Doç. Dr. Fahri VATANSEVER

algoritma geliştirme

ve programlamaya giriş

- Akış diyagramlarıyla birlikte 6 ayrı programlama dilinde (C, C++, C#, Java, MATLAB ve Python) çözülmüş problemler
- 375'i Çözümlü, 680 Örnek/Problem

13. Yenilenmiş Baskı

seçkin

Algoritma Geliştirme
ve
Programlamaya Giriş

Doç. Dr. Fahri VATANSEVER

ALGORİTMA GELİŞTİRME ve PROGRAMLAMAYA GİRİŞ

Akış Diyagramlarıyla Birlikte 6 ayrı Programlama Dilinde
(C, C++, C#, Java, MATLAB ve Python) Çözülmüş Problemler

- ✓ Programlamaya Giriş
- ✓ Algoritma
- ✓ Akış Diyagramı
- ✓ Programlama Dilleri (C, C++, C#, Java, MATLAB, Python, Arduino)
- ✓ Akış Diyagramından Kodlamaya Geçiş
- ✓ Genel Uygulamalar
- ✓ Bir Boyutlu Dizi Uygulamaları
- ✓ İki Boyutlu Dizi Uygulamaları
- ✓ Grafik Uygulamaları
- ✓ Dosyalama Uygulamaları
- ✓ Özel Sayı ve Şekiller
- ✓ Sıralama Algoritmaları
- ✓ Arama Algoritmaları
- ✓ Kodlama/Şifreleme Uygulamaları
- ✓ Sayısal Analiz Uygulamaları
- ✓ Donanım Uygulamaları

375'i Çözümlü, 680 Örnek/Problem

Yenilenmiş 13. Baskı

SEÇKİN | Teknik

seçkin

Algoritma Geliştirme ve Programlamaya Giriş
Fahri VATANSEVER

SEÇKİN | Teknik
No: 13

ISBN 978-975-02-4430-8

Onüçüncü Baskı: Eylül 2017 (Ankara)

Sayfa Tasarımı:
İlhan Eroğlu

640 Sayfa, 16x23,5 cm.

1. Algoritma
2. Akış diyagramı
3. Programlama
4. C
5. C++
6. C#
7. Java
8. MATLAB
9. Python
10. Arduino

Kapak Tasarımı:
Musa Gündoğan

© Seçkin Yayıncılık San. ve Tic. A.Ş.
Bu kitabın her türlü yayın hakkı Seçkin Yayıncılık San. ve Tic. A.Ş.'ye aittir. Yayınevinin yazılı izni olmadan, tanıtım amaçlı toplam bir sayfayı geçmeyecek alıntılar hariç olmak üzere, hiçbir şekilde kitabın tümü veya bir kısmı herhangi bir ortamda yayımlanamaz ve çoğaltılamaz.

Satış ve Dağıtım:

Merkez
Mustafa Kemal Mah.
2158. Sokak No: 13
Çankaya / ANKARA
Tel: (312) 435 30 30
Faks: (312) 435 24 72
satis@seckin.com.tr

Şube
Strazburg Cad. No: 23/B
Sıhhiye / ANKARA
Tel: (312) 230 52 62
ankarasube@seckin.com.tr

Şube
Abide-i Hürriyet Cad.
No: 183/A Şişli / İSTANBUL
Tel: (212) 234 34 77
Faks: (212) 231 24 69
sislisube@seckin.com.tr

Şube
Anadolu Adalet Sarayı
C Blok Zemin Kat No: 29
Kartal / İSTANBUL
Tel-Faks: (216) 303 11 23
kartalsube@seckin.com.tr

Şube
İstanbul Adalet Sarayı
D Blok 2. Bodrum Kat
Çağlayan/İSTANBUL
Tel - Faks: (212) 240 00 15
caglayan@seckin.com.tr

Web Sitesi: www.seckin.com.tr

Seçkin Yayıncılık Sertifika No: 12416

Baskı:
Sözkesen Matbaacılık Tic. Ltd. Şti. – Sertifika No: 13268
İVOGSAN 1518 Sok. Mat-Sit İşmerkezi No: 2/40 Yenimahalle/ANKARA - Tel: (0-312) 395 21 10

Önsöz

Günümüz "*bilgi ve teknoloji toplumunda*"; bilgisayar, insan hayatında vazgeçilmez bir yer edinmiştir. "*Bilgi işleyen bu elektronik makine*", insana birçok olanaklar sunmakta ve hayatını önemli ölçüde kolaylaştırmaktadır. Bilgisayar dünyasındaki gelişmeler; baş döndürücü bir hızla devam ederken bunlar, sadece donanım ile sınırlı kalmayıp yazılıma da yansımaktadır.

Program yazmaya karar veren bilgisayar kullanıcılarının ilk öğrenmesi gereken konu, "*algoritma hazırlama ve geliştirme tekniğidir*". Eğer programın algoritması hazırlanabiliyorsa; o program, bütün programlama dillerinde kolaylıkla kodlanabilir. Çünkü programlama dillerindeki kod karşılıklarında, çok az sayıda komut ve yapı değişikliği oluşacaktır. Bundan dolayı "*programcılığın temelinde, algoritma geliştirme vardır*". Elinizdeki kitapta da değişik birçok problemin akış diyagramı çizilerek, "*algoritma geliştirme mantığı*"nın kavranması amaçlanmıştır.

Kitapta bilgisayar programlaması ile ilgilenen kişilere, başlangıç seviyesindeki bilgilerin öğretilmesi yanında; çok farklı konuların açıklanması ve bilgisayar ortamına aktarılması, özellikle de teknik konularda bilgisayar kullanmak isteyen kişilere/öğrencilere bir ufuk verilmeye çalışılmıştır. Verilen örnek problemler ve çözümleriyle; üniversitelerimizin *lisans* ve *önlisans* bölüm/programlarındaki bilgisayar programlamayla ilgili ders alan öğrencilerden *lise* öğrencilerine kadar ve ayrıca *bilgisayar programlamasını öğrenmek isteyen tüm kişilere* faydalı/kullanılabilir olması hedeflenmiştir. Kitapta yer alan akış diyagramlarının tümü kodlanarak çalıştırılmış, ancak benzer kod/program tekrarından kaçınmak için bazılarının program kodları verilmemiştir.

Kitabın basımında emeği geçen Seçkin Yayıncılık'ın değerli tüm personeline teşekkürlerimi sunarım.

Tüm okuyuculara faydalı olması dileklerimle...

Fahri Vatansever

Ağustos 2017 - Bursa

İçindekiler

Önsöz 5

Bölüm 1
PROGRAMLAMAYA GİRİŞ

I. GİRİŞ 20
II. PROGRAMLAMA DİLLERİ 28
III. İŞLEMLER 31
A. Matematiksel İşlemler 32
B. Karşılaştırma İşlemleri 37
C. Mantıksal İşlemler 39
SORULAR 46

Bölüm 2
ALGORİTMA

I. GİRİŞ 51
II. ALGORİTMALARDA KULLANILAN OPERATÖRLER 54
III. ALGORİTMALARDA KULLANILAN TERİMLER 55
A. Veri 55
B. Tanımlayıcı 56
C. Değişken 56
D. Sabit 58
E. Atama/aktarma 58
F. Sayaç 60
G. Döngü 61
H. Ardışık Toplama 61
I. Ardışık Çarpma 62
IV. ALGORİTMA HAZIRLAMA 63
SORULAR 68

Bölüm 3
AKIŞ DİYAGRAMI

I. GİRİŞ 71
II. AKIŞ DİYAGRAMINDA KULLANILAN ŞEKİLLER 71
A. Başla/Dur 71

B. Veri Girişi 71
C. İşlem 71
D. Döngü 72
E. Karar (Karşılaştırma) 76
F. Yazdırma/Çıktı 78
G. Önceden tanımlı işlem 79
H. Bağlantı 80
I. İşlem Akış Yönleri 80
SORULAR 83

Bölüm 4
PROGRAMLAMA DİLLERİ

I. GİRİŞ 87
II. PROGRAMLAMA DİLLERİNDEKİ VERİ TİPLERİ 87
A. C'deki Veri Tipleri 87
1. Sayısal Veri Tipleri 87
a. Tamsayı Veri Tipleri 87
b. Ondalıklı Sayı Veri Tipleri 88
2. Alfasayısal Veri Tipleri 88
B. C++'daki Veri Tipleri 89
1. Sayısal Veri Tipleri 89
a. Tamsayı Veri Tipleri 89
b. Ondalıklı Sayı Veri Tipleri 89
2. Alfasayısal Veri Tipleri 90
C. C#'daki Veri Tipleri 90
1. Sayısal Veri Tipleri 90
a. Tamsayı Veri Tipleri 90
b. Ondalıklı Sayı Veri Tipleri 90
2. Alfasayısal Veri Tipleri 91
D. Java'daki Veri Tipleri 91
1. Sayısal Veri Tipleri 91
a. Tamsayı Veri Tipleri 91
b. Ondalıklı Sayı Veri Tipleri 91
2. Alfasayısal Veri Tipleri 91
E. MATLAB'taki Veri Tipleri 92
F. Python'daki Veri Tipleri 92
III. PROGRAMLAMA DİLLERİNİN GENEL YAPILARI 92
A. C Dilinin Yapısı 92
1. Program Başlığı 93
2. Tanımlama ve Bildirimler Bölümü 93
a. include 93
b. define 94
c. undef 94
d. Özel Veri Tipi Tanımlama 94
e. Veri Alanı Tanımlama 95

f. Değişken Bildirme 95
g. Sabit Tanımlama veya Başlangıç Değeri Verme 95
3. Alt Programlar Bölümü 95
4. Ana Program Bölümü 97
B. C++ Dilinin Yapısı 97
1. Program Başlığı 98
2. Tanımlama ve Bildirimler Bölümü 98
a. include 99
b. define 100
c. undef 100
d. Özel Veri Tipi Tanımlama 100
e. Veri Alanı Tanımlama 100
f. Sıralama Tipleri Tanımlama 100
g. Değişken Bildirme 101
h. Sabit Tanımlama veya Başlangıç Değeri Verme 101
3. Alt Programlar Bölümü 101
4. Ana Program Bölümü 102
C. C# Dilinin Yapısı 102
1. Program Başlığı veya Açıklamalar 103
2. using 103
3. namespace 103
4. class 104
5. struct 104
6. enum 105
7. Değişken ve Sabit Bildirimleri 105
8. Ana program 105
D. Java Dilinin Yapısı 105
1. Program Başlığı 107
2. Sınıf Çağırma Bölümü 107
3. Sınıflar 107
a. Sınıf Tanımlama 107
b. Değişken Bildirme 107
c. Sabit Tanımlama veya Başlangıç Değeri Verme 108
E. MATLAB Dilinin Yapısı 108
1. Program Başlığı 108
2. Bildirimler/Tanımlamalar 109
a. Değişken Bildirme 109
b. Sabit Tanımlama veya Başlangıç Değeri Verme 109
c. Yapı Tanımlama 109
d. Sınıf Tanımlama 109
3. Fonksiyonlar 110
F. Python Dilinin Yapısı 111
1. Program Başlığı 111
2. Bildirimler/Tanımlamalar 111
a. Değişken/Sabit Tanımlama veya Başlangıç Değeri Verme 111
b. Sınıf Tanımlama 112
c. Modül Çağırma 112
3. Fonksiyonlar 112
SORULAR 116

Bölüm 5
AKIŞ DİYAGRAMINDAN KODLAMAYA GEÇİŞ

I. GİRİŞ **119**

II. VERİ GİRİŞ KOMUTLARI – "Al/Oku/Gir" **121**

A. C'deki Veri Giriş Komutları 122
1. scanf 122
2. gets 124
3. getch 124
4. getche 124
5. getchar 124
6. kbhit 124

B. C++'daki Veri Giriş Komutları 125
1. cin 125
2. get 126
3. getline 126

C. C#'daki Veri Giriş Komutları 126
1. ReadLine 126
2. ReadKey 126
3. Read 127
4. KeyAvailable 127

D. Java'daki Veri Giriş Komutları 127

E. MATLAB'daki Veri Giriş Komutları 128
1. input 128
2. inputdlg 128

F. Python'daki Veri Giriş Komutları 129
1. input 129

III. VERİ ÇIKIŞ KOMUTLARI – "Yaz" **130**

A. C'deki Veri Çıkış Komutları 130
1. printf 130
2. cprintf 133
3. putchar 133
4. puts 133
5. cputs 133

B. C++'daki Veri Çıkış Komutları 133
1. cout 133

C. C#'daki Veri Çıkış Komutları 136
1. Write 136

D. Java'daki Veri Çıkış Komutları 137
1. print 137
2. printf 137

E. MATLAB'daki Veri Çıkış Komutları 139
1. disp 139
2. fprintf 139
3. sprintf 141

F. Python'daki Veri Çıkış Komutları 142
1. print 142

2.Çıkışları biçimlendirme 143
a. format 143
b. center 145
c. ljust 145
d. rjust 145
e. zfill 145

IV. DÖNGÜ (ÇEVRİM) KOMUTLARI 146

A. C/C++/C#/Java'daki Döngü Komutları 146
1. for 147
2. while 148
3. do – while 149
4. foreach 150

B. MATLAB'daki Döngü Komutları 151
1. for 151
2. while 152

C. Python'daki Döngü Komutları 152
1. for 152
2. while 154

V. KARAR (KARŞILAŞTIRMA) KOMUTLARI - "Eğer" 155

A. C/C++/C#/Java'daki Karar Komutları 155
1. if-else if-else 155
2. switch 157
3. ? 159

B. MATLAB'daki Karar Komutları 160
1. if-elseif-else 160
2. switch 161

C. Python'daki Karar Komutları 162
1. if-elif-else 162

VI. DİĞER BAZI KOMUTLAR 163

A. Ekran Komutları 163
B. Matematiksel İşlem Komutları 164
C. Alfasayısal İşlem Komutları 166
D. Diğer Komutlar 167

SORULAR 168

Bölüm 6
GENEL UYGULAMALAR

GENEL UYGULAMALAR 169

SORULAR 238

Bölüm 7
BİR BOYUTLU DİZİ UYGULAMALARI

BİR BOYUTLU DİZİ UYGULAMALARI **243**

SORULAR **304**

Bölüm 8
İKİ BOYUTLU DİZİ UYGULAMALARI

İKİ BOYUTLU DİZİ UYGULAMALARI **307**

SORULAR **376**

Bölüm 9
GRAFİK UYGULAMALARI

I. GİRİŞ **381**

II. GRAFİK KOMUTLARI **381**

A. Grafik Ekranına Geçiş Komutları 382

B. Çizgi 382

C. Daire 383

D. Dikdörtgen 383

E. Nokta Koy 383

F. Diğer Bazı Grafik Ekranı Komutları 384

SORULAR **392**

Bölüm 10
DOSYALAMA UYGULAMALARI

I. GİRİŞ **395**

II. DOSYALAMA KOMUTLARI **396**

A. Dosya Aç 396

B. Dosya Kapat 396

SORULAR **401**

Bölüm 11
ÖZEL SAYI VE ŞEKİLLER

I. e **405**

II. π **409**

III. $\sqrt{2}$ 410
IV. $\sqrt{3}$ 411
V. $\sqrt{5}$ 412
VI. ALTIN ORAN 412
VII. ALTIN ÜÇGEN 413
VIII. GÜMÜŞ ORAN 414
IX. ASAL SAYI 414
X. MÜKEMMEL SAYI 420
XI. CULLEN SAYISI 422
XII. FERMAT SAYISI 422
XIII. LEYLAND SAYISI 423
XIV. HILBERT SAYISI 423
XV. ARMSTRONG SAYISI 423
XVI. DOST SAYILAR 424
XVII. BAĞDAŞIK SAYILAR 424
XVIII. EKSİK SAYI 425
XIX. TAU SAYISI 426
XX. HARMONİK BÖLÜCÜ (ORE) SAYISI 426
XXI. MARKOV (MARKOFF) SAYISI 427
XXII. HETEROMETRİK SAYI 427
XXIII. ARİTMETİK SAYI 427
XXIV. PALINDROM SAYI 428
XXV. EULERIAN SAYISI 428
XXVI. FAKTORION 429
XXVII. BELL SAYISI 430
XXVIII. HERMITE SAYISI 430
XXIX. FIBONACCI SAYISI 431
XXX. PELL SAYISI 436
XXXI. DELANNOY SAYISI 437
XXXII. ÜÇGENSEL SAYI 438
XXXIII. KARE ÜÇGENSEL (ÜÇGEN KARESEL) SAYI 439

XXXIV. FLOYD ÜÇGENİ 440
XXXV. PISAGOR ÜÇLEMESİ 441
XXXVI. BELL ÜÇGENİ 442
XXXVII. STIRLING ÜÇGENİ 444
XXXVIII. PASCAL ÜÇGENİ 445
XXXIX. TOTIENT 448
SORULAR 450

Bölüm 12
SIRALAMA ALGORİTMALARI

I. SEÇME SIRALAMA (SELECTION SORT) 463
II. KABARCIK SIRALAMA (BUBBLE SORT) 467
III. KOKTEYL SIRALAMA (COCKTAIL SORT, BIDIRECTIONAL BUBBLE SORT, COCKTAIL SHAKER SORT, SHAKER SORT, RIPPLE SORT, SHUFFLE SORT, SHUTTLE SORT) 471
IV. TEK-ÇİFT SIRALAMA (ODD–EVEN SORT, ODD–EVEN TRANSPOSITION SORT, BRICK SORT) 472
V. TARAK SIRALAMA (COMB SORT) 474
VI. ARAYA EKLEMELİ SIRALAMA (INSERTION SORT) 477
VII. KABUK SIRALAMA (SHELL SORT) 481
VIII. CÜCE SIRALAMA (GNOME SORT, STUPID SORT) 482
IX. PANKEK SIRALAMA (PANCAKE SORT) 484
X. RASTGELE SIRALAMA (RANDOM SORT, BOGO SORT, MONKEY SORT, SHOTGUN SORT) 488
XI. PERMÜTASYON SIRALAMA (PERMUTATION SORT) 488
XII. BİRLEŞTİRME/BİRLEŞTİRMELİ SIRALAMA (MERGE SORT) 489
XIII. HIZLI SIRALAMA (QUICK SORT) 492
XIV. SAYMA SIRALAMA (COUNTING SORT) 495
XV. HANE/BASAMAK SIRALAMA (RADIX SORT) 495
XVI. KOVA SIRALAMA (BUCKET SORT) 496
XVII. PARÇALI SIRALAMA (STOOGE SORT) 497
XVIII. YIĞIN SIRALAMA (HEAP SORT) 499
SORULAR 503

Bölüm 13
ARAMA ALGORİTMALARI

I. DOĞRUSAL/SIRALI ARAMA (LINEAR/SEQUENTIAL SEARCH) 507
II. ATLAMALI/BLOK ARAMA (JUMP/BLOCK SEARCH) 507
III. İKİLİ ARAMA (BİNARY SEARCH) 508
IV. ENTERPOLASYON ARAMA (INTERPOLATION SEARCH) 511
SORULAR 513

Bölüm 14
KODLAMA/ŞİFRELEME UYGULAMALARI

I. TABAN DÖNÜŞÜMLERİ 517
II. SAYISAL KODLAR 523
A. BCD (8421) kodu 523
B. +3 kodu 527
C. 5'te 2 (74210) kodu 527
D. Aiken (2421) kodu 528
E. Parity (eşlik) kodu 529
F. Gray kodu 530
III. ALFASAYISAL KODLAR 533
IV. ROMEN RAKAMLARI 534
V. MORSE ALFABESİ 538
VI. BAR (ÇUBUK) KODU 541
VII. IBAN/KREDİ KARTI/IMEI KODLAMA 542
VIII. RENK KODLAMALARI 544
A. RGB → Hex 545
B. Hex → RGB 545
C. RGB → Gri 546
D. RGB → HSV 546
E. HSV → RGB 548
F. RGB → CMYK 549
G. CMYK → RGB 549
H. RGB → HSL 550
I. HSL → RGB 551
IX. SEZAR ŞİFRELEME 552
X. ROT13 ŞİFRELEME 554

XI. AFFINE ŞİFRELEME **554**
XII. ATBASH ŞİFRELEME **559**
XIII. HILL ŞİFRELEME **559**
XIV. POLYBIUS KARE ŞİFRELEME **561**
XV. VIGENÈRE ŞİFRELEME **562**
SORULAR **564**

Bölüm 15
SAYISAL ANALİZ UYGULAMALARI

I. ENTERPOLASYON **569**
A. Doğrusal (lineer) enterpolasyon 569
B. Parabolik (kuadratik) enterpolasyon 569
C. Lagrange enterpolasyonu 570
II. SAYISAL TÜREV **573**
III. SAYISAL İNTEGRAL **578**
A. Dikdörtgenler yöntemi 578
B. Yamuklar yöntemi 581
C. Simpson yöntemi 582
IV. KÖK HESAPLARI **583**
A. Yarılama yöntemi 583
B. Kirişler (Regula-False) yöntemi 586
C. Teğetler (Newton-Raphson) yöntemi 587
V. DOĞRUSAL DENKLEM SİSTEMLERİNİN ÇÖZÜMLERİ **591**
A. Ters Matris Yöntemi 591
B. Cramer yöntemi 594
C. Seidel Yöntemi 595
VI. DİFERANSİYEL DENKLEM ÇÖZÜMLERİ **599**
A. Euler Yöntemi 599
B. Heun Yöntemi 602
C. Runge-Kutta Yöntemleri 604
D. Diferansiyel Denklem Sistemleri Çözümleri 607
1. Euler Yöntemiyle Diferansiyel Denklem Sistemi Çözümü 607
2. Runge-Kutta Yöntemiyle Diferansiyel Denklem Sistemi Çözümü 608
SORULAR **611**

Bölüm 16
DONANIM UYGULAMALARI

I. BİLGİSAYAR PORTLARI **615**

II. ARDUINO **618**

A. Arduino programlama 619
1. Arduino program yapısı 620
2. Veri tipleri, sabitler, bildirimleri ve dönüşümleri 620
3. İşlemler 621
4. Karar ve döngü yapıları 622
5. Bazı fonksiyonlar 622
6. Giriş/Çıkış fonksiyonları 622

SORULAR **629**

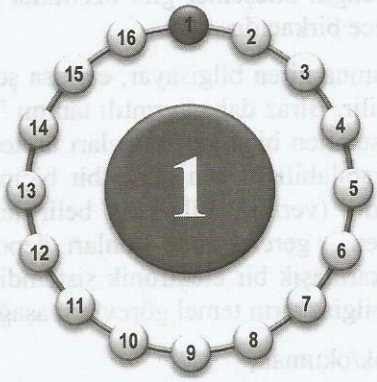
Ek - A: ASCII Kodları **631**

Kaynaklar **633**

Dizin **635**

BÖLÜM 1

PROGRAMLAMAYA GİRİŞ



Bu bölümde

- Bilgisayar ve programalama hakkındaki temel bilgileri,
- Programlardaki temel işlemleri,
- Matematiksel, karşılaştırma ve mantıksal işlemlerin kodlanmasını ve gerçekleştirilmesini öğreneceksiniz.

Bölüm sonunda

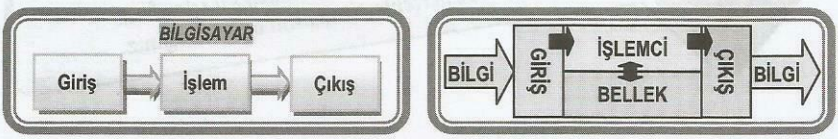
- Bilgisayar ve programalama terimlerini açıklayabilecek,
- Matematiksel ifadeleri kodlayabilecek ve analiz edebilecek,
- Karşılaştırma (karar) ifadeleri oluşturabilecek ve analiz edebilecek,
- Mantıksal ifadeleri oluşturabilecek ve analiz edebileceksiniz.

I. GİRİŞ

Günümüz "bilgi toplumunda" bilgisayar, her alanda kendine yer edinmiş ve insana, birçok işlemde yardımcı olarak büyük kolaylıklar sağlamaktadır. İnsanın elle yaptığı veya yapmakta zorlandığı birçok işlemi (hesaplamalar, kayıtlar, kararlar, kontroller vb.) artık bilgisayar – temelde insan kontrollü olarak – gerçekleştirmektedir. Benzer şekilde artık birçok sürecin yürütülmesi, sistemin işletilmesi ve denetimi de bilgisayarla yapılmaktadır. Çok yüksek işlem hızları, doğruluk oranları, depolama kapasiteleri; otomasyon, güvenilirlik, çok yönlülük, düşük maliyetler, zengin bileşenler gibi özellikler günümüz bilgisayarlarının avantajlarından sadece birkaçıdır.

Hayatımızın her alanına giren bilgisayar, en kısa şekilde "*bilgi işleyen makine*" olarak tanımlanabilir. Biraz daha ayrıntılı tanımı "verileri alabilen, işleyebilen, saklayabilen ve istenilen biçimde sonuçları üreten elektronik veri işleme makinesidir" şeklinde yapılabilir. Daha geniş bir biçimde ise bilgisayar; verilen/alınan giriş değerlerini (verileri, bilgileri), belirtilen yöntemlerle işleyerek istenilen sonuçları üreten - gerektiğinde bunları depolayan/saklayan ve geri çağırabilen/yükleyen - karmaşık bir elektronik sistemdir (Şekil 1.1). Tanımdan da anlaşılacağı gibi bir bilgisayarın temel görevleri, aşağıdaki gibi özetlenebilir:

- Verileri (giriş) almak/okumak,
- Verileri/görevleri depolamak/saklamak ve gerektiğinde kullanmak,
- Verileri işlemek ve kullanılabilir/anlaşılabilir bilgiye dönüştürmek,
- Yeni (çıkış) veriler üretmek,
- Yukarıdaki tüm adımları denetlemek.



Şekil 1.1: Bilgisayardaki temel işlem akışı

Bilgisayarda işlenecek değerler (veriler, bilgiler) ve kullanılacak çözüm yöntemleri; bilgisayara, insan tarafından komutlarla (kodlarla, programlarla) bildirilir. "*Komut*"lar bilgisayarda (mikroişlemcide) çalıştırılabilen temel ifadeler olup bunların ilgili kurallara uygun olarak belirli bir yapıda/düzende, amaca yönelik ve anlamlı bir şekilde bir araya getirilerek düzenlenmesine de "*kodlama*" veya "*program yazma*" denir. Yani bilgisayar; problemleri çözebilmek (istenilen sonuçları üretebilmek) için "*program*"a (komut, kod veya kodlara) ihtiyaç duyar. Talimatlar kümesi olan "*program*"; giriş değerlerini kullanarak istenilen anlamlı çıkış değerlerinin elde edilebilmesi (problemlerin çözülebilmesi, işlemlerin gerçekleştirilebilmesi, istenilen hedeflere ulaşılabilmesi vb.) ve

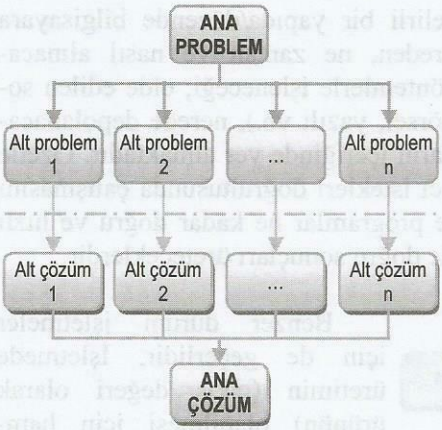
gerektiğinde bunların saklanması için belirli bir yapıda/düzende bilgisayara iletilen komutlar dizisidir. Verilerin nereden, ne zaman ve nasıl alınacağı/okunacağı; hangi amaçlar için ne tür yöntemlerle işleneceği; elde edilen sonuçların kullanıcıya nasıl ulaştırılacağı (görsel, yazılı vb.), nerede depolanacağı/saklanacağı gibi birçok husus programların içeriğinde yer almaktadır. Özetle programlar/kodlar; bilgisayarların, kullanıcı istekleri doğrultusunda çalışmasını sağlayan komutlar bütünüdür. Bu nedenle programlar ne kadar doğru ve hızlı ise, bilgisayarlar da o kadar hızlı bir şekilde doğru sonuçları üreteceklerdir.

Şekil 1.2: İşletme ve bilgisayardaki işlem akışı

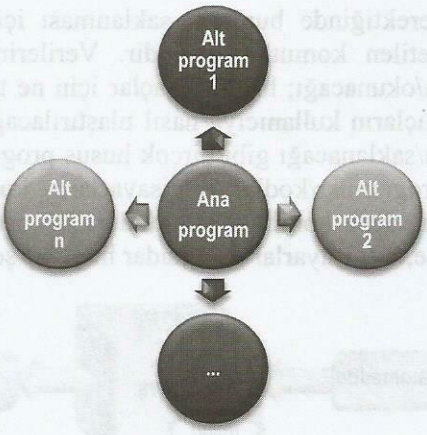
Benzer durum işletmeler için de geçerlidir. İşletmede üretimin (çıkış değeri olarak ürünün) olabilmesi için hammadde (giriş değeri) ve makine/teçhizata ihtiyaç duyulmaktadır (Şekil 1.2). Bilgisayarda olduğu gibi işletmede de işin merkezinde 'insan' vardır. Her iki türde de yapılacak işlemler/işler bilgisayara/makineye insan tarafından bildirilir.

Sonuç olarak bilgisayar, insanın komutlarla verdiği emirleri yerine getiren elektronik bir sistemdir. Kullanıcı; bilgisayara ne öğretirse, bilgisayardan ne talep ederse onun karşılığını almaktadır. Yani program; bilgisayara, arzu edilen sonuçları üretirken (problemi çözerken) hangi giriş verilerini, nasıl işleyeceğini ve bulduğu çıktıları ne yapacağını bildirir. Bunu bilgisayara öğretebilmek için de temel olarak problemin çözüm yolunun programcı tarafından bilinmesi gerekir. Bir problemin, birden çok çözüm yolu vardır. En uygun olanını seçmek, programcının yetenek ve tecrübesine bağlıdır. Eğer verilen programdaki çözüm yolu yanlış ise bilgisayarın bulacağı sonuç/sonuçlar da yanlış olacaktır. Dolayısıyla program yazılmadan önce; çözülecek problem iyice irdelenmeli, uygun çözüm yolu belirlenmeli ve bilgisayara doğru bir şekilde aktarılmalıdır. Eğer problem çok uzun ve karmaşık ise, alt kısımlara (bölümlere/parçalara) ayrılmalıdır. Her kısım ayrı ayrı yazılıp çalıştırılır ve herhangi bir hata yoksa tümü birleştirilir. Bu olaya programcılıkta "*böl ve üstesinden gel*" (*böl ve yönet, parçala ve fethet*) stratejisi (Şekil 1.3) denir. Anlaşılacağı üzere büyük programlar, "*ana program*" ve "*alt program*"lardan meydana gelir. Ana programa yardımcı işlemlerin yapıldığı program parçalarına, "*alt program*" denir. "*Ana program*" ise tüm alt programları denetleyen ve onları belirli düzende çağırıp çalıştıran ana koddur (Şekil 1.4).

Bilgisayar alanındaki bazı gelişmeler, Tablo 1.1'de özetlenmektedir.



Şekil 1.3: "Böl ve üstesinden gel" stratejisi



Şekil 1.4: Ana program ve alt programlar

Tablo 1.1: Bilgisayar alanındaki bazı tarihsel gelişmeler

Yıl	Bilgisayar	Açıklama
MÖ 2400	Abacus	İlk mekanik aritmetik hesaplama makinesi olan "abaküs", Babilliler tarafından icat edildi.
1613	Bilgisayar	Hesaplamaları gerçekleştiren insanı tanımlamak için ilk defa "bilgisayar" sözcüğü kayıtlarda görüldü.
1642	Pascaline	Fransız bilim adamı ve filozof Blaise Pascal (1623-1666) ilk pratik mekanik hesap makinesi olan Pascaline'ı icat etti.
1671, 1685	Leibniz makinesi (çarkı)	Alman matematikçi ve filozof Gottfried Wilhelm Leibniz (1646-1716), Pascaline'ye benzer ancak daha gelişmiş bir makine yapmıştır. 1685'te çarklı hesaplayıcıyı ilk kez tanımladı ve burada ilk bellek (kayıt) özelliği yer almaktadır. Ayrıca hesaplama alanına diğer önemli bir katkısı da ikili kodları ortaya koymasıdır.
1822	Difference engine	Birçoğunun "bilgisayarın babası" olarak kabul ettiği İngiliz matematikçi, filozof ve makine mühendisi Charles Babbage (1791-1871), sayı tablolarını hesaplayabilecek bir buharlı hesap makinesi üzerinde çalışmalar yürüttü. Birçoğuna göre ilk otomatik hesaplama makinesi olarak düşünülen ve birkaç sayı seti hesaplama ve sonuçların basılı kopyalarını oluşturma kapasitesine sahip bu "Difference Engine"ı kavramsallaştırarak geliştirmeye başladı.
1823	Si	İsveçli kimyager Baron Jöns Jacob Berzelius (1779-1848), işlemcilerin temel bileşeni/maddesi olan silikonu (Si) keşfetti.
1837	Analytical engine	Charlse Babbage; ilk mekanik bilgisayar olarak kabul edilen ve ilk genel amaçlı bilgisayar konsepti (aritmetik mantık unitesi, akış kontrolü, dâhili hafıza) olan "Analytical Engine"i önerdi.
1854	Boole cebri	İngiliz matematikçi, eğitimci, filozof ve mantıkçı George Boole (1815-1864), Boole cebrini ortaya koydu.
1911	IBM	1924 yılında ismi IBM (International Business Machines) olarak değiştirilen CTR (Computing-Tabulating-Recording) şirketi kuruldu.
1936	Turing makinesi	İngiliz matematikçi, bilgisayar bilimci ve kriptolog Alan Mathison Turing (1912-1954), hesaplama işlemleri için evrensel makine kavramını ortaya attı.
1936-38	Z1	Alman inşaat mühendisi Konrad Zuse (1910-1995) tarafından programlanabilirliği sınırlı, ancak ilk elektromekanik ikili programlanabilir ve ilk gerçekten işlevsel modern bilgisayar kabul edilen Z1'i geliştirdi.

1937-41	Atanasoff-Berry bilgisayarı(ABC)	Amerikalı fizik ve matematik profesörü John Vincent Atanasoff (1903-1995) ve lisansüstü öğrencisi Clifford Berry, 29 eşitliği eşzamanlı çözebilecek ilk sayısal bilgisayarı tasarladılar.
1940	CNC	1939 yılında Bell Laboratuvarlarında George Robert Stibitz (1904-1995) tarafından tasarlanan karmaşık sayı hesaplayıcısı (complex number calculator - CNC) 1940 yılında tanıtıldı.
1941	Z3	Konrad Zuse, Z3 ismindeki elektromekanik programlanabilen otomatik bilgisayarını geliştirdi.
1942	Z4	Kondrad Zuse, daha sonra ilk ticari bilgisayar olan Z4 üzerinde çalışmaya başladı.
1943	Colossus	İngiliz mühendis Thommy Flowers (1905-1998), II. Dünya Savaşı sırasında Almanların şifreli mesajlarını çözmek için ilk programlanabilir elektronik bilgisayarı tasarladı.
1943	Curta hesaplayıcı	Avusturyalı mühendis Curt Herzstark (1902-1988) mekanik cep hesap makinesi tasarladı.
1943-1944	ENIAC	John William Mauchly (1907-1980) ve John Presper Eckert (1919-1995) isimli profesörler, birçoğunun ilk sayısal genel amaçlı bilgisayar kabul ettiği ENIAC'ı geliştirdiler.
1944	Harvard Mark I (IBM ASCC)	Amerikalı matematikçi Howard Hathaway Aiken (1900-1973) tarafından ilk geniş-ölçekli sayısal bilgisayarı tasarladı.
1945	EDVAC	Macar-Amerikan matematikçi, fizikçi ve bilgisayar bilimci John von Neumann (1903-1957) tarafından ikili tabanı kullanan EDVAC geliştirildi.
1946	UNIVAC	John Mauchly ve Presper Eckert, UNIVAC ticari bilgisayarını geliştirdi.
1946	Williams tüpü	Frederic Calland Williams (1911-1977) tarafından 512-1024 bit arasında veri depolayabilen "Williams tüpü" geliştirildi.
1947	Tranzistör	Bell Laboratuvarından Amerikalı fizikçi William Shockley (1910-1989), fizikçi ve elektrik mühendisi John Bardeen (1908-1991) ve fizikçi Walter Houser Brattain (1902-1987) tranzistörü icat ettiler.
1949	Manchester Mark 1	Frederic Calland Williams (1911-1977), Tom Kilburn (1921-2001) ve Geoff Tootill (1922-) tarafından küçük-ölçekli deney makinesi (bilgisayar) geliştirildi ve 1948 yılında ilk defa bilgisayarda elektronik olarak saklanan bir program çalıştırıldı.
1951	Inkjet baskı	Inkjet baskı geliştirildi.
1953	HDD	IBM tarafından sabit disk sürücüsü icat edildi ve 1956 yılında ticari kullanımları başladı.
1953	701	IBM, ilk ticari bilimsel bilgisayar olan 701'i tanıttı.
1953	COBOL	Amerikalı bilgisayar bilimci ve ABD donanmasında amiral olan Grace Brewster Murray Hopper (1906-1992)) ilk programlama dilini (sonradan COBOL olarak bilinen) ve derleyiciyi geliştirdi.
1954	FORTRAN	"Formula Translation" kelimelerinden türetülmiş olan ve özellikle sayısl ve bilimsel hesaplamalara yönelik olan genel amaçlı programlama dili FORTRAN, IBM'de John Backus (1924-2007) tarafından geliştirildi.
1955	Whirlwind machine	MIT, manyetik çekirdekli RAM ve gerçek zamanlı grafikler içeren ilk sayısal bilgisayarı tanıttı.
1956	TX-O	MIT, ilk tranzistörlü bilgisayarı tanıttı.
1958	Chip	Amerikalı elektrik mühendisi Jack St. Clair Kilby (1923-2005) ve Fairchild Semiconductor (kuruluş 1957) ile Intel (kuruluş 1968)'in kurucularından olan ve "Silikon Vadisi Başkanı" lakaplı Robert Norton Noyce (1927-1990), "bilgisayar yongası" (chip) olarak bilinen entegreyi geliştirdi.
1960	Modem	AT&T veri iletimi için bilinen ilk modemi tanıttı.

1960	PDP-1	Digital Equipment Corporation ilk minibilgisayarı piyasaya sürdü.
1961	SpaceWar	Steve Russell (1937-) tarafından ilk bilgisayar oyunu programlandı.
1962	RS-232	EIA (Electronic Industries Association) tarafından RS-232 ortaya atıldı ve 1969 yılında RS-232-C seri iletişim standardı tanımlandı.
1963-64		Amerikalı mühendis Douglas Carl Engelbart (1925-2013), "insan-bilgisayar etkileşimi" alanının temellerini oluşturan grafiksel kullanıcı arayüzü ve fare (mouse) ile modern bilgisayar prototipini ortaya koydu.
1964	Programma 101	İtalyan elektrik mühendisi Pier Giorgio Perotto (1930-2002) tarafından geliştirilen ve Olivetti tarafından üretilen ilk masaüstü bilgisayar (PC) tanıtıldı.
1968	Intel	Robert Noyce ve Gordon Earle Moore (1929-) tarafından Intel Co. kuruldu.
1968	Nokta baskı	Nokta matris baskı tanıtıldı.
1968	HP 9100A	Hewlett Packard HP9100A masaüstü bilgisayarını piyasaya sürdü.
1969	Lazer yazıcı	Xerox, lazer yazıcı geliştirdi.
1969	AMD	AMD (Advanced Micro Devices) kuruldu.
1969	UNIX	Bell Laboratuvarlarında Ken Thomson (1943-), Dennis Ritchie (1941-2011), Brian Kernighan (1942-), Douglas McIlroy (1932-) ve Joe Ossanna (1928-1977) tarafından UNIX işletim sistemi geliştirildi.
1969	ARPANET	ARPANET üzerinden ilk başarılı mesaj iletildi.
1970	Centronics	"Centronics connector" (paralel iletişim) içeren Centronics Model 101 yazıcısı piyasaya sürüldü.
1970	DRAM	Intel, DRAM çipi olan Intel1103'ü geliştirdi.
1970-71	KENBAK-1	John V. Blankenbaker (1930-) tarafından 1970'te geliştirilen, Kenbak Co. tarafından üretilen ve birçok kişi tarafından ilk kişisel bilgisayar kabul edilen KENBAK-1 üretildi.
1971	ALOHAnet	1971'de ALOHAnet, Hawaii adalarını UHF kablosuz ağı ile bağladı. ALOHAnet ve ALOHA sırasıyla Ethernet ve IEEE 802.11 protokollerinin öncüleri oldular.
1971	Disket	Başında Amerikalı mühendis Alan Shugart (1930-2006)'ın bulunduğu IBM mühendisleri tarafından disket (floppy disk) geliştirildi.
1971-72		Intel dört bitlik (4004) ve sekiz bitlik (8008) işlemcileri üretti.
1973	Ethernet	Amerikalı elektrik mühendisi Robert Metcalfe (1946-) etherneti geliştirdi.
1973	Micral	Vietnamlı-Fransız mühendis André Truong Trong Thi (1936-2005) ve Fransız mühendis François Gernelle (1944-), ilk mikrobilgisayar kabul edilen Micral'i geliştirdiler.
1974	8080	Intel, ilk genel amaçlı mikroişlemcisi olan 8080 işlemcisini üretti.
1974	Xerox Alto	İlk workstation kabul edilen Xerox Alto geliştirildi.
1974-	PC	Kişisel bilgisayarlar piyasaya sunuldu.
1975	Microsoft	Bill Gates (1955-) ve Paul Gardner Allen (1953-) tarafından Microsoft kuruldu.
1975	Altair 8800	Amerikalı mühendis Ed Roberts (1941-2010), Altair 8800'ı tanıttığında "kişisel bilgisayar" ifadesini kullandı.
1975	IBM 5100	İlk taşınabilir bilgisayar kabul edilen IBM 5100 üretildi.
1976	Apple	Steve Jobs (1955-2011) ve Steve Wozniak (1950-) tarafından Apple kuruldu ve ilk bilgisayar modeli olan Apple I satışa sunuldu.
1976	Ethernet kablo	Ethernet kablosu geliştirildi.
1979	68000	Motorola 68000 16/32 bit işlemcisi geliştirildi ve daha sonraları Apple Macintosh ve Amiga bilgisayarlarında kullanıldı.

1981	MS-DOS	Tim Paterson (1956-) tarafından MS-DOS işletim sistemi geliştirildi.
1981	Planar	IBM "Planar", PC'lerde kullanılan ilk anakart oldu.
1981		MS-DOS işletim sistemini kullanan ilk IBM kişisel bilgisayarı tanıtıldı.
1981	Osborne 1	İngiliz-Amerikan yazar ve bilgisayar tasarımcısı Adam Osborne (1939-2003) ilk seri üretim mikroişlemci tabanlı ve birçok kişi tarafından gerçekten taşınabilir bilgisayar (laptop) olarak kabul edilen Osborne 1'i geliştirdi.
1982	CD-ROM	Philips ve Sony tarafından ortaklaşa geliştirilen CD'ler piyasaya sürüldü.
1983		Apple, grafiksel kullanıcı arayüzlü (ikon, açılır menü içeren) ilk kişisel bilgisayarı tanıttı.
1983	Virüs	Fred Cohen (1956-) ilk bilgisayar virüsünü yazdı/programladı.
1984	AT	IMB, AT form faktör anakartı sürdü.
1984	3D baskı	3D baskı geliştirildi.
1985	Windows	Microsoft, Windows işletim sistemini duyurdu.
1985	Alan adı	İlk alan adı kaydettirdi (symbolics.com).
1986	Gigabyte	Yeh Pei-Cheng tarafından Tayvan'da Gigabyte kuruldu.
1986		Sun firması tarafından SPARC (scalable processor architecture) geliştirildi.
1989	ASUS	T.H. Tung, Ted Hsu, Wayne Hsieh ve M.T. Liao tarafından Taipei'de ASUS kuruldu.
1990	HTML	İngiliz bilgisayar bilimci ve W3C'nin kurucusu Sir Timothy John Berners-Lee (1955-) HTML dilini geliştirerek HTTP protokolüyle sunucu ve istemci arasındaki iletişimi gerçekleştirdi.
1991	www	www başlatıldı.
1991		AMD, AM386 mikroişlemci ailesini tanıttı.
1991		Intel 486SX çipi tanıtıldı.
1991	WaveLAN	NCR Co. ile AT&T Co. ilk kablosuz ürünleri ve 802.11'in öncüsü sayılan WaveLAN'ı geliştirdiler.
1994	Webcam	Connectix, ilk ticari web kamerası olan QuickCam'i piyasaya sürdü.
1995	ATX	Intel ATX form faktör anakartı sürdü.
1995	DVD	Philips, Sony, Toshiba ve Panasonic tarafından geliştirildi.
1996	USB 1.0	Orijinal USB (Universal Serial Bus) 1.0 özellikleri tanımlandı.
1996	Google	Sergey Brin (1973-) ve Larry Page (1973-) tarafından google arama motoru geliştirildi.
1997		AMD, K6 işlemcisini (166-300 MHz) geliştirdi.
1997	IEEE 802.11	802.11 protokolünün birinci versiyonu yayınlandı.
1998		AMD, K6-2 işlemcisini (266-550 MHz) geliştirdi.
2000	USB 2.0	USB 2.0 özellikleri yayınlandı.
2003		AMD tarafından ilk 64 bitlik işlemci geliştirildi.
2005		AMD ilk çift-çekirdek işlemcisi Athlon 64 X2 3800+ geliştirdi.
2007		Intel, Core 2 Quad işlemcisi Q6600'ı piyasaya sürdü.
2008	i7	Intel, ilk Core i7 masaüstü işlemcilerini tanıttı.
2008	USB 3.0	USB 3.0 özellikleri yayınlandı.
2010		AMD, ilk Phenom II X6 (8/6 çekirdek) işlemcileri tanıttı.

Tablo 1.1'deki tarihsel bazı gelişmelerden de görüldüğü gibi bilgisayarlar, yapıları itibarıyla - farklı sınıflandırmalar olmasına rağmen - 5 kuşağa (nesil) ayrılabilirler (Şekil 1.5). Ayrıca hesaplama güçleri ve hızlarına göre de beş tür olarak incelenebilirler (Şekil 1.6). Bunun yanında "*tek yongalı (single-chip)*", "*tek kartlı (single-board)*" ve "*çok kartlı, veriyolu tabanlı (multiple-board, bus-based)*" şeklinde de sınıflandırmalar yapmak mümkündür.

