivot,orta;
    i=x;
    j=y;
    orta=(i+j)/2;
    pivot=a[orta];
    while(i<=j)
    {
        while(a[i]<pivot)
            i++;
        while(a[j]>pivot)
            j--;
        if(i<=j)
        {
            temp=a[i];
            a[i]=a[j];
            a[j]=temp;
            i++;
        }
    }

    if(j>x)
        quick(a,x,j);
    if(i<y)
        quick(a,i,y);
}

void main()
{
    int d[6]={10,4,1,6,11,2},i;
    quick(d,1,6);
    for(i=0;i<6;i++)
    {
        printf("%4d",d[i]);
    }
    getch();
}
```

C# Kod:

```
using System;
namespace dmj
{
    class Program
    {
        class dizi
        {
            public void quick(int[] a, int x, int y)
            {
                int i, j, temp, pivot, orta;
                i = x;
                j = y;
                orta = (i + j) / 2;
                pivot = a[orta];
                while (i <= j)
                {
                    while (a[i] < pivot)
                    {
                        i++;
                    }
                    while (a[j] > pivot)
                    {
                        j--;
                    }
                    if (i <= j)
                    {
                        temp = a[i];
                        a[i] = a[j];
                        a[j] = temp;
                        i++;
                    }
                }
                if (j > x)
                {
                    quick(a, x, j);
                }
                if (i < y)
                {
                    quick(a, i, y);
                }
            }
        }
    }
}
```