Şekil 1.5: Bilgisayar kuşakları

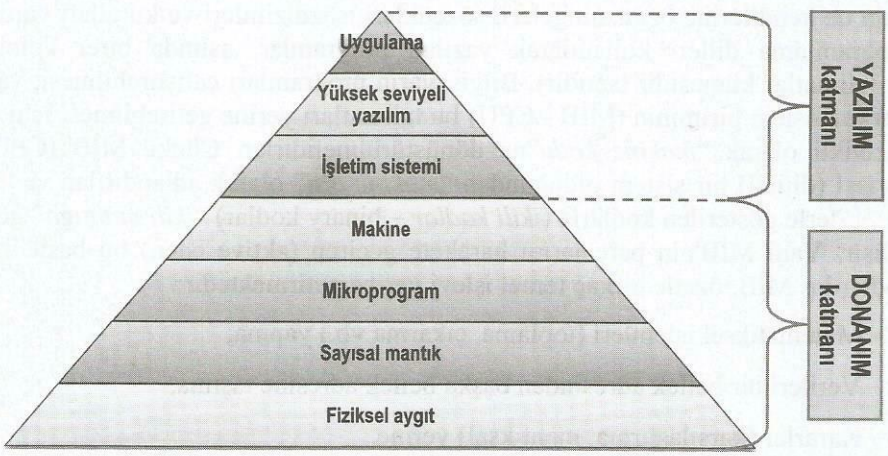
Şekil 1.6: Bilgisayarların genel sınıflandırması

Bilgisayar, temel olarak "*donanım*" ve "*yazılım*" olarak adlandırılan iki kısımdan oluşmaktadır (Şekil 1.7). Bilgisayarın fiziksel yapısı; yani içindeki işlemci, anakart, bellekler, ekran-ses-ağ kartı vb. elektronik aksamlar "*donanım*" kısmını teşkil eder. Donanım kısmının denetimini ve birbiriyle uyum içinde çalışmasını, amaçlar doğrultusunda kullanılmasını sağlayan her türlü kodlar bütününe (programlara) de "*yazılım*" denir. Yazılımlar da kendi içinde iki gruba ayrılabilir: "*sistem yazılımları*" ve "*uygulama yazılımları*". Donanım kısmın çalışmasını, yani bilgisayarın çalışmasını sağlayan işletim sistemleri gibi yazılımlar; "*sistem yazılımları*" olup kullanıcıların genel ve özel uygulama alanlarına yönelik geliştirilmiş olan paket programlar gibi yazılımlar da "*uygulama yazılımları*" grubuna girerler.

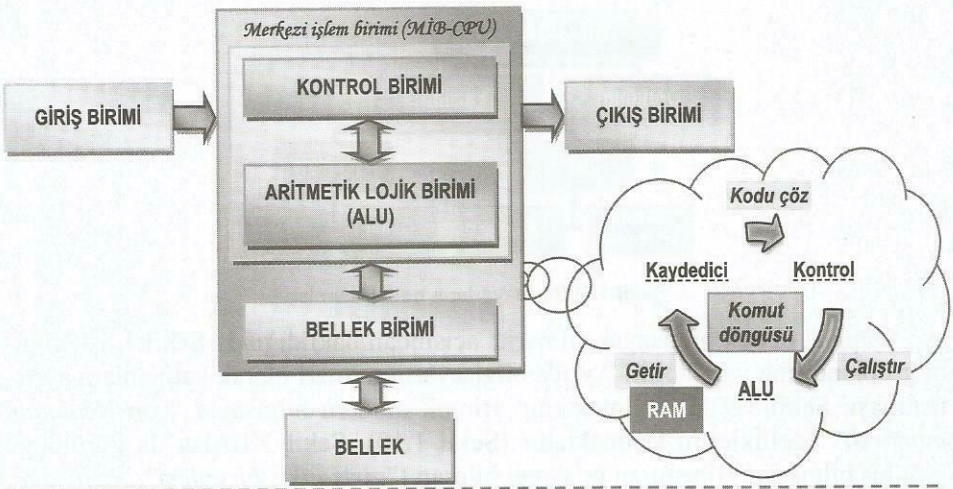
Şekil 1.7: Bilgisayarların genel kısımları

Yapısal olarak bilgisayar mimarisi açısından bakıldığında Şekil 1.8'deki gibi gruplandırma yapılabilir. Ayrıca bilgisayarın mimari olarak katmanlara ayrıştırılması, Şekil 1.9'da verilmektedir. Birçok modern bilgisayar "*von Neumann mimarisi*" özelliklerini taşımaktadır (Şekil 1.10). Şekil 1.10'dan da görüldüğü gibi bir bilgisayarı oluşturan üç temel bileşen ("*giriş-çıkış birimleri*", "*bellekler*" ve "*merkezi işlem birimi*") olup bunlar arasındaki iletişim de "*veriyolları*" ile sağlanmaktadır.

Şekil 1.8: Bilgisayar mimarilerinin genel sınıflandırması



Şekil 1.9: Bilgisayarlardaki mimari katmanlar



	Birim	Görevi	Bileşenler
	Giriş birimi	*Verileri almak*	Klavye, fare, mikrofon, kamera vb.
	Çıkış birimi	*Verileri göstermek*	Monitör, yazıcı, çizici, hoparlör vb.
CPU	*Aritmetik lojik birimi*	*İşlemleri yapmak*	İşlemci
CPU	*Bellek birimi (birincil/ana)*	*Veri ve komutları saklamak*	Kaydedici, cache, ROM (PROM, EPROM, EEPROM), RAM (SRAM, DRAM) vb.
CPU	*Kontrol birimi*	*Kontrol etmek/yönetmek*	
	Bellek (*ikincil*)	*Verileri saklamak*	HDD, FDD, CD-R/RW, DVD, USB bellek vb.

Şekil 1.10: Bilgisayarların genel organizasyonu

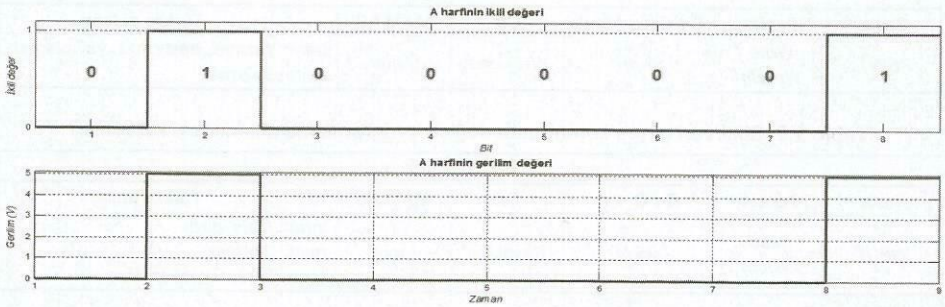
II. PROGRAMLAMA DİLLERİ

"*Programlama dili*"; programcı ile bilgisayar arasındaki iletişimi sağlayan ve programların, komut dosyalarının vb. yazımında kullanılan bir notasyondur. Doğal dillerin (konuşma, yazma) olduğu gibi bilgisayardaki programlama dillerinin de kendilerine özgü simgeleri, sözcükleri, sözdizimleri ve kuralları vardır. Programlama dilleri kullanılarak yazılan programlar, aslında birer komutlar/talimatlar kümesidir (setidir). Bilgisayarın programları çalıştırabilmesi, yani merkezi işlem biriminin (MİB - CPU) bu talimatları yerine getirebilmesi için de öncelikli olarak "*makine kodu*"na dönüştürülmelidirler. Çünkü MİB (CPU), sayısal (dijital) bir sistem olduğundan "*makine dili*" olarak adlandırılan ve "0" ile "1"lerle gösterilen kodlarla (*ikili kodlar* – binary kodlar) "*ikili mantığa*" göre çalışır. Yani MİB'nin parçalarını harekete geçiren (aktive eden) bu basit ikili kodlardır. MİB, özetle birkaç temel işlevi yerine getirmektedir:

- Matematiksel işlemleri (toplama, çıkarma vb.) yapma,
- Verileri bir bellek adresinden başka bellek adresine taşıma,
- Kararlar (karşılaştırma, mantıksal) verme,
- Aldığı kararlara göre yeni komutlar/talimatlar kümesine gitme/atlama.

Sayısal elektronikte "0" gerilim/akım yok, "1" ise gerilim/akım var anlamlarında kullanılır. "0" ve "1"lerin her birine "*bit*" denir. Programların yazımında da kullanılan herbir karakterin (harf, rakam, simge vb.) tamsayı karşılığı vardır (ASCII kodları, bkz. Ek-A). Örneğin "A" harfinin tamsayı karşılığı 65 olup pozitif mantığa göre 1 byte (8 bit)'la ifade edilmesi ve gerilim dalga biçimi, temel olarak Şekil 1.11'deki gibi olacaktır.

$$"A" \rightarrow (65)_{10} = (01000001)_2$$



Şekil 1.11: "A" harfinin örnek gösterimleri/değerleri

Not

1 nibble = 4 bit
1 byte = 8 bit
1 kB = 2^{10} byte = 1024 byte
1 MB = 1024 kB = 2^{20} byte = 1048576 byte
1 GB = 1024 MB = 2^{30} byte = 1073741824 byte
1 TB = 1024 GB = 2^{40} byte = 1099511627776 byte
1 PB = 1024 TB = 2^{50} byte = 1125899906842624 byte
1 EB = 1024 PB = 2^{60} byte = 1152921504606846976 byte

Bilgi

ASCII (*American Standard Code for Information Interchange - Bilgi değişimi için Amerikan standart kodu*), İngiliz/Latin alfabesi tabanlı 7 bitlik karakter kodlama sistemidir. ANSI (*American National Standards Institute* - Amerikan Ulusal Standartlar Enstitüsü) tarafından standartlaştırılan bu karakter/kod kümesinde/setinde/tablosunda 2^7=128 adet karakter ve bunların tamsayı karşılıkları yer almaktadır. Karakterlerden 33 tanesi (10 tabanında tamsayı karşılıkları 0-31,127 olan) yazdırılamayan (ENTER, ESC, DEL vb.) kontrol karakterleri olup 95 tanesi (10 tabanında tamsayı karşılıkları 32-126 olan) ise yazdırılabilen türdendir. 10 tabanında tamsayı karşılıkları 128-255 olan karakterler de eklenerek Genişletilmiş ASCII karakter/kod kümesi/seti/tablosu oluşturulmuştur.

Temelde "*makine tarafından okunabilen*" ve "*insan tarafından okunabilen*" diye ayrılan programlama dilleri üç grupta incelenebilmektedir:

a. *Düşük seviyeli diller*: Makine dillerini içerir.

b. *Orta seviyeli diller*: Makine dilinden biraz daha gelişmiş dilleri (Assembly vb.) içerir.

c. *Yüksek seviyeli diller*: Konuşma dillerine yakın dilleri (Fortran, Basic, Pascal, C, C++, Visual Basic, C#, Delphi, Java vb.) içerir.

Yüksek seviyeli programlama dillerindeki komutlar, genel olarak günlük konuşma dilindeki İngilizce kelimelerden türetilmiştir. Örneğin, ekrana veri/bilgi/yazı yazdırmak için kullanılan komutun değişik programlama dillerindeki karşılıkları Tablo 1.2'de; veri/bilgi girme komutunun değişik programlama dillerindeki karşılıkları ise Tablo 1.3'te verilmektedir.

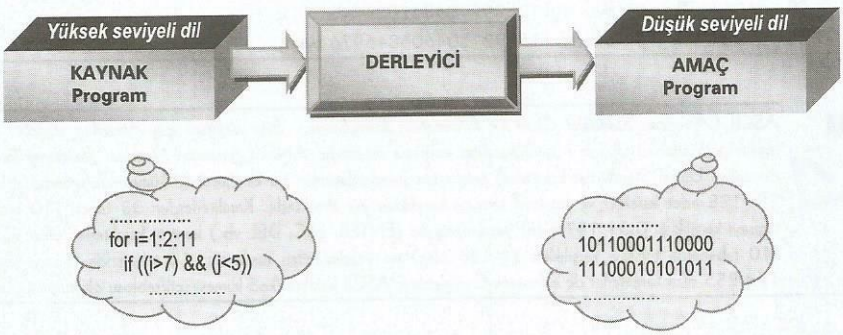
Tablo 1.2: Ekrana yazı yazdırma komutunun farklı programlama dillerindeki karşılıkları

"Yaz" komutu					
Basic	**Pascal**	**C / C ++/Java**	**C#**	**MATLAB**	**Türkçe anlamı**
PRINT	*Write* *Writeln*	*printf*	WriteLine	fprintf	print: Yazmak, bastırmak, yayınlamak write: Yazmak

Tablo 1.3: Bilgi/veri girme komutunun değişik programlama dillerindeki karşılıkları

" Gir" komutu					
Basic	**Pascal**	**C / C ++**	**C#**	**MATLAB**	**Türkçe anlamı**
INPUT	*Read* *Readln*	*scanf*	ReadLine	input	input: Giriş, girdi read: Okumak scan: Taramak, okumak

Programcıların, kodları yazarken 0-1'lerle makine dilinde yazması çok zordur. Bu nedenle yüksek seviyeli ve konuşma diline yakın olan programlama dilleriyle programlar yazılmakta, "*derleyici*" (compiler) ve/veya "*yorumlayıcı*" (interpreter) programlarla da "*makine dili*"ne dönüştürülmektedir, yani çalıştırılabilir hale getirilmektedir (Şekil 1.12). Yüksek seviyeli programlama diliyle yazılmış "*kaynak programın*" (kaynak kodun) içeriğinin yeniden düzenlenme imkânı varken, makine diline dönüştürülmüş ve çalışmaya hazır "*amaç programın*" (makine kodunun) içeriğine müdahale imkânı yoktur.



Şekil 1.12: Yüksek/düşük seviyeli dillerdeki program dönüşümü

Programlar yazılırken genelde iki tür hata yapılır (Şekil 1.13):

a. *Yazım hatası*: Programda yer alan komutların, ifadelerin, sözdizimlerinin vb. ilgili dilin kurallarına uygun olarak yazılmaması durumunda ortaya çıkan hata türüdür. Programın çalıştırılmadan önce derlenmesi esnasında, derleyici bu tür hataları tespit ederek programcıyı uyarmaktadır. Bu nedenle, yazım hataları tehlikeli olmayıp kolaylıkla düzeltilebilir. Örneğin C

programlama dilinde, programcı/yazılımcı "`printf`" komutu yerine yanlışlıkla "`prindf`" yazarsa, derleyici bunu yakalayıp programcıyı uyaracaktır.

b. *Mantık hatası*: İlgili programlama dilinin kurallarına uygun olarak yazılmasına rağmen çözüm yolunda verilen yanlış yönlendirmelerin, program tasarımdaki yanlışlıkların oluşturduğu hata türüdür. Eğer problemin çözüm yolu yanlış aktarılırsa, program herhangi bir hata vermeden çalışacak ancak yanlış sonuçlar üretecektir. Çok dikkat edilmesi gereken hata türüdür. Örneğin programda hesaplanacak matematiksel fonksiyon "`y=3*x+1`" şeklinde olması gerekirken kodlama esnasında yanlışlıkla "`y=2*x+1`" şeklinde yazılırsa program çalışacak, ancak doğru sonuçlar üretmeyecektir.

Hatalar, yakalandıkları/tespit edildikleri an açısından bakıldığında da iki gruba ayrılabilirler (Şekil 1.14):

a. *Derleme zamanı hatası*: Programın derlenmesi sırasında, derleyici tarafından tespit edilen yazım hataları gibi hatalardır.

b. *Çalışma(çalıştırma) zamanı hatası*: Derleyici tarafından algılanamayan ve programın çalıştırılması esnasındaki hatalardır. Örneğin olmayan değerin çağırılması, var olmayan dosyanın açılmaya çalışılması, paydada yer alan değişkene sıfır girilmesi (sıfıra bölme hatası) vb.

Şekil 1.13: Programlama hataları

Şekil 1.14: Tespit anları açısından hatalar

III. İŞLEMLER

Bilgisayarlardaki işlemler, temel olarak üç gruba ayrılırlar (Şekil 1.15):

a. *Matematiksel (aritmetik) işlemler*

b. *Karşılaştırma (karar) işlemleri*

c. *Mantıksal (lojik) işlemler*

Şekil 1.15: Bilgisayardaki işlemler

A. Matematiksel İşlemler

Temel aritmetik işlemler (toplama, çıkarma, çarpma, bölme); üstel, logaritmik, trigonometrik, hiperbolik vb. fonksiyonlar bilgisayarlardaki bu işlem grubunda yer alırlar.

Ondalıklı (gerçel) sayıların tam ve ondalıklı kısımlarını ayırmak için matematikte "," (virgül) karakteri (simgesi) kullanılırken; bu tür sayıların bilgisayar/algoritma dilinde tam ve ondalıklı kısımlarını ayırmak için "." (nokta) kullanılır. Sayının önünde herhangi bir simge yok ise pozitif, "–" varsa negatif kabul edilir. Tablo 1.4'te matematiksel işlemler ve bilgisayar dilindeki veya algoritma/akış diyagramlarındaki karşılıkları, Tablo 1.5'te ise örnek matematiksel işlemler görülmektedir.

Tablo 1.4: Matematiksel işlemler ve karşılıkları

İşlem	Matematik	Algoritma/akış diyagramı/kod
Toplama	a + b	a + b
Çıkarma	a - b	a - b
Çarpma	a . b	a * b
Bölme	a ÷ b	a / b
Üs alma	a^b	a ^ b

Tablo 1.5: Örnek matematiksel işlemler

a	b	a^b	a/b	a*b	a+b	a-b
-2	*6*	64	-0.33333	-12	4	-8
-1	*7*	-1	-0.14286	-7	6	-8
0	*8*	0	0	0	8	-8
1	*7*	1	0.142857	7	8	-6
2	*6*	64	0.333333	12	8	-4

Bilgisayar dilinde yazılmış/kodlanmış matematiksel ifadelerde, "*işlem öncelik sıraları*" bulunmaktadır. Yani herhangi bir ifadede; bilgisayar, bazı işlemleri önce, bazılarını daha sonra yapmaktadır. Dolayısıyla matematiksel ifadeler; program yazımlarında, bilgisayar mantığına göre yeniden düzenlenmelidirler. Matematiksel işlem öncelik sıraları çok önemlidir ve bilgisayar diline aktarılırken (kodlanırken) dikkatli olunmalıdır (Tablo 1.6).

Tablo 1.6: Matematiksel işlemlerde öncelik sıraları

Matematiksel işlemlerde öncelik sıraları		
Sıra	**İşlem**	**Bilgisayar dili**
1	Sayıların negatifliği	- ...
2	Parantezler	(...)
3	Matematiksel fonksiyonlar	`cos , sin , ln , log , exp ,...`
4	Üs alma	`a^b , pow , ...`
5	Çarpma ve bölme	a*b ve a/b
6	Toplama ve çıkarma	a+b ve a-b

Not

Bilgisayarın; aynı önceliğe sahip işlemleri gerçekleştirme sırası, soldan sağa (baştan sona) doğrudur.

Örneğin

```
Y=A*B/C
```

ifadesinde aynı önceliğe sahip çarpma ve bölme işlemleri yer almaktadır. Burada önce " `A*B` " çarpımı yapılır ve bulunan sonuç `C` 'ye bölünür. Benzer şekilde

```
Y=A+B-C+D
```

ifadesinde, bilgisayar sırasıyla `A` ile `B`'yi toplayacak, bulunan toplamdan `C`'yi çıkaracak ve elde ettiği farka `D` ekleyecektir.

Örnek-1.1: Tablo 1.7'de değişik matematiksel ifadelerin bilgisayar/algoritma dilindeki kodlamaları verilmektedir. Verilen tabloda özellikle parantezlerin yerlerine dikkat ediniz. "*Açılan parantezlerin kapatılması*" gerekliliğini gözden kaçırmayınız. Kolaylık olması açısından ifadeleri kodlarken ilgili parantezleri açıp kapatarak içlerine ifadeleri yazabilirsiniz veya ifadeleri sırasıyla parantezleri açarak yazabilir, daha sonra da ilgili parantezleri kapatabilirsiniz.

Not

İşlem önceliğini değiştirmediği sürece; ifadelerde fazladan parantez olması, herhangi bir mantıksal sorun oluşturmamaktadır.

Tablo 1.7: Matematiksel ifadelerin bilgisayar dilinde kodlanması

Matematiksel ifade	Bilgisayar/algoritma dilinde kodlanması
$a+b-c+2abc-7$	a+b-c+2*a*b*c-7
$a+b^2-c^3$	a+b^2-c^3
$a-\frac{b}{c}+2ac-\frac{2}{a+b}$	a-b/c+2*a*c-2/(a+b)
$\sqrt{a+b}-\frac{2ab}{b^2-4ac}$	(a+b)^(1/2)-2*a*b/(b^2-4*a*c)
$\frac{a+b-c}{\sqrt{a^2+b^3}}-\frac{2(ab+ac+bc)}{9}$	(a+b-c)/((a^2+b^3)^(1/2))-2*(a*b+a*c+b*c)/9
$\frac{a^2+b^2}{2ab}$	(a^2+b^2)/(2*a*b)
$a+\frac{b+c^2}{d+\frac{e-f}{3a}}$	a+(b+c^2)/(d+(e-f)/(3*a))
$\frac{\frac{a+b}{c-d}+\frac{3a}{7b-3a}}{\frac{a+2b-3c+4d}{a+b+c+d}}$	((a+b)/(c-d)+3*a/(7*b-3*a))/((a+2*b-3*c+4*d)/(a+b+c+d))
$\frac{a^5+\frac{b^3}{c+d}}{\sqrt[4]{a+b^2+c^3+d^4}}$	(a^5+b^3/(c+d))/((a+b^2+c^3+d^4)^(1/4))
$\sqrt[3]{a^2+b^2}+\frac{abc}{a+\frac{b}{\sqrt{c-\frac{b}{a^3}}}}$	(a^2+b^2)^(1/3)+a*b*c/(a+b/((c-b/a^3)^(1/2)))

$a+\sqrt[7]{abc}-\cfrac{1}{1+\cfrac{1}{a+\cfrac{1}{b+\cfrac{1}{c+\cfrac{1}{abc}}}}}$	a+(a*b*c) ^ (1/7)-1/(1+1/(a+1/(b+1/(c+1/(a*b*c)))))
$\sqrt[3]{\cfrac{\sqrt[5]{a-b}}{\sqrt[4]{a+b-\cfrac{c}{ab}}}}+\cfrac{1}{\sqrt{1+\cfrac{1}{\sqrt{1+\cfrac{1}{abc}}}}}$	((a-b) ^ (1/5)/(a+b-c/(a*b)) ^ (1/4)) ^ (1/3)+1/((1+1/((1+1/ (a*b*c)) ^ (1/2))) ^ (1/2))
$\sqrt[3]{1+\cfrac{a+\cfrac{1}{a+1}}{1+\sqrt{\cfrac{a-1}{a^2+\sqrt{a+1}}}}}$	(1+(a+1/(a+1))/(1+((a-1)/(a ^ 2+(a+1) ^ 0.5)) ^ 0.5)) ^ (1/3)
$\sqrt[5]{a+\cfrac{1+\cfrac{1}{\sqrt[3]{a+1}}}{1+\sqrt{\cfrac{a}{\sqrt{a^3+1}}}}}$	(a+(1+1/(a+1) ^ (1/3))/(1+(a/(a ^ 3+1) ^ .5) ^ .5)) ^ (1/5)
$\sqrt[5]{\cfrac{1}{a+1}+\cfrac{1+\cfrac{a}{\sqrt[3]{a^2+1}}}{1+\sqrt{\cfrac{2a}{a^3+1}}}}$	(1/(a+1)+(1+a/(a ^ 2+1) ^ (1/3))/(1+(2*a/(a ^ 3+1)) ^ 0.5)) ^ (1/5)
$\sqrt[5]{a^2+\cfrac{b+\cfrac{a-1}{\sqrt[4]{1+a}}}{\sqrt{\cfrac{a}{1+a^2}}}}$	(a ^ 2+(b+(a-1)/(1+a) ^ (1/4))/(a/(1+a ^ 2)) ^ (1/2)) ^ (1/5)
$\sqrt[4]{a^2+\sqrt{a+\cfrac{a+b}{\sqrt[3]{1+\cfrac{1}{a+b}}}}}$	(a ^ 2+(a+(a+b)/(1+1/(a+b)) ^ (1/3)) ^ .5) ^ .25
$\sqrt[5]{\cfrac{a^2}{a-b}+\cfrac{a+b}{\sqrt{\cfrac{b}{b^2-a^2}}}}$	(a ^ 2/(a-b)+(a+b)/(b/(b ^ 2-a ^ 2) ^ (1/2))) ^ .2
$\sqrt[3]{a^2+\cfrac{a+b}{1+\cfrac{1}{\sqrt{\cfrac{a}{a^2+b^2}}}}}$	(a ^ 2+(a+b)/(1+1/(a/(a ^ 2+b ^ 2)) ^ .5)) ^ (1/3)

Örnek-1.2: a=4, b=6, c=8, d=10 değerleri için bilgisayar/algoritma dilinde kodlanmış aşağıdaki üç ifadenin sonuçlarını bulunuz.

İfade	Sonuç
c*d/(a*d)+b+c*d/a	28
c*d/a*d+b+c*d/a	226
c*d/a*d+(b+c)*d/a	235

İfadeler matematiksel olarak aynı gibi görünse de, bilgisayar diline kodlanırken parantez farkı oluşturulmuştur. Birinci ifadede - işlem öncelik sırası parantezlerde olduğu için - ilk önce `a*d` işlemi yapılır: `(a*d)=4*10=40`

Daha sonra çarpma ve bölme işlemleri yapılır:

`c*d/(a*d)=8*10/40=80/40=2, c*d/a=8*10/4=80/4=20`

En sonunda da toplama işlemi yapılır: `c*d/(a*d)+b+c*d/a=2+6+20=28`

İkinci ifadede; birinci işlem teriminde 'c' ile 'd' çarpılmakta ve 'a' ya bölünerek bulunan sonuç tekrar 'd' ile çarpılmaktadır:

`c*d/a=8*10/4=80/4=20` ve `c*d/a*d=20*10=200`

Daha sonra `c*d/a=8*10/4=80/4=20` işlemi yapılmakta ve sonuçlar toplanmaktadır: `c*d/a*d+b+c*d/a=200+6+20=226`

Üçüncü ifadede ise önce parantez içindeki toplama `(b+c)=6+8=14` işlemi yapılır ve bulunan bu değer 'd' ile çarpılıp 'a' ya bölünür:

`(b+c)*d/a=14*10/4=140/4=35`

Aynen ikinci ifadede olduğu gibi birinci çarpım terimi

`c*d/a*d=8*10/4*10=80/4*10=20*10=200`

olarak bulunarak sonuçlar toplanır: `c*d/a*d+(b+c)*d/a=200+35=235`

Açıklamalardan görüldüğü gibi; benzer görünen üç ifade, değişik parantez kodlamasıyla farklı sonuçlar vermektedir. İfadelerin matematiksel karşılıkları;

Bilgisayar dilindeki ifade	Matematiksel karşılığı
`c*d/(a*d)+b+c*d/a`	$\frac{cd}{ad}+b+\frac{cd}{a}=\frac{c}{a}+b+\frac{cd}{a}$
`c*d/a*d+b+c*d/a`	$c\frac{d}{a}d+b+\frac{cd}{a}=\frac{cd^2}{a}+b+\frac{cd}{a}$
`c*d/a*d+(b+c)*d/a`	$c\frac{d}{a}d+\frac{(b+c)d}{a}=\frac{cd^2+(b+c)d}{a}$

şeklindedir. Bu örnekte, matematiksel ifade kodlamanın (özellikle parantezlerin yerinin) ne kadar önemli olduğu açıkça görülmektedir.

Örnek-1.3: *a=1, b=2, c=3, d=4, e=-2* değerlerine göre aşağıdaki dört ifadenin sonuçlarını bulunuz.

İfade	Sonuç
`a+d/b+d^2+2*a*b*c/d+e`	*20*
`(a+b)/c+d^2+2*a*b*c/(d+e)`	*23*
`a+b/(c+d)^2+2*a*b*c/d+e`	*2.04*
`(a+b)/(c+d)^2+2*a*b*c/(d+e)`	*6.06*

Aynı gibi görünen bu ifadeler, farklı sonuçlar vermektedir. İşlem adımları karşılaştırmalı olarak Tablo 1.8'de özetlenmektedir.

Tablo 1.8: Örnek-1.3'te verilen ifadeler için işlem adımları

Adım	1. ifade	2. ifade	3. ifade	4. ifade
1	d^2=16	a+b=3 , d+e=2	c+d=7	a+b=3 , c+d=7 , d+e=2
2	2*a*b*c/d=3 , d/b=2	d^2=16	(c+d)^2=49	(c+d)^2=49
3	a+d/b+d^2+2*a*b*c/d +e=20	(a+b)/c=1 , 2*a*b*c/(d+e)=6	b/(c+d)^2=0.04 , 2*a*b*c/d=3	(a+b)/(c+d)^2=0.06 , 2*a*b*c/(d+e)=6
4	---	1+16+6=23	1+0.04+3-2=2.04	0.06+6=6.06

Örnek-1.4: *a=-1, b=7, c=4, d=2* ve *e=6* değerlerini için aşağıdaki üç ifadenin sonuçlarını hesaplayınız.

İfade	Sonuç
a+b+c/d+d-e^2+c+e+d/(b-e)^(1/2)	*-14*
(a+b+c)/d+(d-e)^2+(c+e+d)/((b-e)^(1/2))	*33*
(a+b)+c/d+d-e^2+c+e+d/b-e^1/2	*-18.72*

Tablo 1.9: Örnek-1.4 için işlem adımları

Adım	1. ifade	2. ifade	3. ifade
1	b-e=1	a+b+c=10 , d-e=-4 , c+e+d=12 , b-e=1	a+b=6
2	e^2=36 , (b-e)^(1/2)=1	(d-e)^2=16 , (b-e)^(1/2)=1	e^2=36 , e^1=6
3	c/d=2 , d/(b-e)^(1/2)=2	(a+b+c)/d=5 , (c+e+d)/((b-e)^(1/2))=12	c/d=2 , d/b=0.28 , e^1/2=3
4	-1+7+2+2-36+4+6+2=-14	5+16+12=33	6+2+2-36+4+6+0.28-3=-18.72

Örnek-1.5: *A=8, B=8, C=2, D=4* ve *E=2* değerleri için Tablo 1.10'daki ifadelerin sonuçlarını bulunuz.

Tablo 1.10: Örnek-1.5 için ifadeler ve sonuçları

İfade	Sonuç
(A+B)/C*E^2+(A+D)/(C^2+E)	34
A+B/C*E^2+A+D/C^2+E	35
A+B/(C*E^2)+(A+D)/(C^2+E)	11
(A+B)/C*(E^2)+A+D/(C^2)+E	43
(A+B)/(C*E^2)+(A+D)/(C^2+E)	4

Örnek-1.6: *A=9* ve *B=16* değerleri için Tablo 1.11'deki ifadelerin sonuçlarını bulunuz.

Tablo 1.11: Örnek-1.6 için ifadeler ve sonuçları

İfade	Matematiksel eşdeğeri	Sonuç
A+B^1/2	$A+\frac{B^1}{2}$	$9+\frac{16^1}{2}=9+\frac{16}{2}=9+8=17$
A+B^(1/2)	$A+\sqrt{B}$	$9+\sqrt{16}=9+4=13$
(A+B)^1/2	$\frac{(A+B)^1}{2}$	$\frac{(9+16)^1}{2}=\frac{25^1}{2}=\frac{25}{2}=12.5$
(A+B)^(1/2)	$\sqrt{A+B}$	$\sqrt{9+16}=\sqrt{25}=5$

Örnek-1.7: Tablo 1.12'deki ifadelerin sonuçlarını elde ediniz.

Tablo 1.12: Örnek-1.7 için ifadeler ve sonuçları

Değerler	Bilgisayar/algoritma yazımı	Sonuç
a=2 , b=16 , c=9	a+b/a ^ 1/2+1/1/1/1/a+2*(b+c)/(b+c) ^ (0.5)	16.5
a=25 , b=4 , c=12	100/a ^ 1/2+1/1/1/b*c+2*(b+c) ^ 0.5-(c-b) ^ 1/2	9
a=25 , b=9 , c=16	100/1/a ^ 1/2+1/1/2*b*c+(b+c) ^ 0.5-b-c ^ 1/2	62
a=50 , b=25 , c=3 , d=2	a/b ^ 1/2+1/1/1/1/b ^ (1/2)+a/b*c*d ^ 4/d	49.2
a=32 , b=4 , c=3	(a/1/1/b ^ 1/2) ^ (1/2)+a/1/b*c ^ 1/2	14
a=64, b=32 , c=16	a+b/1/c ^ 1/2+b/1/1/c ^ (1/2)+a/b*c ^ 1/2	89
a=64 , b=16	a/1/b ^ 1/2+a/1/1/1/b ^ (1/2)+a/2*b ^ 1/2	274

Yukarıdaki örneklerin çözümünü bilgisayarın iki tabanında (binary) gerçekleştirdiğini unutmayınız. İki bit arasındaki temel matematiksel işlemler, Tablo 1.13'te özetlenmektedir.

Tablo 1.13: İki tabanındaki temel matematiksel işlemler

Toplama (A+B)

A	B	Elde	Toplam
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Çıkarma (A-B)

A	B	Borç	Fark
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

Çarpma (A*B)

A	B	Çarpım
0	0	0
0	1	0
1	0	0
1	1	1

Örnek-1.8: `A=12` ve `B=4` değerleri için iki tabanında toplama, çıkarma, çarpma ve bölme işlemlerini yapınız.

$A = (12)_{10} = (1100)_2$ ve $B = (4)_{10} = (100)_2$

Tablo 1.14: Örnek-1.8 için işlem adımları ve sonuçları

$A+B$	$A-B$	$A*B$	A/B
1100 + 0100 10000	1100 − 0100 1000	1100 x 100 0000 0000 + 1100 110000	1100 | 100 −100 | 11 100 −100 0
$12+4=16$	$12-4=8$	$12*4=48$	$12/4=3$
$(10000)_2$	$(1000)_2$	$(110000)_2$	$(11)_2$

B. Karşılaştırma İşlemleri

Bilgisayar; temel matematiksel işlemlerin yanında karşılaştırma/karar işlemlerini de yapabilmektedir. Yani iki büyüklükten hangisinin büyük veya küçük olduğu, iki değişken değerinin birbirine eşit olup olmadığı, alfabetik olarak önce veya sonra mı olduğu gibi konularda karar verebilmektedir. Karşılaştırma işlemleri olarak adlandırılan bu işlemler, sayısal ve alfasayısal verilere/bilgilere

uygulanabilmekte ve algoritma/akış diyagramlarındaki genel kullanım sembolleri (karakterleri, simgeleri) toplu halde Tablo 1.15'te verilmektedir.

Tablo 1.15: Karşılaştırma işlemleri

İşlem sembolü	Anlamı
=	Eşittir
<>	Eşit değildir
>	Büyüktür
<	Küçüktür
>= veya =>	Büyük eşittir
<= veya =<	Küçük eşittir

Örnek-1.9: Aşağıdaki algoritmanın çalışmasını inceleyiniz.

```
1. Başla.
2. A sayısını al/oku/gir
3. B sayısını al/oku/gir
4. Eğer A>B ise Yaz "A sayısı, B sayısından büyüktür."
5. Eğer A<B ise Yaz "B sayısı, A sayısından büyüktür."
6. Eğer A=B ise Yaz "İki sayı birbirine eşittir."
7. Dur
```

Verilen algoritma ile bilgisayara klavyeden iki sayı girilmekte (A ve B sayıları) ve bunlar birbiriyle karşılaştırılmaktadır. Karşılaştırma sonuçlarına göre; uygun mesajlar, ekrana yazdırılıp program sona ermektedir. Tablo 1.16'da, girilen değişik sayılarla oluşan program çıktıları verilmektedir.

Tablo 1.16: Örnek-1.9'daki algoritmanın uygulama sonuçları

Girilen A sayısı	Girilen B sayısı	Ekrana yazılan sonuç
3	7	B sayısı, A sayısından büyüktür.
33	11	A sayısı, B sayısından büyüktür.
99	99	İki sayı birbirine eşittir.

Örnek-1.10: Aşağıdaki algoritmanın çalışmasını inceleyiniz.

```
1. Başla.
2. Bir büyük harf veya kelime (A) al/oku/gir
3. İkinci büyük harfi veya kelimeyi (B) al/oku/gir
4. Eğer A>B ise Yaz "1. girilen, alfabetik olarak daha sonra."
5. Eğer A<B ise Yaz "1. girilen, alfabetik olarak daha önce."
6. Eğer A=B ise Yaz "Girilenler aynı"
7. Dur
```

Verilen algoritmada; klavyeden iki karakter veya kelime girilmekte ve birbiriyle karşılaştırılmaktadır. Aslında bu karşılaştırma; karakterler arasında değil, karakterlerin ASCII kodları arasındadır. Örneğin klavyeden "A" karakterine basıldığında; bilgisayar ASCII kod tablosundaki karşılığı olan 65 tamsayısının ikili tabanıyla işlemlerini gerçekleştirir (Tablo 1.17).

Tablo 1.17: Örnek-1.10'daki algoritmanın harf için uygulama sonuçları

1.girilen	2.girilen	1.girilenin ASCII kodu	2.girilenin ASCII kodu	Ekrana yazılan mesaj
B	C	66	67	1. girilen, alfabetik olarak daha önce.
D	A	68	65	1. girilen, alfabetik olarak daha sonra.

Eğer iki kelime girilir ve bu kelimelerin ilk karakterleri (harfleri) aynı ise, ikinci harfleri karşılaştırılır. İlk iki harf aynı ise, üçüncü harfler karşılaştırılır. Yani eşitlik durumunda; ilk farklı karaktere kadar – sözlüklerde olduğu gibi – karşılaştırma devam eder (Tablo 1.18).

Tablo 1.18: Örnek-1.10'daki algoritmanın kelime için uygulama sonuçları

1.girilen	2.girilen	1.girilenin ASCII kodu	2.girilenin ASCII kodu	Ekrana yazılan mesaj
ANA	ANKARA	65-78-**65**	65-78-**75**	1. girilen, alfabetik olarak daha önce.
BURSA	BURADA	66-85-82-**83**	66-85-82-**65**	1. girilen, alfabetik olarak daha sonra.

Not

Büyük harfler ile küçük harflerin ASCII kod tablosundaki tamsayı karşılıkları arasında, 32 fark vardır. Bu nedenle herhangi bir komut kullanmadan büyük-küçük harf dönüşümü yapmak için, ilgili harfin ASCII koduna 32 eklenebilir/çıkarılabilir.

C. Mantıksal İşlemler

Aritmetik ve karşılaştırma işlemlerini sonuçlandırabilen bilgisayar, temel mantık işlemlerini (VE, VEYA, DEĞİL) de yapabilmektedir. Mantıksal işlemler, hem karar ifadelerinde hem de matematiksel hesaplamalarda kullanılırlar. Programlardaki karşılaştırma ifadelerinde, birden fazla koşulun belirli bir özellikte sağlanması istenebilir. Bu durumda, mantıksal işlemlerden faydalanılır. Tablo 1.19'da temel mantıksal işlem sonuçları, Tablo 1.20'de temel mantıksal işlem karşılıkları ve Tablo 1.21'de temel mantıksal işlemler verilmektedir.

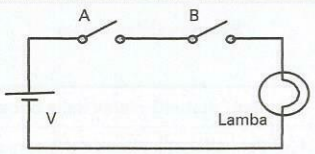
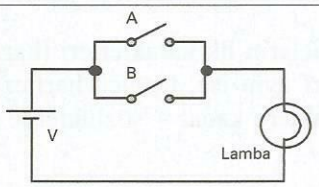
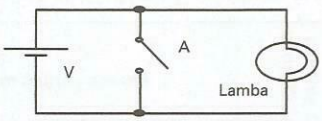
Tablo 1.19: Mantıksal işlem sonuçları

Sonuç	Rakam olarak	Bilgisayar dili
Doğru	1	True
Yanlış	0	False

Tablo 1.20: Temel mantıksal işlem karşılıkları

İşlem	Komut	Sembol
VE	AND	.
VEYA	OR	+
DEĞİL	NOT	`

Tablo 1.21: Temel mantıksal işlemler

İşlem	Elektriksel gösterim eşdeğeri	Doğruluk tablosu
VE (AND)		A B A VE B 0 0 0 0 1 0 1 0 0 1 1 1
VEYA (OR)		A B A VEYA B 0 0 0 0 1 1 1 0 1 1 1 1
DEĞİL (NOT)		A DEĞİL A 0 1 1 0

"VE" işleminde; bütün koşullar doğru ise, sonuç doğrudur. *"VEYA"* işleminde, sonucun doğru olması için koşullardan herhangi birisinin doğru olması yeterlidir. *"DEĞİL"* işleminde ise sonuç, koşulun tersidir.

Not

Bütün koşulların sağlanması gerektiğinde koşulların arasına **'VE'** ,
Koşullardan herhangi birinin sağlanması yeterli olduğunda koşulların arasına **'VEYA'**,
Koşulu sağlamayan gerektiğinde koşulun başına **'DEĞİL'** yazılır.

Matematiksel işlemlerde olduğu gibi, mantıksal işlemlerde de *'öncelik sıraları'* vardır (Tablo 1.22). Ayrıca mantıksal işlemlerin Tablo 1.23'te bazı özellikleri (*"Boole cebri özellikleri"*–*"Boolean cebri özellikleri"*) görülmektedir.

Tablo 1.22: Mantıksal işlemlerde öncelik sıraları

Sıra	İşlem
1	Parantez içindeki işlemler
2	DEĞİL
3	VE
4	VEYA

Tablo 1.23: Bazı mantıksal işlem özellikleri

Boole cebrinin bazı özellikleri	
A+0=0	A.0=0
A+A`=1	A.A`=0
A+A=A	A.A=A
A+1=1	A.1=A
A+B=B+A	A.B=B.A
A+(B+C)=(A+B)+C	A.(B.C)=(A.B).C
A.(B+C)=A.B+A.C	(A`)`=A
(A+B)`=A`.B`	(A.B)`=A`+B`

Not

Mantıksal işlemlerde '*n*' tane koşul/değişken varsa, '*2*ⁿ' tane farklı durum oluşur.

Örnek-1.11: Bir işyerindeki personellerden yalnızca yaşı 25'in üzerinde olup maaş olarak asgari ücret alanların isimleri istenebilir. Burada iki koşul vardır ve her ikisinin de sağlanması gerekir. Dolayısıyla koşul satırı

```
Eğer     Yaş>25    VE  Maaş=Asgari ücret   ise Yaz İsim
         1. Koşul      2. Koşul
```

şeklindedir. '`Yaz`' komutu, 1. ve 2. koşulun her ikisi de sağlanıyorsa çalışacaktır. Çünkü bu iki koşul "`VE`" işlemiyle birbirine bağlıdır. Eğer personelin yaşı 22 olup da asgari ücret alıyorsa ekrana ismi yazılmayacaktır. 1. koşul yanlış *(0)*, 2. koşul doğru *(1)* olduğundan '`0 VE 1 = 0`' sonucunda '`Yaz`' komutu çalışmayacaktır. Asgari ücretin 1000 TL kabul edilmesi durumunda oluşan birkaç değişik durum, Tablo 1.24'te verilmektedir.

Tablo 1.24: Örnek-1.11 için uygulama sonuçları

Yaş	Maaş	1. koşul	2. koşul	Sonuç	Yaz
20	*600*	0	0	0	Çalışmaz
19	*1000*	0	1	0	Çalışmaz
30	*750*	1	0	0	Çalışmaz
27	*1000*	1	1	1	Çalışır

Örnek-1.12: Bir sınıfta "*Bilgisayar*" dersinden `65`'in üzerinde not alıp, "*Türk Dili*" veya "*Yabancı Dil*" derslerinin herhangi birinden `65`'in üzerinde not alanların isimleri istenmektedir.

Burada üç koşul vardır ve "*Bilgisayar*" dersinden `65`'in üzerinde not almış olmak, temel koşuldur. Dolayısıyla "*Bilgisayar*" dersinin notu, diğer derslerinin notlarına '`VE`' işlemiyle bağlanmalıdır. Diğer iki dersin notlarının herhangi birisinin `65`'in üzerinde olması yeterli olduğundan '`VEYA`' işlemi gereklidir. Bu durumda koşul ifadesi;

```
Eğer Bilgisayar notu>65 VE (Türk Dili notu>65 VEYA Yabancı Dil notu>65) ise Yaz İsim
```

veya sembolik olarak;

```
Eğer  BN>65   VE   (TDN>65 VEYA YDN>65) ise Yaz İsim
      1. koşul      2. koşul   3. koşul
Temel koşul         Ortak koşul
```

Koşul	Sembolik
BN>65	A
TDN>65	B
YDN>65	C

şeklinde yazılabilir. Örnek uygulama değerleri Tablo 1.25'te verilmektedir.

Tablo 1.25: Örnek-1.12 için uygulama sonuçları

Bilgisayar	Türk Dili	Yabancı Dil	A	B	C	B+C	A.(B+C)	"Yaz" komutu
50	50	50	0	0	0	0	0	Çalışmaz
30	40	70	0	0	1	1	0	Çalışmaz
45	80	55	0	1	0	1	0	Çalışmaz
35	75	90	0	1	1	1	0	Çalışmaz
95	50	60	1	0	0	0	0	Çalışmaz
100	60	90	1	0	1	1	1	Çalışır
70	70	60	1	1	0	1	1	Çalışır
85	90	80	1	1	1	1	1	Çalışır

Örnek-1.13: Tablo 1.26'da verilen doğruluk tabloları için mantıksal ifadeleri elde ediniz.

Doğruluk tablolarından mantıksal işlem ifadeleri elde edilirken sonucun doğru olduğu durumlar (satırlar) (0 ise DEĞİL'i, 1 ise kendisi VE ile bağlanarak) VEYA ile birleştirilir. Ancak bu şekilde çok uzun mantıksal ifadeler elde edilmektedir. Bunları sadeleştirmek için de ilave işlemlerden (Boole cebri özellikleri, Karnaugh haritaları, tablo (Quine-McCluskey) yöntemi) faydalanılır. Örneğin Tablo 1.26'nın birinci örneğindeki mantıksal ifade, uzun bir şekilde

```
(DEĞİL(A>50) VE DEĞİL(B>20)) VEYA (DEĞİL(A>50) VE (B>20))
```

olacaktır. Bu ifade altı tane mantıksal işlem içermektedir. Ancak Boole cebri özellikleri kullanılırsa

```
(DEĞİL(A>50) VE DEĞİL(B>20)) VEYA (DEĞİL(A>50) VE (B>20))
DEĞİL(A>50) VE (DEĞİL(B>20) VEYA (B>20))
DEĞİL(A>50) VE 1
DEĞİL(A>50)
```

şeklinde bir tane mantıksal işlem içeren çok daha sade (programcılık açısından da hızlı) hale gelmektedir. Aynı sadeleştirme işlemi sembolik olarak

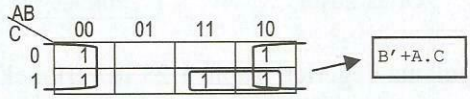
```
A'.B' + A'.B = A'.(B'+B) = A'.1 = A'
```

gösterilebilir.

Not

Programlarınızda ne kadar az işlem varsa, o kadar hızlı çalışacaklarını unutmayınız.

Tablo 1.26'nın ikinci örneğindeki mantıksal ifadenin Karnaugh haritasıyla sadeleştirilmesi aşağıda görülmektedir.



Tablo 1.26: Örnek-1.13 için doğruluk tabloları

Doğruluk tablosu	Mantıksal ifade
A>50 | B>50 | Sonuç 0 | 0 | 1 0 | 1 | 1 1 | 0 | 0 1 | 1 | 0	DEĞİL(A>50)
A>75 | B>75 | C>=75 | Sonuç 0 | 0 | 0 | 1 0 | 0 | 1 | 1 0 | 1 | 0 | 0 0 | 1 | 1 | 0 1 | 0 | 0 | 1 1 | 0 | 1 | 1 1 | 1 | 0 | 0 1 | 1 | 1 | 1	DEĞİL(B>75) VEYA (A>75) VE (C>=75)
A>49 | B>49 | C>=50 | Sonuç 0 | 0 | 0 | 1 0 | 0 | 1 | 0 0 | 1 | 0 | 1 0 | 1 | 1 | 1 1 | 0 | 0 | 1 1 | 0 | 1 | 0 1 | 1 | 0 | 1 1 | 1 | 1 | 0	C'+A'B
A | B | C | Sonuç 0 | 0 | 0 | 0 0 | 0 | 1 | 1 0 | 1 | 0 | 0 0 | 1 | 1 | 0 1 | 0 | 0 | 1 1 | 0 | 1 | 1 1 | 1 | 0 | 0 1 | 1 | 1 | 0	B'(A+C)
A | B | C | Sonuç 0 | 0 | 0 | 1 0 | 0 | 1 | 1 0 | 1 | 0 | 1 0 | 1 | 1 | 1 1 | 0 | 0 | 0 1 | 0 | 1 | 0 1 | 1 | 0 | 0 1 | 1 | 1 | 1	A'+B.C
A | B | C | Sonuç 0 | 0 | 0 | 1 0 | 0 | 1 | 1 0 | 1 | 0 | 0 0 | 1 | 1 | 0 1 | 0 | 0 | 1 1 | 0 | 1 | 1 1 | 1 | 0 | 1 1 | 1 | 1 | 1	A+B'

A	B	C	Sonuç
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

A'+B'

Mantıksal işlemler; koşulları birbirine bağlamanın yanında, aritmetik işlemlerde iki tabandaki sayıların bitleri arasında doğruluk koşullarını da test ederler.

Örnek-1.14: `a=77, b=99` için `c=a.b` ve `d=a+b` mantıksal işlem sonuçlarını hesaplayınız.

Sayıların ikili tabanındaki karşılıkları hesaplanarak 8 bit olarak yazılır.

$a=(77)_{10}=(1001101)_2$, $b=(99)_{10}=(1100011)_2$

Sonra, karşılıklı bitlere aradaki mantıksal işlem uygulanır.

a =	0	1	0	0	1	1	0	1	=11
b =	0	1	1	0	0	0	1	1	=99
c =	0	1	0	0	0	0	0	1	=65
d =	0	1	1	0	1	1	1	1	=111

Oluşan byte'lar 10 tabanına dönüştürülürse, işlem sonuçları görülür. Bu tür işlemler genellikle bazı bitleri maskelemek (örneğin paralel portun bazı pinlerini okumak), bitleri ters çevirmek (örneğin görüntünün negatifini almak) gibi birçok amaçlarla kullanılırlar.

Örnek-1.15: Tablo 1.27'de istenilenleri gerçekleştiriniz.

Tablo 1.27: Örnek-1.15 için istenilenler

Verilen ifadeyi, bilgisayar dilinde kodlayınız.	a=4 , b=16 ve c=2 için işlem sonucunu bulunuz.	Doğruluk tablosuna göre mantıksal ifadeyi yazınız.
$\sqrt[5]{\frac{1}{\frac{1}{a}+\frac{1}{b}}+\frac{1}{a+\frac{a^2+b^2}{\sqrt{a+b}}}}$	`a+b^1/2+1/1/1/c*b^(1/2)/a`	A B C Sonuç 0 0 0 1 0 0 1 1 0 1 0 0 0 1 1 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1 1

`(1/(1/a+1/b)+1/(a+(a^2+b^2)/(a+b)^.5))^.2`	12.5	A+B'

Örnek-1.16: Tablo 1.28'de istenilenleri gerçekleştiriniz.

Tablo 1.28: Örnek-1.16 için istenilenler

Verilen ifadeyi, bilgisayar dilinde kodlayınız.	a=72 , b=9 ve c=2 için işlem sonucunu bulunuz.	Doğruluk tablosuna göre mantıksal ifadeyi yazınız.
$\sqrt[4]{\frac{a}{a+b}+\frac{1}{1+\frac{1}{\sqrt[3]{\frac{1}{a}+\frac{1}{b}}}}}$	`(a/b^1/2)^(1/2)+1/1/1/c*(3*b)^(1/3)`	A B C Sonuç 0 0 0 **1** 0 0 1 **1** 0 1 0 **1** 0 1 1 **1** 1 0 0 **1** 1 0 1 **0** 1 1 0 **1** 1 1 1 **0**

`(a/(a+b)+1/(1+1/(1/a+1/b)^(1/3)))^.25`	3.5	A'+C'

SORULAR

1. Aşağıdaki matematiksel ifadeleri bilgisayar dilinde kodlayınız.

✓ $y1 = \sqrt[3]{\dfrac{1+a}{1+\dfrac{1}{a^2}}} + \dfrac{1}{a\sqrt{a}}$ ✓ $y2 = \dfrac{a+\dfrac{a}{b+1}}{\sqrt[5]{1+\dfrac{\sqrt[3]{a+b^2}}{\dfrac{1}{\sqrt{ab}}}}} + (ab)^2 - 4\dfrac{a}{\sqrt[3]{b^2}}$

✓ $y3 = \sqrt[3]{\dfrac{\dfrac{a+b}{(a^2+b^2)^4}}{1+\dfrac{1}{a+\dfrac{1}{b+\sqrt[5]{ab^2}}}}} + \dfrac{a^2+b^3}{\dfrac{ab}{a^2+b^2+2ab}+3a^2-3b^2} - \dfrac{\sqrt[4]{a^3}}{\dfrac{2}{\sqrt[3]{b^2}}}$

2. A=25 ve B=36 için bilgisayar dilinde kodlanmış aşağıdaki ifadelerin sonuçlarını bulunuz.

İfade	Sonuç
A^1/2+B^1/2	
A^(1/2)+B^1/2	
A^1/2+B^(1/2)	
A^(1/2)+B^(1/2)	
(A^1/2+B)^1/2	
(A^(1/2)+B)^1/2	
(A^(1/2)+B)^(1/2)	
(A+B)^(1/2)	

3. A=64, B=16 ve C=2 için aşağıdaki ifadelerin sonuçlarını bulunuz.

İfade	Sonuç
A^1/2+32/B*C+A/B^1/2	
A^(1/2)+32/B*C+A/B^1/2	
A^(1/2)+32/(B*C)+A/B^1/2	
A^(1/2)+32/(B*C)+A/B^(1/2)	
A^1/2+32/(B*C)+A/B^(1/2)	
A^1/2+32/B*C+A/B^(1/2)	
A^(1/2)+32/B*C+A/B^(1/2)	

4. Bilgisayar dilinde kodlanmış aşağıdaki ifadelerin matematiksel karşılıklarını yazınız.

✓ `y= (A+B)/C*D + A^1/2*B`

✓ `y= A+B/C*D + A^1/(2*B)`

✓ `y= (A+B)/(C*D) + A^(1/2*B)`

✓ `y= A+B/(C*D) + A^(1/2*B)`

✓ `y= A+B/C*D + (A^(1/2))*B`

5. **A=5, B=6, C=7 ve D=8 için aşağıdaki ifadelerin sonuçlarını "Doğru(1)" veya "Yanlış(0)" olarak bulunuz.**

İfade	Sonuç
(A>B) VE (C<D)	
(A<B) VE ((C<=D) VEYA (A>D))	
((D>B) VEYA (C>A)) VE ((A<C) VEYA (C>D))	
((D<=B) VEYA (A>C)) VE ((DEĞİL(A>D)) VEYA (B>C))	
DEĞİL(DEĞİL(DEĞİL(C>D)))	
DEĞİL((A>B) VEYA (B>D)) VEYA (A>D)	
DEĞİL(DEĞİL((A<D) VE (DEĞİL(C>B))) VE (DEĞİL(A>=C)))	

6. **Aşağıdaki alfasayısal bilgiler için, karşılaştırma ifadelerinin sonuçlarını "Doğru" veya "Yanlış" şeklinde bulunuz.**

1. karakter/kelime (A)	2. karakter/kelime (B)	İfade	Sonuç
"Ankara"	"Bursa"	A>B	
"program"	"programcı"	A<B	
"1225"	"1226"	A>B	
"soru-1"	"soru:1"	A<B	
"örnek 2"	"örnek2"	A>B	
"BURSA"	"Bursa"	A<B	
"Balkan"	"Balkanlar"	A>B	

7. **A="Ankara", B="Bursa", C="İstanbul" ve D="İzmir" için Soru-5'teki ifadelerin sonuçlarını tekrar bulunuz.**

8. **Klavyeden girilen bir sayının 50'den büyük, 50'den küçük veya 50'ye eşit olup olmadığını bulup ilgili mesajı yazdıran programın algoritmasını yazınız.**

9. **Bir sınıftaki öğrencilerden ancak Matematik (Mt) ile Türkçe (Tr) notu 50'ye eşit veya üzerinde olup Fizik (Fzk) veya Tarih (Th) derslerinin herhangi birisinden notu 50'ye eşit veya üzerinde olanlar "Başarılı" sayılmaktadır. Buna göre aşağıdaki komut satırında koşullar arasına uygun parantez ve mantıksal operatörleri yerleştiriniz.**

Eğer Mt>=50 ... Tr>=50 ... Fzk>=50 ... Th>=50 ise Yaz "Başarılı"

10. **A=33 ve B=77 değerleri için aşağıdaki işlemlerin sonuçlarını hesaplayınız.**

İşlem	Sonuç
C = A VE B	
D = A VEYA B	
E = DEĞİL (B)	

11. Aşağıdaki doğruluk tablolarına göre karar komut satırlarını tamamlayınız (ilgili mantıksal işlemleri ve parantezleri yerleştiriniz)

1. koşul	2. koşul	3. koşul	Sonuç
A>50	B>50	C>50	
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

```
Eğer ... A>50 ... B>50 ... C>50 ... ise Yaz "Doğru"
```

1. koşul	2. koşul	3. koşul	Sonuç
A>=45	B>=50	C>65	
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

```
Eğer ... A>=45 ... B>=50 ... C>65 ... ise Yaz "Doğru"
```

1. koşul	2. koşul	3. koşul	4. koşul	Sonuç
A>=50	B>49	C>64	D>=65	
0	0	0	0	**0**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	**0**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**
1	0	0	0	**0**
1	0	0	1	**1**
1	0	1	0	**1**
1	0	1	1	**1**
1	1	0	0	**0**
1	1	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**1**

```
Eğer ...A>=50...B>49...C>64...D>=65... ise Yaz "Doğru"
```

12. Mantıksal ifadeleri sadeleştirmede kullanılan yöntemleri araştırınız ve sadeleştirme örnekleri çözünüz.

BÖLÜM 2

ALGORİTMA

Bu bölümde

- Algoritma hakkındaki temel bilgileri,
- Algoritmalardaki operatörleri ve terimleri,
- Algoritma analizi ve tasarımını öğreneceksiniz.

Bölüm sonunda

- Algoritma, operatörlerini ve terimlerini açıklayabilecek,
- Algoritma analizleri yapabilecek,
- Temel algoritma tasarımları geliştirebileceksiniz.

I. GİRİŞ

İnsan, yaşamı boyunca herhangi bir işi gerçekleştirmeden önce genellikle plan yapar. Plan; işin nasıl yapılacağını, hangi aşamalardan geçileceğini, ne tür adımlar izleneceğini, neler elde edileceğini düzenli bir şekilde belirtir ve hedefe ulaşmada hem yol gösterir hem de büyük kolaylıklar sağlar. Planlı bir şekilde gerçekleştirilen işlerde, başarıya ulaşmak çok daha kolay ve hızlıdır.

Tablo 2.1: Okula gidip-gelme örnek adımları (planı)

(a)

```
1. Sabah kalk
2. Hazırlıklarını yap
3. Kahvaltı yap
4. Okula git
5. Derslere katıl
6. Eve dön
```

(b)

```
1. Sabah kalk
2. Hazırlıklarını yap
3. Kahvaltı malzemeleri var mı?
   3.1. Yoksa gidip al
4. Kahvaltı yap
5. Otobüs durağına git
6. Otobüsü bekle
7. Otobüse binerek okula git
8. Derslere katıl
9. Eve dön
```

(c)

```
1. Sabah kalk
2. Hazırlıklarını yap
3. Kahvaltı malzemeleri var mı?
   3.1. Yoksa gidip al
4. Kahvaltı yap
5. Otobüs durağına git
6. Otobüsü bekle
7. Otobüs geldi mi?
   8.1. Geldiyse otobüse binerek okula git
   8.2. Gelmediyse metro durağına git
        8.2.1. Metroyu bekle
        8.2.2. Metroya binerek okula git
8. Derslere katıl
9. Eve dön
```

Örneğin okula gidip-gelecek bir öğrencinin izlemesi gereken adımlar, Tablo 2.1'deki gibi sıralanabilir. Tablo 2.1a'da ayrıntıya inilmeden temel adımlar görülmektedir. Ancak plan yapılırken (hedefe ulaşılmaya çalışılırken) tüm olasılıklar göz önüne alınmalıdır, şansa bağlı durumlara bırakılmamalıdır. Örneğin evde kahvaltı malzemesi (ekmek, peynir, çay vb.) yoksa, öncelikle bunların temin edilmesi gerekmektedir. Tablo 2.1b'deki adımlarla, bu durum ilave edilmiştir. Benzer şekilde Tablo 2.1a'daki dördüncü adımda "Okula git" işlemi yer almaktadır. Ancak okula "ne zaman", "nereden", "nasıl" ("ne ile"), "kim" ile gidilecek gibi hususları içermemektedir. Oysaki düzgün bir planın, en ince ayrıntıları içermesi gerekmektedir. Bu duruma ilişkin biraz daha ayrıntılı adımlar, Tablo 2.1c'de verilmektedir. Ancak bu adımların (planın) da ideal olmadığını unutmayınız. Örneğin kahvaltı malzemesi alınmaya gidildiğinde bak-

kal/marketin kapalı olması, metro ulaşımının olmaması veya okula doğrudan metro hattının olmaması, otobüs/metro durağında inildiğinde durak ile okul arasında uzun mesafenin olması gibi birçok olasılık mevcut olabilir. Aynı şekilde Tablo 2.1a'daki altıncı adımda yer alan "Eve dön" işlemi de çok kapalıdır (ayrıntı içermemektedir). Planlar; problemlerin/sorunların en açık, en kolay ve en hızlı çözüm yollarını içermelidirler.

Bilgisayar dünyasında, insanın yaşamı boyunca yaptığı "plan" kavramına eşdeğer "algoritma" kavramı vardır. Bilgisayardaki işlemin/işlemlerin gerçekleştirilmesinde izlenecek mantıklı adımlara (adımlar dizisine, aşamalara), "*algoritma*" denir. Yani algoritma, işlemleri yaptırabilmek (problemleri çözdürebilmek, kontrolleri gerçekleştirebilmek vb.) için bilgisayara öğretilen/iletilen işlem basamaklarıdır. Bilgisayarın; bir problemi çözerken hangi ve neredeki verileri/giriş değerlerini alacağı, bunları işlerken ne tür yöntemleri kullanacağı, ne tür sonuçlar üreteceği ve bu sonuçları nerede göstereceği veya saklayacağı gibi adımların hepsi, hazırlanan algoritmanın herhangi bir programlama dilinin kurallarına uygun olarak yazılmış komutlarıyla (programlarla) bilgisayara iletilmektedir. Algoritmanın özel geometrik şekillerle çizilmiş şekli de "*akış diyagramı*" olarak adlandırılmaktadır.

Bilgi

"Algoritma" kelimesi Orta Asya'daki Harezm'de doğan matematikçi *Ebu Abdullah Muhammed bin Musa el-Harezmi*'nin isminden gelmektedir. IX. yüzyılda cebir alanında yayınladığı "Hisab el-cebir ve el-mukabala" kitabı, dünyadaki cebir alanında ilk kitaplardandır. Latince'ye tercüme edilmesiyle çok büyük ilgi görür ve Avrupalılar "el-Harezmi" (Arapça'da Al-Khwârizmî) yerine "*algorizm*" kelimesini 'Arap rakamlarını kullanarak aritmetiksel problemleri çözme kuralları' anlamında kullanırlar ve zamanla bu kelime "*algoritma*"ya dönüşür.

Bilgi

"Algoritma" kelimesinin Türk Dil Kurumu'ndaki tanımı "*Orta Çağ'da ondalık sayı sistemine göre, son zamanlarda ise iyi tanımlanmış kuralların ve işlemlerin adım adım uygulanmasıyla bir sorunun giderilmesi veya sonuca en hızlı biçimde ulaşılması işlemi, Harezmi yolu*" şeklindedir.

İnsanın günlük yaşantısında yaptığı planlar veya hedeflerine ulaşmak için belirlediği yollarda olduğu gibi, bilgisayarda da bir problemin çözümündeki adımlar Şekil 2.1'deki gibi özetlenebilir. Program yazılırken aşağıdaki adımlara dikkat edilmesi, programcının yararına olacaktır.

Program yazma adımları

i. Problem/iş iyice irdelenir/incelenir/anlaşılır (analiz edilir).

ii. Çözüm yolları ortaya konularak programlamaya en uygun (en az komutla, en kısa sürede, en doğru-hassas sonuç veren) çözüm yolu seçilir/belirlenir.

iii. Programın algoritması hazırlanır veya akış diyagramı çizilir.

iv. Algoritma veya akış diyagramı, programcının bildiği veya tercih ettiği bir programlama diliyle kodlanır.

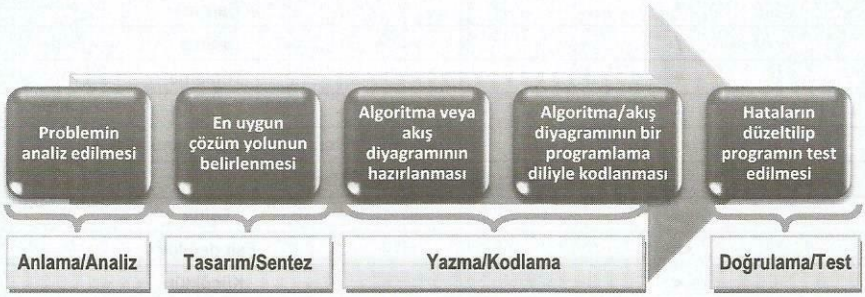
v. Program, kodlandığı editörde çalıştırılarak hataları düzeltilir.

vi. Bilinen giriş-çıkış değerleriyle programın doğru çalışıp çalışmadığı test edilir (doğrulama yapılır).

Bir programın yazımındaki temel işlem, programlama mantığına uygun algoritmanın hazırlanması veya akış diyagramının çizilmesidir. Program yazılırken/kodlanırken her ne kadar farklı programlama dilleri kullanılsa da, bu programın algoritma/akış diyagramı hiçbir zaman değişmemektedir. Bu nedenle algoritması hazırlanmış/yazılmış veya akış diyagramı çizilmiş bir program/problem, tüm programlama dillerinde kolaylıkla kodlanabilmektedir.

Not

Programlama mantığına uygun algoritmanın hazırlanması/akış diyagramının çizilmesi, programlamanın temelidir.



Şekil 2.1: Program yazma adımları

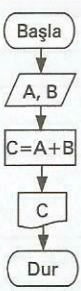
Algoritmalar genel olarak üç şekilde ifade edilebilirler (gösterilebilirler-sunulabilirler):

a. *Algoritmanın metin olarak yazılması*: Problemin çözüm adımları (program), düz metin olarak açık cümlelerle ifade edilebilir.

b. *Algoritmanın sözde kodlarla (pseudo-code) yazılması*: Problemin çözüm adımları (program), komut benzeri anlaşılır metinlerle veya kısaltmalarla ifade edilebilir.

c. *Algoritmanın akış diyagramının çizilmesi*: Problemin çözüm adımları (program), geometrik şekillerle gösterilebilir.

Örnek-2.1: Klavyeden girilen iki sayıyı toplayıp ekranda gösteren programın algoritmasının, üç değişik şekilde gösterilmesi.

Metin olarak yazılması	Sözde kodlarla yazılması	Akış diyagramının çizilmesi
1. Başla 2. Birinci sayıyı klavyeden okuyunuz (alınız, giriniz) 3. İkinci sayıyı klavyeden okuyunuz (alınız, giriniz) 4. Girilen/alınan sayıları toplayarak sonucu oluşturunuz 5. Sonucu, ekrana yazdırınız 6. Dur	1. Başla 2. A oku (al, gir) 3. B oku (al, gir) 4. C=A+B 5. C yaz (göster) 6. Dur	

II. ALGORİTMALARDA KULLANILAN OPERATÖRLER

İşlemleri belirten simgelere/sembollere, bilgisayar terminolojisinde "*operatör*" denir. Algoritmalarda/akış diyagramlarında kullanılan operatörler, Tablo 2.2'de toplu olarak verilmektedir.

Tablo 2.2: Algoritmalarda/akış diyagramlarında kullanılan operatörler

OPERATÖRLER	
Matematiksel işlem operatörleri	
^	Üs alma
*	Çarpma
/	Bölme
+	Toplama
-	Çıkarma
.	Tam ve ondalıklı kısımları ayırma
Karşılaştırma operatörleri	
=	Eşittir
<>	Eşit değildir
<	Küçüktür
>	Büyüktür
>= *veya* =>	Büyük eşittir
<= *veya* =<	Küçük eşittir
Mantıksal işlem operatörleri	
`	DEĞİL
.	VE
+	VEYA
Küme işlem operatörleri	
+	Birleşim
-	Fark
*	Kesişim
=	Eşit
<>	Eşit değil
=< *veya* <=	Soldaki küme, sağdakinin alt kümesi
>= *veya* =>	Sağdaki küme, soldakinin alt kümesi
Alfasayısal operatörler	
+	Birleştirme
Genel işlem operatörleri	
=	Aktarma
( )	Parantez

Not

Her programlama dilinde bu operatörlerin yazımında farklılıklar olabileceği gibi, daha az veya daha fazla sayıda da bulunabilmektedirler.

III. ALGORİTMALARDA KULLANILAN TERİMLER

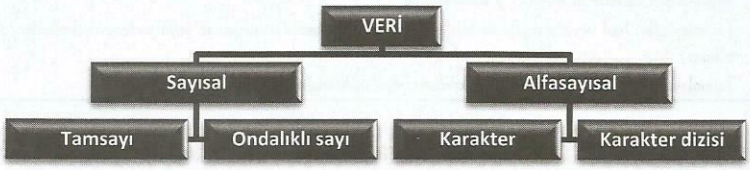
Algoritmalarda (programlarda) sık sık '*veri*', '*tanımlayıcı*', '*değişken*', '*sabit*', '*aktarma*', '*sayaç*', '*döngü*' vb. terimler kullanılır. Bundan sonraki kısımda ilgili terimler, sırasıyla ayrıntılı olarak açıklanacaktır.

A. Veri

Bilgisayarlarda işlenen tüm bilgiler "*veri*" olarak adlandırılırlar ve temel/genel olarak iki gruba ayrılırlar (Şekil 2.2).

a. ***Sayısal veriler***: Sayısal ağırlığı (değeri) olan ve üzerinde matematiksel işlemler yapılabilen verilerdir. Tamsayılar, ondalıklı sayılar vb. bu grupta yer alırlar. Sayısal veriler, bilgisayara herhangi bir tabanda veya üstel biçimde aktarılabilirler (girilebilirler).

b. ***Alfasayısal veriler***: Herhangi bir sayısal ağırlığı (değeri) olmayan verilerdir. Bu grupta karakterler (harfler, simgeler, rakamlar) ve karakter dizileri (kelimeler, cümleler) yer alır. Alfasayısal veriler, programlarda veya algoritma/akış diyagramlarında tek/çift tırnak içinde verilirler.



Şekil 2.2: Bilgisayarda işlenen temel veriler

Not

Programlardaki sayısal verilerde herhangi bir taban belirtme simgesi yoksa 10 tabanında (decimal) kabul edilirler.

Not

Üstel belirtme biçimindeki "*sayıEüs*" veya "*sayıE üs*" anlamı "*sayı*.$10^{üs}$" tür.

Aşağıda sayısal ve alfasayısal verilere ilişkin birkaç örnek verilmektedir.

Sayısal veri	Alfasayısal veri
2004	Bursa
-555	a
6.78	;
2013	"2013"
3.9E2	©
1000e-3	"½"

Not

Programlarda/algoritmalarda sayısal veriler doğrudan, alfasayısal veriler ise tek/çift tırnak içinde aktarılırlar/gösterilirler.

Bu temel verilerin yanında "zaman" (saat, tarih vb.), "multimedya" (ses, görüntü vb.), "boolean" (doğru, yanlış) gibi veri sınıflandırmaları da yapılabilmektedir.

B. Tanımlayıcı

Programı yazan kişi tarafından düşünülüp oluşturulan ve programdaki değişkenleri, sabitleri, paragrafları, kayıt alanlarını, özel veri/bilgi tiplerini, fonksiyonları, alt programları vb. adlandırmak için kullanılan karakterlere/kelimelere/ifadelere "*tanımlayıcı*" denir. Prensip olarak; programdaki tanımlayıcı karakterlerin veya karakter dizilerinin (kısaltmaların, kelimelerin vb.), yerini tuttukları ifadelere/işlemlere/görevlere çağrışım yapacak şekilde seçilmesi/oluşturulması daha uygundur. Böylece diğer programcıların bu kodları incelemesi, anlaması ve geliştirmesi/düzenlemesi daha da kolaylaşmaktadır. Tanımlayıcılar oluşturulurken, aşağıdaki "*isimlendirme kuralları*"na uyulmalıdır.

İsimlendirme kuralları

i. İngiliz alfabesindeki A-Z veya a-z arası 26 harf kullanılabilir.
ii. 0-9 arası rakamlar kullanılabilir.
iii. Simgelerden sadece alt çizgi(_) kullanılabilir.
iv. Tanımlayıcılar, harf veya alt çizgi ile başlayabilir; ancak rakamla başlayamaz veya sadece rakamlardan oluşamaz (sayı olamaz).
v. Tanımlayıcı, ilgili programlama dilinin komutu veya saklı/anahtar kelimesi olamaz.

Yazılımcılar; kurallara uygun olarak oluşturulan tanımlayıcıları, istedikleri şekilde yazabilirler veya programcılıkta kullanılan standart gösterimlerden (notasyonlardan) birisini tercih edebilirler. En çok kullanılan standartlar/notasyonlar "Pascal case" ve "Camel case" olmasına karşın bazıları genel şekilleriyle aşağıda özetlenmektedir:

i. *Pascal case/casing/notation*: Kelimelerin ilk harfleri büyük, diğerleri küçük yazılır. Örneğin `AdSoyad`, `EvTel`, `CepTelNo` vb.

ii. *Camel case/casing/notation*: Birinci kelimenin ilk harfi hariç diğer kelimelerin ilk harfleri büyük, diğerleri küçük yazılır. Örneğin `adSoyad`, `evTel`, `cepTelNo` vb.

iii. *Upper (upperscore) case/notation*: Kelimenin tüm harfleri büyük yazılır. Örneğin `ADSOYAD`, `EVTEL`, `CEPTELNO` vb.

iv. *Underscore case/notation*: Kelimelerin arasına alt çizgi yazılır. Örneğin `Ad_Soyad`, `Ev_Tel`, `Cep_Tel_No` vb.

C. Değişken

Program içinde farklı değerler alabilen/aktarılabilen bellek/veri/bilgi alanlarını gösteren tanımlayıcılar, "*değişken*" olarak adlandırılırlar. "İsimlendirme kurallarına" uygun olarak oluşturulan değişken isimlerinin; yerini tuttukları/aldıkları verilere çağrışım yapacak şekilde olması, programın anlaşılırlığı açısın-

dan önemlidir. Örneğin; bir kişinin adının aktarıldığı değişkenin 'Ad', adı ve soyadının aktarıldığı 'Adsoy', telefonunun aktarıldığı 'tel', ev adresinin aktarıldığı 'EvAdres', işyeri adresinin aktarıldığı 'is_adres', vb. şeklinde oluşturulması daha anlaşılır olmaktadır.

Örnek-2.2:

```
1. Başla
2. Birinci sayıyı (A) al
3. İkinci sayıyı (B) al
4. C=A+B işlemini yap
5. Sonucu (C) yaz
6. Dur
```

Yandaki algoritma, klavyeden (dışarıdan) girilen/okunan/alınan iki sayının toplamını yapıp sonucu ekrana yazdırır. Algoritmada 'A', 'B' ve 'C' değişkendir.

Program her çalıştırıldığında 'A' ve 'B' değişkenlerine farklı değerler girilir ve sonucu temsil eden 'C' de farklı değerler oluşur (Tablo 2.3).

Tablo 2.3: Örnek-2.2 için uygulama sonuçları

Girilen A sayısı	Girilen B sayısı	C=A+B	Yazılan C sayısı	
-3	3	0	0	*1. çalıştırma*
3	11	14	14	*2. çalıştırma*
33	99	132	132	*3. çalıştırma*
1	2	3	3	*4. çalıştırma*
-100	200	100	100	*5. çalıştırma*
50	-83	-33	-33	*6. çalıştırma*

Örnek-2.3:

```
1. Başla
2. Öğrencinin numarasını (No) al
3. Öğrencinin adını ve soyadını (Adsoy) al
4. Öğrencinin vize notunu (Vize) al
5. Öğrencinin final notunu (Final) al
6. Ort = 0.3*Vize + 0.7*Final
7. Numara (No) ve ortalamayı (Ort) yaz
8. Dur
```

Verilen algoritmada; öğrencinin numarası, adı ve soyadı, vize notu ve final notu girilmektedir. Verilere göre vize notunun %30'ü ile final notunun %70'i alınarak ortalaması hesaplanmakta ve ekrana yazdırılmaktadır. Burada değişken olarak *No*, *Adsoy*, *Vize*, *Final* ve *Ort* vardır. Programın çalıştırılmasıyla oluşabilecek bazı sonuçlar Tablo 2.4'te verilmektedir.

Tablo 2.4: Örnek-2.3 için uygulama sonuçları

No	Adsoy	Vize	Final	Ort	Ekrana yazılan	
9801.0033	Ahmet	30	70	58	9801.0033 – 58	*1. çalıştırma*
9804.0099	Mehmet	70	90	84	9804.0099 – 84	*2. çalıştırma*
9712.0123	Ali	60	30	39	9712.0123 – 39	*3. çalıştırma*

Yapısal dillerdeki programlarda, kullanılacak olan değişkenler ve bunlara aktarılacak olan veri tipleri önceden (program belirli bloklarında, kullanıldıkların yerden önce, kullanıldıkları yerde vb.) bildirilmektedirler. Değişken bildirimi yapıldığında; veri türüne göre bellekte yer açılmakta/ayrılmakta ve program içinde bunlara değerler aktarılarak kullanılabilmektedirler. Bu bildirimler iki şekilde gerçekleştirilebilmektedir: değişken bildirimi ve *işaretçi* (*pointer*) bildirimi.

Şekil 2.3: Bellek ve değer

Yukarıda değişken konusu incelenmiştir. İşaretçiler ise bellek adreslerini gösteren/saklayan veya bellek adreslerini göstererek değeri kullanan/işleyen değişkenlerdir. Programlama dillerindeki operatörler ile işaretçinin bellek adresini mi yoksa bellekteki değeri mi gösterdiği belirtilmektedir. Örneğin Şekil 2.3'teki gibi "m=3" şeklinde programda değişken bildirimi yapılırsa, "m" değişkeni hafızadaki "3" değerini gösterecektir. Ancak işaretçi bildirimi yapılırsa "3" değerinin kayıtlı olduğu bellek/hafıza adresi olan "0x0000FFFF"i (16 tabanında) gösterecektir.

D. Sabit

Bilim/uygulama alanlarına ait birçok sabitler bulunmaktadır. Örneğin geometride π sayısı, matematikte doğal logaritma tabanı olan e sayısı bunlara ilişkin bazı örneklerdir. Programlardaki değeri değişmeyen veriler (veri alanları) "*sabit*" olarak adlandırılmakta ve değişkenlerde olduğu gibi "isimlendirme kuralları"na uygun olarak oluşturulmaktadırlar.

Örnek-2.4: Yarıçapı girilen dairenin çevresiyle alanını hesaplayıp yazdıran programın algoritması.

```
1. Başla
2. pi_sayisi=3.14
3. Dairenin yarıçapını (r) al
4. cevre=2*pi_sayisi*r
5. alan=pi_sayisi*r*r
6. Çevre(cevre) ve alanı(alan) yaz
7. Dur
```

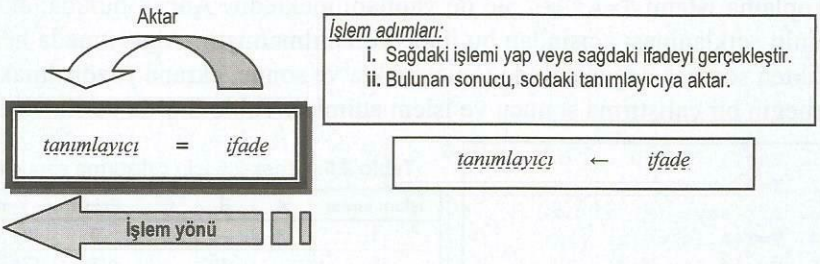
E. Atama/aktarma

Herhangi bir veri alanına, bilgi yazma/kaydetme; herhangi bir işlemin veya ifadenin sonucunu başka bir değişkende gösterme vb. görevlerde "*atama*" veya "*aktarma*" operatörü kullanılır.

```
tanımlayıcı = ifade
```

satırında '*tanımlayıcı*' ile gösterilen kısım, herhangi bir değişkenin, sabitin vb. ismidir. '*ifade*' ile belirtilen kısımda ise; matematiksel, mantıksal veya

alfasayısal ifade olabilir. Aradaki ' = ' sembolü, "*atama operatörü*" veya "*aktarma operatörü*" olarak adlandırılır ve sağdaki ifadenin/işlemin sonucunu, soldakine aktarır. Bu durumda tanımlayıcının - varsa - bir önceki değeri (eski değeri) silinir (Şekil 2.4).



Şekil 2.4: Aktarma/atama işlemi

Tablo 2.5: Aktarma işlemi uygulama sonuçları

a	b	Eski c	Yeni c	
Matematiksel ifadelerde (*c = a + b*)				
3	7	---	*10*	*1. çalıştırma*
5	7	*10*	*12*	*2. çalıştırma*
20	33	*12*	*53*	*3. çalıştırma*
Mantıksal ifadelerde (*c = a + b*)				
0	1	---	*1*	*1. çalıştırma*
0	0	*1*	*0*	*2. çalıştırma*
1	1	*0*	*1*	*3. çalıştırma*
Alfasayısal ifadelerde (*c = a + b*)				
Galata	saray	---	Galatasaray	*1. çalıştırma*
Beşik	taş	*Galatasaray*	*Beşiktaş*	*2. çalıştırma*
Fener	bahçe	*Beşiktaş*	*Fenerbahçe*	*3. çalıştırma*

Aktarma işlemiyle ilgili örnekler Tablo 2.5'te görülmektedir. Burada her ifade 3 kez arka arkaya çalıştırılmaktadır. Bulunan her yeni sonuçta, eski sonucun hiçbir etkisi yoktur. Çünkü bulunan her yeni sonuç; eskisinin üzerine yazılmakta, dolayısıyla onu silmektedir.

Tablo 2.5'te, matematiksel mantığa aykırı bir durum yoktur. Ancak

```
TPL = TPL + 1
```

işlemine, matematiksel mantıkla yaklaşılırsa;

```
TPL = TPL+1        ,        TPL - TPL = 1        ,        0 = 1
```

şeklinde yanlış sonuç çıkmaktadır. Bilgisayar mantığında/dilinde durum böyle değildir. Önce sağ taraftaki ifadede değişkenin eski/mevcut değeriyle işlem gerçekleştirilmekte (1 arttırılmakta, 1 ile toplanmakta); sonra bulunan sonuç, sol tarafa yeni değer olarak aktarılmaktadır (eskisinin üzerine yazılmaktadır).

Not

Atama operatörü, eşitlik operatörü ile karıştırılmamalıdır. "=" atama operatörü, "←" anlamındadır.

Not

Atama operatörünün sol tarafında kesinlikle işlem yer almamalıdır!
Sol tarafta her zaman sonucun aktarıldığı tanımlayıcı olmalıdır !

Örnek 2.5: Aşağıdaki algoritmada; klavyeden üç sayı girilip toplanmaktadır. Toplama işlemi `T=A+B+C` ile de yapılabilmektedir. Ancak burada, aktarma işleminin açıklanması açısından bu ifade kullanılmamıştır. Algoritmada her sayı girildikten sonra `T` değişkeninde toplanmakta ve sonuç, ekrana yazdırılmaktadır. Bu örneğin bir çalıştırma sonucu ve işlem adımları Tablo 2.6'da verilmektedir.

```
1. Başla
2. T=0
3. Bir sayı (A) al
4. T=T+A işlemini yap
5. Başka bir sayı (B) al
6. T=T+B işlemini yap
7. Başka bir sayı (C) al
8. T=T+C işlemini yap
9. Yaz T
10. Dur
```

Tablo 2.6: Örnek-2.5 için çalıştırma sonuçları

İşlem sırası	A	B	C	Eski T	Yeni T
1	3	---	---	0	0+3=3
2	---	4	---	3	3+4=7
3	---	---	5	7	7+5=12
					T=12

F. Sayaç

Programlarda; bazı işlemlerin belirli sayıda yapılması veya işlenen/üretilen değerlerin sayılması gerekebilir. Örneğin klavyeden girilen bir cümlede kaç tane sesli harf olduğunu bulan programda, cümlenin her harfi sırasıyla çağırılır ve sesli harfler kümesine ait olup olmadığı araştırılır. Eğer sıradaki çağırılan harf bu kümeye ait ise, bunları sayacak olan değişkenin değeri bir arttırılır. `s=s+1` şeklindeki işlemde sağdaki ifadede değişkenin eski (mevcut/önceki) değerine '`1`' eklenmekte; bulunan sonuç yine kendisine, yeni değer olarak aktarılmaktadır. Bu tür sayma işlemlerine, algoritmada "*sayaç*" veya "*sayıcı*" adı verilir. Yani "*sayaç*"; işlem akışı kendisine her geldiğinde, belirtilen adım değeri kadar artan/azalan değişkendir (Şekil 2.5).

Şekil 2.5: Sayaç/sayıcı (sayma işlemi)

Örnek-2.6: Aşağıdaki algoritmada 1-5 arası sayılar, ekrana yazdırılmaktadır. 1-5 arası sayıları oluşturmak için sayaç (`S=S+1`) kullanılmakta ve algoritmanın çalışma prensibi Tablo 2.7'de verilmektedir.

```
1. Başla
2. S=0
3. Eğer S>4 ise git 7
4. S=S+1
5. Yaz S
6. Git 3
7. Dur
```

Tablo 2.7 Örnek-2.6'daki sayacın çalışması

Eski S	Yeni S	Ekrana yazılan
0	0+1=1	1
1	1+1=2	2
2	2+1=3	3
3	3+1=4	4
4	4+1=5	5

G. Döngü

Birçok programda; bazı işlemler, belirli ardışık değerlerle gerçekleştirilmekte veya belirli sayıda yapılmaktadır. Programlardaki belirli işlem bloklarını (kod parçalarını); aynı veya farklı değerlerle, verilen sayıda gerçekleştiren çevrim yapılarına "*döngü*" denir. Örneğin 1 ile 1000 arasındaki tek sayıların toplamını hesaplayan programda `T=1+3+5+7+9+11+...` gibi uzun uzun yazmak yerine 1 ile 1000 arasında ikişer ikişer artan bir döngü açılır ve döngü değişkeni ardışık olarak toplanır.

Döngü oluşturma kuralları

i. Döngü değişkenine başlangıç değeri verilir.
ii. Döngünün artma veya azalma miktarı belirlenir.
iii. Döngünün bitiş değeri belirlenir.
iv. Eğer döngü, karar/karşılaştırma ifadeleriyle oluşturuluyorsa; karar/karşılaştırma işleminden önce döngü değişkenine başlangıç değeri verilmiş olmalı ve döngü içinde adım miktarı kadar arttırılmalı/azaltılmalıdır.

Örnek-2.7: Aşağıdaki algoritma 1-10 arası tek sayıların toplamını hesaplamakta ve çalışma prensibi Tablo 2.8'de verilmektedir.

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

Döngü

Tablo 2.8 Örnek-2.7'deki döngünün çalışması

Eski J	Eski T	Yeni T	Yeni J
1	0	0+1=1	3
3	1	1+3=4	5
5	4	4+5=9	7
7	9	9+7=16	9
9	16	16+9=25	11
11	-	-	-

H. Ardışık Toplama

Ardışık toplama işleminin çalışma prensibi, sayacınkine benzemekte olup programlarda aynı değerin üzerine yeni değerler eklemek için kullanılır. Genel kullanım şekli aşağıdaki gibidir:

```
Toplam değişkeni = Toplam değişkeni + Sayı
```

Not

Toplam değişkenine, başlangıç değeri olarak toplama işleminin etkisiz elemanı olan 0 (sıfır) atanır.

Örnek-2.8: Klavyeden girilen 5 sayının ortalamasını bulan programın algoritması.

N adet sayının ortalaması, bu sayıların toplamının N'e bölünmesiyle elde edilir. Aşağıdaki algoritmada girilen sayıların toplamı için 'T' değişkeni; beş sayının girilip girilmediğini kontrol etmek için ise 'S' sayacı kullanılmıştır. Koşul 'S>N-1' olabileceği gibi 'S=N' de yazılabilirdi. Algoritmaya ilişkin örnek işlem adımları Tablo 2.9'da verilmektedir.

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

Tablo 2.9 Örnek-2.8 uygulama sonuçları

Eski S	Yeni S	Sayı	Eski T	Yeni T	Ort.
0	0+1=1	3	0	0+3=3	---
1	1+1=2	1	3	3+1=4	---
2	2+1=3	5	4	4+5=9	---
3	3+1=4	11	9	9+11=20	---
4	4+1=5	35	20	20+35=55	---
Ortalama=55/5=11					

I. Ardışık Çarpma

Ardışık çarpma işleminde; aynı değer, yeni değerlerle çarpılarak eskisinin üzerine aktarılmaktadır. Genel kullanım şekli aşağıdaki gibidir:

```
Çarpım değişkeni = Çarpım değişkeni * Sayı
```

Not

Çarpım değişkenine, başlangıç değeri olarak çarpma işleminin etkisiz elemanı olan 1 (bir) atanır.

Örnek-2.9: Klavyeden girilen N sayısının faktöriyelini hesaplayan programın algoritması.

Algoritmada, klavyeden N sayısı girilerek faktöriyeli hesaplanmaktadır. N=5 için programın çalışması Tablo 2.10'da verilmektedir.

```
1. Başla
2. N sayısını al
3. F=1
4. S=0
5. Eğer S>N-1 ise git 9
6. S=S+1
7. F=F*S
8. Git 5
9. Yaz F
10. Dur
```

Tablo 2.10: Örnek-2.9 için uygulama sonuçları

Eski S	Yeni S	Eski F	Yeni F	Sonuç
0	0+1=1	1	1*1=1	---
1	1+1=2	1	1*2=2	---
2	2+1=3	2	2*3=6	---
3	3+1=4	6	6*4=24	---
4	4+1=5	24	24*5=120	120

IV. ALGORİTMA HAZIRLAMA

Daha önce de belirtildiği gibi, "*algoritma, programın temelidir*". Algoritması hazırlanmış veya akış diyagramı çizilmiş programın kodunu yazmak çok kolaydır. İlgili programlama dilinin kaynaklarına bakılarak kodlamalar rahatlıkla gerçekleştirilebilir. Algoritma hazırlanırken aşağıdaki adımlar izlenebilir:

- Problem/sorun/iş tanımlanmalı, iyice analiz edilmelidir.
- Problem/sorun/iş büyük, karmaşık ve zorsa; daha küçük ve çözümü daha kolay parçalara ayrıştırılmalıdır.
- Tüm olasılıklar göz önünde bulundurularak çözüm yolları kapsamlı araştırılmalı ve uygun olan bir tanesi seçilmelidir. Bir problemin birçok çözüm yolunun olabileceğini unutmayınız. Ayrıca her yazılımcının/programcının algoritma geliştirme ve program kodlama tarzının da farklı olması gayet doğaldır.
- Algoritmada kullanılacak olan tanımlayıcılara, alt programlara, fonksiyonlara, modüllere, dâhili ve harici bileşenlere vb. karar verilmelidir.
- Problemin/sorunun/işin bildirimi/çözümü için gerekli verilerin/bilgilerin girişi/alınması/okunması belirlenmelidir. Yani algoritmanın kesinlikle bir girdisinin olması gerekmektedir.
- Problemin/sorunun/işin çözümü için belirlenen/seçilen yol açık, kesin ve net bir şekilde sunulmalıdır.
- Algoritmada adımları (çözüm aşamaları); şansa bağlı faktörler içermemelidir, tüm ihtimaller göz önünde bulundurulmalıdır. Örneğin bölme işlemi olan bir programda "paydadaki değer sıfır çıkmayabilir", "kullanıcı paydaya sıfır atamayabilir/girmeyebilir" gibi iyi niyetlerde (temennilerde) bulunulmamalıdır.