```
static void Main(string[] args)
{
    dizi d = new dizi();
    int[] dizi = new int[6] { 1, 4, 1, 6, 11, 2 };
    d.quick(dizi, 1, 6);
    for (int i = 0; i < 6; i++)
    {
        Console.WriteLine(dizi[i]);
    }
    Console.ReadLine();
}
}
```

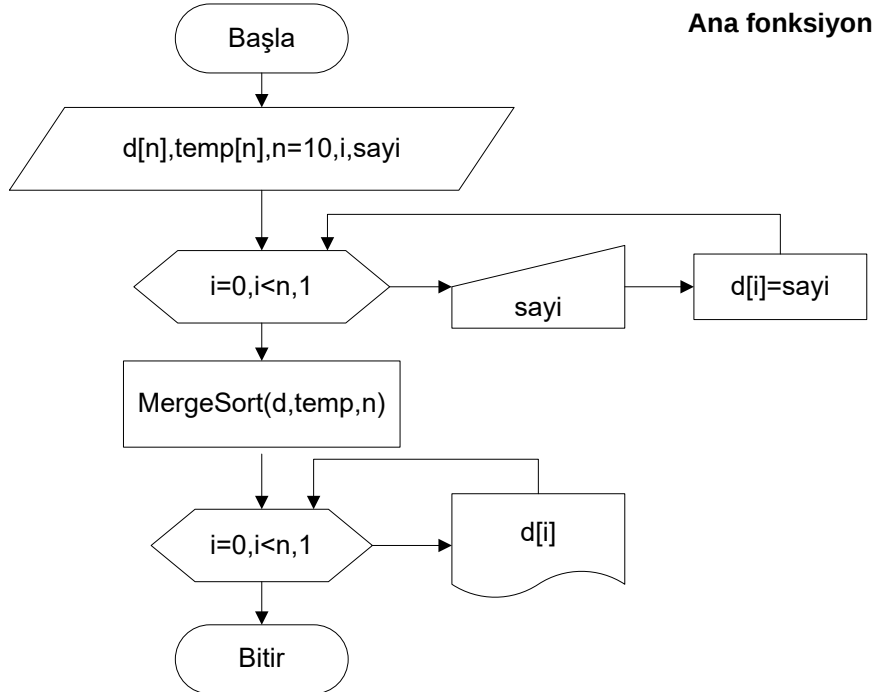
100. Merge Sort (birleşme sıralaması) algoritmasını yapan programın algoritma ve akış diyagramını çiziniz.

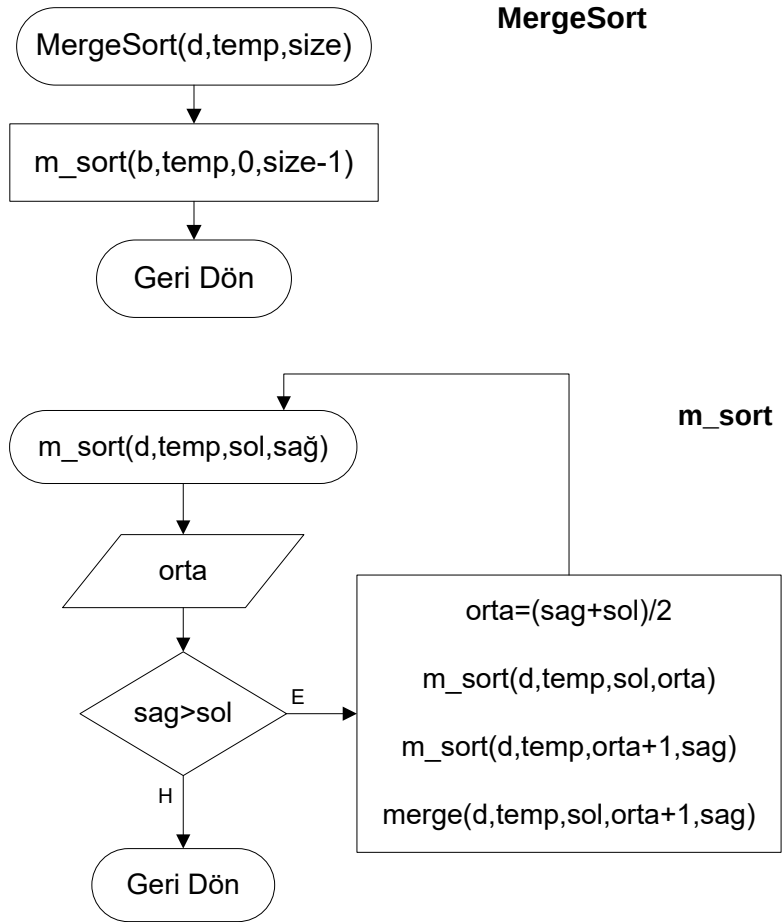
Algoritma:

- 1- Başla
- 2- d[n],temp[n],n=10,i,sayi
- 3- $i < 0$ olduğu sürece 4. adımı yap
- 4- d[i]=sayi gir
- 5- MergeSort(d,temp,n)
- 6- i=0
- 7- $i < 0$ olduğu sürece 8.adımı yap
- 8- Yazdır d[i]
- 9- Bitir
- 10- fonk(Başla)(MergeSort)
- 11- size=n
- 12- m_sort(d,temp,0,size-1)
- 13- fonk.(Bitir)
- 14- fonk(Başla)(m_sort)
- 15- sol,sag
- 16- Eğer (sag>sol) ise devam et
değilse 22'e git
- 17- orta=(sag+sol)/2
- 18- m_sort(dizi,temp,sol,orta) 14'e git
- 19- m_sort(dizi,temp,orta+1,sag) 14'e git
- 20- merge(dizi,temp,sol,orta+1,sag) 22'e git
- 21- fonk(Bitir)
- 22- fonk(Başla)(merge)
- 23- i,sol_son,eleman_sayisi,tmp_pos
- 24- sol_son=orta-1,
tmp_pos=sol,eleman_sayisi=sag-sol+1
- 25- (sol<=sol_son) && (orta<=sag) şartı doğru olduğu sürece devam et değilse 29'a git
- 26- Eğer (dizi[sol]<=dizi[orta])ise devam et,değilse 28'e git

- 27- $\text{temp}[\text{tmp_pos}] = \text{dizi}[\text{sol}]$, $\text{tmp_pos} = \text{tmp_pos} + 1$, $\text{orta} = \text{orta} + 1$ 26'ya git
- 28- $\text{temp}[\text{tmp_pos}] = \text{dizi}[\text{orta}]$, $\text{tmp_pos} = \text{tmp_pos} + 1$, $\text{orta} = \text{orta} + 1$ 26'ya git
- 29- $(\text{sol} \leq \text{sol_son})$ olduğu sürece devam et, değilse 31'e git
- 30- $\text{temp}[\text{tmp_pos}] = \text{dizi}[\text{orta}]$, $\text{orta} = \text{orta} + 1$, $\text{tmp_pos} = \text{tmp_pos} + 1$ 29'a git
- 31- $(\text{orta} \leq \text{sag})$ olduğu sürece devam et, değilse 33'e git
- 32- $\text{temp}[\text{tmp_pos}] = \text{dizi}[\text{orta}]$, $\text{orta} = \text{orta} + 1$, $\text{tmp_pos} = \text{tmp_pos} + 1$ 31'e git
- 33- $i = 0$
- 34- $i \leq \text{eleman_sayisi}$ olduğu sürece devam et, değilse 36'ya git
- 35- $\text{dizi}[\text{sag}] = \text{temp}[\text{sag}]$, $\text{sag} = \text{sag} - 1$, $i++$ 34'e git
- 36- fonk(Bitir)

Akış Diyagramı:







Açıklama:

(Merge Sort) Bilgisayar bilimlerinde $O(n \log(n))$ derecesinde karmaşıklığa sahip bir sıralama algoritmasıdır. Bu algoritma John von Neumann tarafından 1945 yılında bulunmuştur. Girdi olarak aldığı diziyi en küçük hale gelene kadar ikili gruplara böler ve karşılaştırma yöntemi kullanarak diziyi sıralar.

Algoritmanın çalışması şöyledir:

Böl, parçala, sırala, birleştir mantığı da denilebilir. Verilen değerler iki aralığa ayrılır. Daha sonra elde edilen bu aralıklar tekrar ikiye bölünür. Bu işleme aralık boyu yeterince küçük olana kadar devam edilir. Aralık boyu yeterince küçük olduğu zaman, elde edilen bu parçaların birleştirme işlemine geçilir. Parçalar, ikiye ikiye, sıralı bir şekilde birleştirilir. Oluşan yeni parçalar da yeniden sıralı bir şekilde birleştirilir. Bu işleme tüm parçalar birleşene kadar devam edilir. Quick sort ile karmaşıklığı aynıdır.

Algoritma Testi:

<u>5</u> <u>2</u> <u>4</u> <u>6</u> <u>1</u> <u>3</u> <u>2</u> <u>6</u>	böl
<u>5</u> <u>2</u> <u>4</u> <u>6</u> <u>1</u> <u>3</u> <u>2</u> <u>6</u>	böl
<u>5</u> <u>2</u> <u>4</u> <u>6</u> <u>1</u> <u>3</u> <u>2</u> <u>6</u>	böl
<u>2</u> <u>5</u> <u>4</u> <u>6</u> <u>1</u> <u>3</u> <u>2</u> <u>6</u>	sırala ve birleştir
<u>2</u> <u>4</u> <u>5</u> <u>6</u> <u>1</u> <u>2</u> <u>3</u> <u>6</u>	sırala ve birleştir
<u>1</u> <u>2</u> <u>2</u> <u>3</u> <u>4</u> <u>5</u> <u>6</u> <u>6</u>	sıralı hali

C Kod:

```
#define N 10
#include <conio.h>
#include <stdio.h>
void mergeSort(int sayilar[], int temp[], int size);
void m_sort(int dizi[], int temp[], int sol, int sag);
void merge(int dizi[], int temp[], int sol, int orta, int sag);
int dizi[N];
int temp[N];
int main()
{
    int i;
    for (i = 0; i < N; i++)
    {
        printf( "sayinizi giriniz : " );
        scanf("%d",&dizi[i]);
    }
    mergeSort(dizi, temp, N);
    printf("Siralama basariyla tamamlandi.\n");
    for (i = 0; i < N; i++)
        printf("%4d", dizi [i] );
    getch();
}
void mergeSort(int dizi[], int temp[], int size)
{
    m_sort(dizi, temp, 0, size - 1);
}
void m_sort(int dizi[], int temp[], int sol, int sag)
{
    int orta;
    if (sag > sol)
    {
        orta = (sag + sol) / 2;
        m_sort(dizi, temp, sol, orta);
        m_sort(dizi, temp, orta+1, sag);
```

```
merge(dizi, temp, sol, orta+1, sag);
}
}
void merge(int dizi[], int temp[], int sol, int orta, int sag)
{
int i, sol_son, eleman_sayisi, tmp_pos;
sol_son = orta - 1;
tmp_pos = sol;
eleman_sayisi = sag - sol + 1;
while ((sol <= sol_son) && (orta <= sag))
{
if (dizi[sol] <= dizi[orta])
{
temp[tmp_pos] = dizi[sol];
tmp_pos = tmp_pos + 1;
sol = sol + 1;
}
else
{
temp[tmp_pos] = dizi[orta];
tmp_pos = tmp_pos + 1;
orta = orta + 1;
}
}

while (sol <= sol_son)
{
temp[tmp_pos] = dizi[sol];
sol = sol + 1;
tmp_pos = tmp_pos + 1;
}

while (orta <= sag)
{
temp[tmp_pos] = dizi[orta];
orta = orta + 1;
tmp_pos = tmp_pos + 1;
}
```

```
for (i=0; i <= eleman_sayisi; i++)  
{  
    dizi[sag] = temp[sag];  
    sag = sag - 1;  
}  
  
}
```

C# Kod:

```
using System;
namespace dmğ
{
    class Program
    {
        class dizi
        {
            public void mergeSort(int[] dizi, int[] temp, int size)
            {
                m_sort(dizi, temp, 0, size - 1);
            }
            public void m_sort(int[] dizi, int[] temp, int sol, int sag)
            {
                int orta;
                if (sag > sol)
                {
                    orta = (sag + sol) / 2;
                    m_sort(dizi, temp, sol, orta);
                    m_sort(dizi, temp, orta + 1, sag);
                    merge(dizi, temp, sol, orta + 1, sag);
                }
            }

            public void merge(int[] dizi, int[] temp, int sol, int orta, int sag)
            {
                int i, sol_son, eleman_sayisi, tmp_pos;

                sol_son = orta - 1;
                tmp_pos = sol;
                eleman_sayisi = sag - sol + 1;

                while ((sol <= sol_son) && (orta <= sag))
                {
                    if (dizi[sol] <= dizi[orta])
                    {
                        temp[tmp_pos] = dizi[sol];
                        tmp_pos = tmp_pos + 1;
                        sol = sol + 1;
                    }
                }
            }
        }
    }
}
```