Örnek işlem	Yanlış	Doğru
Bölme işlemi	"Paydadaki değer sıfır çıkmayabilir (olmayabilir, girilmeyebilir)"	Paydanın sıfır olma ihtimalini dikkate almak
Dosyadan okuma	"Belki programın çalıştırılacağı bilgisayarda dosya vardır"	Dosyanın varlığı kontrol etmek ve yoksa uyarmak/bilgilendirmek
Karttan okuma	"Belki programın çalıştırılacağı bilgisayarda bu model kart vardır"	Kartın varlığını kontrol etmek ve yoksa uyarmak/bilgilendirmek
Aşamalı işlem	"Kullanıcı adım adım ilerleyecektir"	Adım/aşama atlamasını engellemek ve bilgilendirmek

- Algoritmadaki adımlar, kesin ifadelerle belirtilmelidir. Örneğin "Değişkenin değerini *biraz* arttırın" şeklinde kesin olmayan ifadeler içermemelidir. Buradaki arttırma miktarı bir, iki vb. gibi kesin olmalıdır.

Yanlış	Doğru
Değişkenin değerini *biraz* arttır	Değişkenin değerini bir arttır
Bulunan sonuç *büyük* ise ...	Bulunan sonuç 100'den büyük ise ...
Hava *sıcak* ise ...	Hava sıcaklığı 33 dereceden büyük ise ...
Ağırlık *50'den küçük* ise ...	Ağırlık 50 kg dan küçük ise ...

- Algoritmadaki adımlar, programın sonlu sayıda işlem yapmasını sağlamalıdır. Yani her program; sonucu/çözümü bulma, belirli bir koşulu sağlama, belirtilen değere ulaşma vb. durumlarda sonlanmalıdır. Bir programın, sonsuz tane işlem yapamayacağını göz ardı etmeyiniz.

Yanlış	Doğru
1. Başla 2. t=0 , s=0 3. s=s+1 4. t=t+s 5. Git 3 6. Yaz t 7. Dur	1. Başla 2. t=0 , s=0 3. s=s+1 4. t=t+s 5. Eğer s>1000 ise Git 7 6. Git 3 7. Yaz t 8. Dur

- Algoritmadaki adımlar sıralı ve mantıklı olmalıdır. Örneğin önceki satırlarda hesaplanmamış/tanımlanmamış bir değeri mevcut satırda kullanma; önceki satırlarda veya önceden yazılmamış bir alt programı çağırma gibi işlemler yapılmamalıdır.

Yanlış	Yanlış
1. Başla 2. Al (gir, oku) x 3. y=2*x 4. t=x+y+z 5. Yaz t 6. Dur --- z değeri yok	1. Başla 2. "topla(x, y)" alt programını oluştur/tanıt 3. Al a, b 4. m=topla(a,b) 5. n=cikar(a,b) 6. Yaz m, n 7. Dur --- "cikar(x, y)" alt programı yok

- Algoritmalarda tekrar eden işlemlerden programın hızı ve etkinliği açısından kaçınılmalıdır. Örneğin önceki adımlarda hesaplanmış olan bir ifadeyi mevcut adımda tekrar hesaplamak, programın hızını yavaşlatacaktır.

İyi	Daha iyi
1. Başla 2. Al a, b 3. toplam=a+b 4. fark=a-b 5. ortalama=(a+b)/2 6. Yaz toplam, fark, ortalama 7. Dur	1. Başla 2. Al a, b 3. toplam=a+b 4. fark=a-b 5. ortalama=toplam/2 6. Yaz toplam, fark, ortalama 7. Dur

- Algoritmalarda farklı değerlerle gerçekleştirilen aynı işlemleri, tekrar tekrar yazmak yerine bunları alt program/fonksiyon/modül olarak oluşturmak daha uygundur.

İyi	Daha iyi
1. Başla 2. Al a, b 3. ta=0, tb=0, sa=1, sb=1 4. Eğer sa>a ise Git 8 5. ta=ta+sa 6. sa=sa+1 7. Git 4 ***8. Eğer sb>b ise Git 12*** ***9. tb=tb+sb*** ***10. sb=sb+1*** ***11. Git 8*** 12. Yaz ta, tb 13. Dur	***tpl=toplam(x)*** ***1. Başla*** ***2. tpl=0, s=1*** ***3. Eğer s>x ise Git 6*** ***4. tpl=tpl+s , s=s+1*** ***5. Git 3*** ***6. Dur*** 1. Başla 2. Al a, b 3. ta=toplam(a), tb=toplam(b) 4. Yaz ta, tb 5. Dur

- Algoritmalarda gereksiz işlemlerden kaçınılmalıdır. Örneğin bir tamsayının kendisi hariç tam bölenlerini bulabilmek için 1'den o tamsayıya kadar döngü açılır ve bunun sıradaki döngü değişkenine tam bölünüp bölünmediği kontrol edilir. Ancak hiçbir tamsayı, kendi yarısından daha büyük tamsayıya tam bölünemez. Bu durumda örneğin 1000 sayısının tam bölenlerini bulmak için 1-1000 arası döngü (1000 tane işlem) yerine 1-500 arası döngü (500 tane işlem) açmak çok daha doğrudur. Ayrıca tüm tamsayılar, bire tam bölünürler. Bu durumda "1" başlangıçta işlem yapılmadan yazdırılıp döngü 2'den açılabilir. Böylece gereksiz işlemler yapılmayıp programın çalışma hızı arttırılır.

Örnek işlem	İyi	Daha iyi
A'dan C'ye kopyalama	1. A'dan B'ye kopyala 2. B'den C'ye kopyala	1. A'dan C'ye kopyala
Kendisi hariç tam bölenlerini bulma	1. Başla 2. Al a 3. s=1 4. Eğer s>a ise Git 8 5. Eğer (a/s) tamsayı ise Yaz s 6. s=s+1 7. Git 4 8. Dur	1. Başla 2. Al a 3. s=2, yari=a/2 4. Yaz "1" 5. Eğer s>yari ise Git 9 6. Eğer (a/s) tamsayı ise Yaz s 7. s=s+1 8. Git 5 9. Dur

- Algoritmalarda etkisiz işlemlerden kaçınılmalıdır. Örneğin bir değeri 0 ile toplama veya bir değeri 1 ile çarpma, sonuçta herhangi bir değişikliğe yol açmamasına rağmen programda bir işlem zamanı alacaktır.
- Algoritmalarda bellek verimliliği açısından fazladan veya gereksiz tanımlamaların yapılmaması daha uygundur. Özellikle sonuçtaki asıl çıktıda yer almayacak veya daha sonra kullanılmak üzere saklanmayacak verilerin, ayrı tanımlayıcılarda tutulmaması daha uygun olacaktır.

İyi	Daha iyi
1. Başla 2. Al a, b 3. toplam=a+b 4. fark=-a-b 5. ortalama=toplam/2 6. Yaz toplam, fark, ortalama 7. Dur	1. Başla 2. Al a, b 3. toplam=a+b 4. Yaz toplam, (-1)*toplam, toplam/2 5. Dur

- Algoritma adımlarındaki işlemlerle bir veya birden fazla çıktı (sonuç, ürün, süreç tamamlama vb.) elde edilmelidir. Yani her algoritmanın çıktısı olmalıdır. Buradaki çıktı veya çıktılar; ekranda görüntülenen veya yazıcıdan yazdırılan veriler (sonuçlar, grafikler, resimler vb.), kayıtlar (verileri dosya oluşturarak veya başka çevre birimlere kaydetme, kes-kopyala-yapıştırma vb. işlemler), ses çalma, video oynatma, donanımsal veri alışverişi yapma (örneğin USB porta bağlı bir karttan veri alma-gönderme) gibi somut ve anlamlı/anlaşılır bilgiler veya süreçler olmalıdır.

Bilgi

Fransız filozof, matematikçi ve bilim adamı René Descartes (1596-1650), 1637 yılında yayınladığı "*Discours de la méthode*" (İngilizce: "*Discourse on the Method*"; Türkçe: Yöntem Üzerine Söylem; Tam ismi: "*Discours de la méthode pour bien conduire sa raison, et chercher la vérité dans les sciences*") adlı eserinde; çözüme/sonuca/bilgiye ulaşmak için izlenecek yolu dört başlıkta özetler:

i. Tahmin ve önyargılardan kaçınarak, acele etmeden, doğruluğu kanıtlanmadıkça, hiçbir şeyi (çözüm, bilgi, vb.) kesin olarak doğru kabul etmeyiniz.

ii. Büyük problemleri/bilgileri (güçlükleri), mümkün olduğunca küçük parçalara bölününüz.

iii. Çözüme/bilgiye; çözülmesi/anlaşılması en kolay olan parçalarla başlayıp adım adım daha karmaşık ve zor olanlara doğru - düzenli ve sistematik düşünerek - ilerleyiniz.

iv. Hazırlanan ayrıntılı çözüme/bilgiye ulaşma adımlarının ve genel değerlendirmenin (çözümün/bilginin) - hiçbir olasılığı göz ardı etmeyecek şekilde - eksiksiz ve mükemmel olduğundan emin olunuz.

Algoritma hazırlama adımları (özet)

i. Yapılacak iş/çözülecek problem iyice irdelenir. Tüm olasılıklar gözden geçirilir. İş/problem büyük ise daha küçük parçalara bölünür.

ii. En az işlemle (komutla), en kısa sürede, en doğru-hassas sonuca ulaştıracak çözüm yolu/yöntem veya sadece çözüm yolu/yöntem belirlenir.

iii. Tanımlayıcı isimleri belirlenir.

iv. Algoritmada her işlem adımına bir numara verilir.

v. Problemin çözümü için gerekli olan veriler/işlenecek veriler girilir veya başka ortamlardan alınır.

vi. Yapılacak sonlu sayıdaki işlemler/kullanılacak yöntemler açık, net ve kesin şekilde verilir.

vii. İşlemler sonucunda süreç tamamlanır veya bulunan sonuçlar anlamlı/anlaşılır hale getirilir (görüntüleme, yazdırma, kaydetme vb.).

Algoritma yazma kuralları

i. Algoritmadaki tüm satırlar, 1'den başlayarak tamsayılarla numaralandırılmalıdır.

ii. Bütün algoritmalarda birinci satır "1. Başla" şeklindedir.

iii. Bütün algoritmalar "Dur" ile biter.

iv. Algoritmalarda kullanılan "Git" işlem akışı yönlendirme komutu satır numarasıyla birlikte kullanılmalıdır.

v. Algoritmalarda kullanılabilecek alt program veya fonksiyonlar, tanımlayıcı isimleri ve parametreleriyle birlikte verilmelidir.

vi. Algoritmalardaki adımlar sınırlı sayıda, açık, net ve kesin olmalıdır.

vii. Algoritmalardaki ifadeler anlaşılır ve mümkün olduğunca sade (az ve öz) olmalıdır.

viii. Algoritmalardaki ifadeler; herhangi bir programlama diline, işletim sistemine, donanıma vb. bağımlı olmamalıdır.

Algoritmanın/akış diyagramının avantajları

i. Program yazmayı kolaylaştırır.

ii. Hatalı kodlama oranını azaltır.

iii. Program yazımı için geçen süreyi kısaltır.

iv. Program kontrolünü ve hata takibini kolaylaştırır.

v. Sonradan yapılacak düzenlemelerde kolaylıklar sağlar.

Not

Algoritma;

i. Bilgisayarın gerçekleştireceği tüm işlem adımlarını açık/ayrıntılı ve tam olarak, sıralı bir şekilde anlatmalıdır.
ii. Algoritmadaki adımlar, tüm olasılıklar göz önünde bulundurularak kesin/belirgin olmalıdır.
iii. Algoritmada veri girişi (alınması) olmalıdır.
iv. Algoritmadaki işlemler sınırlı/sonlu olmalıdır.
v. Algoritmada anlaşılır/anlamlı veri çıkışı olmalıdır.
vi. Esnek yapıda, etkin ve hızlı olmalıdır.
vii. Değişikliklere/güncellemelere/geliştirmelere açık olmalıdır.
viii. Programlamaya uygun olup herhangi bir programlama diline bağımlı olmamalıdır.

Not

İyi bir program;

i. İşlemleri, hatasız gerçekleştirip doğru sonuçlar üretmelidir.
ii. Hızlı olmalıdır.
iii. Türüne göre bellekte fazla yer kaplamamalıdır.
iv. Sistem kaynaklarını gereksiz kullanmamalıdır.
v. Kodları veya modülleri sade olup değişiklikler/güncelleştirmeler/geliştirmeler kolaylıkla yapılabilmelidir.
vi. Modüler olup diğer programlarla da uyumlu bir şekilde çalışabilmelidir.

Not

Algoritma;

i. Bilgisayarın gerçekleştireceği tüm işlem adımlarını açık/ayrıntılı ve tam olarak, sıralı bir şekilde anlatmalıdır.
ii. Algoritmadaki adımlar, tüm olasılıklar göz önünde bulundurularak kesin/belirgin olmalıdır.
iii. Algoritmada veri girişi (alınması) olmalıdır.
iv. Algoritmadaki işlemler sınırlı/sonlu olmalıdır.
v. Algoritmada anlaşılır/anlamlı veri çıkışı olmalıdır.
vi. Esnek yapıda, etkin ve hızlı olmalıdır.
vii. Değişikliklere/güncellemelere/geliştirmelere açık olmalıdır.
viii. Programlamaya uygun olup herhangi bir programlama diline bağımlı olmamalıdır.

Not

İyi bir program;

i. İşlemleri, hatasız gerçekleştirip doğru sonuçlar üretmelidir.
ii. Hızlı olmalıdır.
iii. Türüne göre bellekte fazla yer kaplamamalıdır.
iv. Sistem kaynaklarını gereksiz kullanmamalıdır.
v. Kodları veya modülleri sade olup değişiklikler/güncelleştirmeler/geliştirmeler kolaylıkla yapılabilmelidir.
vi. Modüler olup diğer programlarla da uyumlu bir şekilde çalışabilmelidir.

SORULAR

1. Algoritma nedir? Algoritmanın gerekliliğini ve avantajlarını açıklayınız.

2. Algoritma hazırlanırken dikkat edilmesi gereken hususları açıklayınız.

3. Değişken nedir? Programlarda neden değişkenlere ihtiyaç duyulmaktadır?

4. Sayaçlar, nerelerde ve niçin kullanılmaktadır?

5. Aşağıdaki algoritmanın sonucu nedir?

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

6. Aşağıdaki algoritmanın sonucunu hesaplayınız?

```
1. Başla
2. F=1
3. S=20
4. Eğer S<1 ise Git 9
5. S= S-3
6. F= F+S
7. F= F+2
8. Git 4
9. Yaz F
10. Dur
```

7. Girilen üç sayıdan en büyüğünü bulan programın algoritmasını hazırlayınız.

8. Girilen üç sayıyı küçükten büyüğe doğru sıralayan programın algoritmasını hazırlayınız.

9. 1-99 arasındaki tek ve çift sayıların toplamları ile çarpımlarını ayrı ayrı hesaplayan programın algoritmasını hazırlayınız.

10. Klavyeden girilen pozitif bir tamsayının tek mi ve çift mi olduğunu tespit eden programın algoritmasını hazırlayınız.

11. Klavyeden girilen bir tamsayının pozitif, negatif veya sıfır olduğunu tespit eden programın algoritmasını hazırlayınız.

BÖLÜM 3

AKIŞ DİYAGRAMI

Bu bölümde

- Akış diyagramlarının çizimindeki temel şekilleri,
- Akış diyagramlarını okuyup analiz etmeyi,
- Akış diyagramlarının çizimindeki temel adımları öğreneceksiniz.

Bölüm sonunda

- Akış diyagramındaki temel şekilleri açıklayabilecek,
- Akış diyagramlarını anlayıp (okuyup) analiz edebilecek,
- Basit işlemlere ait akış diygramlarını çizebileceksiniz.

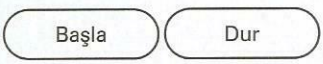
I. GİRİŞ

Programlamanın temeli olan algoritmanın, özel geometrik şekillerle çizilmiş gösterimine '*akış diyagramı*' denir. Yazılımcı; programını herhangi bir dilde kodlamadan önce onun algoritmasını yazabilir veya akış diyagramını çizebilir. Akış diyagramı şekilleri, en genel gösterim biçimleriyle bu bölümde ayrıntılı olarak anlatılacak ve daha sonraki bölümlerde program kodlarıyla birlikte verilecektir.

II. AKIŞ DİYAGRAMINDA KULLANILAN ŞEKİLLER

Akış diyagramında kullanılan temel şekiller Tablo 3.1'de toplu olarak verilmektedir.

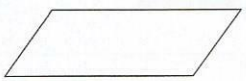
A. Başla/Dur



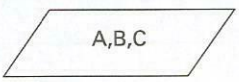
Şekil 3.1: Başla/Dur şekilleri

'`Başla`' ve '`Dur`' işlemleri standarttır. Her akış diyagramı; '`Başla`' şekli ile başlar, '`Dur`' şekli ile biter (Şekil 3.1). İşlemler, bu iki şekil arasında açıklanır.

B. Veri Girişi



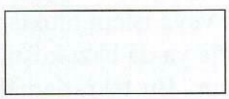
Şekil 3.2: Veri girişi şekli



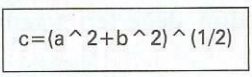
Şekil 3.3: Veri girişi örneği

Bilgisayara klavyeden (dışarıdan) veri/bilgi girişini temsil eden şekildir (Şekil 3.2). Bu şeklin içine, girilen verinin aktarılacağı değişkenin ismi yazılır. Örneğin; şeklin içinde '`N`' yazıyor ve program çalıştırıldığında '`10`' girilirse, bu işlem '`N=10`' anlamındadır. Veri girişi şeklinin içine, birden fazla değişken ismi de yazılabilir. Şekil 3.3 programlama dili ile kodlanıp çalıştırılır ve işlem (program akışı) bu satıra geldiğinde klavyeden '`3`', '`11`' ve '`Teknik`' ifadeleri girilirse; `A=3`, `B=11` ve `C='Teknik'` atamaları gerçekleştirilmiş olur.

C. İşlem



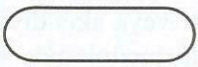
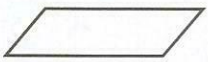
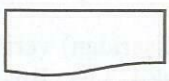
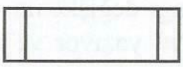
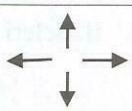
Şekil 3.4: İşlem şekli



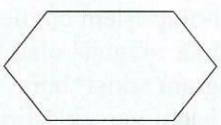
Şekil 3.5: İşlem örneği

Programın çalışması sırasında yapılacak işlemleri ifade etmek için kullanılan şekil olup işlem cümleleri/ifadeleri bilgisayar mantığına uygun olarak aynen yazılır (Şekil 3.4). Program akışı buraya geldiğinde, şeklin içinde yazılı işlem gerçekleştirilir. Birden fazla işlem; aynı şekil içinde, aralarına virgül konularak veya alt alta yazılarak gösterilebilir. Şekil 3.5'teki örnekte; işlem akışı bu şekle/satıra geldiğinde, $c=\sqrt{a^2+b^2}$ işlemi yapılır.

Tablo 3.1: Akış diyagramlarında kullanılan genel şekiller

Şekil	Anlamı
	Başla / Dur
	Veri/bilgi girişi
	İşlem
	Döngü
	Karar/Karşılaştırma
	Veri/bilgi yazma
	Önceden tanımlı işlem (alt program, fonksiyon...)
	Bağlantı
	İşlem akış yönü

D. Döngü



Şekil 3.6: Döngü şekli

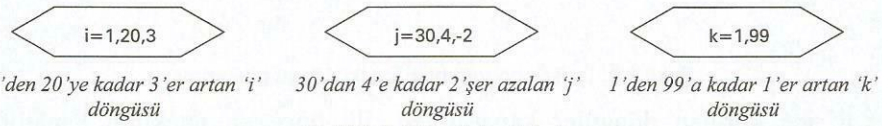
Birçok programda; belirli işlem veya işlem blokları aynı veya farklı/ardışık değerlerle ya da bazı koşullar sağlanıncaya kadar tekrarlanır. Bu tekrarlamalı işlemler '*döngü*' (*çevrim*) olarak isimlendirilir. Döngü şeklinin içine; döngü (çevrim, kontrol) değişkeninin başlangıç-bitiş- adım değerleri yazılır (Şekil 3.6-3.7). Genel yazım şekli, aşağıdaki gibidir:

```
Döngü değişkeni = Başlangıç değeri, Bitiş değeri, Adım değeri
```

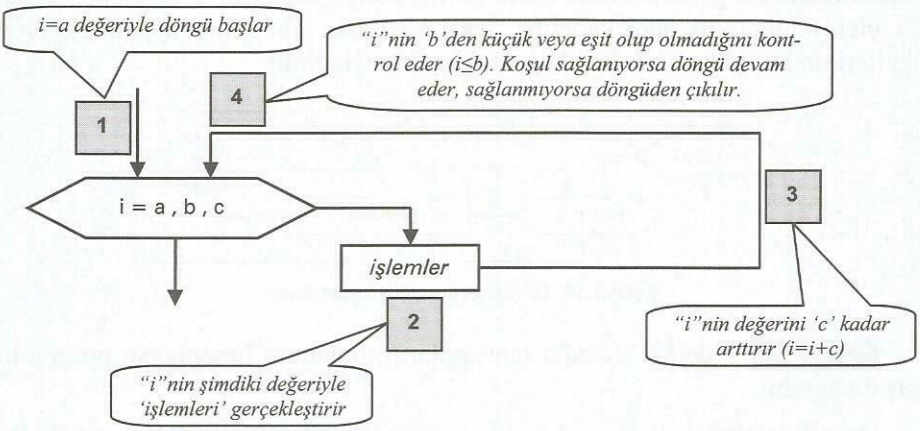
Döngüler, iki şekilde oluşturulabilir:

a. *Artan döngü*: Başlangıç değeri, bitiş değerinden küçüktür (adım değeri pozitiftir).

b. *Azalan döngü*: Başlangıç değeri, bitiş değerinden büyüktür (adım değeri negatiftir).



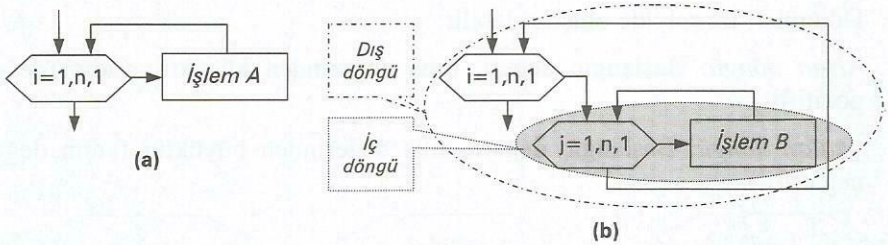
Şekil 3.7: Döngü örnekleri



Şekil 3.8: Uygun değerlerle(a<b ve c>0) örnek artan bir döngünün çalışma mantığı

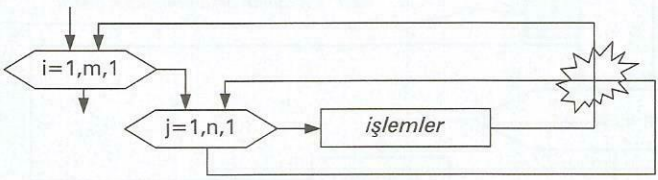
Örnek olarak Şekil 3.8'de artan bir döngünün çalışma mantığı verilmektedir. Döngü değişkeni `i=a` başlangıç değerini alarak döngü devam/çalışma koşulunu (`i≤b`) kontrol eder. Eğer sağlanmıyorsa hiç döngüye girmeden sonraki program adımlarına geçer. Koşul uygunsa/sağlanıyorsa döngü bloğu içindeki işlemler gerçekleşir. Sonraki aşamada döngü değişkeni değerini, adım miktarı kadar arttırır (`i=i+c`). Son aşamada ise döngü değişkeninin, bitiş değerine ulaşıp ulaşmadığını (`i≤b`) kontrol eder. Eğer bitiş değeri, aşıldıysa döngü sonlanır (döngüden çıkılır); aşılmadıysa döngü devam eder. Aslında Şekil 3.8'de "4" ile gösterilen otomatik işlem adımı, "1" ile gösterilen döngü başlangıcında da gerçekleştirilir. Konu anlatımının bütünlüğü açısından; döngünün başlangıç değerinin uygun olduğu varsayımıyla, döngü girişinde "4" ile gösterilen işlem tekrar belirtilmemiştir. Azalan döngünün de çalışması benzer şekildedir; sadece döngü değişkenine arttırma yerine azaltma işlemi uygulanmakta, koşul olarak da döngü değişkeninin bitiş değerinden büyük veya eşit olup olmadığı kontrol edilmektedir.

Birçok programlarda iç içe birden fazla döngü kurulmaktadır. Örnek olarak Şekil 3.9.a'da tek döngü yapısı verilmekte ve bu döngüdeki '*İşlem A*', "`n`" defa gerçekleştirilmektedir. Şekil 3.9.b'de ise iç içe iki döngü bulunmaktadır. Önce en içteki döngü (`J`) tamamlanmakta ve '*İşlem B*', `n.n=n`^{2} defa gerçekleştirilmektedir.



Şekil 3.9: (a) Tek döngü (b) İç içe iki döngü gösterimi

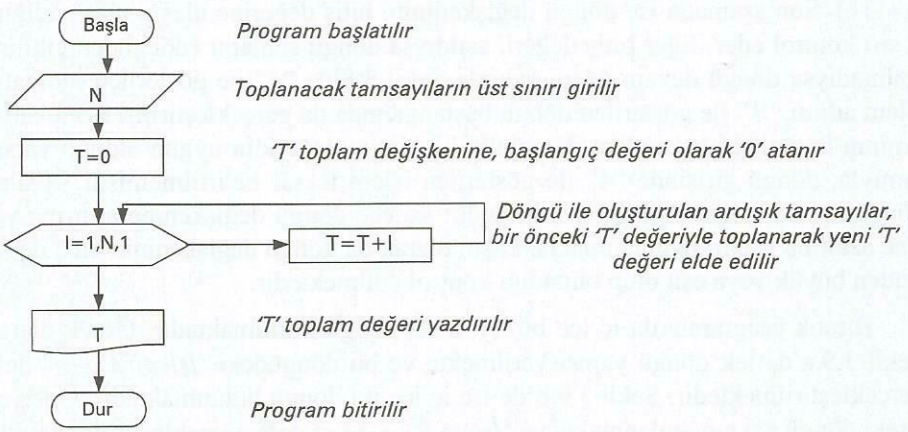
İç içe kurulan döngüler kapatılırken, ilk önce en içtekinin kapatılması/tamamlanması gerekir. Daha sonra bir üst döngü kapatılır. Yani en son açılan (en içteki) döngü, ilk önce kapatılır. Şekil 3.10'daki gibi döngü kapatma (döngü çizgilerinin kesişmesi) kesinlikle gerçekleştirilmemelidir.



Şekil 3.10: Döngülerin yanlış kapatılması

Örnek-3.1: 1'den N'e kadar tamsayıların toplamını hesaplayan programın akış diyagramı.

Öncelikle çizilecek olan akış diyagramında kullanılacak değişken isimleri belirlenir: toplanacak tamsayıların üst sınırı 'N', 1'den üst sınıra kadar olan tamsayıların toplamı 'T' ve tamsayıları üreten döngü değişkeni de 'I'. Birinci adımda toplanacak olan tamsayıların üst sınırının klavyeden girilmesi istenir. İkinci adımda ardışık toplama değişkenine sıfır değeri atanır. Daha sonra 1 ile bu üst sınır arasındaki tamsayıları üretecek olan döngü açılır ve döngü içinde ardışık toplama işlemi gerçekleştirilir.



Klavyeden 'N' için 5 girilirse; programın çalışması Tablo 3.2'deki gibidir.

Tablo 3.2: Örnek-3.1'in çalışmasında oluşan değerler

I	Eski T	Yeni T
1	0	0+1=1
2	1	1+2=3
3	3	3+3=6
4	6	6+4=10
5	10	10+5=*15*

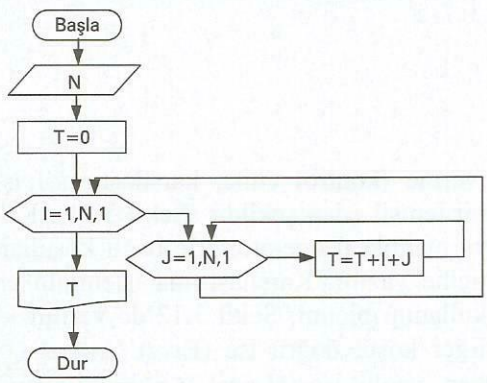
Eğer işlem kutusundaki 'T=0' yerine 'F=1' ve döngü içindeki 'T=T+I' yerine 'F=F*I' yazılırsa, bu program; klavyeden girilen pozitif bir tamsayının faktöriyelini hesaplar (Tablo 3.3).

Tablo 3.3: Faktöriyel alma işleminde oluşan değerler

I	Eski F	Yeni F
1	1	1*1=1
2	1	1*2=2
3	2	2*3=6
4	6	6*4=24
5	24	24*5=*120*

Örnek-3.2: İç içe iki döngülü akış diyagramındaki işlemlerin incelenmesi.

İç içe döngülerde, dıştaki döngünün her çalıştırılma adımında içteki döngü yeniden baştan başlar. Örnekteki 'I' dış döngüsünün her bir değerinde 'J' iç döngüsü tekrar baştan başlar ve içindeki 'T=T+I+J' işlemini yapar. Akış diyagramının N=3 değeri için çalışması, Tablo 3.4'teki gibi olur. Tablo 3.4'te de görüldüğü gibi, 'I' nın her değerinde ilk önce en içteki döngü (J) tamamlanmakta, daha sonra dış döngüye (I) çıkılmaktadır.



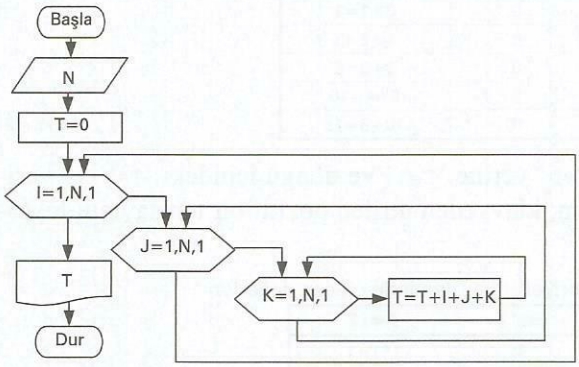
Tablo 3.4: Örnek-3.2'nin çalışma prensibi

I	J	Eski T	Yeni T
1	1	0	0+1+1=2
1	2	2	2+1+2=5
1	3	5	5+1+3=9
2	*1*	*9*	*9+2+1=12*
2	*2*	*12*	*12+2+2=16*
2	*3*	*16*	*16+2+3=21*
3	1	21	21+3+1=25
3	2	25	25+3+2=30
3	3	30	30+3+3=36

Örnek-3.3: İç içe üç döngülü akış diyagramının incelenmesi.

Böyle bir programda her bir 'I' değeri için 'N' sayıda 'J' döngüsü ve her bir 'J' değeri için de 'N' sayıda 'K' döngüsü tekrarlanır. İç içe üç döngüsü bulunan akış diyagramında N=2 değeriyle programın çalışması durumunda, Tablo 3.5'teki değerler oluşur. Tablo 3.5'ten de görüldüğü gibi 'I' döngüsü 'J' döngü-

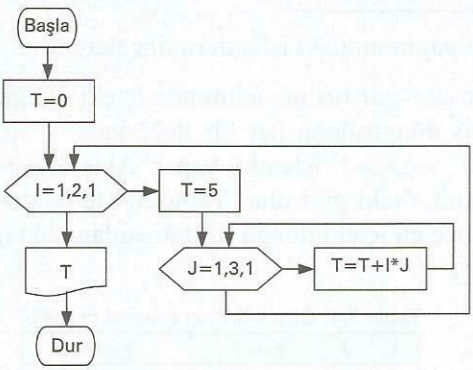
sünü, 'J' döngüsü de 'K' döngüsünü başlatmaktadır. En içteki 'K' döngüsü tamamlandığında bir üst döngü olan 'J'ye çıkılmakta ve 'J' döngüsü tamamlanmadıysa sıradaki değeriyle yeniden 'K' döngüsünü başlatmaktadır.



Tablo 3.5: Örnek-3.3'teki akış diyagramının çalışma prensibi

I	J	K	Eski T	Yeni T
1	1	1	0	0+1+1+1=3
1	1	2	3	3+1+1+2=7
1	2	1	7	7+1+2+1=11
1	2	2	11	11+1+2+2=16
2	1	1	16	16+2+1+1=20
2	1	2	20	20+2+1+2=25
2	2	1	25	25+2+2+1=30
2	2	2	30	30+2+2+2=36

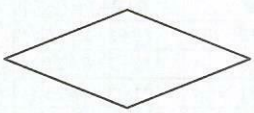
Örnek-3.4: Aşağıdaki akış diyagramının sonucunu bulunuz.



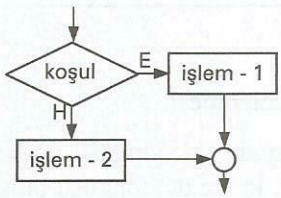
Tablo 3.6: Örnek-3.4'teki akış diyagramının çalışma prensibi

I	J	Eski T	Yeni T
1	1	5	5+1*1=6
1	2	6	6+1*2=8
1	3	8	8+1*3=11
2	1	5	5+2*1=7
2	2	7	7+2*2=11
2	3	11	11+2*3=**17**

E. Karar (Karşılaştırma)



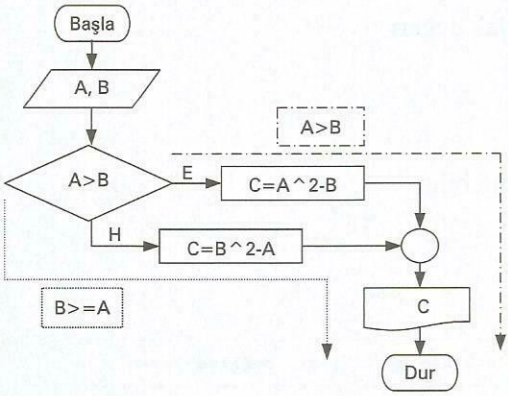
Şekil 3.11: Karar (karşılaştırma) şekli



Şekil 3.12: Karşılaştırma işlemi şekli

Karar verme (kontrol etme, karşılaştırma) işlemlerini temsil eden şekildir (Şekil 3.11). Koşul veya mantıksal operatörlerle bağlı koşullar, şeklin içine yazılır. Karşılaştırma işleminin en genel kullanım biçimi, Şekil 3.12'de verilmektedir. Eğer koşul doğru ise (**E***vet*) `'işlem-1'` yapılırken, yanlış ise (**H***ayır*) `'işlem-2'` yapılır. İşlemlerden sonra, program akışı, bağlantı noktasında birleşerek alt satırlara gider. İşlem; sadece koşulun doğru (veya yanlış) olduğu durumda mevcutsa, diğer işlem kutusunu çizmeye gerek yoktur (Tablo 3.7).

Örnek-3.5: Girilen iki değerin karşılaştırılması sonucuna göre farklı işlemlerin gerçekleştirilmesinin istendiği bir programın akış diyagramının incelenmesi.

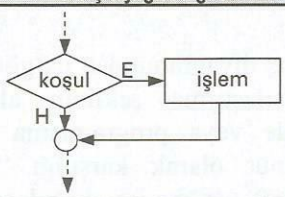
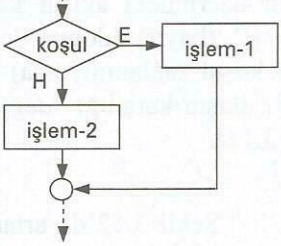
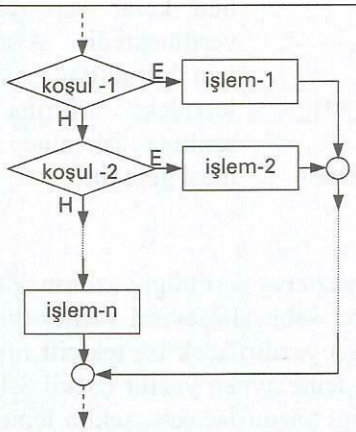


Akış diyagramında, girilen `A` ve `B` sayılarından:

- `A` büyükse; $C=A^2-B$ hesaplanıp, işlem sonucu yazdırılır.
- `B` büyük veya eşitse; $C=B^2-A$ hesaplanıp, işlem sonucu yazdırılır.

Örneğin; `A=5, B=4` için ekrana *'21'*; `A=3, B=6` için de *'33'* değeri yazılır.

Tablo 3.7: Karşılaştırma ifadelerinin genel kullanım şekilleri

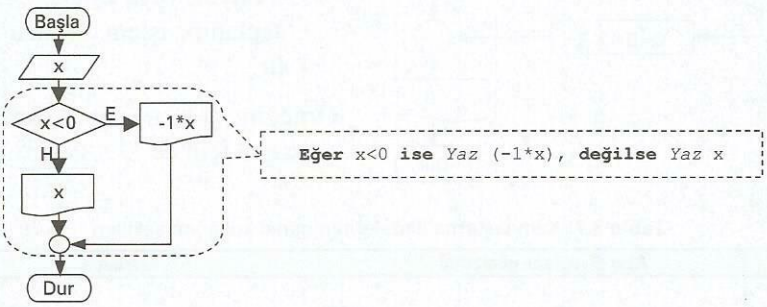
Biçim	Akış diyagramı gösterimi	Açıklama
Yarım		"*koşul*" doğru ise (*E*) "*işlem*" yapılır, değilse (*H*) akış devam eder.
Tam		"*koşul*" doğru ise (*E*) "*işlem-1*", değilse (*H*) "*işlem-2*" yapılır.
Çok koşullu		"*koşul-1*" doğru ise (*E*) "*işlem-1*", değilse (*H*) "*koşul-2*" kontrol edilir ve doğru ise (*E*) "*işlem-2*" yapılır. "*koşul-2*" yanlış ise (*H*) sıradaki koşul kontrol edilir ve doğru olup olmamasına göre işlem devam eder. Eğer tüm koşullar yanlış ise "*işlem-n*" yapılır.

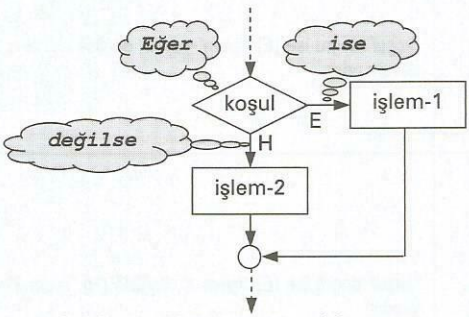
Örnek-3.6: Klavyeden girilen bir sayının mutlak değerini ekrana yazdıran programın akış diyagramının incelenmesi.

Herhangi bir x sayısının $|x|$ mutlak değeri

$$|x| = \begin{cases} -1.x & , \ x < 0 \\ x & , \ x \geq 0 \end{cases}$$

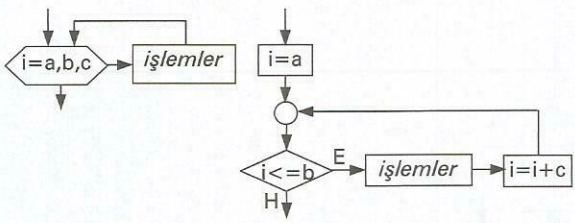
şeklinde matematiksel olarak ifade edilebilir.





Şekil 3.11: Karşılaştırma şekli ve algoritmadaki karşılığı

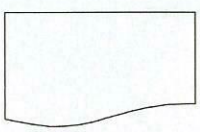
Akış diyagramından da görüldüğü gibi karşılaştırma şeklinin algoritmadaki ifade veya programlama dillerindeki komut olarak karşılığı "*Eğer*"; 'E' (evet) (doğru ise, koşul sağlanıyorsa) kolu üzerindeki akışın karşılığı "*ise*" ve 'H' (hayır) (doğru değilse, yanlış ise, koşul sağlanmıyorsa) kolu üzerindeki akışın karşılığı "*değilse*"dir (Şekil 3.11).



Şekil 3.12: Artan döngünün karar yapısı eşdeğeri

Şekil 3.12'de artan bir döngünün karar yapısıyla eşdeğeri verilmektedir. Azalan döngü için koşul ifadesinin yönü ve içerideki arttırma işleminin azaltma işlemine dönüştürülmesi gerekir.

F. Yazdırma/Çıktı



Şekil 3.13 Çıktı şekli

Ekrana veya yazıcıya veri/bilgi yazdırmak (belge) için kullanılır. Eğer sabit alfasayısal veriler (karakter, kelim, cümle vs.) yazdırılacak ise tek/çift tırnak arasına çıktı şeklinin içine aynen yazılır (Şekil 3.13). Bir değişkenin içeriği yazdırılacaksa, şeklin içine ismi yazı-

lır. Birden fazla değişken içeriği aralarına virgül konularak tek şekil içinde gösterilebilir. Şeklin içinde son karakter olarak 'virgül' veya "noktalı virgül" varsa; veri yazdırılacak ve imleç aynı satırda kalacak anlamındadır. Çıktı şekli içine atama operatörü olmadan işlem de yazılabilir. Bunun anlamı "işlemi yap, sonucunu yazdır" şeklindedir.

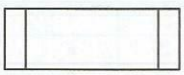
Örnek-3.7: Tablo 3.8'de yazdırma/çıktı örnekleri yer almaktadır.

Tablo 3.8: Örnek yazdırma işlemleri

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

G. Önceden tanımlı işlem

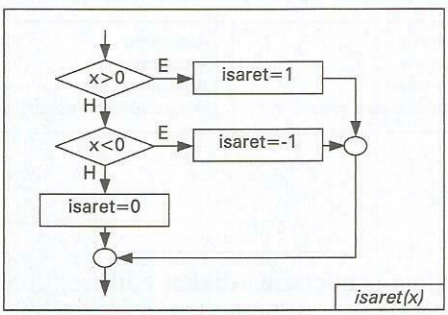
Büyük programlar, birçok küçük programın (modülün) birleşiminden meydana gelmektedir. Benzer şekilde küçük programlarda da bazı işlemler ayrı olarak tanımlanıp (alt program, fonksiyon vb.) kullanılmaktadırlar. Böylece belirli işlem veya işlemlerin yapılması bir defa kodlanarak parametresiz veya parametreli olarak ana programdan çağırılmaktadırlar. Böylece programın işlevselliği, modülerliği, hızı gibi birçok özelliği olumlu yönde etkilenmektedir.

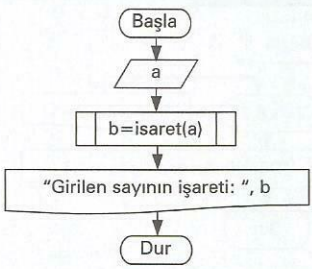


Şekil 3.14 Önceden tanımlı işlem

Önceden tanımlı işlemler, isimleri ve varsa parametreleriyle birlikte Şekil 3.14'tekinin içine yazılır.

Örnek-3.8: Bir sayının işaretini (pozitif, negatif veya sıfır olup olmadığını) bulan alt program yazılıp ana programda aşağıdaki gibi kullanılabilir.





H. Bağlantı

Şekil 3.15 Bağlantı şekli

Şekil 3.16 Bağlantı yapma

Bağlantı şekli (daire) (Şekil 3.15), genel anlamda işlem akışlarını birleştiren bir yer olup

i. Farklı yerlere dallanan işlem akışlarını toplamak,

ii. Akış diyagramı bir sayfaya sığmadığında diğer sayfadaki akış diyagramıyla bağlantı kurmak,

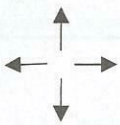
iii. Parça parça çizilen akış diyagramları arasında bağlantı kurmak için kullanılır.

Parçalı program yazılması veya diyagramın çizilen yere sığmaması durumlarında; bağlantı yapılacak uçlara, bağlantı şekli çizilir ve içine aynı harf/karakter dizisi veya rakam/sayı yazılır (Şekil 3.16).

Not

Genel olarak algoritmalarda/akış diyagramlarında/programlarda işlem akışı adım adım ve doğrusal bir şekilde olmalıdır.

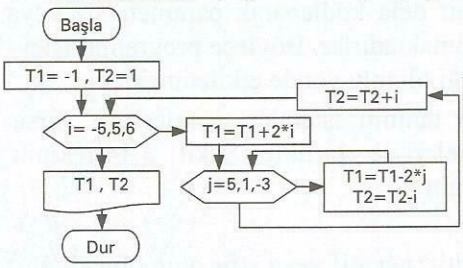
I. İşlem Akış Yönleri



Şekil 3.17 İşlem akış yönü şekilleri

Akış diyagramlarında işlem akışının hangi yönde olduğunu gösteren oklardır. İşlem akış yönüne uygun olan ok kullanılır (Şekil 3.17).

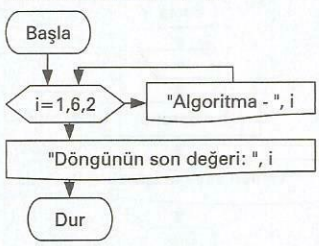
Örnek-3.9: Aşağıdaki akış diyagramının sonuçlarını elde ediniz.



Tablo 3.9: Örnek-3.9'un çalıştırma adımlarının sonuçları

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

Örnek-3.10: Aşağıdaki akış diyagramının ekran çıktısını elde ediniz.



Algoritma - 1
Algoritma - 3
Algoritma - 5
*Döngünün son değeri: **7***

Algoritma - 1
Algoritma - 3
Algoritma - 5
*Döngünün son değeri: **5***

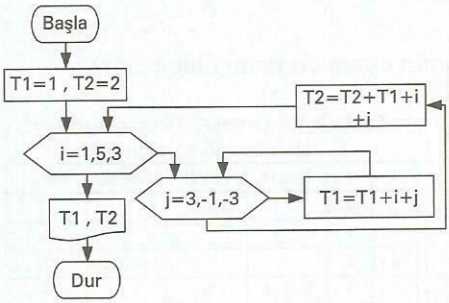
Yukarıdaki algoritmanın çalıştırılma sonuçlarına dikkat edilirse, döngünün son değeri 7 veya 5 şeklinde farklı olmaktadır. Aslında döngü değişkeni en son

7 değerini almakta ve böylece döngünün devam koşulu sağlanmadığından döngü sonlanmaktadır. Ancak bazı programlama dillerinde döngü çıkışında, son değer olarak döngünün son olarak çalıştığı (devam koşulunun sağlandığı) değer atanmaktadır. Bu bölümdeki akış diyagramların analizinde; ikinci durum, yani çıkış değeri olarak döngünün çalıştığı son değer kullanılacaktır.

Not

Programlama dillerine göre döngü çıkışındaki değerlere dikkat ediniz. Bazı dillerde son değer olarak koşulu sağlatmayan değer (adım değeri kadar arttırılmış olan son değer) atanırken bazılarında da koşulu sağlatan (döngünün çalıştığı) son değer atanır.

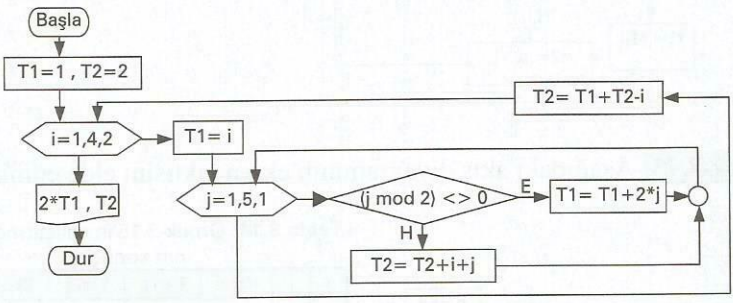
Örnek-3.11: Aşağıdaki akış diyagramının sonuçlarını elde ediniz.



Tablo 3.10: Örnek-3.11'in çalıştırma adımlarının sonuçları

i	j	Eski T1	Yeni T1	Eski T2	Yeni T2
1	3	1	1+1+3=5	2	-
	0	5	5+1+0=6	2	2+6+1+0=9
4	3	6	6+4+3=13	9	-
	0	13	13+4+0=17	9	9+17+4+0=30

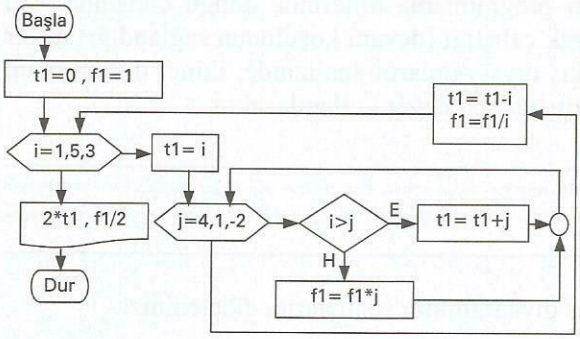
Örnek-3.12: Aşağıdaki akış diyagramının ekran çıktısını elde ediniz.



Tablo 3.11: Örnek-3.12'nin çalıştırma adımlarının sonuçları

i	j	Eski T1	Yeni T1	Eski T2	Yeni T2
1	1	~~1~~ 1	3		
	2			2	5
	3	3	9		
	4			5	10
	5	9	19	10	28
3	1	~~19~~ 3	5		
	2			28	33
	3	5	11		
	4			33	40
	5	11	21	40	58
					Ekran çıktısı: 42 58

Örnek-3.13: Aşağıdaki akış diyagramının ekran çıktısını elde ediniz.

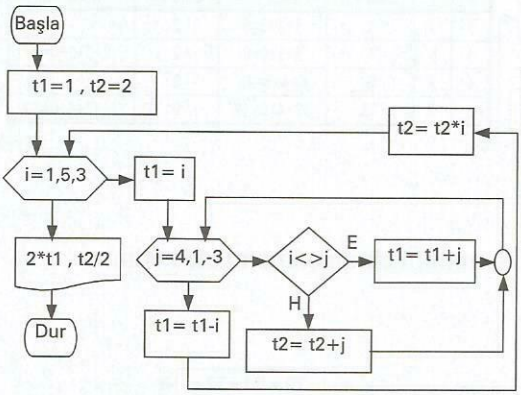


Tablo 3.12: Örnek-3.13'ün çalıştırma adımlarının sonuçları

i	j	Eski t1	Yeni t1	Eski f1	Yeni f1
1	4	~~0~~ 1		1	4
	2			4	8
4	4	~~0~~ 4		8	32
	2	4	6		

Ekran çıktısı: **4 4**

Örnek-3.14: Aşağıdaki akış diyagramının ekran çıktısını elde ediniz.

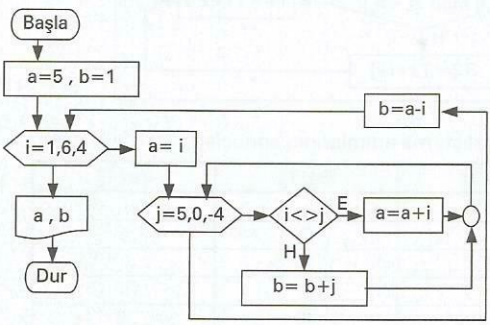


Tablo 3.13: Örnek-3.14'ün çalıştırma adımlarının sonuçları

i	j	Eski t1	Yeni t1	Eski t2	Yeni t2
1	4	~~1~~ 1	5		
	1			2	3
4	4			~~3~~ 3	7
	1	~~5~~ 4 4	5		
		~~5~~ 1		~~7~~ 28	

Ekran çıktısı: **2 14**

Örnek-3.15: Aşağıdaki akış diyagramının ekran çıktısını elde ediniz.



Tablo 3.14: Örnek-3.15'in çalıştırma adımlarının sonuçları

i	j	Koşul	Eski a	Yeni a	Eski b	Yeni b
1	5	1<>5	~~5~~ , 1	2	1	
	1	1<>1				~~2~~ , 1
5	5	5<>5	5		1	6
	1	5<>1		10	6	5

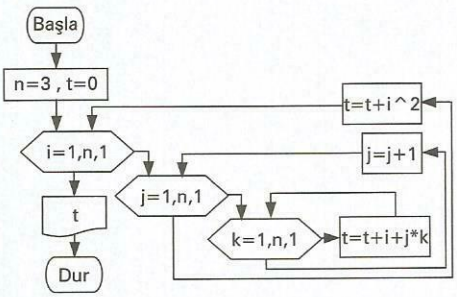
Ekran çıktısı: **10 5**

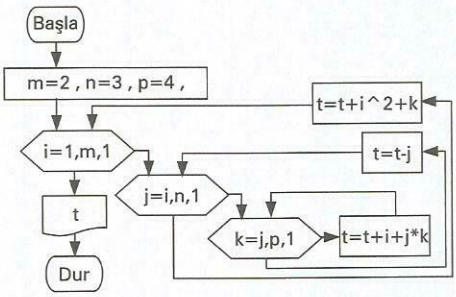
SORULAR

1. Algoritma ile akış diyagramı arasındaki farklar nelerdir? Birbirlerine göre avantaj ve dezavantajlarını açıklayınız.

2. Akış diyagramı hazırlarken dikkat edilmesi gereken hususlar nelerdir?

3. Aşağıdaki akış diyagramlarının sonuçlarını hesaplayınız.





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

5. Klavyeden girilen ismi, yine klavyeden istenen sayı kadar alt alta yazdıran programın akış diyagramını çiziniz?

6. Klavyeden metre (m) cinsinden girilen uzunluğu, kilometre (km) ve santimetre (cm) cinsine dönüştürüp yazdıran programın akış diyagramını çiziniz?

7. Klavyeden üç kenarı girilen üçgenin türünü (eşkenar, ikizkenar veya çeşitkenar) tespit edip yazdıran programın akış diyagramını çiziniz?

8. Klavyeden girilen bir sayının yine klavyeden istenen yüzdesini hesaplayıp yazdıran programın akış diyagramını çiziniz?

9. Klavyeden bir ürünün fiyatı ve KDV oranı istenmektedir. Buna göre ürünün KDV'li satış fiyatını hesaplayıp yazdıran programın akış diyagramını çiziniz?

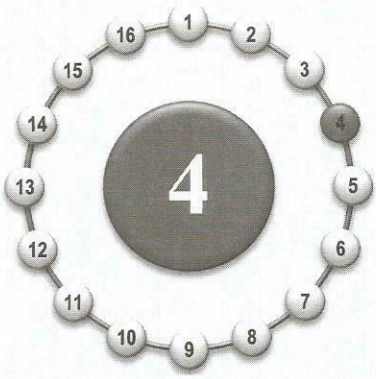
10. Klavyeden bir ürünün fiyatı ve kâr/zarar oranı istenmektedir. Buna göre ürünün satış fiyatını hesaplayıp yazdıran programın akış diyagramını çiziniz?

11. Klavyeden 1-7 arasında bir tamsayı istenmektedir. Bu tamsayıya göre haftanın ilgili gününü kelime/isim olarak yazdıran programın akış diyagramını çiziniz?

12. Klavyeden girilen 10 sayının ortalamasını hesaplayıp yazdıran programın akış diyagramını çiziniz?

BÖLÜM 4

PROGRAMLAMA DİLLERİ



Bu bölümde

- C, C++, C#, Java, MATLAB ve Python'daki veri tiplerini,
- C, C++, C#, Java, MATLAB ve Python program yapılarını,
- C, C++, C#, Java, MATLAB ve Python programlarındaki benzerlik ve farklılıkları öğreneceksiniz.

Bölüm sonunda

- C, C++, C#, Java, MATLAB ve Python'daki veri tiplerini ve program yapılarını kavrayacak
- C, C++, C#, Java, MATLAB ve Python programlarını karşılaştırabileceksiniz.

I. GİRİŞ

Bu bölümde fazla ayrıntıya inilmeden C, C++, C#, Java, MATLAB ve Python programlama dillerinin genel yapıları anlatılacak ve beşinci bölümde de akış diyagramlarından kodlamaya nasıl geçileceği sunulacaktır. Altıncı bölümden itibaren ise çözülen problemlerin algoritma/akış diyagramlarının yanında, bu altı programlama dilindeki kodları (programları/yazılımları) da verilecektir.

II. PROGRAMLAMA DİLLERİNDEKİ VERİ TİPLERİ

Bilgisayarda işlenen veriler temelde iki türdür: Sayısal ve alfasayısal. Sayısal veriler bilgisayara, belirli bir tabanda veya üstel biçimde aktarılabilirler/girilebilirler. Sayının hangi tabanda olduğunu belirtmek için; programlama dillerinde, sayının önüne belirli simgeler/karakterler yazılır (Tablo 4.1). Alfasayısal veriler ise programlama dillerinde tek/çift tırnak içinde verilirler (Tablo 4.2).

Tablo 4.1: Sayısal veri tiplerinde taban/biçim belirtme simgeleri

Taban/biçim	C / C++ / C# / Java	MATLAB / Python
10 (decimal)	-	-
16 (hexadecimal)	0x*sayı* veya 0X*sayı*	-
Üstel	e*üs* veya E*üs*	e*üs* veya E*üs*

Tablo 4.2: Programlama dillerinde alfasayısal verilerin gösterilmesi/aktarılması

Alfasayısal veri	C / C++ / C# / Java	MATLAB	Python
Karakter	Tek tırnak içinde	Tek tırnak içinde	Tek, çift veya üç tane çift tırnak içinde
Karakter dizisi	Çift tırnak içinde	Tek tırnak içinde	Tek, çift veya üç tane çift tırnak içinde

A. C'deki Veri Tipleri

C dilinde de kullanılacak değişkenler ve veri tipleri, programda önceden bildirilmek zorundadır.

1. Sayısal Veri Tipleri

a. Tamsayı Veri Tipleri

Tamsayı veriler, sayının alabileceği en küçük ve en yüksek değerlere bağlı olarak değişik şekillerde tanımlanırlar (Tablo 4.3).

Tablo 4.3: C'deki temel tamsayı veri tipleri

Veri tipi	Aktarılabilecek en küçük değer	Aktarılabilecek en büyük değer	Bellekte kapladığı alan (byte)
char	-2^7	2^7-1	1
int	-2^{15}	$2^{15}-1$	2

b. Ondalıklı Sayı Veri Tipleri

Ondalıklı sayısal değişkenler, Tablo 4.4'deki veri tiplerinin herhangi birisiyle tanımlanabilirler.

Tablo 4.4: C'deki temel ondalıklı sayı veri tipleri

Veri tipi	Alt sınır	Üst sınır	Bellekte kapladığı alan (byte)
float	$3{,}4.10^{-38}$	$3{,}4.10^{38}$	4
double	$1{,}7.10^{-308}$	$1{,}7.10^{308}$	8

Tablo 4.3-4.4'te verilen tamsayı ve ondalıklı sayı tipleri, C'nin temel sayısal veri tipleridir. Bunların yeterli olmadığı veya kullanılacak bellek alanının önemli olduğu durumlarda tip sıfatları kullanılır. Tablo 4.5'deki tip sıfatları, C'deki temel sayısal veri tiplerinin önlerine yazılarak değişik veri tipleri elde edilir. Tip sıfatları kullanılarak oluşturulabilen sayısal veri tiplerinden bazıları Tablo 4.6'da toplu olarak verilmektedir.

C

```
/* Sayısal veri tipleri */
#include <stdio.h>
#include <conio.h>
#include <math.h>
int a; float b; double c;
void main()
{   clrscr();printf("Pi sayısı:\n"); a=M_PI; b=M_PI; c=M_PI;
    printf("int tipinde       => %i\n",a);
    printf("float tipinde     => %0.20f\n",b);
    printf("double tipinde    => %0.20f\n",c);    getch();  }
```

```
Pi sayısı:
int tipinde       => 3
float tipinde     => 3.14159274101257324200
double tipinde    => 3.14159265358979311600
```

Tablo 4.5: C'deki tip sıfatları

Tip sıfatı	Anlamı
signed	İşaretli
unsigned	İşaretsiz
short	Kısa
long	Uzun

Tablo 4.6: Tip sıfatları kullanılarak oluşturulmuş bazı sayısal veri tipleri

Veri tipi	Kısa yazılışı	Alt sınır	Üst sınır	Uzunluk (Byte)
unsigned char	-	0	2^8-1	1
signed char	**char**	-2^7	2^7-1	1
enum	-	-2^{15}	2^{15}-1	2
unsigned int	**unsigned**	0	2^{16}-1	2
short int	**short**	-2^{15}	2^{15}-1	2
unsigned long	-	0	2^{32}-1	4
long	-	-2^{31}	2^{31}-1	4
long double	-	$3{,}4.10^{-4932}$	$1{,}1.10^{4932}$	10

2. Alfasayısal Veri Tipleri

C'deki alfasayısal veri tipi, Tablo 4.7'de verilen "`char`" olup veriler tek/çift tırnak içinde verilirler ve "\0" ile sonlanırlar. Bu veriler, dizi şeklinde kullanılabilir ve işlenebilirler.

Tablo 4.7: C'deki alfasayısal veri tipi

Veri tipi	Tanımlama
char	char *değişken*[n]

Aşağıdaki programda 'a' için 5 karakterlik yer ayrılmakta ve "strcpy" komutu ile 'SAÜ' verisi aktarılarak oluşan bellek yerleşimi verilmektedir.

C

```
/* Sayısal veri tipleri */
#include <stdio.h>
#include <conio.h>
#include <string.h>
char a[5];
void main() { clrscr(); strcpy(a,"SAÜ"); getch(); }
```

0. byte	1. byte	2. byte	3. byte	4. byte
a[0]	a[1]	a[2]	a[3]	a[4]
S	A	Ü		

B. C++'daki Veri Tipleri

C dillerinde olduğu gibi C++'da da kullanılacak olan değişkenler ve veri tipleri, önceden bildirilmek zorundadır. C'deki veri tipleri ayrıntılı bir şekilde anlatıldığı için C++'takiler özetlenecektir.

1. Sayısal Veri Tipleri

a. Tamsayı Veri Tipleri

C++'taki temel tamsayı veri tipleri ve sınırları Tablo 4.8'de verilmektedir.

Tablo 4.8: C++'daki temel tamsayı veri tipleri

Veri tipi	Aktarılabilecek en küçük değer	Aktarılabilecek en büyük değer	Bellekte kapladığı alan (byte)
char	-2^7	2^7-1	1
short	-2^{15}	$2^{15}-1$	2
int	-2^{31}	$2^{31}-1$	4
long	-2^{31}	$2^{31}-1$	4

b. Ondalıklı Sayı Veri Tipleri

Ondalıklı sayılar için Tablo 4.9'daki veri tipleri kullanılabilir.

Tablo 4.9: C++'daki temel ondalıklı sayı veri tipleri

Veri tipi	Alt sınır	Üst sınır	Bellekte kapladığı alan (byte)	Duyarlılık (hane)
float	$3{,}4.10^{-38}$	$3{,}4.10^{38}$	4	~ 7
double	$1{,}7.10^{-308}$	$1{,}7.10^{308}$	8	~ 15

Tablo 4.10'daki tip sıfatları kullanılarak oluşturulabilen sayısal veri tiplerinden bazıları Tablo 4.11'de toplu olarak verilmektedir.

Tablo 4.10: C++'daki tip sıfatları

Tip sıfatı	Anlamı
signed	İşaretli
unsigned	İşaretsiz
short	Kısa
long	Uzun

Tablo 4.11: Tip sıfatları kullanılarak oluşturulmuş bazı sayısal veri tipleri

Veri tipi	Kısa yazılışı	Alt sınır	Üst sınır	Uzunluk (Byte)
unsigned short	-	0	2^{16}-1	2
unsigned int	unsigned	0	2^{32}-1	4
unsigned long	-	0	2^{32}-1	4
long double		1,7.10^{-308}	1,7.10^{308}	8

2. Alfasayısal Veri Tipleri

C'de olduğu gibi C++'daki alfasayısal veri tipi, Tablo 4.12'de verilen "`char`"dır.

Tablo 4.12: C++'daki alfasayısal veri tipi

Veri tipi	Tanımlama
char	char *değişken*[n]

C. C#'daki Veri Tipleri

C ve C++ dillerinde olduğu gibi C#'da da kullanılacak olan değişkenler ve veri tipleri, önceden (başta veya kullanılmadan önce) bildirilmek zorundadır.

1. Sayısal Veri Tipleri

a. Tamsayı Veri Tipleri

C#'taki temel tamsayı veri tipleri ve sınırları Tablo 4.13'te verilmektedir.

Tablo 4.13: C#'daki temel tamsayı veri tipleri

Veri tipi	.NET tipi	Sonek	Aktarılabilecek en küçük değer	Aktarılabilecek en büyük değer	Bellekte kapladığı alan (byte)
byte	System.Byte		0	2^8-1	1
sbyte	System.SByte		-2^7	2^7-1	1
short	System.Int16		-2^{15}	2^{15}-1	2
ushort	System.UInt16		0	2^{16}-1	2
int	System.Int32		-2^{31}	2^{31}-1	4
uint	System.UInt32	U, u	0	2^{32}-1	4
long	System.Int64	L , *(l)*	-2^{63}	2^{63}-1	8
ulong	System.UInt64	UL,ul,Ul,uL,LU, lu,Lu,lU,*U,u,L,l*	0	2^{64}-1	8

b. Ondalıklı Sayı Veri Tipleri

Ondalıklı sayılar için Tablo 4.14'teki veri tipleri kullanılabilir.

Tablo 4.14: C#'daki temel ondalıklı veri tipleri

Veri tipi	.NET tipi	Sonek	Aktarılabilecek en küçük değer	Aktarılabilecek en büyük değer	Duyarlılık (hane)	Bellek alanı (byte)
float	System.Single	F, f	±1,5.10^{-45}	±3,4.10^{38}	7	4
double	System.Double	D, d	±5,0.10^{-324}	±1,7.10^{308}	15-16	8
decimal	System.Decimal	M, m	±1,0.10^{-28}	±7,9.10^{28}	28-29	16

2. Alfasayısal Veri Tipleri

C#'taki alfasayısal veri tipleri, Tablo 4.15'te verilmektedir.

Tablo 4.15: C#'daki alfasayısal veri tipleri

Veri tipi	.NET tipi	Aktarılabilecek karakterler	Bellek alanı (byte)
char	System.Char	U+0000 ile U+ffff aralığında UNICODE tek karakter	2
string	System.String	Karakter dizisi	-

D. Java'daki Veri Tipleri

Java programlarında da C, C++ ve C# dillerinde olduğu gibi kullanılacak olan değişkenler ve veri tipleri bildirilmelidir.

1. Sayısal Veri Tipleri

a. Tamsayı Veri Tipleri

Tamsayılar için mevcut veri tipleri ve sınırları Tablo 4.16'da verilmektedir.

Tablo 4.16: Java'daki tamsayı veri tipleri

Veri tipi	Aktarılabilecek en küçük değer	Aktarılabilecek en büyük değer	Bellekte kapladığı alan (byte)
byte	-2^7	2^7-1	1
short	-2^{15}	$2^{15}-1$	2
int	-2^{31}	$2^{31}-1$	4
long	-2^{63}	$2^{63}-1$	8

b. Ondalıklı Sayı Veri Tipleri

Ondalıklı sayılar için Tablo 4.17'deki veri tipleri kullanılabilir.

Tablo 4.17: Java'daki ondalıklı sayı veri tipleri

Veri tipi	Alt sınır	Üst sınır	Bellekte kapladığı alan (byte)
float	$3{,}4.10^{-38}$	$3{,}4.10^{38}$	4
double	$1{,}7.10^{-308}$	$1{,}7.10^{308}$	8

2. Alfasayısal Veri Tipleri

Java'da tek karakter ve karakter grubu (kelime, cümle vb.) için kullanılabilecek iki veri tipi olup (Tablo 4.18) aynı C/C++/C# programlarında olduğu gibi bir boyutlu dizi olarak işlenebilmektedirler.

Tablo 4.18: Java'daki alfasayısal veri tipleri

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

E. MATLAB'taki Veri Tipleri

MATLAB kodlarında kullanılacak olan değişkenlerin ve tiplerinin, önceden bildirilme zorunluluğu yoktur. Burada da tamsayı (`int8`, `int16`, `int32`, `int64`, `uint8`, `uint16`, `uint32`), ondalıklı sayı (`single`, `double`), alfasayısal (`char`) gibi birçok veri tipi bulunmaktadır.

F. Python'daki Veri Tipleri

MATLAB'ta olduğu gibi Python'da da kullanılacak değişkenlerin ve tiplerinin, özel olarak önceden bildirilme zorunluluğu yoktur. Burada da sayısal (`int`, `long`, `float`, `complex`) ve alfasayısal gibi veri tipleri yer almaktadır.

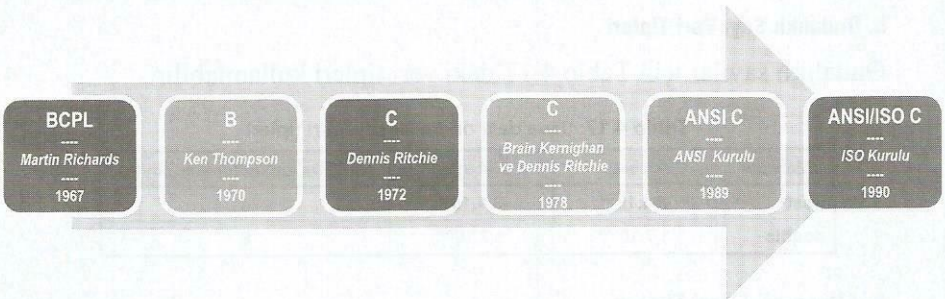
Not

Yukarıda belirtilen veri tipi aralıklarında/sınırlarında; kullanılan yazılımın sürümüne, işletim sistemine vb. göre değişiklikler olabilir.

III. PROGRAMLAMA DİLLERİNİN GENEL YAPILARI

A. C Dilinin Yapısı

C programlama dili, 1972 yılında AT&T Bell Telefon Laboratuvarları'nda; Harvard Üniversitesi fizik lisans, uygulamalı matematik yüksek lisans ve matematik alanında doktora derecelerini alan ABD'li *Dennis Ritchie* (1941-2011) tarafından geliştirilmiş genel amaçlı bilgisayar programlama dilidir. C dilinin tarihi gelişim süreci, Şekil 4.1'deki gibi özetlenebilir. Günümüzde de çok popüler olan C ve C++ programlama dili; UNIX, Microsoft Windows, Linux gibi birçok işletim sistemimin de yazımında kullanılmıştır.



Şekil 4.1: C dilinin tarihsel özet gelişimi

C dilinin temel yazım özellikleri

i. Program yazımı belirli kalıpta, bloklar halinde olur.
ii. Bloklar, { } parantezleri ile oluşturulur.
iii. Komutlar aynı veya alt alta satırlara yazılabilirler.
iv. Tüm komutlar, noktalı virgül (;) ile biter. Yalnız blok başlatan ifadelerden sonra noktalı virgül kullanılmaz.
v. Programda kullanılan tüm değişkenler ve veri tipleri bildirilir.
vi. Programda kullanılacak olan komutların bulunduğu kütüphaneler aktifleştirilir/çağırılır.

1. Program Başlığı

Genellikle program hakkındaki açıklamaları içeren kısımdır.

Kullanım şekli:

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

2. Tanımlama ve Bildirimler Bölümü

Bu alt bölümde önişlemci komutları, değişken ve yapı tanımlamaları, sabit değer atamaları gibi bildirimler yer alır. Kitaptaki uygulamalar için yeterli olacak bazı bildirimler, burada özetlenecektir.

a. include

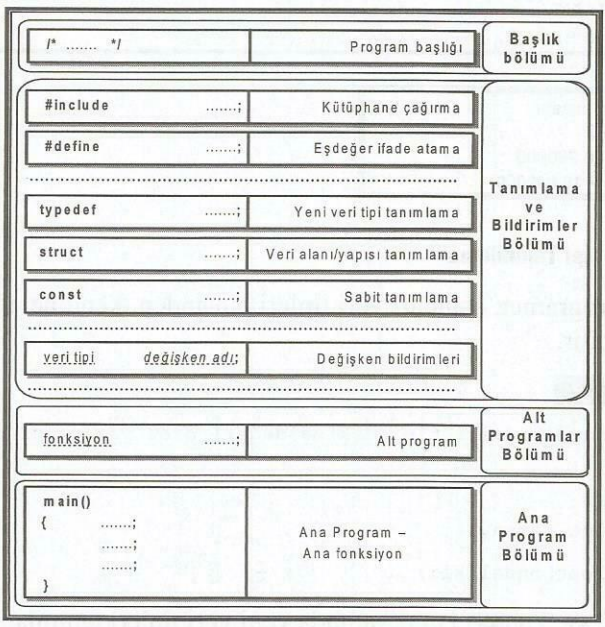
C dilinde komutlar/fonksiyonlar, belirli kütüphanelerde toplanmıştır. Kullanılacak komutlar için "*başlık dosyaları*" olarak adlandırılan ve bunları içeren kütüphaneler (Tablo 4.19) çağırılmalıdır.

Kullanım şekli:

```
#include <  kütüphane adı  >
```

Tablo 4.19: C'deki bazı kütüphaneler ve içerdikleri komutlar/fonksiyonlar

Kütüphane	İçeriği	Kütüphane	İçeriği
stdio.h	Standart giriş/çıkış	**graphics.h**	Grafik ortamı
conio.h	DOS destekli giriş/çıkış	**dos.h**	DOS fonksiyonları
math.h	Matematiksel fonksiyonlar	**ctype.h**	Karakter dönüşüm ve sınıflandırma
stdlib.h	Dönüşüm, sıralama, arama vb.	**string.h**	Alfasayısal ve bazı bellek yönetimi



b. define

Bazı ifadelerin veya sabitlerin, sembolik bir isme aktarılmasını sağlayan önişlemci komutudur.

Kullanım şekli:

```
#define   sembolik_isim   eşdeğer_ifade
```

c. undef

Programda "define" ile önceden tanımlanmış ifade veya sabitlerin iptal edilmesini sağlayan önişlemci komutudur.

Kullanım şekli:

```
#undef   sembolik_isim
```

C

```
/* Önişlemci komutları */
#include <stdio.h>
#include <conio.h>
#define PISAY 3.14
#define dairealan(r) PISAY*r*r
float alan,cevre,yaricap;
main()
{   clrscr();printf("Dairenin alan hesabı\n");
    printf("Yarıçapı giriniz => "); scanf("%f",&yaricap);
    alan=dairealan(yaricap); printf("Dairenin alanı => %f\n",alan);
    printf("Dairenin çevresi => %f\n",2*PISAY*yaricap);
    #undef PISAY
    getch();return 0; }
```

```
Dairenin alan-çevre hesabı
Yarıçapı giriniz => 3
Dairenin alanı => 28.260000
Dairenin çevresi => 18.840000
```

d. Özel Veri Tipi Tanımlama

C'de de programcı; standart veri tipleri cinsinden, kendine ait özel veri tiplerini oluşturabilir.

Kullanım şekli:

```
typedef standart_veri_tipindeki_eşdeğeri özel_veri_tipi_adı;
```

Örneğin;

```
typedef  int tamsayi;
typedef float ondalikli;
```

ile " tamsayi " ve " ondalikli " adında yeni veri tipleri tanımlanmaktadır.

e. Veri Alanı Tanımlama

Birden fazla içeriğe sahip yapıların oluşturulmasında "struct" kullanılır.

Kullanım şekli:

```
struct  veri alanı adı {
            veri tipi 1   veri alt alanı 1;
            ..............................
            veri tipi N   veri alt alanı N;
        } veri alanı değişkenleri ;
```

Örneğin;

```
struct saat_tipi {
        int saat; int dakika; int saniye; int salise; };
```

ile "saat", "dakika", "saniye" ve "salise" alanlarına sahip "saat_tipi" isminde yapı tanımlamasıdır.

f. Değişken Bildirme

C'de tüm değişkenler, isim ve veri tipi olarak bildirilmelidir.

Kullanım şekli:

```
veri tipi    değişken adı;
```

g. Sabit Tanımlama veya Başlangıç Değeri Verme

C programlarında sabit tanımlamak için "const" da kullanılabilmektedir.

Kullanım şekli:

```
const   sabit adı = değeri;
```

Ayrıca değişken bildirimlerinde, değişkenlere doğrudan başlangıç değer ataması yapılabilir.

Kullanım şekli:

```
veri tipi    değişken adı = değeri ;
static  char  karakter dizisi adı[] = değer ;
static  char  karakter dizisi adı[ uzunluk ] = değer ;
```

3. Alt Programlar Bölümü

C programları, "function" olarak adlandırılan birçok alt fonksiyonun/programın birleşimi olup ana fonksiyonun adı "main" dir.

Kullanım şekli

```
Fonksiyonun veri tipi   Fonksiyon adı ( Parametreleri )
Parametre veri tipi bildirimleri ;
{
      Yerel tanımlamalar ve bildirimler;
      ..................................;
      ..................................;  } işlemler
      ..................................;
}
```

C

```
/* Fonksiyon örneği */
#include <stdio.h>
#include <conio.h>
float carp(a,b)
int a,b;
{   float c;
    c=a*b; return c;   }
main()
{   int k,l; float m;
    clrscr();printf("Birinci sayıyı giriniz=> ");scanf("%d",&k);
    printf("İkinci sayıyı giriniz=> ");scanf("%d",&l);
    m=carp(k,l);
    printf("Girilen iki sayının çarpımı =>%f",m);
    getch(); return 0;
}
```

```
Birinci sayıyı giriniz=> 9
İkinci sayıyı giriniz=> 11
Girilen iki sayının çarpımı =>99.000000
```

Yukarıdaki programda "carp" fonksiyonu (alt programı) tanımlanmıştır. Bu fonksiyonun tipi, yani geri göndereceği değer ondalıklı sayı (float) ve parametreleri de tamsayı (int) cinsinden 'a' ve 'b' dir. Yerel değişken olarak da ondalıklı sayı tipinde 'c' bildirilmiştir. Fonksiyon; kedisine gelen iki tamsayıyı çarparak, sonucu ve işlem akışını "return" komutu ile çağırılan yere gönderir.

```
Fonksiyonun veri tipi   Fonksiyon adı (tip ve parametreleri)
{
   Yerel tanımlamalar ve bildirimler;
   ..................................;
   ..................................;   } işlemler
   ..................................;
}
```

şeklinde de fonksiyonlar oluşturulabilir.

C

```
/* Fonksiyon örneği-1 */
#include <stdio.h>
#include <conio.h>
float carp(int a,int b)
{   float c;
    c=a*b; return c;
}
void main()
{   int k,l; float m;
    clrscr();printf("Birinci sayıyı giriniz=> ");scanf("%d",&k);
    printf("İkinci sayıyı giriniz=> ");scanf("%d",&l);
    m=carp(k,l);
    printf("Girilen iki sayının çarpımı =>%f",m); getch();
}
```

Yukarıdaki fonksiyon ile bir önceki fonksiyon, bildirimleri dışında tamamen aynıdır. Programcı, istediği yazım tarzını kullanabilir. Ana fonksiyon olan "`main`"in veri tipi "`void`" olarak bildirilmiştir. "`void`", geriye değer göndermeyen fonksiyon anlamındadır. Birinci programda "`main`" ana fonksiyonunda "`void`" kullanılmadığı için program sonunda "`return 0`" kullanılmıştır.

Not

C programlarında fonksiyonlar, "main" ana fonksiyonundan sonra yazılırsa/yazılacaksa; "main"den önce prototip bildirimi yapılmalıdır.

4. Ana Program Bölümü

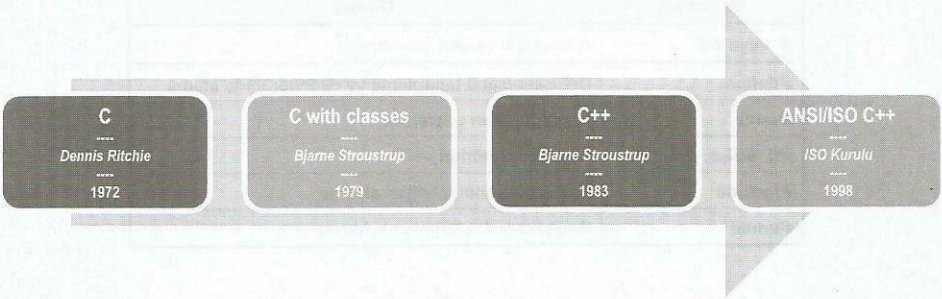
Temel işlem ve kontrollerin bulunduğu ana program kısmı, C'de "`main()`" ana fonksiyonu olup genel kullanım şekli aşağıdaki gibidir.

Kullanım şekli:

```
tip main()
{
        .....
        .....   işlemler
        .....
}
```

B. C++ Dilinin Yapısı

1979 yılında Bell Laboratuvarı araştırmacılarından, Aarhus Üniversitesi matematik alanında yüksek lisans (1975) ve Cambridge Üniversitesi bilgisayar bilimleri alanında doktora (1979) derecesine sahip Danimarkalı *Bjarne Stroustrup* (1950-) tarafından geliştirilmeye başlanılan, C programlama dilinin temel özelliklerinin yanında "*nesne yönelimli programlama*" özelliklerini de taşıyan ve C dilini kapsayan genel amaçlı bir programlama dilidir. 1980'lerin başında "*C With Classes*" (Sınıflarla C) olarak adlandırılan bu programlama dilinin adı, 1983 yılında "*C++*" olarak değiştirilmiştir (Şekil 4.2).



Şekil 4.2: C++ dilinin tarihsel özet gelişimi

C++ dilinin temel yazım özellikleri

i. Nesne yönelimli ve genel amaçlı bir programlama dilidir.
ii. C programlama dilinin birçok özelliklerini taşır.
iii. Program yazımı belirli kalıpta, bloklar halinde olur.
iv. Bloklar, { } parantezleri ile oluşturulur.
v. Komutlar aynı veya alt alta satırlara yazılabilirler.
vi. Tüm komutlar, noktalı virgül (;) ile biter. Yalnız blok başlatan ifadelerden sonra noktalı virgül kullanılmaz.
vii. Programda kullanılan tüm değişkenler ve veri tipleri bildirilir.
viii. Programdaki komutların bulunduğu kütüphaneler aktifleştirilir/çağırılır.

Not

Her C programı, C++'ta çalışabilir.

1. Program Başlığı

Program hakkındaki açıklamaları ya da ismini içeren ifade veya ifadelerdir.

Kullanım şekilleri:

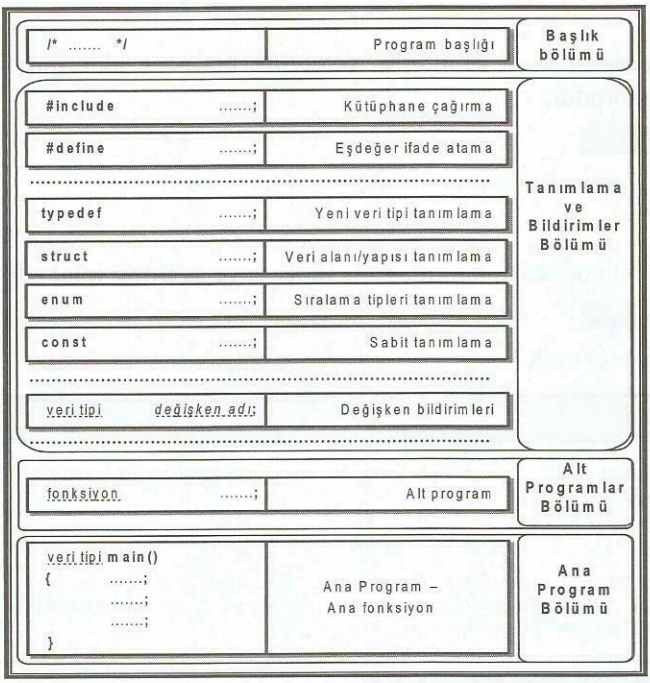
```
/*  açıklamalar veya program başlığı  */
//  açıklamalar veya program başlığı
```

2. Tanımlama ve Bildirimler Bölümü

C++ programlarının bu bölümünde, C programlarında olduğu gibi önişlemci komutları, değişken ve veri yapıları tanımlamaları, sabit değer atamaları gibi bildirimler yapılır. Burada yine C'de anlatılan ve daha sonraki bölümlerde yeterli olacak olan "`include`", "`define`" ve "`undef`" önişlemci komutları (Tablo 4.20); yeni veri tipleri tanımlama ("`typedef`"), yapı oluşturma ("`struct`") ve değişken/sabit bildirimleri özetlenecektir.

Tablo 4.20: C/C++'daki bazı önişlemci komutları

Önişlemci	Görevi
# veya ##	Alfasayısal verileri yönetmek
#define	Değişken/sabit tanımlama ve eşdeğer ifade atama
#error	Hata mesajları gösterme
#if, #else, #endif ...	Koşul operatörleri
#include	Başka dosyanın içeriğini ekleme
#undef	Değişken/sabit tanımını kaldırma



a. include

C programlarında olduğu gibi C++ programlarında da ilgili kütüphane (başlık/kitaplık) dosyalarının bildirilmesi gerekmektedir. C kütüphane dosyalarına ilave olarak bazı C++ kütüphane dosyaları Tablo 4.21'de verilmektedir. Örneğin C'deki "`math`" kütüphanesi C++'ta "`cmath`"; "`string`" kütüphanesi "`cstring`"; "`stdlib`" kütüphanesi "`cstdlib`" gibi çok küçük değişikliklerle veya doğrudan da kullanılabilmektedir.

Kullanım şekli:

```
#include < kütüphane adı >
```

Tablo 4.21: C++'daki bazı kütüphaneler ve içerdikleri komutlar/fonksiyonlar

Kütüphane	İçeriği	Kütüphane	İçeriği
iostream	Temel işlemler ve ekran komutları	**new**	Dinamik bellek kullanımı ve işlemleri
windows	Windows uygulamaları ve makroları	**complex**	Karmaşık sayı işlemleri
vectors	Dizi işlemleriyle ilgili fonksiyonlar	**algorithms**	Özellikle eleman dizileri fonksiyonları

Piyasada birçok C++ yazılım geliştirme editörleri bulunmakta ve bunlar, dilin değişik zamanlarındaki standartlarını kullandıklarından yazımlarda küçük değişiklikler oluşabilmektedir. Örneğin kullanılan editöre göre "*iostream*" kütüphanesini çağırmak aşağıdaki üç şekilden biriyle olabilmektedir:

```
➢ #include <iostream.h>
➢ #include <iostream>
➢ #include <iostream>
  using namespace std;
```

b. define

Bazı ifadelerin veya sabitlerin, sembolik bir isme aktarılmasını sağlayan önişlemci komutudur.

Kullanım şekli:

```
#define   sembolik isim   eşdeğer ifade
```

c. undef

"`define`" ile önceden tanımlanmış ifade veya sabitleri iptal eder.

Kullanım şekli:

```
#undef   sembolik isim
```

C++

```
// Önişlemci komutları
#include <iostream>
 using namespace std;
#define mutlak(a) ((a<0) ? -a : a)
float x;
int main()
{    system("cls"); cout << "Bir sayi giriniz: "; cin >> x;
     cout << "Girilen sayinin mutlak degeri: " << mutlak(x) << endl;
     system("pause"); return 0;
}
```

```
Bir sayi giriniz: -2009
Girilen sayinin mutlak degeri: 2009
```

d. Özel Veri Tipi Tanımlama

Temel veri tipleri cinsinden özel veri tipleri "`typedef`" ile tanımlanır.

Kullanım şekli:

```
typedef temel veri tipindeki eşdeğeri özel veri tipi adı;
```

e. Veri Alanı Tanımlama

C+ programlarında yapı oluşturmak için yine C'deki "`struct`" kullanılır.

Kullanım şekli:

```
struct  veri alanı adı {
          veri tipi 1   veri alt alanı 1;
          ..............................
          veri tipi N   veri alt alanı N;
          } veri alanı değişkenleri ;
```

f. Sıralama Tipleri Tanımlama

C++ programlarında, C programlarında olduğu gibi belli aralıktaki veya belirli değerleri alabilen "*sıralama tipleri*" tanımlanabilir.

Kullanım şekli:

```
enum  sıralama tipi adı { değerler } değişkenler ;
```

Örneğin;

```
enum gun{pazartesi,sali,carsamba,persembe,cuma,cumartesi,pazar} bugun;
```

g. Değişken Bildirme

C++ programlarındaki tüm değişkenler de kullanılacakları yere göre genel veya yerel olarak bildirilmelidirler.

Kullanım şekli:

```
veri tipi    değişken adı;
```

h. Sabit Tanımlama veya Başlangıç Değeri Verme

C programlarında olduğu gibi C++ programlarında da sabit tanımlamak için "`const`" kullanılmaktadır. Ayrıca C'de olduğu gibi değişken bildirimlerinde, değişkenlere doğrudan başlangıç değer ataması yapılabilir.

Kullanım şekli:

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

3. Alt Programlar Bölümü

C programları gibi C++ programları da birçok fonksiyonun birleşiminden oluşur. Ana fonksiyonun adı "`main`" dir.

Kullanım şekli:

```
Fonksiyonun veri tipi   Fonksiyon adı ( Parametreleri )
{
      Yerel tanımlamalar ve bildirimler;
      ..................................;
      ..................................;   } işlemler
      ..................................;
}
```

C++

```
// Fonksiyon örneği
#include <iostream>
using namespace std;
long faktoriyel (long x)
{    if (x>1) return (x * faktoriyel (x-1)); else return (1); }
int main ()
{ long sayi; system("cls"); cout << "Tamsayi giriniz: "; cin >> sayi;
  cout << sayi << "!=" << faktoriyel(sayi) <<endl;
  system("pause");    return 0; }
```

```
Tamsayi giriniz: 5
5!=120
```

Yukarıdaki programda " `faktoriyel` " "*yinelemeli (rekürsif) fonksiyon*"u tanımlanmıştır. Fonksiyonun tipi, yani çalışması sonucu oluşan verinin tipi uzun tamsayı (`long`) ve giriş parametresi de uzun tamsayı (`long`) cinsinden '`x`' dir. Fonksiyon; kendisine gelen uzun tamsayıyı yinelemeli olarak çarparak, bulduğu sonucu ve işlem akışını "`return`" komutu ile çağırılan yere gönderir.

4. Ana Program Bölümü

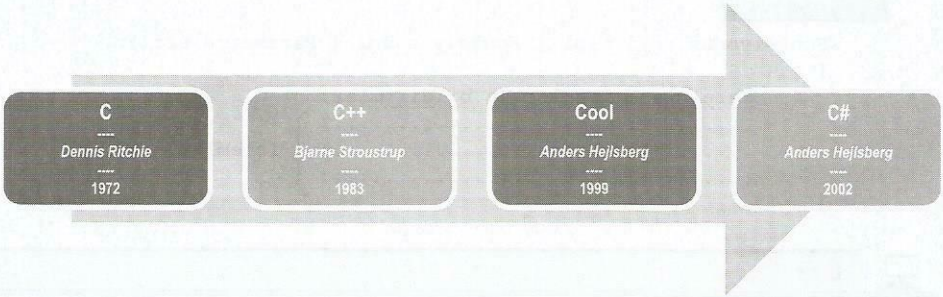
Ana program kısmı, C++'da da C'deki gibi "`main()`" ana fonksiyonudur.

Kullanım şekli:

```
veri tipi  main()
{
        ..................................
        .................................. } işlemler
        ..................................
}
```

C. C# Dilinin Yapısı

C# Microsoft tarafından .NET Framework (platform, teknoloji) ve ortak dil altyapısı (Common Language Infrastructure) için geliştirilmiş olan basit ama güçlü, fonksiyonel, modern, genel amaçlı, nesne yönelimli ve bileşen tabanlı yeni nesil programlama dillerindendir. C, C++ ve Java ile benzer kod yapısına sahip; ECMA ve ISO tarafından da standartlaştırılan C# dilinin tasarımına Danimarkalı yazılım mühendisi *Anders Hejlsberg* (1960-) öncülük etmiştir. Kendisi aynı zamanda Turbo Pascal'ın yazarı ve Delphi'nin baş mimarıdır. Ocak 1999 yılında kurduğu grup ile Cool (C-like Object Oriented Language) dilini geliştirmeye başladılar. Ancak Temmuz 2000 yılında marka nedenlerinden dolayı Microsoft bu dilin ismini C# olarak değiştirdi (Şekil 4.3).