```
        else
        {
            temp[tmp_pos] = dizi[orta];
            tmp_pos = tmp_pos + 1;
            orta = orta + 1;
        }
    }

    while (sol <= sol_son)
    {
        temp[tmp_pos] = dizi[sol];
        sol = sol + 1;
        tmp_pos = tmp_pos + 1;
    }

    while (orta <= sag)
    {
        temp[tmp_pos] = dizi[orta];
        orta = orta + 1;
        tmp_pos = tmp_pos + 1;
    }

    for (i = 0; i <= eleman_sayisi; i++)
    {
        if (sag == -1)
        {
            sag++;
        }
        dizi[sag] = temp[sag];
        sag = sag - 1;
    }
}

static void Main(string[] args)
{
    dizi sirala = new dizi();
    int N = 10;
    int[] d = new int[N];
    int[] temp = new int[N];
    int i;
```

```
for (i = 0; i < N; i++)
{
    Console.Write(i + ".Sayinizi giriniz :");
    d[i] = Convert.ToInt32(Console.ReadLine());
}
siralama.mergeSort(d, temp, N);
Console.WriteLine("Siralama basariyla tamamlandi.");
for (i = 0; i < N; i++)
{
    Console.WriteLine(d[i]);
}
Console.ReadLine();
}
```

Sıralama Algoritmalarının karşılaştırılması

Adı	Ortalama	En Kötü	Bellek	Kararlı mı?	Yöntem
Kabarcık Sıralaması	—	$O(n^2)$	$O(1)$	Evet	Değiştirme
Seçmeli Sıralama	$O(n^2)$	$O(n^2)$	$O(1)$	Hayır	Seçme
Eklemeli Sıralama	$O(n + d)$	$O(n^2)$	$O(1)$	Evet	Ekleme
Kabuk Sıralaması	—	$O(n \log^2 n)$	$O(1)$	Hayır	Ekleme
Birleştirmeli Sıralama	$O(n \log n)$	$O(n \log n)$	$O(n)$	Evet	Birleştirme
Hızlı Sıralama	$O(n \log n)$	$O(n^2)$	$O(\log n)$	Hayır	Bölümlendirme

KAYNAKLAR

Kitaplar

1. Turbo Pascal ve Programlama Sanatı , “Ömer AKGÖBEK”, Beta Basım Yayım Dağıtım A.Ş. İstanbul, 1996
2. Veri Yapıları ve Algoritmalar ,”Dr.Rıfat Çölkesen”, Papatya Yayıncılık Eğitim, İzmir, Ankara, İstanbul , 2007
3. Programlamaya Giriş ve Algoritmalar, “Soner ÇELİKKOL”, Academic Book Publishing , Trabzon, 2007
4. Algoritma Geliştirme ve Programlamaya Giriş , “Fahri VATANSEVER”, Seçkin Yayıncılık 2005
5. A’dan Z’ye C klavuzu , “Kaan ASLAN” , Pusula Yayıncılık ,2002
6. Algoritma Geliştirme ve Veri Yapıları, ”Bülent ÇOBANOĞLU”, Pusula Yayıncılık, 2008

Web Adresleri

1. C ve Sistem Programcıları Derneği , www.csystem.org
2. <http://ankmyo.fatih.edu.tr/~esenyurek/algoritmaornek.htm>
3. <http://tbagriyanik.googlepages.com/algo-modul5.pdf>
4. <http://e-bergi.com/2008/Subat/Algoritma>
5. <http://e-bergi.com/2008/Mart/Siralama-Algoritmaları>
6. <http://www.ceturk.com/Ornekyukle.asp?id=205>
7. <http://aycanayhan.wordpress.com/2008/11/07/secmeli-siralama-selection-sort/>
8. <http://www.bydigi.net/java-jsp/318592-insertion-sort-uygulamasi.html>
9. <http://tr.wikipedia.org/>
10. <http://www.bilimfeneri.gen.tr>
11. <http://www.gurcanbanger.com/yaz/yaz4/genprog.pdf>
12. http://www.c.happycodings.com/Sorting_Searching/code13.html
13. <http://compsources0.tripod.com/download/olympiad/olimpiyatques.pdf>
14. <http://haydut.cmpe.boun.edu.tr/olimpiyat/bilgisayar/sorular.html>
15. <http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Sorting/Gifs/mergeSort.gif>