Şekil 4.3: C# dilinin tarihsel özet gelişimi

C# dilinin temel yazım özellikleri

i. Nesne yönelimli, bileşen tabanlı ve genel amaçlı bir programlama dilidir.
ii. Program yazımı belirli kalıpta, bloklar halinde olur.
iii. Bloklar, { } parantezleri ile oluşturulur.
iv. Komutlar aynı veya alt alta satırlara yazılabilirler.
v. Tüm komutlar, noktalı virgül (;) ile biter. Yalnız blok başlatan ifadelerden sonra noktalı virgül kullanılmaz.
vi. Büyük-küçük harf duyarlılığı vardır.
vii. Programda kullanılan tüm değişkenler ve veri tipleri bildirilir.
viii. Sınıflar, yapılar vb. için ilgili ad alanlarının bildirilmesi gerekmektedir.

```
// .....                                   // program başlığı veya açıklamalar
using ..... ;                              // namespace(ad alanı) dâhil etme
namespace .....                            // namespace hazırlama
{
        class .....
        {       sınıf oluşturma       }
        struct .....
        {       yapı oluşturma        }
        interface .....
        {       arabirim tanımlama    }
        ..... delegate .....              // yöntem bildirimi
        enum .....
        {       numaralandırma/sıralama tipleri tanımlama      }
        namespace .....
        {
                struct .....
                        {       yapı oluşturma        }
        }
        class .....                       // ana sınıf
        {
                ..... Main( ..... )
                        {       // Ana program         }
        }
}
```

1. Program Başlığı veya Açıklamalar

Program hakkında açıklayıcı bilgilerin yer aldığı ve kullanımı keyfi olan satırlardır.

Kullanım şekli:

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

2. using

Programa kapsama alanı ismini dahil eder. .NET Framework'teki "*class*" (sınıf)lar, "*struct*" (yapı)lar, "*interface*" (arabirim)ler vb. "*namespace*"(ad alanı) olarak isimlendirilen yapılarda saklanırlar. Dolayısıyla programda bunları kullanabilmek için bulundukları ad alanlarının bildirilmesi gerekmektedir.

Kullanım şekli:

```
using  alan adı  ;
```

3. namespace

.NET Framework'te çok sayıda sınıf, yapı, arabirim vb. bir araya getirilerek ad alanlarında toplanmakta ve programlarda referans verilerek kullanılmaktadırlar. C# programları, ad alanları kullanılarak düzenlenmekte ve bunlar programın hem iç hem de dış organizasyonunda yer almaktadırlar.

Kullanım şekli:

```
namespace  ad alanı ismi
{
    içeriği
}
```

Aşağıdaki programda "using" ve "namespace" kullanımı verilmektedir. Eğer "using System;" kullanılmamış olsaydı "*The name 'Console' does not exist in the current context*" hata mesajı oluşacaktı. Çünkü "Console" sınıfı, "System" ad alanında yer almaktadır. Bu hatayı düzeltmek için "using System;" ad alanı dahil edilir veya "Console" sınıfı kullanılırken "System.Console" şeklinde yazmak gerekir. Dolayısıyla ad alanları; büyük proje kodlarının düzenli olmasını sağlarlar, her sınıf için isim belirtme ve "." operatörü kullanma gereksinimi kaldırarak kodları daha da kısaltırlar.

C#

```
// Ad alanı kullanımı
using System;
namespace ConsoleApplication1
{ class Program
  { static void Main(string[] args)
    { Console.WriteLine("Ad alanı..."); Console.ReadLine(); } } }
```

```
Ad alanı...
```

4. class

Sınıflar; veri üyeleri (sabitler ve alanlar), fonksiyon üyeleri (özellikleri, olayları, yöntemleri vb.) ve içindeki türleri barındıran veri yapılarıdır.

Kullanım şekli:

```
..... class sınıf ismi
{
    İçeriği (alanlar, özellikler, yöntemler, olaylar...)
}
```

C#

```
// Sınıflar
using System;
namespace ConsoleApplication1
{ class class_1
  { public static void islem()
    { Console.WriteLine("class_1 kullanıldı"); Console.ReadLine(); }
  }
  class class_2
  { static void Main(string[] args)
    { class_1.islem(); } } }
```

```
class_1 kullanıldı
```

Yukarıdaki programda "class_1" ile "class_2" şeklinde iki sınıf tanımlanmakta ve "class_1", "class_2" de kullanılmaktadır.

5. struct

Yapı tanımlamalarının yapıldığı kısımdır. Bunlar da sınıflar gibi veri ve fonksiyon üyelerini içeren veri yapılarıdır.

Kullanım şekli:

```
..... struct yapı ismi
{
    İçeriği (alanlar, özellikler, yöntemler, olaylar...)
}
```

C#

```
// Yapı kullanımı
using System;
namespace ConsoleApplication1
{   public struct kompleks
    {   public float a, b;
        public kompleks (float gercel, float sanal)
        {   a = gercel; b = sanal;   }
    }
    class ana_sinif
    {   static void Main()
        {   kompleks sayi1=new kompleks();
            kompleks sayi2=new kompleks(1,2);
            Console.Write("1. karmaşık sayı: ");
            Console.WriteLine("z={0}+j{1}", sayi1.a, sayi1.b);
            Console.Write("2. karmaşık sayı: ");
            Console.WriteLine("z={0}+j{1}", sayi2.a, sayi2.b);
            Console.ReadLine();          }     }     }
```

```
1. karmaşık sayı: z=0+j0
2. karmaşık sayı: z=1+j2
```

6. enum

Numaralandırma/sıralama tiplerinin (kümelerin) tanımlandığı kısımdır.

Kullanım şekli:

```
enum  küme ismi
{
   elemanları
}
```

Örneğin `enum temel_renkler {kırmızı, yeşil, mavi}` ile bir küme tanımı yapılmaktadır.

7. Değişken ve Sabit Bildirimleri

Değişken ve sabit bildirimleri C/C++'taki gibi yapılabilmektedir.

Kullanım şekli:

```
veri tipi  değişken ismi ;
veri tipi  değişken ismi = başlangıç değeri ;
const veri tipi  isim = değeri ;
```

8. Ana program

C# programlarında ana program (ana sınıf) "`Main`"dir.

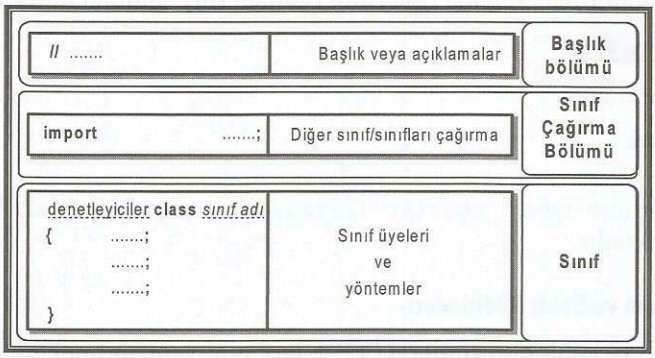
D. Java Dilinin Yapısı

1991 yılında Sun Microsystems mühendislerinden Calgary Üniversitesi Bilgisayar Bilimleri'nden lisans ve Carnegie Mellon Üniversitesi Bilgisayar Bilimleri'nden doktora derecesine sahip Kanadalı *James Gosling* (1955-)'in kurduğu "Yeşil takım" (Green Team) (James Gosling, Mike Sheridan ve Patrick Naughton) tarafından geliştirilmeye başlanmış olan Java'nın bir genel amaçlı, nesne yönelimli ve yüksek seviyeli programlama dili olarak ilk sürümü 1996

yılında piyasaya çıkmıştır. İlk önce "*Greentalk*", sonra "*Oak*" ve nihayetinde "*Java*" olarak adlandırılan platformdan bağımsız, yani kendisi bir sistem platformu olan Java programlarını çalıştırmak için bilgisayarda "*Java sanal makinesi*"(JVM) kurulu olmalıdır. Java sanal makinesi ve geliştirme araçları http://java.sun.com/ adresinden indirilebilir. C++ programlama diliyle yapı-işleyiş-komut benzerlikleri olan Java programları, işletim sisteminden bağımsız olarak JVM'nin kurulabildiği her ortamda çalışabilmektedirler.

Java dilinin temel yazım özellikleri

i. Nesne yönelimli ve genel amaçlı bir programlama dilidir.
ii. Sistem platformudur. Belirli bir bilgisayar veya işletim sistemi mimarisinden bağımsızdır.
iii. Çoklu kullanım (multithreaded) ve dağıtık (distributed) programlama/çalışma desteği olup JVM'nin kurulabildiği her ortamda çalışırlar.
iv. Program yazımı belirli kalıpta, bloklar halinde olur.
v. Bloklar, { } parantezleri ile oluşturulur.
vi. Komutlar aynı veya alt alta satırlara yazılabilirler.
vii. Tüm komutlar, noktalı virgül (;) ile biter. Yalnız blok başlatan ifadelerden sonra noktalı virgül kullanılmaz.
viii. Programda kullanılan tüm değişkenler ve veri tipleri bildirilir.
ix. Komutlar için ilgili kütüphaneler, modüller, sınıflar aktifleştirilir/çağırılır.



Java programları değişik şekillerde çalıştırılabilirler:

i. Metin *editörü ve MS-DOS komut istemi kullanarak*: Metin editöründe (örneğin Not Defteri) program yazılıp "*javac*" derleyicisinin bulunduğu dizine "`.java`" uzantılı olarak kaydedilir. Daha sonra MS-DOS komut isteminden "*javac*" derleyicisinin olduğu dizine gidilerek "`javac program adı.java`" komut satırı ile derlenir. Herhangi bir hata yoksa program derlenerek "`program adı.class`" oluşur ve komut satırından "`java program adı`" ile çalıştırılır.

ii. *Java editörü programı kullanarak*: Herhangi bir Java yazılım editörüyle programlar doğrudan çalıştırılabilirler.

iii. *Appletviewer veya html programları kullanarak*: Java applet program kodu hazırlanarak html olarak kaydedilir, komut satırından "`appletviewer program adı.html`" komutu ile veya doğrudan web tarayıcıda çalıştırılabilir.

Java dilindeki program yazımı, büyük oranda C++ dilinden esinlenilerek türetilmiştir. Bu nedenle Java programlarıyla C/C++ programları yazım olarak benzerlikler gösterirler.

1. Program Başlığı

Program hakkındaki açıklamaları ya da ismini içeren ifade veya ifadelerdir.

Kullanım şekli:

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

2. Sınıf Çağırma Bölümü

Java dilinde "*sınıf*"lar (class), "*paket*" (package) olarak adlandırılan dosyalarda (Tablo 4.22) toplanmışlardır. Diğer sınıfların, ilgili sınıfta (programda) kullanılabilmesi için önceden çağırılması gerekir. Bu olay, C ve C++'taki kütüphane/başlık dosyalarını çağırma/aktifleştirme durumuna benzetilebilir. Herhangi bir Java programı yazıldığında, Java standart kütüphanesi (*java.lang* paketi) otomatik olarak yüklenir/çağırılır. İlgili programda diğer paketlere ait sınıflar, nesneler, fonksiyonlar, komutlar kullanılacaksa, "`import`" ile çağırılması gerekir.

Kullanım şekli:

```
import paket . sınıf ;
```

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

Tablo 4.22: Java'daki bazı paketler

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

3. Sınıflar

a. Sınıf Tanımlama

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

Kullanım şekli:

```
denetleyiciler  class  sınıf adı {
                    üyeler veya yöntemler
}
```

b. Değişken Bildirme

Java programlarında da kullanılacak değişkenler bildirilmelidir.

Kullanım şekli:

```
veri tipi    değişken adı;
```

c. Sabit Tanımlama veya Başlangıç Değeri Verme

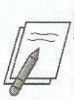
Java programlarında sabit tanımlamak için "`final`" kullanılmaktadır. Ayrıca C/C++'da olduğu gibi değişken bildirimlerinde, değişkenlere doğrudan başlangıç değer ataması yapılabilir.

Kullanım şekli:

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

E. MATLAB Dilinin Yapısı

"*MATrix LABoratory*" (matris laboratuvarı) ifadesinden oluşturulan MATLAB kısaltması, New Mexico Universitesi'nde Bilgisayar Bilimleri bölüm başkanı olan ABD'li *Cleve Moler* (1939-) tarafından 1970'li yılların sonunda geliştirilmeye başlanmıştır. Başlangıçta öğrencilere FORTRAN bilmeden LINPACK (doğrusal denklem sistemlerinin çözümü) ve EISPACK (özdeğer-özvektör çözümleri)'ı kullanmayı/erişmeyi sağlamak için tasarlanan MATLAB, 1980'li yıllarda C ile yeniden yazılmıştır. MATLAB'ın yeni sürümleri Cleve Moler, Jack Little ve Steve Bangert tarafından 1984'te kurulan MathWorks Inc. tarafından geliştirilmektedir. İşlemleri temelde matris tabanlı olarak hızlı ve kolay bir şekilde gerçekleştiren bu yazılım; aynı zamanda sunduğu birçok zengin programlama fonksiyonları içeriği, nesne tabanlı programlama imkânları, grafiksel çözümler, arayüz tasarımları (Graphical User Interface – GUI), web servisleri, harici arabirim/arayüz ve donanım destekleri, birçok uygulama alanına ait SIMULINK araçları gibi çok sayıda bileşenleriyle popülaritesi gün geçtikçe artmaktadır. Yüksek performanslı sayısal hesaplamalar ve grafiksel çizimlerin gerçekleştirilebildiği MATLAB'ın özel uygulama alanlarına ait çözümleri de bulunmaktadır. İçinde barındırdığı çok sayıda araç kutusu (toolbox) ile özellikle mühendislik problemlerinin çözümünde çok popüler durumdadır.



MATLAB dilinin temel yazım özellikleri

i. Nesne yönelimli ve dinamik tipli bir programlama dilidir.
ii. Esnek bir program yazım şekline sahiptir.
iii. Komutlar aynı veya alt alta satırlara yazılabilirler.
iv. Tüm komutlar/komut satırları, noktalı virgül (;) ile biter. Noktalı virgül kullanılmaması durumunda komut sonucu görülür.
v. Kullanılan değişkenlerin ve tiplerinin önceden bildirilme zorunluluğu yoktur.

1. Program Başlığı

Program hakkındaki açıklamaları ya da ismini içeren ve kullanımı zorunlu olmayan ifade veya ifadelerdir.

Kullanım şekli:

```
% açıklamalar veya program başlığı
```

2. Bildirimler/Tanımlamalar

a. Değişken Bildirme

"İsimlendirme kurallarına" uygun olarak oluşturulan değişken isimlerinin bildirilmesine gerek yoktur. Oluşturulan değişken isimlerinin ilk 31 karakteri anlamlıdır. Eğer işlem sonucu herhangi bir değişkene aktarılmazsa, otomatik olarak "ans" değişkenine atanır. Ancak değişken değerleri başka programlarda kullanılacaksa genel bildirim/tanımlama yapılmalıdır. Genel (global) bildirim/tanımlama "global" ile yapılmaktadır. Benzer şekilde değişkenlerin sembolik olarak kullanılması durumunda "syms" ile bildirilmesi gerekmektedir.

Kullanım şekli:

```
global  değişken        →   Değişkenin global olarak bildirilmesi/tanımlanması
syms    değişken        →   Değişkenin sembolik olarak bildirilmesi/tanımlanması
```

b. Sabit Tanımlama veya Başlangıç Değeri Verme

MATLAB programlarında C/C++'da olduğu gibi tanımlayıcılara doğrudan sabit veya başlangıç değer ataması yapılır.

c. Yapı Tanımlama

MATLAB programlarında diğer dillerdeki gibi yapılar oluşturulabilir.

Kullanım şekli:

```
yapı_adı = struct( alan , değer );
```

MATLAB

```
% Komut satırında
>> alan='gerilim';
>> deger={'genlik','frekans','faz'};
>> y=struct(alan,deger);
```

```
>> y

y = 1x3 struct array with fields:
    gerilim
```

```
>> y.gerilim

ans = genlik
ans = frekans
ans = faz
```

d. Sınıf Tanımlama

MATLAB, nesne yönelimli programlama dili olduğunda sınıfları da destelemektedir. Sınıf oluşturma genel olarak aşağıdaki şekilde yapılmaktadır.

Kullanım şekli:

```
classdef sınıf adı
   properties
      özellikler
   end
   methods
      yöntemler
   end
   events
      olaylar
   end
   enumeration
      listeler
   end
end
```

MATLAB

```
classdef selam
    methods
        function yaz(this)
            disp('Merhaba !')
        end
    end
end
% selam.m olarak kaydedilir
```

MATLAB

```
% Komut satırında
>> x=selam;
>> x.yaz
```

```
Merhaba !
```

3. Fonksiyonlar

Alt program veya fonksiyonların oluşturulması MATLAB'ta "`function`" ile yapılmaktadır. İsimleriyle kaydedilen fonksiyonlar, istenilen programlardan çağırılabilirler.

Kullanım şekli:

```
function  sonuc = fonksiyon adı ( parametreleri )
    işlemler
end
```

MATLAB

```
function y=mutlak(x)
    if (x<0)
        y=-1*x;
    else
        y=x;
    end
end
% mutlak.m olarak kaydedilir
```

MATLAB

```
% Komut satırında veya başka bir programda
>> mutlak(-5)
```

```
ans =
   5
```

F. Python Dilinin Yapısı

Yüksek seviyeli, genel amaçlı, nesne tabanlı, modüler, etkileşimli, yorumlayıcı ve otomatik bellek yönetimine sahip bir dinamik programlama dili olan Python, Amsterdam Üniversitesi Matematik ve Bilgisayar Bilimleri alanında yüksek lisans derecesine sahip Hollandalı programcı *Guido van Rossum* (1956-) tarafından geliştirilmiş ve ilk versiyonu 1991'de piyasaya sürülmüştür (Şekil 4.4). C, Java ve Perl ile birçok benzerlikleri olan Python dili, hemen hemen bütün platformlarda (Windows, Linux, Unix, Mac, .NET vb.) çalışmaktadır. Ayrıca sahip olduğu çok sayıdaki modüllerle, birçok alanda (bilimsel hesaplamalar, grafiksel kullanıcı arayüz tasarımları, web ve mühendislik uygulamaları vb.) yaygın olarak kullanılmaktadır.

Python dilinin temel yazım özellikleri

i. Nesne yönelimli, modüler, etkileşimli, yorumlanabilen, dinamik, ücretsiz ve açık kaynak kodlu bir programlama dilidir.

ii. Program yazımı, temelde girintilerle ve hizalamalarla oluşturulan basit bir yapıda yapılır.

iii. Komutlar alt alta satırlara yazılırlar. Aynı satıra yazılması durumunda noktalı virgül (;) ile ayrılırlar.

iv. Bloklar, hizalamalarla belirlenirler.

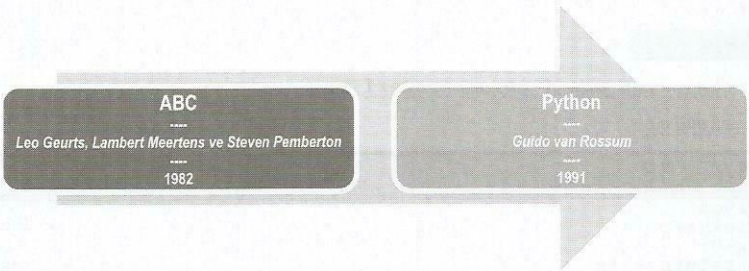
v. Büyük-küçük harf duyarlılığı vardır.

1. Program Başlığı

Program hakkındaki açıklamaları ya da ismini içeren ve kullanımı zorunlu olmayan ifade veya ifadelerdir.

Kullanım şekli:

```
# açıklamalar veya program başlığı
```



Şekil 4.4: Python dilinin tarihsel özet gelişimi

2. Bildirimler/Tanımlamalar

a. Değişken/Sabit Tanımlama veya Başlangıç Değeri Verme

"İsimlendirme kurallarına" uygun olarak oluşturulan değişken isimlerinin bildirilmesine gerek yoktur. Tekli veya çoklu şekilde değişkenlere, başlangıç değer ataması yapılabilmektedir. Değişkeni (referansı) silmek için `"del"` kullanılmaktadır.

Kullanım şekli:

```
değişken = değer                          → Tekli değer ataması
değişken1 = değişken2 = ... = değer       → Çoklu değer ataması
değişken1 , değişken2 , ... = değer1, değer2 , ...   → Çoklu değer ataması
del değişken                              → Referans silme
```

b. Sınıf Tanımlama

Python programlarında sınıflar (class) oluşturulabilmektedir.

Kullanım şekli:

```
class sınıf_adı :
    ifade-1
    ifade-2
    ...
```

c. Modül Çağırma

Modüller, programlarda "`import`" ile çağırılmaktadırlar (Tablo 4.23).

Kullanım şekli:

```
import modül
import modül as tanımlayıcı
from modül import bileşen
```

Tablo 4.23: Python'daki bazı modüller ve içerikleri

Modül	İçeriği
scipy	Bilimsel hesaplamalar
numpy	Matematiksel/sayısal hesaplamalar
matplotlib	Çizimler

3. Fonksiyonlar

Alt program veya fonksiyonların oluşturulması Python'da "`def`" ile yapılmaktadır.

Kullanım şekli:

```
def fonksiyon_adı ( parametreleri ):
    işlemler
```

MATLAB

```
# Fonksiyon
def mutlak(x):
    if x<0:
        return -1*x
    else:
        return x
a=eval(input("Bir tamsayı giriniz: "))
print("\nMutlak değeri: %d"%mutlak(a))
```

```
Bir tamsayı giriniz: -2017

Mutlak değeri: 2017
```

Kitaptaki C programlarının kodlanmasında/yazımında Borland/Turbo C (www.borland.com) (Turbo C++'ın en az 3.0 sürümü) veya başka yazılım geliştirme editörü kullanılabilir. C++ programlarının yazımında Bloodshed Software tarafından geliştirilen ve GPL lisansı (GNU General Public License) ile dağıtı-

lan açık kaynak kodlu Dev-C++ editöründen (en az 4.9.9.2 sürümü) faydalanılabilir. Bloodshed Dev-C++ yazılım geliştirme editörü http://www.bloodshed.net/ sitesinden indirilebilir. Aynı şekilde C# için de Visual Studio (http://msdn.microsoft.com/vstudio) kullanılabilir. Kitapta, C# konsol uygulamaları/kodlamaları yapılacaktır. Java (Java'nın en az 1.6 sürümü (Java 6.0)) programlarının yazımında, Xinox Software tarafından geliştirilen JCreator LE yazılım geliştirme editörü (en az 4.50.010 sürümü) kullanılabilir. JCreator'un ücretsiz olan öğrenci sürümü JCreator LE http://www.jcreator.com/ sitesinden indirilebilir. MATLAB kodlamaları için de MATLAB (http://www.mathworks.com/) programını edinmelisiniz. Kitaptaki MATLAB programları "m-file" olarak düzenlenecektir. Python kodlamaları için de gerekli yazılımı, https://www.python.org/ adresinden (en az Python 3.6.1) indirebilir veya internette mevcut olan diğer editörleri kullanabilirsiniz.

Örnek-4.1: Ekrana "Merhaba" yazan programı C, C++ , C#, Java, MATLAB ve Python dillerinde kodlayınız.

Bu bölümde son olarak karşılaştırma amacıyla, ekrana "Merhaba" yazdıran programların C, C++ , C#, Java, MATLAB ve Python dillerindeki kodlarını yazıp yapılarını özetleyelim.

C

```
/* Merhaba */
#include <stdio.h>
#include <conio.h>
void main()
{   clrscr(); printf("Merhaba"); getch();  }
```

C programlarında da kullanımı serbest olan açıklama veya program başlıkları "`/*  */`" arasında verilir. Programda kullanılacak olan komutların tanımlı olduğu kütüphaneler/başlık dosyaları "`include`" önişlemci komutuyla çağırılır. Ana program/fonksiyon "`main()`" olup "`clrscr()`" komutuyla ekran temizlenir, "`printf`" komutu parantez içinde çift tırnak arasında verilen alfasayısal veriyi yazdırır. Sonuç ekranını bekletmek için ise klavyeden tek karakter girilmesini sağlayan "`getch()`" kullanılabilir.

C++

```
// Merhaba
#include <iostream>
using namespace std;
main()
{     system("cls"); cout << "Merhaba"; system("pause"); return 0;  }
```

C++ programlarındaki açıklama veya program başlıkları "`//`" den sonra veya "`/*  */`" arasında verilir. "`include`" önişlemci komutuyla kütüphane/başlık dosyaları çağırılır. C'deki gibi ana program/fonksiyon "`main()`" olup "`system`" komutuyla sistem komutları çalıştırılabilir. "`system("cls")`" ekranı temizler, "`cout`" ile çift tırnak içindeki alfasayısal veri, ekrana gönderilir (yazdırılır). Sonuç ekranını bekletmek için ise yine sistem komutlarından "`system("pause")`" kullanılabilir.

C#

```
// Merhaba
using System;
namespace merhaba
{   class Program
    {   static void Main(string[] args)
        {   Console.WriteLine("Merhaba"); Console.ReadLine(); } } }
```

C# programlarındaki açıklama veya program başlıkları "//" den sonra verilir. "using" ile ad alanı bildirimi yapılır. "Main()"de "Console" sınıfının "WriteLine" yöntemiyle ekrana yazı yazdırılır ve sonuç ekranı "ReadLine" ile bekletilebilir.

JAVA

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

Java programlarında da C++/C# programlarındaki gibi açıklama veya program başlıkları "//" den sonra verilir. Başka sınıflar kullanılacaksa sınıf tanımından önce "import" ile çağırılır. "class" ile sınıf tanımlanır. Dosya adı ile sınıf adının aynı olması gerekir. "main"deki "public" deyimi, sınıfın veya yöntemin herkese açık (dışarıdan erişilebilir) olduğunu belirtir. "static" deyimi sınıf tarafından paylaşıldığını, "void" de bir değer geri göndermediğini (dönmediğini) belirtir.

MATLAB

```
% Merhaba
clc
fprintf('Merhaba\n'); % disp('Merhaba')  kullanılabilir
```

MATLAB programlarında herhangi bir zorunlu yapı olmadığından, işlemler/komutlar sırayla yazılır. Kullanımı zorunlu olmayan açıklamalar ve başlık ifadeleri "%" simgesinden sonra yapılır. "clc" komutu ekranı (komut satırını) temizlerken, "fprintf" komutu da; kendisinden sonra gelen tek tırnak içindeki alfasayısal veriyi yazdırır.

PYTHON

```
# Merhaba
print("Merhaba")
```

Python programlarında kullanımı zorunlu olmayan açıklamalar ve başlık ifadeleri "#" simgesinden sonra yapılır. "print" komutu da; kendisinden sonra gelen tırnak içindeki alfasayısal veriyi yazdırır.

Son olarak; C, C++, C#, Java, MATLAB ve Python arasındaki bazı program yazım özellikleri karşılaştırmalı olarak Tablo 4.24'te verilmektedir.

Tablo 4.24: C, C++, C#, Java, MATLAB ve Python arasındaki karşılaştırmalı bazı program yazım özellikleri

Kısım	C	C++	C#	Java	MATLAB	Python
Program başlığı	/* ... */	// ... veya /* ... */	// ...	// ...	%	# veya """ """
Kütüphane / modül çağırma	#include ...	#include ...	using ...	import ...		import ...
Değişken bildirme	*tip değişken adı* ;	*tip değişken adı* ;	*tip değişken adı* ;	*tip değişken adı* ;		
Bloklar	{ ... }	{ ... }	{ ... }	{ ... }	... end	Girintilerle
Komut sonlandırma	;	;	;	;	; veya ENTER (yeni satır)	; veya ENTER (yeni satır)
Açıklamalar	/* ... */	// ... veya /* ... */	// ...	// ...	%	# veya """ """

Fonksiyonlar	*tip ad* (*parametreler*)
	tip parametreler;
	{
	.....
	}
		tip ad (*parametreler*)
		{
		.....
		}
			class *ad*
			{
			.....
			}
				denetleyiciler tip yöntemin adı (*parametreler*)
				{
				.....
				}
					function *sonuç* = *fonksiyon adı* (*parametreleri*)
					işlemler
					end
						def *ad* (*parametreler*):

Ana program	main() { ... }	main() { ... }	main()	main()		
Alt satırlardan devam ettirme					...	\

SORULAR

1. Neden farklı programlama dilleri ortaya çıkmıştır? Tartışınız.
2. Neden programlama dillerinde farklı veri tiplerine ihtiyaç duyulmuştur?
3. Veri tiplerinde "*duyarlılık*" nedir? Açıklayınız.
4. C, C++, C#, Java, MATLAB ve Python dillerini temel özellikleri bakımından kıyaslayınız.
5. Fazla veri tipi, bir programlama dili için avantaj mıdır? Açıklayınız.
6. Değişkenleri önceden bildirmenin avantajları nelerdir?
7. Programlarda neden "alt program" kullanılır?
8. C, C++, C# ve Java'da birbirinin benzeri olan veri tiplerini tespit ediniz.
9. Ekrana isminizi yazdıran programı C, C++, C#, Java, MATLAB ve Python'da kodlayınız.
10. C, C++, C#, Java, MATLAB ve Python dillerini daha ayrıntılı araştırarak sunumlar gerçekleştiriniz.

BÖLÜM 5

AKIŞ DİYAGRAMINDAN KODLAMAYA GEÇİŞ

Bu bölümde

- C, C++, C#, Java, MATLAB ve Python'daki operatörleri,
- Akış diyagramı şekillerine karşılık gelen C, C++, C#, Java, MATLAB ve Python'daki temel ve alternatif komutları/fonksiyonları öğreneceksiniz.

Bölüm sonunda

- Genel olarak akış diyagramını C, C++, C#, Java, MATLAB ve Python'da kodlayabilecek
- C, C++, C#, Java, MATLAB ve Python'da kodlanmış programları analiz edebilecek ve akış diyagramlarını çizebileceksiniz.

I. GİRİŞ

Algoritması hazırlanmış/tasarlanmış (metin olarak yazılmış, sözde kodlarla kodlanmış veya akış diyagramı çizilmiş) bir problemi/işlemi, bilgisayar programlama dillerinden birisiyle kodlamak son derece kolaydır. Kodlanacak programlama dilinin kaynaklarından (kitaplar, yazım editöründeki yardım menüleri, web vb.) faydalanılarak kod yazımı yapılır. Genellikle, temel programlama dillerinin program yapısında ve komut yazımında fazla farklılıklar oluşmamaktadır. Oluşan farklılıkları kavramak ve gidermek de oldukça basittir. Bu nedenle; programlama dilinin yapısını ve komutlarını ezberlemek yerine, "*programlama mantığını*" kavramak daha önemlidir. Programcılığın özü; "*programlama mantığı*"dır, "*algoritma geliştirme*"dir. Yazılımcı programın algoritması geliştirilebiliyorsa; o programı, tüm programlama dillerinde kodlamış sayılabilir. Algoritma geliştirme yeteneği de ancak ve ancak birçok değişik problemler çözerek arttırılabilir. Yazılımcı; çözdüğü her problemde, karşılaştığı her sorunda farkında olarak veya olmayarak bir şeyler kazanacaktır. Dolayısıyla birçok problemler ortaya koymak, bunları sorgulamak, parametrelerini değiştirmek, onları sadece bir yoldan değil de değişik yollardan çözmek gibi adımlar algoritma geliştirme mantığını arttıracaktır.

Not

Programlama dilleri arasında fark olsa da, problemin çözümünde izlenen algoritma ve algoritmaya göre çizilen akış diyagramı hep aynı kalır.

Bölüm 6'dan itibaren çözülecek olan problemlerin çözüm yolları anlatılarak akış diyagramları çizilecek ve altı ayrı programlama dilinde kodlanacaktır: C, C++, C# (konsol), Java, MATLAB ve Python. Kodlar yazılırken, farklı kullanım şekilleri ve farklı programlama teknikleri sunulmaya çalışılacaktır. Bir problemin birden fazla çözüm yolu olduğu gibi bir programın da çok farklı kodlama şekilleri olabilmektedir.

Algoritma/akış diyagramlarında kullanılan operatörlerin C, C++, C#, Java, MATLAB ve Python'daki karşılıkları Tablo 5.1'de, akış diyagramı şekillerine karşılık gelen genel komutları ise toplu halde Tablo 5.2'de verilmektedir. Programlama dillerinde, bunlara karşılık kullanılabilecek başka komutlar mevcuttur. Yazılımcının tecrübesi arttıkça; en uygun komutları, en uygun yerlerde en verimli şekilde kullanabilecektir.

Not

Her programlama dillinin yapısında fazladan operatörler bulunabilir.

Not

Her programlama dillinde; benzer veya birbirine yakın görevlerde birçok komut bulunabilir.

Tablo 5.1: Operatörler ve programlama dillerindeki karşılıkları

OPERATÖRLER							
Operatör	**Anlamı**	**C**	**C++**	**C#**	**Java**	**MATLAB**	**Python**
Matematiksel operatörler							
^	*Üs alma*					^	**
*	*Çarpma*	*	*	*	*	*	*
/	*Bölme*	/	/	/	/	/	/
	Sağdan bölme					\	
	Tamsayı bölme						//
+	*Toplama*	+	+	+	+	+	+
-	*Çıkarma*	-	-	-	-	-	-
	Mod	%	%	%	%		%
	Arttırma	++	++	++	++		
	Azaltma	--	--	--	--		
	Topla ve aktar	+=	+=	+=	+=		+=
	Çıkar ve aktar	-=	-=	-=	-=		-=
	Çarp ve aktar	*=	*=	*=	*=		*=
	Böl ve aktar	/=	/=	/=	/=		/=
	Mod ve aktar	%=	%=	%=	%=		%=
	Tam böl ve aktar						//=
	Üs al ve aktar						**=
Karşılaştırma operatörleri							
=	*Eşittir*	==	==	==	==	==	==
<>	*Eşit değildir*	!=	!=	!=	!=	~=	!=
<	*Küçüktür*	<	<	<	<	<	<
>	*Büyüktür*	>	>	>	>	>	>
>=	*Büyük eşittir*	>=	>=	>=	>=	>=	>=
<=	*Küçük eşittir*	<=	<=	<=	<=	<=	<=
	Koşullu anlatım	?:	?:	?:	?:		
Mantıksal operatörler							
`	*DEĞİL*	!	!	!	!	~	~
.	*VE*	&&	&&	&&	&&	&	&
+	*VEYA*	||	||	||	||	|	|
	ÖZEL VEYA						^
Bitsel işlem operatörleri							
	DEĞİL	~	~	~	~		~
	VE	&	&	&	&		&
	VEYA	|	|	|	|		|
	ÖZEL VEYA	^	^	^	^		^
	Sağa kaydır	>>	>>	>>	>>		>>
	Sola kaydır	<<	<<	<<	<<		<<
Alfasayısal operatörler							
+	*Birleştirme*	+	+	+	+	+	+
İşaretçiler							
	Adres	@	&	&	&		
	Adresteki değer	^	*	*	*		
Genel operatörler							
=	*Aktarma/atama*	=	=	=	=	=	=
( )	*Parantez*	( )	( )	( )	( )	( )	( )

Tablo 5.2: Akış diyagramı şekilleri ve programlama dillerindeki temel karşılıkları

Şekil	C	C++	C#	Java	MATLAB	Python
Başla	Açıklamalar Bildirimler	Açıklamalar Bildirimler	Açıklamalar Bildirimler	Açıklamalar Bildirimler	Açıklamalar Bildirimler	Açıklamalar Bildirimler
değişken	**scanf**(*biçim ifadesi* , *&değişken*)	**cin**>>*değişken*	*değişken* =**Console.ReadLine**();	*değişken* =**nextInt**(); *değişken* =**nextLine**(); ...	*değişken* =**input**(*mesaj*);	*değişken* =**input**(*mesaj*);
değişken	**printf**(*biçim ifadesi* , *değişken*)	**cout**<<*değişken*	**Console.Write**(*değişken*)	**System.out.print**(*değişken*)	**fprintf**(*biçim ifadesi* , *değişken*)	**print**(*biçim ifadesi* % *değişken*)
işlem	*işlem*	*işlem*	*işlem*	*işlem*	*işlem*	*işlem*
değişken= başla,dur,adım	**for** (*başla*; *şart*; *adım*)	**for** (*başla*; *şart*; *adım*)	**for** (*başla*; *şart*; *adım*)	**for** (*başla*; *şart*; *adım*)	**for** *değişken* = *başla* : *adım* : *dur*	**for** *değişken* **in** *değerler* :
koşul E *işlem1* H *işlem2*	**if** (*koşul*) **then** *işlem1*; **else** *işlem2*;	**if** (*koşul*) **then** *işlem1*; **else** *işlem2*;	**if** (*koşul*) **then** *işlem1*; **else** *işlem2*;	**if** (*koşul*) **then** *işlem1*; **else** *işlem2*;	**if** (*koşul*) *işlem1*; **else** *işlem2*;	**if** *koşul* : *işlem1* **else:** *işlem2*
Dur						

II. VERİ GİRİŞ KOMUTLARI – "Al/Oku/Gir"

Bilgisayar, işlemleri gerçekleştirebilmek için bilgilere/verilere ihtiyaç duyar. Bilgilerin/verilerin dışarıdan (klavye, port, dosya vb.) alınmasını sağlayan komutlar, 'veri giriş komutları' olarak adlandırılırlar.

A. C'deki Veri Giriş Komutları

1. scanf

C programlama dilindeki temel veri giriş komutudur. Klavyeden; belirtilen değişkene, veri aktarılmasını sağlar.

Kullanım şekli:

```
scanf(" biçim ifadesi ",& değişkenler listesi );
```

$$\%\ \left|*\right|\ \left|n\right|\ \left|\begin{matrix}F\\N\\h\\l\\L\end{matrix}\right|\ \left|\begin{matrix}d & D\\i & I\\u & U\\o & O\\x & X\\e & E\\g & G\\f\\s\\c\\\%\\n\\p\end{matrix}\right|$$

Buradaki *"biçim ifadesi"*, veri girişinin hangi biçimde olacağını; *"değişkenler (adres) listesi"* de verilerin aktarılacağı değişkenleri belirtir. *Biçim ifadesi*'nin genel şekli yandaki gibi olup iki kısımdan oluşur: *Biçim* kısmı ve *tip belirtme* kısmı. Daha kısa olarak biçim ifadesi ⟨% *biçim tip*⟩ şeklinde de gösterilebilir. Görüldüğü gibi biçim ifadesi; çift tırnak içinde, **%** karakteriyle (sembolüyle) başlamaktadır. Biçim ve tip karakterlerinin anlamları ise Tablo 5.3 ile Tablo 5.4'te verilmektedir.

Tablo 5.3: Biçim karakterleri

Karakter		Anlamı	Kullanımı
*	*Gizleme*	Değer okunur, fakat belirtilen değişkene aktarılmaz.	Seçimli
n	*Genişlik*	Girilecek karakterlerin maksimum sayısını belirtir.	Seçimli
F	*Pointer*	*'far'*(uzak) pointer olduğunu belirtir.	Seçimli
N		*'near'*(yakın) pointer olduğunu belirtir.	Seçimli
h	*Tip niteleme*	Tamsayı tipleriyle (*d, i, u, o, x*) birlikte kullanıldığında *"short int"* anlamını verir.	Seçimli
l		Tamsayı tipleriyle (*d, i, u, o, x*) birlikte kullanıldığında *"long int"* anlamını; ondalıklı sayı tipleriyle (*e, f, g*) birlikte kullanıldığında ise *"double"* anlamını verir.	Seçimli
L		Ondalıklı sayı tipleriyle (*e, f, g*) kullanıldığında *"long double"* anlamını verir.	Seçimli

Tablo 5.4: Tip karakterleri

Karakter	Girilmesi gereken değer	Tipi
	Sayılar için	
d	10 tabanında tamsayı	*int*
D	10 tabanında tamsayı	*long*
i	10, 8 veya 16 tabanında tamsayı	*int*
I	10, 8 veya 16 tabanında tamsayı	*long*
u	10 tabanında işaretsiz tamsayı	*unsigned int*
U	10 tabanında işaretsiz tamsayı	*unsigned long*
o	8 tabanında tamsayı	*int*
O	8 tabanında tamsayı	*long*
x	16 tabanında tamsayı	*int*
X	16 tabanında tamsayı	*long*
e	Ondalıklı sayı	*float*
E	Ondalıklı sayı	*float*
g	Ondalıklı sayı	*float*
G	Ondalıklı sayı	*float*
f	Ondalıklı sayı	*float*
	Karakterler için	
s	Alfasayısal veri(karakter dizisi)	*char* cumle*[...]* veya *char* *cumle
c	Karakter	*char*
%	%	-
	Pointer'ler için	
n	-	*int*
p	16 tabanında biçim(YYYY:ZZZZ veya ZZZZ)	Object (*far** veya *near**)

C

```
/* veri girişi 1*/
#include <stdio.h>
#include <conio.h>
int a,b,c,d,e,f;
main()
{  clrscr();printf("Bir pozitif tamsayı giriniz: ");scanf("%u",&a);
   printf("Bir boşluk bırakarak iki tamsayı giriniz: ");
   scanf("%d %d",&b,&c);
   printf("Araya virgül koyarak iki tamsayı giriniz: ");
   scanf("%d,%d",&d,&e);  f=a+b+c+d+e;
   printf("\nGirilen sayıların toplamı %d dir.",f); getch();return 0;}
```

```
Bir pozitif tamsayı giriniz: 5
Bir boşluk bırakarak iki tamsayı giriniz: 4 3
Araya virgül koyarak iki tamsayı giriniz: 2,1

Girilen sayıların toplamı 15 dir.
```

Yukarıdaki programda beş tane tamsayı, değişik biçimlerde girilmekte ve toplanarak ekrana yazdırılmaktadır.

C

```
/* veri girişi 2*/
#include <stdio.h>
#include <conio.h>
float a,b,c;
main()
{  clrscr();printf("Bir ondalıklı sayı giriniz: ");scanf("%f",&a);
   printf("Bir başka ondalıklı sayı giriniz: ");scanf("%3f",&b);
   c=a+b;printf("\nGirilen sayıların toplamı %0.3f dir.",c);
   getch();return 0; }
```

```
Bir ondalıklı sayı giriniz: 100.125
Bir başka ondalıklı sayı giriniz: 123456789.125

Girilen sayıların toplamı 223.125 dir.
```

Yukarıdaki örnekte görüldüğü gibi belirtilen uzunluğu aşan bir giriş yapılırsa; girilen değerin, belirtilen uzunluktan fazla olan kısmı atılır. Ondalıklı sayılarda "." da bir basamak/hane (dijit) olarak kabul edilir. Eğer ikinci ondalıklı sayı, biçimli giriş ile alınmasaydı; sonuç, 123456889.25 şeklinde doğru olarak hesaplanacaktı. Ancak "%3f" biçimiyle girilen 123456789.125 sayısının sadece 123 kısmı (üç haneli/basamaklı kısım) alınabildiğinden sonuç da 100.125+123=223.125 olarak bulunur. Biçim ifadesi, alfasayısal veriler üzerinde de aynı etkiyi yapmaktadır.

2. gets

Klavyeden, alfasayısal veri (karakter, kelime, cümle vb.) girilmesini sağlar.

Kullanım şekli:

```
gets( değişken );
```

```
/* gets ile scanf arasındaki fark */
#include <stdio.h>
#include <conio.h>
char a[255],b[255];
void main()
{ clrscr(); printf("Cümleyi giriniz: ");gets(a);
  printf("Cümleyi tekrar giriniz: ");scanf("%s",&b);
  printf("\ngets ile alınan cümle: %s",a);
  printf("scanf ile alınan cümle: %s",b); getch(); }
```

```
Cümleyi giriniz: 17 Ağustos 1999, Adapazarı Depremi
Cümleyi tekrar giriniz: 17 Ağustos 1999, Adapazarı Depremi

gets ile alınan cümle: 17 Ağustos 1999, Adapazarı Depremi
scanf ile alınan cümle: 17
```

Örnek programdan da görüldüğü gibi "`gets`" komutu, '`Enter`' tuşuna basılıncaya kadar girilen karakter/karakter grubunun tamamını alırken; "`scanf`" komutu, ilk özel karaktere kadar olan kısmı alabilmektedir/okuyabilmektedir.

3. getch

Ekranda görülmeden, klavyeden tek karakterin girilmesini sağlar.

Kullanım şekli:

```
değişken = getch();
```

4. getche

Ekranda görünerek, klavyeden tek karakterin girilmesini sağlar.

Kullanım şekli:

```
değişken = getche();
```

5. getchar

Klavyeden veya dosyadan tek karakterin okunmasını/alınmasını sağlar.

Kullanım şekli:

```
değişken = getchar();
```

6. kbhit

Klavyede, herhangi bir tuşa basılıp basılmadığını verir. Basılması durumunda "`True`" (`1`) üretir (sonucunu verir).

Kullanım şekli:

```
kbhit();
```

B. C++'daki Veri Giriş Komutları

1. cin

C++ programlama dilindeki temel veri giriş komutudur.

Kullanım şekilleri:

```
cin >> değişken ;
cin >> değişken1  >> değişken2  >> değişkenN;
cin >> biçimleyici >> değişken  ;
cin.biçim ayarı ;
```

Bu komutla alfasayısal veriler girilirken, ilk özel karaktere (boşluk) kadar olan kısmı alınır. Araya boşluk koyularak, birden fazla değişkene de tek komutla değer aktarılabilir. "*biçimleyici*", veri girişinin biçimini belirtir ve alabileceği bazı değerler Tablo 5.5'te verilmektedir. Karakter sayısı (genişliği) gibi ayarlamaların yapıldığı "*biçim ayarı*" ise "*biçimleyici*" gibi "`cout`" kısmında daha ayrıntılı olarak anlatılacaktır.

Tablo 5.5: C++'daki biçimleyici değerleri ve anlamları

Biçim ifadesi	Anlamı
boolalpha	Alfasayısal mantıksal değer kullan
noboolalpha	Sayısal mantıksal değer kullan
skipws	Özel karakterleri (boşluk, tab, vb.) atla
noskipws	Özel karakterleri atlama
ws	İlgili konumdaki özel karakterleri iptal et
dec	10 tabanını kullan
hex	16 tabanını kullan
oct	8 tabanını kullan

Not

C++ programlama dilinde, C dilinde anlatılan veri giriş komutları, ilgili kütüphaneler çağırılarak kullanılabilir.

C++

```
// Veri girişinde biçim ifadeleri
#include <iostream>
using namespace std;
bool a=true;
int b,c,d;
char e[20],f[10];
int main ()
{   cout << boolalpha << a << endl; cout << noboolalpha << a << endl;
    cout << "Tamsayi giriniz: "; cin >> hex >> b;
    cout << "10 tabanindaki degeri: " << b << endl << endl;
    cout << "Iki tamsayi ve kelime giriniz: ";cin>>dec>>c>>d>>e;
    cout << "Girdiginiz degerler: " << c <<"\t"<< d <<"\t"<< e <<endl;
    cout << "Bir kelime giriniz: "; cin.width(5); cin >> f;
    cout << "Girdiginiz kelime: " << f << endl;
    system("pause"); return 0; }
```

```
true
1
Tamsayi giriniz: 20
10 tabanindaki degeri: 32

Iki tamsayi ve kelime giriniz: 30 40 Bursa
Girdiginiz degerler: 30  40      Bursa
Bir kelime giriniz: Sakarya
Girdiginiz kelime: Saka
```

2. get

Klavyeden tek karakter okuyarak ilgili değişkene aktarır.

Kullanım şekli:

```
cin.get( değişken );
```

3. getline

Klavyeden belirtilen uzunluğa kadar karakter dizisi okur.

Kullanım şekli:

```
cin.getline( değişken , uzunluk );
getline(cin, değişken );
```

C++

```
// Alfasayısal veri girişleri
#include <iostream>
using namespace std;
char a[50],b[50];
int main ()
{ cout<<"1. cümle: ";cin.getline(a,50);
  cout<<"2. cümle: ";cin.getline(b,5);
  cout<<"\n1. getline ile alınan: "<<a<<endl;
  cout<<"2. getline ile alınan: "<<b<<endl; system("pause");return 0;}
```

```
1. cümle: Uludağ Universitesi
2. cümle: Uludağ Universitesi

1. getline ile alınan: Uludağ Universitesi
2. getline ile alınan: Ulud
```

C. C#'daki Veri Giriş Komutları

1. ReadLine

C# konsol uygulamalarındaki temel veri giriş/okuma komutudur. Standart girişteki tüm satır verisini alfasayısal olarak alır/okur. Bu nedenle sayısal veri girişi yapıldığında - "`ReadLine`" alfasayısal olarak değerlendirdiğinden - sayısala dönüştürülmesi gerekir.

Kullanım şekli:

```
değişken = Console.ReadLine();
```

2. ReadKey

C# konsol uygulamalarında, ekranda görünerek tek karakteri veya fonksiyonu alır/okur. "`ConsoleKeyInfo`" tipinde değer geri döndürmekte olup okunan karakteri elde etmek için "`Key`", "`KeyChar`" gibi özelliklerinden faydalanılır.

Kullanım şekli:

```
değişken = Console.ReadKey();
```

3. Read

Standart girişten bir sonraki karakteri alır/okur.

Kullanım şekli:

```
değişken = Console.Read();
```

4. KeyAvailable

Bu konsol özelliği, tuşa basılıp basılamayacağını belirtir.

Kullanım şekli:

```
Console.KeyAvailable;
```

C#

```
// Veri girişleri
using System;
namespace ornek
{ class Program
   { static void Main(string[] args)
      {  String ad;
         ConsoleKeyInfo h;
         Console.Clear(); Console.Write("İsim: ");
         ad=Console.ReadLine();Console.WriteLine("\nMerhaba " + ad);
         do
          {  Console.Write("\nÇıkış için X tuşu... ");
             h=Console.ReadKey();
          } while (h.KeyChar != 'X');  }    }   }
```

```
İsim: Uludağ

Merhaba Uludağ

Çıkış için X tuşu... x
Çıkış için X tuşu... X
```

D. Java'daki Veri Giriş Komutları

Java programlarında klavyeden veri girişi için "`java.util`" paketinde yer alan "`Scanner`" sınıfının yöntemleri ("`Sistem.in`") kullanılır. Bu nedenle programın başında "`import java.util.Scanner`" ile sınıf çağırılır. "`Scanner`" sınıfının bazı yöntemleri Tablo 5.6'da özetlenmektedir.

Tablo 5.6: "Scanner" sınıfının bazı yöntemleri

Yöntem	Görevi
findInLine("*patern*")	Girilen ifadede "patern" ile belirtilen ifadeyi alır
findInLine(".").charAt(0)	Klavyeden girilen ifadenin ilk karakterini alır
next()	Klavyeden ifadeyi ilk özel karakterine (boşluk) kadar alır
nextBoolean()	Klavyeden girilen ifadeyi "boolean" tipinde alır
nextByte()	Klavyeden girilen değeri "byte" veri tipinde alır
nextDouble()	Klavyeden girilen değeri "double" veri tipinde alır
nextFloat()	Klavyeden girilen değeri "float" veri tipinde alır
nextInt()	Klavyeden girilen değeri "int" veri tipinde alır
nextLine()	Klavyeden girilen tüm satırı alır
nextLine ().charAt(0)	Klavyeden girilen ifadenin ilk karakterini alır
nextLong()	Klavyeden girilen değeri "long" veri tipinde alır
nextShort()	Klavyeden girilen değeri "short" veri tipinde alır

JAVA

```
// Veri girişleri
import java.util.Scanner;
public class Verigiris {
public static void main(String[] args)
{ String a,b;
  Scanner klavye=new Scanner(System.in);
  System.out.print("Bir cümle giriniz: ");  a=klavye.nextLine();
  System.out.println("Girdiğiniz cümle: "+a);
  System.out.print("Bir kelime giriniz: ");b=klavye.findInLine("ak");
  System.out.println("Girilen kelimeden: "+b); }}
```

```
Bir cümle giriniz: Java programlama dili
Girdiğiniz cümle: Java programlama dili
Bir kelime giriniz: Sakarya
Girilen kelimeden: ak
```

Yukarıdaki programda "`Scanner klavye=new Scanner(System.in);`" ile "`klavye`" isminde yeni bir nesne oluşturularak daha sonraki adımlarda "`Scanner`" sınıfının yöntemleri kullanılmaktadır.

E. MATLAB'daki Veri Giriş Komutları

1. input

MATLAB uygulamalarında, ekrana belirli mesaj yazdırarak veri girişi/okuma sağlar.

Kullanım şekli:

```
değişken = input(' mesaj ');          → Sayısal veri girişi
değişken = input(' mesaj ','s');      → Alfasayısal veri girişi
```

MATLAB

```
% Veri girişi
clear all; clc; a=input('Bir tamsayı giriniz: ');
b=input('Bir şehir ismi giriniz:','s'); disp(a); disp(b);
```

```
Bir tamsayı giriniz: 99
Bir şehir ismi giriniz: Bursa
99

Bursa
```

2. inputdlg

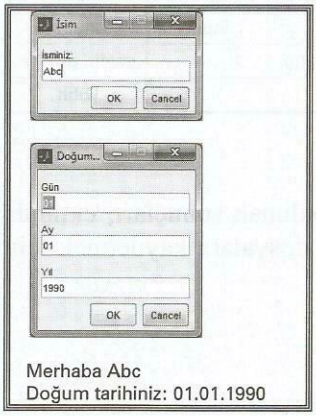
Veri girişi iletişim penceresiyle değer alınmasını sağlar.

Kullanım şekli:

```
değişken = inputdlg('Açıklama', 'Başlık', Satır sayısı, 'Varsa-
yılan değer',);
```

MATLAB

```
clear all; clc
a=inputdlg('İsminiz: ','İsim');
tarih=inputdlg({'Gün','Ay','Yıl'},'Doğum tarihi',1,{'01','01','1990'});
fprintf('Merhaba %s\nDoğum tarihiniz: %s.%s.%s\n',
char(a),char(tarih(1)),char(tarih(2)),char(tarih(3)));
```



F. Python'daki Veri Giriş Komutları

1. input

Python programlarında, ekrana belirli mesaj yazdırarak klavyeden alfasayısal veri girişi/okuma sağlar. Aşağıdaki kullanım şekillerinde çift tırnak yerine tek tırnak da kullanılabilmektedir.

Kullanım şekli:

```
değişken = input(" mesaj ");          → Alfasayısal giriş
değişken = int(input(" mesaj "))      → Tamsayı giriş
değişken = float(input(" mesaj "))    → Ondalıklı sayı giriş
değişken = eval(input(" mesaj "))     → Uygun giriş
```

PYTHON

```
# Veri girişi
import datetime
a=input("İsminiz: "); b=input("Doğum yılınız: ")
t=datetime.datetime.now(); yas=t.year-int(b)
print ("\nMerhaba "+a+"\nYaşınız: "+str(yas))
```

```
İsminiz: Abc
Doğum yılınız: 1999

Merhaba Abc
Yaşınız: 18
```

Python'da "msvcrt"(Microsoft Visual C/C++ Runtime Library) modülü kullanılarak `getch`, `getche`, `kbhit` gibi komutlardan/fonksiyonlardan faydalanılabilir.

Tablo 5.7'de; anlatılan altı programlama dilindeki klavyeden veri giriş komutları, karşılaştırmalı olarak verilmektedir.

Tablo 5.7: C, C++, C#, Java, MATLAB ve Python'daki eşdeğer görevdeki veri giriş komutları

Algoritma	C	C++	C#	Java	MATLAB	Python
Gir (genel)	scanf	cin	ReadLine()	next...()	input	input
Gir *cümle*	gets	getline	ReadLine()	nextLine()	input	input
Gir *harf*	getch, getche	get	ReadKey()	nextLine().charAt(0)		getch, getche
Tuş	kbhit	kbhit	KeyAvailable			kbhit

III. VERİ ÇIKIŞ KOMUTLARI – "Yaz"

Bilgisayarda işlemleri gerçekleştirdikten sonra bulunan sonuçları; ekranda görüntülemek, yazıcıdan çıktı almak veya bellekteki dosyalara kaydetmek için kullanılan komutlardır.

A. C'deki Veri Çıkış Komutları

1. printf

Sonuçları, ekrana yazdırmayı sağlar.

Kullanım şekli:

```
printf(" biçim ifadesi ", değişkenler );
```

Çift tırnak arasında yer alan '*biçim ifadesi*' genel olarak üç kısımdan oluşur: açıklama, biçim ve kontrol/kaçış.

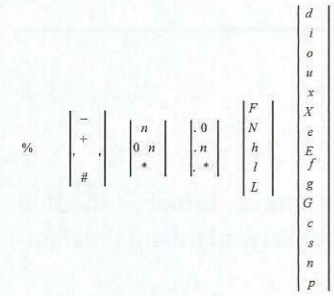
a. ***Açıklama***: Çift tırnaklar arasında verilip ekrana doğrudan yazılır.

C

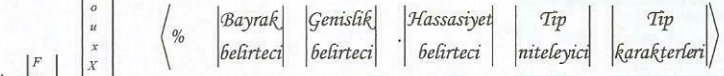
```
/* Yazdırma - 1 */
#include <stdio.h>
#include <conio.h>
void main()
{ printf("Ankara"); getch();  }
```

Ankara

b. ***Biçim***: % sembolüyle başlayan ve çıkış biçiminin belirlendiği kısımdır.



Bu kısım daha kısa olarak;



şeklinde de gösterilebilir. "*Tip karakterleri*" hariç diğerlerinin kullanımı zorunlu değildir. Bu alt kısımlardaki karakterlerin anlamları Tablo 5.8 - 5.12 arasında verilmektedir.

Tablo 5.8: Bayrak belirteci karakterleri

Karakter	Anlamı	
-	Sonucu sola dayalı olarak yazıp sağ tarafı boşluklarla doldurur.	
+	Sonucun önüne, durumuna göre '+' veya '-' işaretini koyar.	
boşluk	Pozitif sonuçlarda '+' işareti yerine boşluk koyar.	
#	*Ardındaki karakter*	*Anlamı*
	c, s, d, i, u	Herhangi bir etkisi yok.
	0	Sıfırdan farklı değerin önüne '0' ekler.
	x, X	Değerin önüne '0x' (veya 0X) ekler.
	e, E, f	Ondalık noktasından sonra hane olmasa bile, ondalık noktası daima olur.
	g, G	'e' ve 'E' deki gibi etkisi vardır ve ondalık noktasından sonraki izleyen sıfırlar atılmaz.

Tablo 5.9: Genişlik belirteci karakterleri

Karakter	Anlamı
n	En az "n" tane karakter yazdırılır. Eğer yazdırılacak değer; "n" karakterden az ise, eksik kısım boşluklarla doldurulur.
0n	En az "n" tane karakter yazdırılır. Eğer yazdırılacak sayısal değer; "n" karakterden az ise, eksik kısım(sayının önü) sıfır(0)larla doldurulur.
*	Genişlik için gerekli değer, değişkenler listesinden okunur.

Tablo 5.10: Hassasiyet belirteci karakterleri

Karakter	Anlamı
yok	Hassasiyet, varsayılan olarak: - *d, i, o, u, x, X* tipleri için =1 - *e, E, f* tipleri için =6 - *g, G* tipleri için tüm anlamlı haneler - *s* tipleri için ilk "NULL" karakterine kadar - *c* tipinde etkisi yok
0	- d, i, o, u, x tipleri için varsayılan hassasiyet değeri - e, E, f tipleri için ondalık noktası yok
n	"n" tane karakter veya "n" hane yazdırılır.
*	Hassasiyet değeri, değişkenler listesinden okunur.

Tablo 5.11: Tip niteleyici karakterleri

Karakter	Nitelediği tip	Anlamı
F	Pointer(p, s ve n)	far pointer
N		near pointer
h	d, i, o, u, x, X	short int
l	d, i, o, u, x, X	long int
	e, E, f, g, G	double
L	e, E, f, g, G	long double

Tablo 5.12: Tip karakterleri

Karakter	Gerekli giriş değeri	Çıkış biçimi
	Sayılar için	
d	Tamsayı	İşaretli 10 tabanında tamsayı
i	Tamsayı	İşaretli 10 tabanında tamsayı
o	Tamsayı	İşaretsiz 8 tabanında tamsayı
u	Tamsayı	İşaretsiz 10 tabanında tamsayı
x	Tamsayı	İşaretsiz 16 tabanında tamsayı (harfler küçük)
X	Tamsayı	İşaretsiz 16 tabanında tamsayı (harfler büyük)
f	Ondalıklı sayı	[-]hhhh.hhhh formunda işaretli değer
e	Ondalıklı sayı	[-]h.hhhh veya e[+/-]hhh biçiminde(üstel formda) işaretli sayı
g	Ondalıklı sayı	Belirtilen değer ve hassasiyete göre *e* veya *f* formunda işaretli sayı
E	Ondalıklı sayı	*e* ile aynı, üs belirteci "E"
G	Ondalıklı sayı	*g* ile aynı, *e* formatı kullanılırsa üs belirteci "E"
	Karakterler için	
c	Karakter	Tek karakter
s	Karakter dizisi	Karakter dizisi
%		%
	İşaretçiler (pointer) için	
n	Tamsayı işaretçi	Çok uzaktaki karakterlerin sayısı
p	İşaretçi	XXXX:YYYY veya YYYY(sadece offset)

c. ***Kontrol/kaçış***: "\" işaretiyle başlayan bu karakterlerin anlamları Tablo 5.13'te verilmektedir.

Tablo 5.13: Kontrol/kaçış karakterleri

Karakter	Anlamı	Karakter	
\n	Yeni satır	\\	\
\r	Satırbaşı	\"	"
\b	Backspace	\'	'
\a	Zil (Beep) sesi	\?	?
\t	Yatay TAB	\ddd	8 tabanındaki sayının ASCII karşılığı
\v	Düşey TAB (Yazıcıda yeni satır)	\xddd	16 tabanındaki sayının ASCII karşılığı
\f	Yazıcı için yeni sayfa başı yap.	\0	NULL

C

```
/* Yazdırma-2 */
#include <stdio.h>
#include <conio.h>
void main()
{  int a,b,c;
   clrscr(); printf("Birinci sayıyı giriniz: "); scanf("%d",&a);
   printf("İkinci sayıyı giriniz: "); scanf("%d",&b); printf("\n");
   c=a+b; printf("%d+%d=%d dir.",a,b,c);
   getch(); }
```

```
Birinci sayıyı giriniz: 15
İkinci sayıyı giriniz: 30

15+30=45 dir.
```

2. cprintf

"printf" komutu renkli yazımlarda kullanılamazken, yerine "cprintf" ile renkli çıktılar oluşturulabilir.

Kullanım şekli:

```
cprintf( biçim ifadesi , değişkenler listesi );
```

3. putchar

Belirtilen karakteri ekrana yazar.

Kullanım şekli:

```
putchar( karakter değişken );
```

4. puts

Belirtilen alfasayısal veriyi ekrana yazar ve bir alt satıra geçer.

Kullanım şekli:

```
puts( değişken );
```

5. cputs

Belirtilen alfasayısal veriyi ekrana, ayarlanan renkte yazar.

Kullanım şekli:

```
cputs( değişken );
```

B. C++'daki Veri Çıkış Komutları

1. cout

C++ programlama dilindeki temel veri çıkış/yazdırma komutudur.

Kullanım şekilleri:

```
cout << değişken ;
cout << " ifade ";
cout << biçimleyici << değişken ;
```

"<<" işaretinden sonra yer alan '*biçimleyici*' ifadeleri belirli gruplar altında toplanarak özetlenebilir ve bunların bazıları Tablo 5.14-5.19'da verilmektedir.

Not

Bazı biçimleyicileri kullanabilmek için "*iomanip*" kütüphanesi çağırılmalıdır.

Not

C++'daki "*cout*" ile birlikte C'deki kontrol/çıkış karakterleri yer alabilir. Ayrıca C'deki veri çıkış komutları, ilgili kütüphaneler çağırılarak kullanılabilir.

Tablo 5.14: Sonlandırıcı biçimleyicileri

Biçimleyici	Görevi
endl	Satırı sonlandır ve yeni satırbaşı yap
ends	"NULL" (\0) karakteri ekle

Tablo 5.15: Hizalama biçimleyicileri

Biçimleyici	Görevi
internal	İlgili konuma karakterler ekleyerek alanı hizalama
left	Sola hizalama
right	Sağa hizalama

Tablo 5.16: Çıkış alanı biçimleyicileri

Biçimleyici	Görevi
width(*n*)	Minimum çıkış alanını ayarla
setw(*n*)	Minimum çıkış alanını ayarla
setfill('*k*')	Çıkış alanındaki boş kalan yerlere doldurulacak olan karakteri ayarla

Tablo 5.17: Gösterim biçimleyicileri

Biçimleyici	Görevi
boolalpha	Mantıksal alfasayısal değer göster
noboolalpha	Mantıksal sayısal değer göster
showbase	Sayısal taban önekini göster
noshowbase	Sayısal taban önekini gösterme
showpoint	Ondalık noktasını göster
noshowpoint	Ondalık noktasını gösterme
showpos	Pozitif işaretini göster
noshowpos	Pozitif işaretini gösterme
skipws	Özel karakterleri atla
noskipws	Özel karakterleri atlama
uppercase	Büyük harfe dönüştür
nouppercase	Büyük harfe dönüştürme

Tablo 5.18: Sayısal taban biçimleyicileri

Biçimleyici	Görevi
dec	10 tabanı
hex	16 tabanı
oct	8 tabanı
setbase(*taban*)	Taban ayarlar (taban 10, 16 ve 8 olabilir)

Tablo 5.19: Kayan-nokta biçimleyicileri

Biçimleyici	Görevi
fixed	Sabit noktalı gösterim kullan
scientific	Bilimsel gösterim kullan
setprecision(*n*)	Ondalık basamak sayısını ayarla

C++

```
// Yazdırma-1
#include <iostream>
#include <iomanip>
using namespace std;
int main ()
{ int n=2009;
  cout.width(10); cout << left << n << endl;
  cout.width(15); cout << internal << n << endl;
  cout.width(10); cout << right << n << endl;
  cout << setfill('x') << setw(10) << n << endl;
  system("pause");return 0; }
```

```
2009
      2009
   2009
xxxxxx2009
```

C++

```
// Yazdırma-2
#include <iostream>
#include <iomanip>
using namespace std;
int main ()
{ double a=1234.56789;
  cout << scientific << a << endl;
  cout << setprecision (15) << a << endl;
  cout << fixed;
  cout << setprecision (3) << a << endl;
  cout << setprecision (15) << a << endl;
  system("pause");return 0; }
```

```
1.234568e+003
1.234567890000000e+003
1234.568
1234.567890000000000
```

C++

```
// Yazdırma-3
#include <iostream>
#include <iomanip>
using namespace std;
int main ()
{ char a,b,c;
  bool d=1;
  cout << setbase(16) << 33 << endl;
  cout << boolalpha << d << endl;
  cout << "Onunde bosluklar olan sayi: ";
  cin >> skipws >> a >> b >> c;
  cout <<"a= "<<a<<"\t"<<"b= "<<b<<"\t"<<"c= "<<c<<"\n";
  system("pause");return 0; }
```

```
21
true
Onunde bosluklar olan sayi:    987
a= 9   b= 8   c= 7
```

C. C#'daki Veri Çıkış Komutları

1. Write

C# konsol uygulamalarındaki temel veri çıkışı/yazdırma komutudur. Belirtilen veriyi, istenilen biçimde standart çıkışa aktarır/yazdırır. "`WriteLine()`" yöntemiyse, yazdırma işlemini tamamlandıktan sonra, imleci yeni satıra geçirir (satırbaşı yaptırır).

Kullanım şekilleri:

```
Console.Write( değişken );
Console.Write(" ifade-biçim ");
```

"`Write`" veya "`WriteLine`"deki veri biçimleyicisi genel olarak *{indeks [,hizalama] [: biçim]}* şeklindekidir.

a. ***İndeks***: Sıfır(0)'dan başlayarak değişkenler listesindeki sıradaki/ilgili veriyi belirtir.

b. ***Hizalama***: Kullanımı zorunlu olmayan bu parametre, yazdırma alanını (hizalama şeklini) belirten işaretli tamsayıdır. Sağa hizalama için pozitif değer kullanılırken, sola hizalamada negatif değerlerden faydalanılır. Eğer yazdırılacak olan biçimlendirilmiş verinin uzunluğu (karakter sayısı), belirtilen hizalama değerinden büyükse; hizalama parametresi görevini gerçekleştiremez.

c. ***Biçim***: Kullanımı zorunlu olmayan bu parametre ile biçimlendirilecek verinin tipi belirtilir. Tablo 5.20'de bu genel biçimleyiciler verilmektedir. Ayrıca sayısal ("0", "#", ".", ",", ";", "\" vb.) ve zaman verileri için ilave biçimleyiciler bulunmaktadır.

Tablo 5.20: Sayısal, zaman ve sayılabilir biçimleyiciler

Sayısal biçimler		Zaman biçimleri		Sayılabilir biçimler	
Harf	**Biçimi**	**Harf**	**Biçimi**	**Harf**	**Biçimi**
C , c	Para birimi	d	Kısa tarih	G , g	Alfasayısal
D , d	On tabanı	D	Uzun tarih	F , f	Alfasayısal
E , e	Üstel (bilimsel)	f	Tam tarih/saat (kısa)	D , d	Tamsayı
F , f	Sabit noktalı	F	Tam tarih/saat (uzun)	X , x	Onaltı tabanı
G , g	Genel	g	Genel tarih/saat (kısa)		
N , n	Sayı	G	Genel tarih/saat (uzun)		
P , p	Yüzde	M, m	Ay/gün		
R , r	Round-trip	O,o	Round-trip tarih/saat		
X , x	Onaltı tabanı	R,r	RFC1123		
		s	Sıralanabilir tarih/saat		
		t	Kısa saat		
		T	Uzun saat		
		u	Evrensel sıralanabilir tarih/saat		
		U	Evrensel tam tarih/saat		
		Y,y	Ay/yıl		

C#

```
// Veri çıkışları/yazdırma
using System;
namespace ornek
{ class Program
    { enum renk { Kırmızı=1, Yeşil, Mavi };
      static DateTime b=DateTime.Now;
      static void Main(string[] args)
        { double a=1234.5678d; Console.Clear();
          Console.WriteLine("Bazı veri biçimlendirmeleri:");
          Console.WriteLine(a); Console.WriteLine(b);
          Console.WriteLine("{0:E}", a);
          Console.WriteLine("{0:U}", b);
          Console.WriteLine("{0:F}", renk.Mavi);
          Console.WriteLine("{0:F0}\t{1:D}\t{2:D}", a, b, renk.Yeşil);
          Console.ReadLine();          }    }    }
```

```
Bazı veri biçimlendirmeleri:
1234,5678
09.09.2013 14:43:31
1,234568E+003
09 Eylül 2013 Pazartesi 11:43:31
Mavi
1235    09 Eylül 2013 Pazartesi 2
```

D. Java'daki Veri Çıkış Komutları

1. print

Java programlarındaki temel ekrana yazdırma yöntemidir. Parantez içindeki değişkenin değerini veya çift tırnak içindeki ifadeyi yazdırır. "`println`" da aynı görevde olup yazma işlemini gerçekleştirdikten sonra, imleci bir alt satıra geçirir (satır başı yapar).

Kullanım şekilleri:

```
System.out.print( değişken );
System.out.print(" ifade ");
```

2. printf

Ekranda belirli bir biçimde değerleri/ifadeleri yazdırmak (formatlı çıkışlar) için Java programlarında "`printf`" kullanılır.

Kullanım şekli:

```
System.out.printf( biçim ifadesi , değişken );
```

"*biçim ifadesi*" C'deki "`printf`" komutundaki gibidir. Tablo 5.21-22'de de bazı temel tip karakterleri verilmektedir.

Tablo 5.21: Java'daki bazı tip karakterleri

Karakter	Giriş	Çıkış biçimi
	Sayılar	
d	Tamsayı	10 tabanında tamsayı
o	Tamsayı	8 tabanında tamsayı
x	Tamsayı	16 tabanında tamsayı (harfler küçük)
X	Tamsayı	16 tabanında tamsayı (harfler büyük)
f	Ondalıklı sayı	Ondalıklı sayı
e	Ondalıklı sayı	Üstel gösterimde (bilimsel) ondalıklı sayı (e)
E	Ondalıklı sayı	Üstel gösterimde (bilimsel) ondalıklı sayı (E)
g	Ondalıklı sayı	Belirtilen değer ve hassasiyete göre *e* veya *f* biçiminde ondalıklı
G	Ondalıklı sayı	*g* ile aynı, *e* kullanılırsa üs belirteci "E" olur
a	Ondalıklı sayı	16 tabanında ve üstel gösterimli sayı (e)
A	Ondalıklı sayı	16 tabanında ve üstel gösterimli sayı (E)
	Karakterler	
c	Karakter	Tek karakter
C	Karakter	Tek karakter (büyük harf)
s	Karakter dizisi	Karakter dizisi
S	Karakter dizisi	Karakter dizisi (büyük harflerle)
%		%
	Tarih ve saat	
t veya T	Tarih / saat	Tarih ve saat gösterimleri (Tablo 5.24)

Tablo 5.22: Bazı tarih/saat tip karakterleri

Karakter	Biçim
R	saat:dakika (24 saat)
T	saat:dakika:saniye (24 saat)
r	saat:dakika:saniye öğleden önce/sonra (12 saat)
D	ay:gün:yıl
F	yıl:ay:gün
c	tarih ve saat

JAVA

```
// Yazdırma işlemleri-1
public class yazdirma1 {
public static void main(String[] args)
{ System.out.print("C "); System.out.println("C++");
  System.out.printf("%X\r",27); System.out.printf("%.3f\n",123.456789);
  System.out.printf("%e\n",123.45); System.out.printf("%S\n","java");
  System.out.printf("%5.4s\n","PROGRAMLAMA"); }}
```

```
C C++
1B
123,457
1.234500e+02
JAVA
  PROG
```

JAVA

```
// Yazdırma işlemleri-2
import java.util.Date;
public class yazdirma2 {
public static void main(String[] args)
{ Date zaman=new Date();
  System.out.printf("%tR\n",zaman); System.out.printf("%tT\n",zaman);
  System.out.printf("%tr\n",zaman); System.out.printf("%tD\n",zaman);
  System.out.printf("%tF\n",zaman);System.out.printf("%tc\n",zaman); }}
```

```
17:10
17:10:06
05:10:06 PM
08/16/09
2009-08-16
Paz Ağu 16 17:10:06 EEST 2009
```

E. MATLAB'daki Veri Çıkış Komutları

1. disp

Kullanıcının belirlediği değişkenin içeriğini veya mesajı, değişken ismini göstermeden görüntüler. MATLAB'ta değişken içerikleri herhangi bir komut/fonksiyon kullanılmadan da – sadece değişken ismi yazılarak – gösterilebilmektedir.

Kullanım şekilleri:

```
disp( ' mesaj ' );
disp( değişken );
```

MATLAB

```
% Yazdırma-1
clear all; clc; a=16; disp('Bursa'); disp(a); a
```

```
Bursa
   16

a =
   16
```

MATLAB

```
% Yazdırma-2  Hyperlink
clear all; clc;
disp('<a href = "http://www.fahriv.com">Fahri Vatansever</a>')
```

Fahri Vatansever

2. fprintf

Belirtilen veriyi, istenilen biçimde ekrana veya dosyaya yazdırır.

Kullanım şekilleri:

```
fprintf( ' biçim ' , değişken );
fprintf( dosya değişkeni , ' biçim ' , değişken );
sayı = fprintf( dosya değişkeni , ' biçim ' , değişken );
```

"%" karakteriyle başlayan biçim ifadesinin genel yapısı

%	Tanımlayıcı	$	Bayrak belirteci	Genişlik belirteci	.	Hassasiyet belirteci	Tip niteleyici	Tip (dönüşüm) karakteri

şeklindedir. Buradaki "*Tanımlayıcı*", işleme alınacakların/yazdırılacakların sırasını belirten sayıdır ve "$" karakteriyle biter. Bayrak-genişlik-hassasiyet belirteci, tip niteleyici ve tip (dönüşüm) karakterleri sırasıyla Tablo 5.23-5.26'da verilmektedir. Tablo 5.27'de ise kontrol/kaçış karakterleri yer almaktadır. Üçüncü kullanım şeklindeyse "*sayı*" değişkenine, yazdırılan verilerin (Byte) sayısı aktarılmaktadır.

Tablo 5.23: Bayrak belirteci karakterleri

Karakter	Anlamı	
-	Sola hizalar.	
+	İşaret karakterini (+ veya -) yazar.	
boşluk	Değerden önce boşluk ekler (boşluklarla doldurur).	
0	Sıfırlarla doldurur.	
#	*Ardındaki karakter*	*Anlamı*
	o , x , X	0, 0x veya 0X önekini ekler.
	f , e , E	Hassasiyet belirteci 0 olsa bile ondalık noktasını yazar.
	g, G	Ondalık noktası ve izleyen sıfırlar atılmaz.

Tablo 5.24: Genişlik belirteci karakterleri

Karakter	Anlamı
n	En az "n" tane karakter yazdırılır.
*	Genişlik için gerekli değer, değişkenler (argümanlar) listesinden okunur.

Tablo 5.25: Hassasiyet belirteci karakterleri

Karakter	Tip	Anlamı
n	f , e , E	"n" tane ondalık basamak (hane) yazdırılır.
	g , G	"n" tane anlamlı basamak yazdırılır.
*		Genişlik için gerekli değer, değişkenler listesinden okunur.

Tablo 5.26: Tip karakterleri

Veri tipi	Karakter	Anlamı
İşaretli tamsayı	d	İşaretli 10 tabanında tamsayı
	i	İşaretli 10 tabanında tamsayı
İşaretsiz tamsayı	u	İşaretsiz 10 tabanında tamsayı
	o	İşaretsiz 8 tabanında tamsayı
	x	İşaretsiz 16 tabanında tamsayı (harfler küçük)
	X	İşaretsiz 16 tabanında tamsayı (harfler büyük)
Ondalıklı sayı (Kayan noktalı sayı)	f	Kayan noktalı gösterim
	e	Üstel gösterim
	E	*e* ile aynı, üs belirteci "E"
	g	Anlamsız sıfırlar olmadan *e* veya *f* formunda işaretli sayı
	G	*g* ile aynı, *e* formatı kullanılırsa üs belirteci "E"
	bu	Çift duyarlıklı 10 tabanında sayı
	bx	Çift duyarlıklı 16 tabanında sayı
	bX	Çift duyarlıklı 16 tabanında sayı
	bo	Çift duyarlıklı 8 tabanında sayı
	tu	Tek duyarlıklı 10 tabanında sayı
	tx	Tek duyarlıklı 16 tabanında sayı
	tX	Tek duyarlıklı 16 tabanında sayı
	to	Tek duyarlıklı 8 tabanında sayı
Karakter	c	Tek karakter
	s	Karakter dizisi

Tablo 5.27: Kontrol/kaçış karakterleri

Karakter	Anlamı	Karakter	Anlamı
\n	Yeni satır	\a	Alarm (beep) sesi
\t	Yatay TAB	\f	Yeni sayfa başı
\v	Düşey TAB	\\	\
\r	Satır başı	\"	"
\b	Backspace	%%	%

MATLAB

```
% Yazdırma-1
fprintf('%3$s\t%1$s\t%2$s\n','A','B','C')
```

```
C       A       B
```

MATLAB

```
% Yazdırma-2
clear all; clc ; a=123.456;
fprintf('%- 20.2f\n',a); fprintf('%+015.5f\n',a);
printf('%#5.0f\n',a); fprintf('%*.*f\n',0,4,a);
```

```
123.46
+00000123.45600
123.
123.4560
```

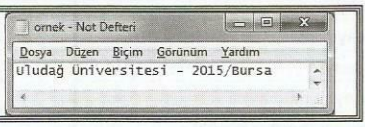
MATLAB

```
% Yazdırma-3 Hyperlink
fprintf('<a href = "http://fahriv.home.uludag.edu.tr/">Fahri Vatansever
kişisel web sayfası</a>\n');
```

Fahri Vatansever kişisel web sayfası

MATLAB

```
% Dosyaya yazdırma
clear all; clc; dosya=fopen('ornek.txt','wt');
a='Uludağ Üniversitesi'; b=2015 ; c='Bursa';
fprintf(dosya,'%s - %d/%s\n',a,b,c); fclose(dosya);
```



3. sprintf

Belirtilen veriyi, alfasayısal değişkene yazdırır.

Kullanım şekilleri:

```
sprintf( ' biçim ' , değişken );
[ alfasayısal ]=sprintf( ' biçim ' , değişken );
[ alfasayısal , hata mesajı ]=sprintf( ' biçim ' , değişken );
```

MATLAB

```
% Altın oranı yazdırma
clear all; clc; sprintf('%0.15f\n',(1+sqrt(5))/2)
```

```
ans =
  1.618033988749895
```

F. Python'daki Veri Çıkış Komutları

1. print

Ekrana veya dosyaya yazmak için Python'da kullanılan temel komuttur.

Kullanım şekilleri:

```
print(' mesaj ') veya print(" mesaj ") veya print(""" mesaj """)
print( değişken )
print( " ifade " % değişken )
print( " biçim " %( değişkenler ))
print( ifade/değer, ...,sep="...",end="...",file=...,flush )
```

Diğer dillerde de olduğu gibi biçim ifadesinin genel yapısı

şeklindedir. "sep" parametresi yazdırılan verilerin hangi karakterle ayrılacağını belirlerken, "end" parametresi de yazdırılan verilerin sonuna gelecek karakteri belirtmektedir (Tablo 5.28-5.29).

Tablo 5.28: Bayrak belirteci karakterleri

Karakter	Anlamı
-	Sola hizalar.
+	İşaret karakterini (+ veya -) yazar.
boşluk	Yazdırılacak işaret yoksa değerden önce boşluk ekler.
0	Sıfırlarla doldurur.
#	o, x veya X ile birlikte kullanıldığında 0, 0o, 0O, 0x veya 0X öneklerini oluşturur.

Tablo 5.29: Tip/dönüştürme karakterleri

Veri tipi	Karakter	Anlamı
İşaretli tamsayı	d	İşaretli 10 tabanında tamsayı
	i	İşaretli 10 tabanında tamsayı
İşaretsiz tamsayı	u	İşaretsiz 10 tabanında tamsayı
	o	İşaretsiz 8 tabanında tamsayı
	x	İşaretsiz 16 tabanında tamsayı (harfler küçük)
	X	İşaretsiz 16 tabanında tamsayı (harfler büyük)
Ondalıklı sayı (Kayan noktalı sayı)	f	Kayan noktalı gösterim
	F	Kayan noktalı gösterim
	e	Üstel gösterim (küçük harf)
	E	Üstel gösterim (büyük harf)
	g	Üs değeri -4'ten büyük veya hassasiyetten küçükse "*e*" ile, aksi durumda "*f*" ile aynı
	G	*g* ile aynı (büyük harf)
Karakter	c	Tek karakter
	s	Karakter dizisi (*str()* ile dönüştürme)
	r	Karakter dizisi (*repr()* ile dönüştürme)
Yok	%	%

PYTHON

```
# Yazdırma - 1
print("Çift tırnak içinde yazdırma")
print('Tek tırnak içinde yazdırma')
print("""Üç çift tırnak içinde yazdırma""")
print("Python'u öğreniyorum") # farklı tırnak kullanımı
print('Pytnon\'u öğreniyorum') # \ yeni satırda ifadenin devamı
print("16","Bursa","2017",sep="-") # aralarına - ekler
print("Uludağ Üniversitesi\nElektrik-Elektronik Müh.\nBursa\tTürkiye")
# kaçış karakterleri
```

```
Çift tırnak içinde yazdırma
Tek tırnak içinde yazdırma
Üç çift tırnak içinde yazdırma
Python'u öğreniyorum
Pytnon'u öğreniyorum
16-Bursa-2017
Uludağ Üniversitesi
Elektrik-Elektronik Müh.
Bursa	Türkiye
```

PYTHON

```
# Yazdırma - 2
a=99;b=123.4567;c="Bursa"
print("%d"%a) # Tamsayı yazdırma
print("%d"%b) # Tamsayı yazdırma
print("%X"%a) # 16 tabanında yazdırma
print("%10.4o"%a) # 8 tabanında yazdırma
print("%0.2f"%b) # Ondalıklı sayı yazdırma
print("%0.8f"%b) # Ondalıklı sayı yazdırma
print("%10.4e"%b) # Ondalıklı sayı yazdırma
print("%10.4g"%b) # Ondalıklı sayı yazdırma
print(a,b,c)
print("%d\t%f\t%f"%(a,b,a+b))
print("%c\t%s\t%d"%(c[0],c,a/3))
d="%s'nın plaka kodu %d'dır"%(c,16)
print(d)
```

```
99
123
63
    0143
123.46
123.45670000
1.2346e+02
   123.5
99 123.4567 Bursa
99	123.456700	222.456700
B	Bursa	33
Bursa'nın plaka kodu 16'dır
```

2.Çıkışları biçimlendirme

Veri çıkışlarını biçimlendirmek için Python, farklı yöntemler sunmaktadır.

a. format

İstenilen değeri, belirtilen biçime dönüştürür.

Kullanım şekilleri:

```
format( değer , biçim )
```

Parametrelerin gösterimi açısından kolaylık sağlaması için Python'daki "`print`" kullanımını C'deki "`printf`" kullanımına; Python'daki "`format`" kullanımını da C#'taki "`Write`" kullanımına benzetebilirsiniz (Tablo 5.30-5.33).

Tablo 5.30: Hizalama seçenekleri

Seçenek	Anlamı
<	Sola hizalama
>	Sağa hizalama
^	Ortala
=	Sayılarda dolgunun işaretten sonra ve basamaklardan önce gelmesi

Tablo 5.31: İşaret seçenekleri

Seçenek	Anlamı
+	Hem pozitif hem negatif sayılarda işaret
-	Sadece negatif sayılarda işaret
Boşluk	Pozitif sayılarda boşluk, negatif sayılarda işaret

Tablo 5.32: Hizalama seçenekleri

Seçenek	Anlamı
#	İlgili tamsayı (ikili, sekizli, onaltılı) çıkışlara 0b, 0o veya 0x öneklerini oluşturur
,	Sayısal çıkışlarda virgül ile üçlü gruplandırmalar
Genişlik	Minimum alan genişliğini belirler
Hassasiyet	Kayan noktalı (ondalıklı) sayılarda ondalık basamak sayısını belirler

Tablo 5.33: Tip/dönüştürme karakterleri

Veri tipi	Karakter	Anlamı
Tamsayı	**b**	2 tabanı
	c	Karakter
	d	10 tabanı
	o	8 tabanı
	x	16 tabanı (harfler küçük)
	X	16 tabanı (harfler büyük)
	n	*d* ile aynı, ancak geçerli yerel ayarlara göre ayırıcı karakter
	Yok	*d* ile aynı
Ondalıklı sayı (Kayan noktalı sayı)	**f**	Kayan noktalı gösterim
	F	Kayan noktalı gösterim
	e	Üstel gösterim (küçük harf)
	E	Üstel gösterim (büyük harf)
	g	Genel gösterim (küçük harf)
	G	Genel gösterim (büyük harf)
	n	*g* ile aynı, ancak geçerli yerel ayarlara göre ayırıcı karakter
	%	Yüzde gösterim
	Yok	*g* ile aynı
Karakter	**c**	Tek karakter
	s	Karakter dizisi (*str*)
	r	Karakter dizisi (*repr*)
	Yok	*s* ile aynı

PYTHON

```
# Yazdırma - 3
a=99;b=123.4567;c="Bursa"
print("1. değer: {}, 2. değer: {}".format(a,b))
print("1. değer: {0}, 2. değer: {1}".format(a,b))
print("Dikkat! 1. değer: {1}, 2. değer: {0}".format(a,b))
print("1. değer: {0}, 2. değer: {1:0.2f}".format(a,b))
print("1. değer: {0:03d}, 2. değer: {1:<10.1f}, 3. değer:
{2:>10s}".format(a,b,c)); print("{:.2}".format(c))
print("{:^15}".format(c)); print("{:+<10}".format(c))
```

```
1. değer: 99, 2. değer: 123.4567
1. değer: 99, 2. değer: 123.4567
Dikkat! 1. değer: 123.4567, 2. değer: 99
1. değer: 99, 2. değer: 123.46
1. değer: 099, 2. değer: 123.5    , 3. değer:
Bursa
Bu
   Bursa
Bursa+++++
```

b. center

Alfasayısal veriyi, belirtilen değere göre ortalamakta ve istenildiğinde seçilen karakterlerle doldurmaktadır.

Kullanım şekilleri:

```
veri.center( genişlik ," doldurma karakteri ")
```

c. ljust

Alfasayısal veriyi, belirtilen değere göre sola hizalamakta ve istenildiğinde seçilen karakterlerle doldurmaktadır.

Kullanım şekilleri:

```
veri.ljust( genişlik ," doldurma karakteri ")
```

d. rjust

Alfasayısal veriyi, belirtilen değere göre sağa hizalamakta ve istenildiğinde seçilen karakterlerle doldurmaktadır.

Kullanım şekilleri:

```
veri.rjust( genişlik ," doldurma karakteri ")
```

e. zfill

Alfasayısal verinin sol tarafını (önünü) sıfırlarla belirtilen uzunlukta doldurmaktadır.

Kullanım şekilleri:

```
veri.zfill( genişlik )
```

PYTHON

```
# Yazdırma - 3
a="Bursa"; b="16"
print(a.center(11,"*")); print(a.ljust(11,"~"))
print(a.rjust(11,":")); print(b.zfill(11))
```

```
***Bursa***
Bursa~~~~~~
::::::Bursa
00000000016
```

Tablo 5.34'te C, C++, C#, Java, MATLAB ve Python dillerindeki temel veri çıkış/yazdırma komutları karşılaştırmalı olarak verilmektedir.

Tablo 5.34: C, C++, C#, Java, MATLAB ve Python'daki veri çıkış komutları

Algoritma	C	C++	C#	Java	MATLAB	Python
Yaz	printf	cout	Write	printf	fprintf	print

IV. DÖNGÜ (ÇEVRİM) KOMUTLARI

Ardışık veya tekrarlı işlemlerin yapılmasını sağlayan komutlardır. Döngüler, üçe ayrılırlar:

i. *Sayıcılı döngüler*: Döngü işlemleri bir sayaca bağlı olarak gerçekleştirilir.

ii. *Ön koşullu döngüler*: Döngü içindeki işlemler koşula/koşullara bağlı olarak gerçekleştirilir ve bu koşul/koşullar döngü öncesinde kontrol edilir.

iii. *Son koşullu döngüler*: Döngü içindeki işlemler koşula/koşullara bağlı olarak gerçekleştirilir ve bu koşul/koşullar döngü sonunda kontrol edilir.

Not

Ön koşullu ve son koşullu döngülerde;

- ✓ Döngü değişkenine, döngü öncesinde başlangıç değeri verilmelidir.
- ✓ Döngü değişkeni, döngü içerisinde arttırılmalı/azaltılmalıdır (değiştirilmelidir). Değiştirilmemesi durumunda; döngü, sonsuz çevrime girer.

Not

Ön koşullu ve son koşullu döngüler arasındaki temel farklar:

- ➢ Döngü koşulu/koşulları;
  - ❖ *Ön koşullu döngüde, döngü başında*
  - ❖ Son koşullu döngüde, döngü sonunda kontrol edilir.
- ➢ Döngü işlemi/işlemleri;
  - ❖ Ön koşullu döngüde, koşul/koşullar döngünün başında kontrol edildiğinden hiç gerçekleştirilmeyebilir.
  - ❖ Son koşullu döngüde, koşul/koşullar döngünün sonunda kontrol edildiğinden en az bir defa gerçekleştirilir.

A. C/C++/C#/Java'daki Döngü Komutları

C, C++, C# ve Java programlama dillerinde üç döngü yapısı da bulunmaktadır: `'for'`, `'while'` ve `'do-while'`. Her üç dildeki komutlar aynı olduğu için bu döngü komutları, ayrı başlık altında incelenmeyip sadece var olan küçük farklılıkları belirtilecektir.

1. for

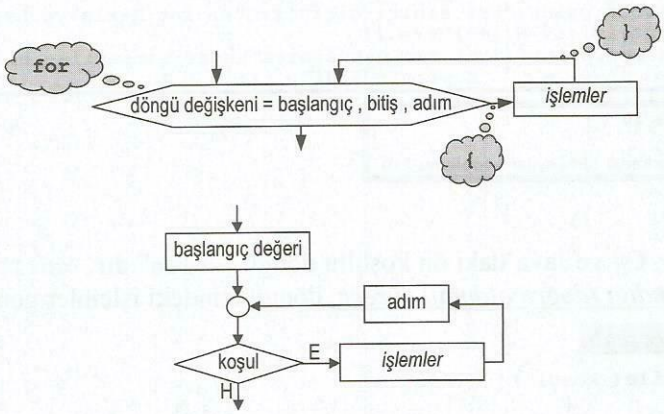
Tüm dillerde olduğu gibi C/C++/C#/Java'da sayıcılı döngü komutudur. "`for`" döngüsünde; koşul *sağlandığı (doğru olduğu) sürece*, döngü bloğu işlemleri yapılır.

Kullanım şekli (C/C++/C#/Java):

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

Kullanım şekli (C++/C#/Java):

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



C

```
/*Döngü 1*/
#include <stdio.h>
#include <conio.h>
void main()
{  int i,N; float t=0;
   clrscr();printf("Üst sınırı giriniz: "); scanf("%u",&N);
   for(i=1;i<=N;i++) t+=i;
   printf("\n1 den üst sınıra kadar tamsayıların toplamı: %0.0f",t);
   getch(); }
```

C++

```
// Döngü 1 -> C kodunun karşılığı
#include <iostream>
using namespace std;
int main()
{   int N; float t=0;
    system("cls");cout<<"Üst sınırı giriniz:";cin >> N;
    for(int i=1;i<=N;i++) t+=i;
    cout <<"\n1 den üst sınıra kadar tamsayıların toplamı: "<<t<<endl;
    system("pause");return 0; }
```

C#

```
// Döngü 1 -> C kodunun karşılığı
using System;
namespace ornek
{ class Program
   { static void Main(string[] args)
      { float t=0;
        Console.Clear();Console.Write("Üst sınırı giriniz: ");
        int N=Convert.ToInt32(Console.ReadLine());
        for (int i=1;i<=N;i++) t+=i;
        Console.Write("\n1 den üst sınıra kadar tamsayıların
                       toplamı: {0:F0}",t);   Console.ReadLine(); }}}
```

JAVA

```
// Döngü 1 -> C kodunun karşılığı
import java.util.Scanner;
public class ornek {
  public static void main(String[] args) {
    float t=0; Scanner klavye=new Scanner(System.in);
    System.out.print("Üst sınırı giriniz: "); int N=klavye.nextInt();
    for (int i=1;i<=N;i++) t+= i;
    System.out.printf("\n1 den üst sınıra kadar tamsayıların
                       toplamı: %.0f", t);     }   }
```

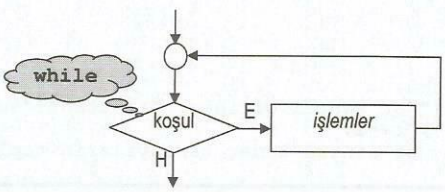
```
Üst sınırı giriniz: 20

1 den üst sınıra kadar tamsayıların toplamı: 210
```

2. while

C, C++, C# ve Java'daki ön koşullu döngü "`while`" dır. "`while`" ile verilen koşul *sağlandığı (doğru olduğu) sürece*, döngü içindeki işlemler gerçekleştirilir.

Kullanım şekli:

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



C

```
/* Döngü 2 */
#include <stdio.h>
#include <conio.h>
void main()
{ int i,N; float t=0;
  clrscr();printf("Tek sayıların üst sınırı: ");scanf("%u",&N); i=1;
  while(i<=N) { t+=i; i+=2; }
  printf("\nToplamı: %0.0f",t); getch(); }
```

C++

```
// Döngü 2 -> C kodunun karşılığı
#include <iostream>
using namespace std;
int main()
{   int N; float t=0;
    system("cls"); cout << "Tek sayıların üst sınırı: "; cin >> N;
    int i=1;
    while(i<=N) { t+=i; i+=2; }
    cout << "\nToplamı: " << t << endl;
    system("pause");return 0; }
```

C#

```
// Döngü 2 -> C kodunun karşılığı
using System;
namespace ornek
{ class Program
   { static void Main(string[] args)
       { float t=0; Console.Write("Tek sayıların üst sınırı: ");
         int N=Convert.ToInt32(Console.ReadLine());   int i=1;
         while (i<=N) { t+=i; i+=2; }
         Console.Write("\nToplamı: {0:F0}", t);
         Console.ReadLine();          }    }   }
```

JAVA

```
// Döngü 2 -> C kodunun karşılığı
import java.util.Scanner;
public class ornek {
  public static void main(String[] args) {
    float t=0; Scanner klavye=new Scanner(System.in);
    System.out.print("Tek sayıların üst sınırı: ");
    int N=klavye.nextInt(); int i=1;
    while (i<=N) { t+=i; i+=2; }
    System.out.printf("\nToplamı: %.0f", t);     }}
```

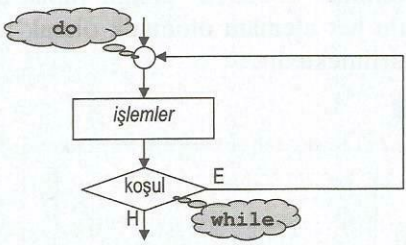
```
Tek sayıların üst sınırı: 10

Toplamı: 25
```

3. do – while

C, C++, C# ve Java'daki son koşullu döngüdür. "`while`" ile belirtilen koşul *sağlandığı (doğru olduğu) sürece*, döngüdeki işlemler yapılır.

Kullanım şekli:

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



C

```
/* Döngü 3 */
#include <stdio.h>
#include <conio.h>
void main()
{ int i,N; float t=0;
  clrscr(); printf("Çift sayıların üst sınırı: "); scanf("%u",&N); i=2;
  do { t+=i; i+=2; } while(i<=N);
  printf("\nToplamı: %0.0f",t);  getch(); }
```

C++

```
// Döngü 3 -> C kodunun karşılığı
#include <iostream>
using namespace std;
int main()
{   int N; float t=0;
    system("cls");cout<<"Çift sayıların üst sınırı: ";cin>>N;int i=2;
    do { t+=i; i+=2; } while(i<=N);
    cout << "\nToplamı: " << t << endl;
    system("pause");return 0; }
```

C#

```
// Döngü 3 -> C kodunun karşılığı
using System;
namespace ornek
{ class Program
  {static void Main(string[] args)
    { float t=0; Console.Write("Çift sayıların üst sınırı: ");
      int N=Convert.ToInt32(Console.ReadLine()); int i=2;
      do { t+=i; i+=2; } while (i<=N);
      Console.Write("\nToplamı: {0:F0}", t); Console.ReadLine();}}}
```

JAVA

```
// Döngü 3 -> C kodunun karşılığı
import java.util.Scanner;
public class ornek {
  public static void main(String[] args) {
    float t=0; Scanner klavye=new Scanner(System.in);
    System.out.print("Çift sayıların üst sınırı: ");
    int N=klavye.nextInt(); int i=2;
    do { t+=i; i+=2; } while (i<=N);
    System.out.printf("\nToplamı: %.0f", t);     }}
```

Çift sayıların üst sınırı: 10

Toplamı: 30

4. foreach

C# programlama dilinde "`foreach`" döngü yapısı bulunmaktadır. Bu yapı ile çoklu veri kümesinin her elemanı otomatik olarak tek tek alınmakta ve işlem/işlemler gerçekleştirilmektedir.

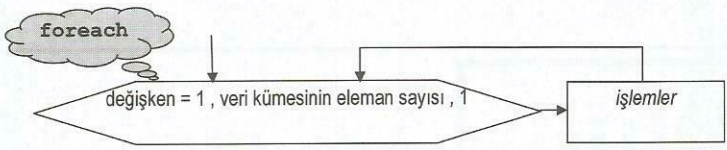
Kullanım şekli (C#):

```
foreach( aktarılacak değişken in veri kümesi )
        { ......................
          ......................   } işlemler
          ......................
        }
```

Kullanım şekli (Java eşdeğeri):

```
for( aktarılacak değişken : veri kümesi )
        { ......................
          ......................
          ......................
        }
```

işlemler



C#

```
// Döngü 4
using System;
namespace ornek
{ class Program
    { static void Main(string[] args)
        {  int[] veri={3,11,33,99};
           foreach (int a in veri)
               Console.Write("{0}\t", a);  Console.ReadLine(); } } }
```

JAVA

```
// Döngü 4 -> C# kodunun eşdeğeri
public class ornek {
    public static void main(String[] args) {
       int veri[]={3,11,33,99};
          for(int a:veri)
          System.out.printf("%d\t",a);     }}
```

```
3 11 33 99
```

B. MATLAB'daki Döngü Komutları

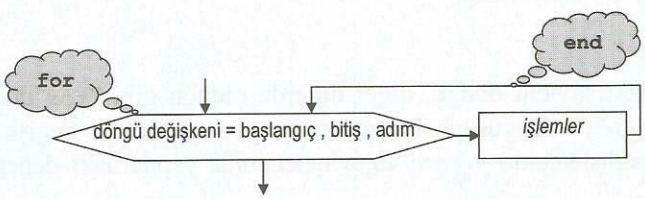
1. for

MATLAB'daki sayıcılı döngüdür. "`for`" döngüsünde "`adım`" değeri 1 ise belirtmeye gerek yoktur. Ayrıca döngü değişkeni, değerlerini bir diziden/kümeden de – C#'taki "`foreach`" ve Java'daki gibi - alabilir.

Kullanım şekli:

```
for döngü değişkeni=başlangıç : adım : bitiş
    ......................
    ......................
    ......................
end;
```

işlemler



MATLAB

```
% Döngülerle çarpım tablosu
clear all; clc
for i=1:10
    for j=1:1:10
        a(i,j)=i*j;
    end
end
a
```

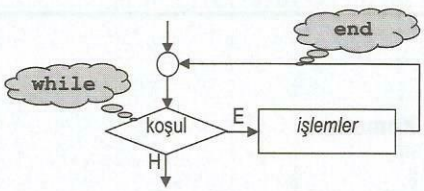
```
a =

   1   2   3   4   5   6   7   8   9  10
   2   4   6   8  10  12  14  16  18  20
   3   6   9  12  15  18  21  24  27  30
   4   8  12  16  20  24  28  32  36  40
   5  10  15  20  25  30  35  40  45  50
   6  12  18  24  30  36  42  48  54  60
   7  14  21  28  35  42  49  56  63  70
   8  16  24  32  40  48  56  64  72  80
   9  18  27  36  45  54  63  72  81  90
  10  20  30  40  50  60  70  80  90 100
```

2. while

MATLAB'daki ön koşullu döngüdür. "`while`" ile verilen koşul *sağlandığı (doğru olduğu) sürece*, döngü içindeki işlemler gerçekleştirilir.

Kullanım şekli:

```
while koşul
      ..........................
      ..........................  } işlemler
      ..........................
end;
```



MATLAB

```
% Ön koşullu döngü
clear all; clc; i=1; t=0;
while t<=55
    fprintf('%d\t',t); t=t+i; i=i+1;
end
```

```
0  1  3  6  10  15  21  28  36  45  55
```

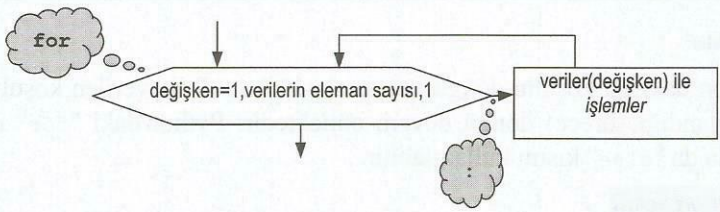
C. Python'daki Döngü Komutları

1. for

Python'daki sayıcılı döngü, diğer dillerde olduğu gibi "`for`"dur. Daha çok C#'taki "`foreach`" döngüsüne benzeyen bu yapıda, döngü değişkeni belirtilen sıra/dizi/kelime/liste/çoklu veriler vb. yinelenebilir yapılardaki değerleri sırasıyla almaktadır.

Kullanım şekli:

```
for döngü değişkeni in sıra :
    .....
    .....        } işlemler
    .....

for döngü değişkeni in sıra :
    .....
    .....        } işlemler
    .....
else:
    .....
    .....        } işlemler
    .....
```



PYTHON

```
# Döngü-1
a="Bursa"
for i in a:
    print(i)
```

```
B
u
r
s
a
```

PYTHON

```
# Döngü-2
a=["C","C++","C#","Java","MATLAB","Python"]
for i in a:
    print(i+" ~ ",end="")
```

```
C ~ C++ ~ C# ~ Java ~ MATLAB ~
Python ~
```

PYTHON

```
# Döngü-3
for i in range(10):
    print("%d\t"%i,end="")
print()
for i in range(1,10):
    print("%d\t"%i,end="")
print()
for i in range(1,10,2):
    print("%d\t"%i,end="")
print()
for i in range(10,0,-2):
    print("%d\t"%i,end="")
```

0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9	
1	3	5	7	9					
10	8	6	4	2					

Tablo 5.35'te C, C++, C#, Java, MATLAB ve Python dillerindeki eşdeğer görevdeki örnek "`for`" kullanımları verilmektedir.

Tablo 5.35: C, C++, C#, Java, MATLAB ve Python'daki örnek "for" kullanımları

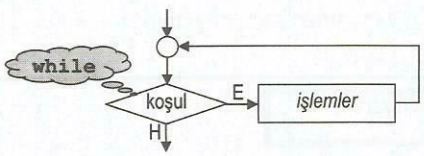
Örnek	C / C++ / C# / Java	MATLAB	Python
i=0,10	for (i=0;i<11;i++)	for i=0:10	for i in range(11):
i=1,10,2	for (i=1;i<11;i+=2)	for i=1:2:10	for i in range(1,11,2):
i=20,2,-4	for (i=20;i>1;i-=4)	for i=20:-4:2	for i in range(20,1,-4):
i=a,b,c	for (i=a;i<b+1;i+=c)	for i=a:c:b	for i in range(a,b+1,c):

2. while

Python'daki ön koşullu döngü yapısı olan "`while`" ile verilen koşul doğru olduğu (sağlandığı sürece) döngü devam etmektedir. Python'daki "`for`" da olduğu gibi burada da "`else`" kısmı kullanılabilir.

Kullanım şekli:

```
while koşul :
        ........................
        ........................  } işlemler
        ........................

while koşul :
        ......................
        ......................  } işlemler
        ......................
else:
        ......................
        ......................  } işlemler
        ......................
```



PYTHON

```
# Döngü-1
s=0
while s<5:
    print(s," < 5")
    s+=1
else:
    print("Döngü koşulu sağlanmadı.",s," = 5")
```

```
0 < 5
1 < 5
2 < 5
3 < 5
4 < 5
Döngü koşulu sağlanmadı. 5 = 5
```

Tablo 5.36'da C, C++, Java, MATLAB ve Python'daki döngü yapıları, karşılaştırmalı olarak verilmektedir. Ayrıca bu dillerdeki döngü kontrol komutları da Tablo 5.37'de yer almaktadır. "*Sonlandır*" komutu, döngüden (döngü bloğundan) çıkışı sağlarken, "*Devam ettir*" de işlem akışını döngü koşuluna yönlendirir (bir sonraki döngü değeriyle devam etmesini sağlar).

Tablo 5.36: C, C++, C#, Java, MATLAB ve Python'daki eşdeğer görevdeki döngü komutları

Döngü türü	C	C++	C#	Java	MATLAB	Python
Sayıcılı	for	for	for	for	for	for
Ön koşullu	while	while	while	while	while	while
Son koşullu	do-while	do-while	do-while	do-while		

Tablo 5.37: C, C++, C#, Java, MATLAB ve Python'daki döngü kontrol komutları

Döngüyü	C	C++	C#	Java	MATLAB	Python
Sonlandır	break	break	break	break	break	break
Devam ettir	continue	continue	continue	continue	continue	continue

V. KARAR (KARŞILAŞTIRMA) KOMUTLARI - "Eğer"

Koşulların kontrolünde kullanılan komutlardır. Koşulların doğru olup olmamasına (sağlanıp sağlanmamasına) göre işlem akışını yönlendirirler. Karar komutları dört farklı yapıda (biçim, form) olabilirler:

i. *Yarım form*: Sadece koşul/koşullar doğru olduğunda (doğru ise – evet ise) yapılacak işlem/işlemler vardır.

ii. *Tam form*: Koşul/koşullar doğru ve yanlış olduğunda yapılacak işlem/işlemler vardır.

iii. *Çok koşullu form*: Birçok koşulun durumuna göre yapılacak işlem/işlemler vardır.

iv. *Seçimli form*: Seçim/kontrol değişkeninin değerine göre yapılacak işlem/işlemler vardır. Bu form; "çok koşullu form"un sade biçimidir.

Not

Döngülerinde olduğu gibi; C, C++, C# ve Java karar yapılarındaki işlem tek komuttan oluşuyorsa, blok açmaya gerek yoktur.

A. C/C++/C#/Java'daki Karar Komutları

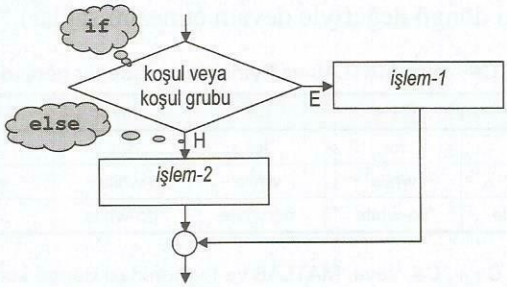
1. if-else if-else

C/C++/C#/Java'da "`if-else`" yapısının tüm biçimleri mevcuttur.

Kullanım şekilleri (C/C++/C#/Java):

```
if ( koşul veya koşul grubu )  işlem;

if ( koşul veya koşul grubu )  işlem-1;
else işlem-2 ;
```

```
if ( koşul1 veya koşul grubu1 )  işlem-1;
else if ( koşul2 veya koşul grubu2 )  işlem-2;
else if ( koşul3 veya koşul grubu3)   işlem-3;
.....................................
else  işlem-n ;
```



C

```
/* Karar 1 */
#include <stdio.h>
#include <conio.h>
void main()
{  int a;
   clrscr();printf("Bir tamsayı giriniz => ");scanf("%d",&a);
   if (a>0) printf("Pozitif");
   else if (a<0) printf("Negatif");
   else printf("Sıfır");
   getch(); }
```

C++

```
// Karar 1 -> C kodunun karşılığı
#include <iostream>
using namespace std;
main()
{   int a;
    system("cls"); cout << "Bir tamsayi giriniz => "; cin >> a;
    if (a>0) cout << "Pozitif";
    else if (a<0) cout << "Negatif";
    else cout << "Sifir";
    system("pause");return 0; }
```

C#

```
// Karar 1 -> C kodunun karşılığı
using System;
namespace ornek
{ class Program
    { static void Main(string[] args)
        {  Console.Clear(); Console.Write("Bir tamsayı giriniz => ");
           int a=Convert.ToInt32(Console.ReadLine());
           if (a>0) Console.Write("Pozitif");
           else if (a<0) Console.Write("Negatif");
           else Console.Write("Sıfır");
           Console.ReadLine();
}}}
```

JAVA

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

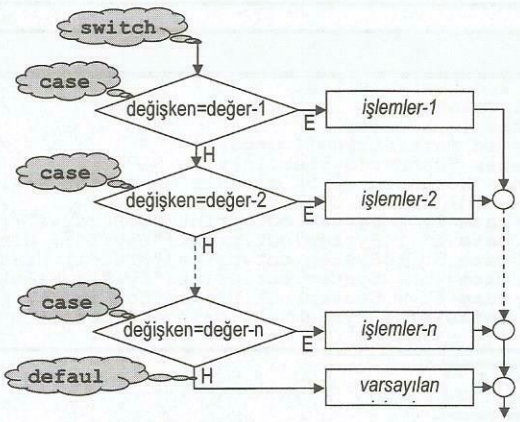
```
Bir tamsayı giriniz => -5
Negatif
```

2. switch

Seçimli form yapısıdır. "switch" ten sonra belirtilen 'değişken', "case"ten sonraki hangi değeri alırsa, karşılığındaki işlem yapılır. Eğer "değişken"e karşılık "case"te değer yoksa, "default"taki "varsayılan işlemler" gerçekleştirilir.

Kullanım şekilleri (C/C++/C#/Java):

```
switch ( değişken )
        {
            case değer-1 : işlemler-1 ;
            case değer-2 : işlemler-2 ;
            case değer-3 : işlemler-3 ;
            ..........................
            case değer-n : işlemler-n ;
        }

switch ( değişken )
        {
            case değer-1 : işlemler-1 ;
            case değer-2 : işlemler-2 ;
            case değer-3 : işlemler-3 ;
            ......................
            case değer-n : işlemler-n ;
            default: varsayılan işlemler ;
        }
```



C

```
/* Karar 2 */
#include <stdio.h>
#include <conio.h>
void main()
{   int a;clrscr();printf("Notunuzu giriniz(1-5) => ");scanf("%d",&a);
    switch (a) {
       case 1: { printf("Çok zayıf"); break; }
       case 2: { printf("Zayıf"); break; }
       case 3: { printf("Orta"); break; }
       case 4: { printf("İyi"); break; }
       case 5: { printf("Çok iyi"); break; }
       default: { printf("Geçersiz not"); break; }
} getch(); }
```

C++

```
// Karar 2 -> C kodunun karşılığı
#include <iostream>
using namespace std;
main()
{       int a;
    system("cls"); cout << "Notunuzu giriniz (1-5) => "; cin >> a;
    switch (a) {
       case 1: { cout << "Çok zayıf"; break; }
       case 2: { cout << "Zayıf"; break; }
       case 3: { cout << "Orta"; break; }
       case 4: { cout << "İyi"; break; }
       case 5: { cout << "Çok iyi"; break; }
       default:{ cout << "Geçersiz not"; break; }
    } system("pause");return 0; }
```

C#

```
// Karar 2 -> C kodunun karşılığı
using System;
namespace ornek
{ class Program
    { static void Main(string[] args)
       { Console.Clear(); Console.Write("Notunuzu (1-5) giriniz => ");
         int a=Convert.ToInt32(Console.ReadLine());
         switch (a)
           {   case 1: { Console.Write("Çok zayıf"); break; }
               case 2: { Console.Write("Zayıf"); break; }
               case 3: { Console.Write("Orta"); break; }
               case 4: { Console.Write("İyi"); break; }
               case 5: { Console.Write("Çok iyi"); break; }
               default: { Console.Write("Geçersiz not"); break; }
           } Console.ReadLine();  }    }   }
```

JAVA

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

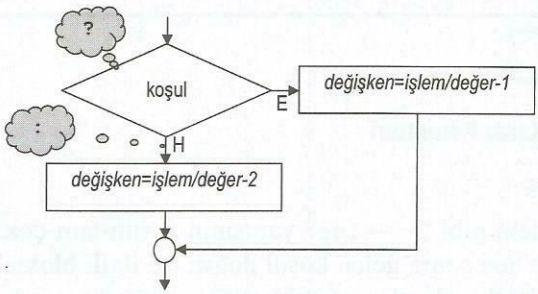
```
Notunuzu giriniz (1-5) => 4
Iyi
```

3. ?

Koşulun durumuna göre ilgili değeri veya işlem sonucunu, belirtilen değişkene aktarır.

Kullanım şekli (C/C++/C#/Java):

```
değişken = ( koşul ) ?  işlem/değer-1 : işlem/değer-2 ;
```



Yukarıdaki karar yapısında, "koşul" doğru ise "değişken"e "işlem/değer-1" ; yanlış ise "işlem/değer-2" atanır.

C

```
/* Karar 3 */
#include <stdio.h>
#include <conio.h>
void main()
{   int a,b;
    clrscr();printf("Bir tamsayı giriniz: ");scanf("%d",&a);
    b=(a<0) ? -1*a : a; printf("Mutlak değeri: %u\n",b);  getch(); }
```

C++

```
// Karar 3 -> C kodunun karşılığı
#include <iostream>
using namespace std;
main()
{   int a,b;
    system("cls"); cout << "Bir tamsayı giriniz: "; cin >> a;
    b=(a<0) ? -1*a : a;
    cout << "Mutlak değeri: " << b << endl; system("pause");return 0; }
```

C#

```
// Karar 3 -> C kodunun karşılığı
using System;
namespace ornek
{ class Program
   { static void Main(string[] args)
      {Console.Clear(); Console.Write("Bir tamsayı giriniz: ");
       int a=Convert.ToInt32(Console.ReadLine());
       int b=(a < 0) ? -1 * a : a;
       Console.Write("Mutlak değeri: {0}\n", b);Console.ReadLine();}}}
```

JAVA

```
// Karar 3 -> C kodunun karşılığı
import java.util.Scanner;
public class ornek {
 public static void main(String[] args) {
  Scanner klavye=new Scanner(System.in);
  System.out.print("Bir tamsayı giriniz: "); int a=klavye.nextInt();
  int b=(a<0) ? -1 * a : a;
  System.out.printf("Mutlak değeri: %d\n", b);   }}
```

```
Bir tamsayı giriniz: -25
Mutlak değeri: 25
```

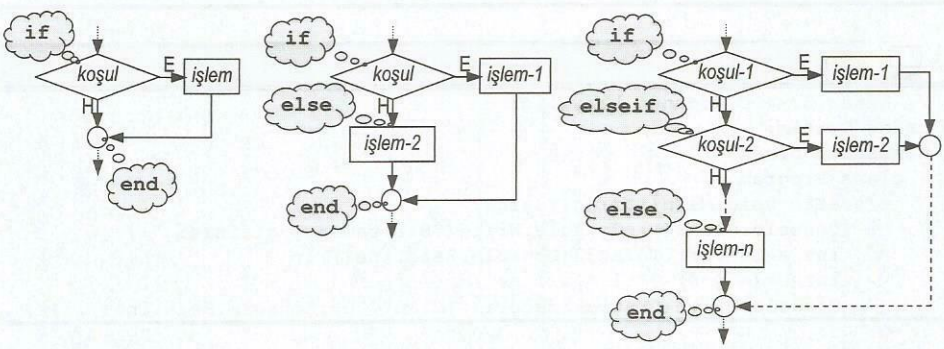
B. MATLAB'daki Karar Komutları

1. if-elseif-else

Diğer dillerdeki gibi "`if-else`" yapısının yarım-tam-çok koşullu biçimleri kullanılabilir. "`if`"ten sonra gelen koşul doğru ise ilgili bloktaki işlemler, yanlış ise "`else`" bloğundaki işlemler gerçekleştirilir ve karar yapısı "`end`" ile sonlandırılır. Eğer birden çok ve ortak olmayan koşullar varsa "`if-elseif-else`" yapısı kullanılır.

Kullanım şekilleri:

```
if ( koşul veya koşul grubu )
   işlem/işlemler;
end;

if ( koşul veya koşul grubu )
   işlem/işlemler-1;
else
   işlem/işlemler-2;
end;

if ( koşul veya koşul grubu - 1 )
   işlem/işlemler-1;
elseif ( koşul veya koşul grubu - 2 )
   işlem/işlemler-2;
elseif ( koşul veya koşul grubu - 3 )
   işlem/işlemler-3;
   ..................
else
   işlem/işlemler-n;
end;
```



MATLAB

```
% Karar
clear all; clc; a=input('Bir sayı giriniz: ','s'); v=findstr(a,'.');
if isempty(v)
    disp('Girilen sayı TAMSAYI');
else
    disp('Girilen sayı ONDALIKLI SAYI')
end;
```

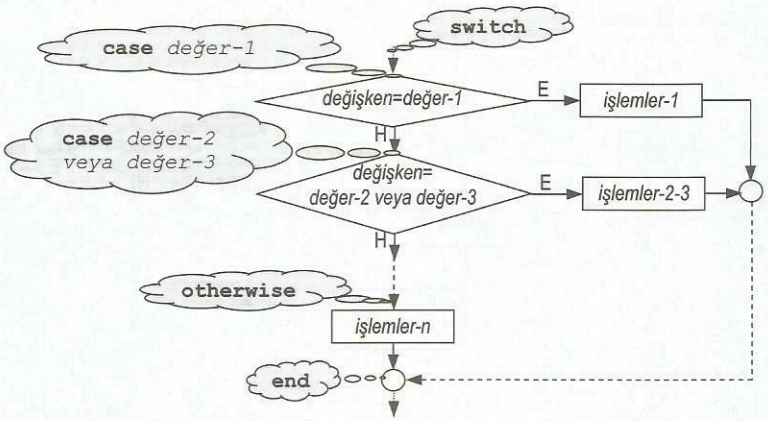
```
Bir sayı giriniz: 2015
Girilen sayı TAMSAYI
```

2. switch

Seçimli form karar yapısı için MATLAB'da "`switch-case-otherwise`" kullanılmaktadır.

Kullanım şekli:

```
switch değişken
   case değer-1
      işlemler-1 ;
   case { değer-2 , değer-3 }
      işlemler-2-3 ;
   ..........................
   otherwise
      işlemler-n ;
end;
```



MATLAB

```
% Karar
clear all; clc;
not=upper(input('Harf notunuzu giriniz: ','s'));
switch not
    case {'AA','BA','BB','CB','CC'}
        disp('Geçtiniz...');
    case {'DC','DD'}
        disp('Şartlı geçtiniz...');
    case {'FD','FF'}
        disp('Kaldınız...')
    otherwise
        disp('Geçersiz harf notu...');
end
```

```
Harf notunuzu giriniz: AAA
Geçersiz harf notu...
```

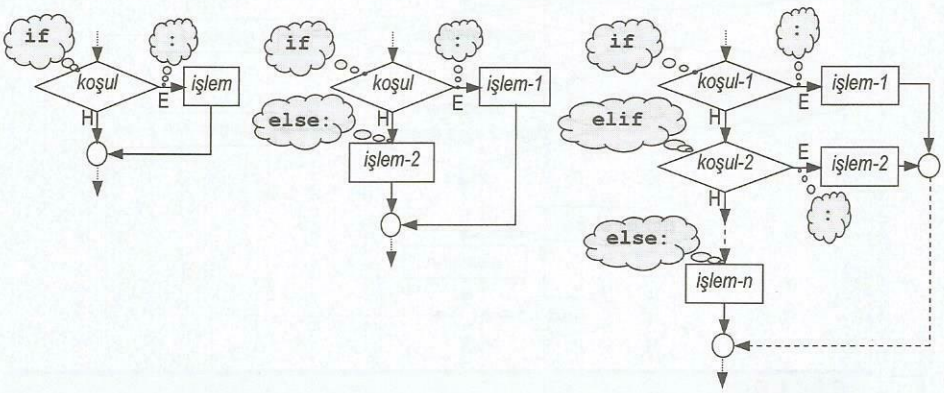
C. Python'daki Karar Komutları

1. if-elif-else

Python'daki yarım-tam-çok koşullu yapı "`if-elif-else`" dir.

Kullanım şekilleri:

```
if ( koşul veya koşul grubu ):
   işlem/işlemler
if ( koşul veya koşul grubu ):
   işlem/işlemler-1
else:
   işlem/işlemler-2;
doğru ise sonuç if ( koşul veya koşul grubu ) else yanlış ise sonuç
if ( koşul veya koşul grubu - 1 ):
   işlem/işlemler-1
elif ( koşul veya koşul grubu - 2 ):
   işlem/işlemler-2
elif ( koşul veya koşul grubu - 3 ):
   işlem/işlemler-3;
   .....
else:
   işlem/işlemler-n
```



PYTHON

```
# Karar yapısı-1
a=eval(input("Bir sayı giriniz: "))
if (a>0):
    print("Pozitif")
elif (a<0):
    print("Negatif")
else:
    print("Sıfır")
```

```
Bir sayı giriniz: -5
Negatif
```

PYTHON

```
# Karar yapısı-2
a=eval(input("Bir sayı giriniz: "))
x=-1*a if (a<0) else a
print(x)
```

```
Bir sayı giriniz:
100
100
```

Tablo 5.38'de C, C++, C#, Java, MATLAB ve Python'daki karar yapıları, karşılaştırmalı olarak verilmektedir.

Tablo 5.38: C, C++, C#, Java, MATLAB ve Python'daki eşdeğer görevdeki karar yapıları

Form	C	C++	C#	Java	MATLAB	Python
Yarım	if	if	if	if	if	if
Tam	if-else	if-else	if-else	if-else	if-else	if-else
Çok koşullu	if-else if-else	if-else if-else	if-else if-else	if-else if-else	if-elseif-else	if-elif-else
Seçimli	switch	switch	switch	switch	switch	
Operatör	?	?	?	?		

VI. DİĞER BAZI KOMUTLAR

Bu alt başlıklarda, daha sonraki bölümlerdeki örneklerde kullanılacak olan bazı fonksiyonlar/komutlar tanıtılacaktır.

A. Ekran Komutları

Ekranın belirtilen yerine imleci yönlendirme, ekranı temizleme, renk ayarları gibi işlemlerin yapılmasını sağlayan komutlardır (Tablo 5.39-5.40).

Tablo 5.39: Renk kodları

Kod	Renk	Kod	Renk
0	Siyah	*8*	Koyu gri
1	Mavi	*9*	Açık mavi
2	Yeşil	*10*	Açık yeşil
3	Turkuaz	*11*	Açık turkuaz
4	Kırmızı	*12*	Açık kırmızı
5	Pembe	*13*	Açık pembe
6	Kahverengi	*14*	Sarı
7	Gri	*15*	Beyaz
Bu kodlara 16 tamsayısının eklenmesiyle yazı renginin yanıp sönmesi sağlanır.			

Tablo 5.40: Ekranla ilgili bazı fonksiyonlar

Alg.	Görevi	C	C++	C#
Sil	Ekranı temizler	**clrscr()**	**system("cls")**	**Console.Clear()**
Konum(*sat,sut*)	İmleci, ekranın belirtilen konumuna getirir	**gotoxy(*sut,sat*)**	**COORD xy = {*sut,sat* }; SetConsoleCursorPosition (GetStdHandle (STD_OUTPUT_HANDLE),xy);**	**Console.SetCursorPosition(*sut,sat*)**
Renk(*yz,zmn*)	Yazı ve zemin rengini ayarlar	**textcolor(*yz*); textbackground (*zmn*);**	**system("color *zmn yz* ")** veya **SetConsoleTextAttribute (GetStdHandle (STD_OUTPUT_HANDLE), *yz*)**	**Console.ForegroundColor=*yz* Console.BackgroundColor=*zmn***

sat: Satır konumu , sut: Sütun konumu , yz: Yazı rengi , zmn: Zemin rengi

B. Matematiksel İşlem Komutları

Matematiksel işlemlerin yapılmasını sağlayan komutlardır (Tablo 5.41-53).

Tablo 5.41: Matematiksel sabitler

Sabit	C	C++	C#	Java	MATLAB	Python
π	M_PI	M_PI	PI	PI	pi	pi
e	M_E	M_E	E	E		e
τ						tau
∞					Inf	inf
Sayı değil					NaN	nan

Tablo 5.42: Karmaşık (kompleks) sayılarla ilgili fonksiyonlar

İşlem	C	C++	C#	Java	MATLAB	Python
Oluşturma / belirtme					i *veya* j	j *veya* complex
Mutlak değer					abs	abs
Açı					angle	phase
Gerçel kısım					real	real
Sanal kısım					imag	imag
Eşlenik					conj	conjugate

Tablo 5.43: Açı dönüşüm fonksiyonları

İşlem	C	C++	C#	Java	MATLAB	Python
radyan → derece				toDegrees	radtodeg	degrees
derece → radyan				toRadians	degtorad	radians

Tablo 5.44: Üstel/logaritmik fonksiyonlar

İşlem	C	C++	C#	Java	MATLAB	Python
e^x	exp(x)	exp(x)	Exp(x)	exp(x)	exp(x)	exp(x)
Ln(x)	log(x)	log(x)	Log(x)	log(x)	log(x)	log(x)
Log(x)	log10(x)	log10(x)	Log10(x)	log10(x)	log10(x)	log10(x)

Tablo 5.45: Trigonometrik fonksiyonlar

İşlem	C	C++	C#	Java	MATLAB	Python
Sin(x)	sin(x)	sin(x)	Sin(x)	sin(x)	sin(x)	sin(x)
Cos(x)	cos(x)	cos(x)	Cos(x)	cos(x)	cos(x)	cos(x)
Tan(x)	tan(x)	tan(x)	Tan(x)	tan(x)	tan(x)	tan(x)

Tablo 5.46: Ters trigonometrik fonksiyonlar

İşlem	C	C++	C#	Java	MATLAB	Python
ArcSin(x)	asin(x)	asin(x)	Asin(x)	asin(x)	asin(x)	asin(x)
ArcCos(x)	acos(x)	acos(x)	Acos(x)	acos(x)	acos(x)	acos(x)
Arctan(x)	atan(x)	atan(x)	Atan(x)	atan(x)	atan(x)	atan(x)

Tablo 5.47: Hiperbolik fonksiyonlar

İşlem	C	C++	C#	Java	MATLAB	Python
Sinh(x)	sinh(x)	sinh(x)	Sinh(x)	sinh(x)	sinh(x)	sinh(x)
Cosh(x)	cosh(x)	cosh(x)	Cosh(x)	cosh(x)	cosh(x)	cosh(x)
Tanh(x)	tanh(x)	tanh(x)	Tanh(x)	tanh(x)	tanh(x)	tanh(x)

Tablo 5.48: Ters hiperbolik fonksiyonlar

İşlem	C	C++	C#	Java	MATLAB	Python
ArcSinh(x)	asinh(x)	asinh(x)			asinh(x)	asinh(x)
ArcCosh(x)	acosh(x)	acosh(x)			acosh(x)	acosh(x)
ArcTanh(x)	atanh(x)	atanh(x)			atanh(x)	atanh(x)

Tablo 5.49: Üs alma komutları

İşlem	C	C++	C#	Java	MATLAB	Python
x^y	pow(x,y)	pow(x,y)	Pow(x,y)	pow(x,y)	^	**, pow
$\sqrt{x}$	sqrt(x)	sqrt(x)	Sqrt(x)	sqrt(x)	sqrt(x)	sqrt(x)

Tablo 5.50: Rastgele sayı üretme komutları

İşlem	C	C++	C#	Java	MATLAB	Python
Rastgele başlat	randomize() srand()	randomize() srand()				
Rastgele(x)	random(x+1) rand()%(x+1)	random(x+1) rand()%(x+1)	Next(x+1)	random()*(x+1)	x*rand	randint(0,x) randrange(0,x+1)
x: Rastgele üretilecek sayının maksimum değeri (Sayılar 0-x arasında)						

Tablo 5.51: Taban dönüşüm fonksiyonları

İşlem	C	C++	C#	Java	MATLAB	Python
10 → 2	itoa(...,...,2)	itoa(...,...,2)	toString(...,2)	toString(...,2)	dec2bin	bin
2 → 10	strtol(...,...,2)	strtol(...,...,2)	toInt32(...,2)	parseInt(...,2)	bin2dec	int(...,2)
10 → 8	itoa(...,...,8)	itoa(...,...,8)	toString(...,8)	toString(...,8)		oct
8 → 10	strtol(...,...,8)	strtol(...,...,8)	toInt32(...,8)	parseInt(...,8)	oct2dec	int(...,8)
10 → 16	itoa(...,...,16)	itoa(...,...,16)	toString(...,16)	toString(...,16)	dec2hex	hex
16 → 10	strtol(...,...,16)	strtol(...,...,16)	toInt32(...,16)	parseInt(...,16)	hex2dec	int(...,16)
10 → n	itoa(...,...,n)	itoa(...,...,n)			dec2base	
n → 10	strtol(...,...,n)	strtol(...,...,n)			base2dec	int(...,n)

Tablo 5.52: Sayı biçimlendirme fonksiyonları

İşlem		C	C++	C#	Java	MATLAB	Python
Yuvarla	*+∞*	ceil	ceil	Ceiling	ceil	ceil	ceil
	-∞	floor	floor	Floor	floor	floor	floor
	0	trunc	trunc	Truncate		fix	trunc
	Tam	round	round	Round	round	round	round

Tablo 5.53: Diğer bazı matematiksel komutlar

İşlem	C	C++	C#	Java	MATLAB	Python
Mutlak	abs	abs	Abs	abs	abs	abs
Mod	%	%	%	%	mod	%
Tam	div	div		divide...		//
Faktöriyel					factorial	factorial
Toplam					sum	sum
Çarpım					prod	
En büyük		max	Max	max	max	max
En küçük		min	Min	min	min	min
Sırala	qsort	sort	Sort	sort	sort	sort

C. Alfasayısal İşlem Komutları

Karakter ve karakter grupları (kelime, cümle, vb.) üzerinde işlem yapan komutlardır (Tablo 5.54).

Tablo 5.54: Bazı alfasayısal işlem komutları

İşlem		C	C++	C#	Java	MATLAB	Python
Uzunluk		strlen	length	Length	length	length	len
Kopyala			copy	Substring	substring		
Sil				Remove		erase	
ASCIIkod						double	ord
ASCIIkar		toascii	toascii			char	chr
Büyüt		toupper	toupper	ToUpper	toUpperCase	upper	upper
Küçült		tolower	tolower	ToLower	toLowerCase	lower	lower
Ters		strrev	strrev	Reverse	reverse		
Bul		strstr	strstr find	IndexOf Contains	indexOf contains	strfind	find
Değiştir			replace	Replace	replace	strrep	replace
Sayısal	*Tam*	atoi	atoi	ToInt32 Int32.Parse	parseInt	str2num str2double	int
	Ond.	atof	atof	ToDouble Double.Parse	parseFloat	str2num str2double	float
Alfasayısal	*Tam*	itoa	itoa	ToString	toString	num2str int2str	str
	Ond.	fcvt, ecvt, ...	fcvt, ecvt, ...	ToChar, ToString ...	toString	num2str	str

D. Diğer Komutlar

Bu alt başlıkta; daha sonraki bölümlerde yazılan programlarda kullanılan bazı komutlar verilmektedir(Tablo 5.55-59).

Tablo 5.55: Fonksiyon tanımlama

C	C++	C#	Java	MATLAB	Python
define	define			function inline	def

Tablo 5.56: Dizi tanımlama/belirtme ve başlangıç indisi

C	C++	C#	Java	MATLAB	Python
[]	[]	[]	[]	()	[]
0	*0*	*0*	*0*	*1*	*0*

Tablo 5.57: Geciktirme(bekletme) komutları

Algotirma	Görevi	C	C++	C#	Java	MATLAB	Python
Bekle	İşlem akışını, belirtilen süre kadar geciktirir.	delay	sleep	Thread. Sleep	Sleep	pause	sleep

Tablo 5.58: İşlem akışını yönlendirme komutları

Algoritma	Görevi	C/C++/C#	MATLAB	Python
Git	İşlem akışını, belirtilen yere yönlendirir.	goto		

Tablo 5.59: Port komutları

Algotirma	Görevi	C	C++	C#	Java	MATLAB	Python
port	Porta veri gönder veya al	inp inport outp outport	inp outp	Read Write	read write	fscanf fprintf	read write

Not

Kitapta asıl amaç algoritma öğretmek olduğundan, programların yazımında mümkün olduğunca hazır fonksiyonlar/komutlar kullanılmayacaktır. Ancak ilgili fonksiyon/komutun işleyişi önceki programlarda/örneklerde açıklanmışsa veya çözümde kullanılması büyük kolaylıklar sağlıyorsa faydalanılacaktır.

SORULAR

1. C, C++, C#, Java, MATLAB ve Python programlama dilindeki klavyeden veri giriş komutlarını karşılaştırınız.

2. C, C++, C#, Java, MATLAB ve Python programlama dilindeki ekrana yazdırma komutlarını karşılaştırınız.

3. C, C++, C#, Java, MATLAB ve Python programlama dilindeki döngü komutlarını karşılaştırınız.

4. C, C++, C#, Java, MATLAB ve Python programlama dilindeki karar komutlarını karşılaştırınız.

5. C/C++/C# programlama dilindeki "for" döngü yapısını karar ifadeleriyle gerçekleyiniz.

6. C/C++/C# programlama dilindeki "for" döngü yapısını "while" ile gerçekleyiniz.

7. C/C++/C# programlama dilindeki "for" döngü yapısını "do-while" ile gerçekleyiniz.

8. Java programlama dilindeki "for" döngü yapısını karar ifadeleriyle gerçekleyiniz.

9. Java programlama dilindeki "for" döngü yapısını "while" ile gerçekleyiniz.

10. Java programlama dilindeki "for" döngü yapısını "do-while" ile gerçekleyiniz.

11. MATLAB programlama dilindeki "for" döngü yapısını karar ifadeleriyle gerçekleyiniz.

12. MATLAB programlama dilindeki "for" döngü yapısını "while" ile gerçekleyiniz.

13. Python programlama dilindeki "for" döngü yapısını karar ifadeleriyle gerçekleyiniz.

14. Python programlama dilindeki "for" döngü yapısını "while" ile gerçekleyiniz.

BÖLÜM 6

GENEL UYGULAMALAR

Bu bölümde

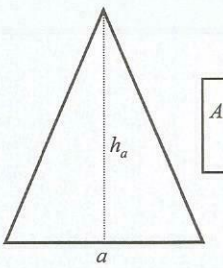
- Temel algoritma geliştirme tekniklerini,
- Temel kodlama tekniklerini,
- Genel/temel problemlerin algoritmalarını, akış diyagramlarını, 6 dildeki kodlamalarını ve alternatif çözümlerini öğreneceksiniz.

Bölüm sonunda

- Genel/temel problemlerin algoritmalarını hazırlayabilecek,
- Genel/temel problemlerin akış diyagramlarını çizebilecek,
- Genel/temel problemleri kodlayabilecek,
- Genel/temel problemler için alternatif çözümler üretebileceksiniz.

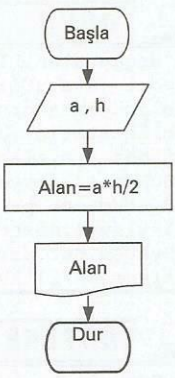
Teknik derslerde yeni öğrenilen bir konuyu anlamanın/kavramanın en iyi yolu; bu konu ile ilgili çok sayıda alıştırma/uygulama yapmaktır. Yapılan her yeni alıştırma/uygulamayla konunun belirli noktaları/özellikleri/ayrıntıları daha açık ve net olarak ortaya çıkarak anlaşılmaktadır. Bu durum, programcılık için de geçerlidir. Karşılaşılan her problem, bu problemin çözümü için geliştirilen algoritma ve bu algoritmanın kodlanması; programcıya birçok teknikler kazandırmaktadır. Bu noktadan hareketle; farklı alanlara ait çok sayıda problem belirlenerek (örnek oluşturularak), çözümü için gerekli algoritma ve akış diyagramıyla birlikte bilgisayar ortamındaki kodları C, C++, C# (konsol uygulaması), Java, MATLAB ve Python programlama dilleriyle yazılacaktır. Ancak benzerlik arz eden problemlerin sadece algoritma/akış diyagramları hazırlanarak kod kısmı okuyucuya bırakılacaktır. Ayrıca birçok problemin birden fazla çözüm yolları olduğundan alternatif algoritma geliştirme/programlama teknikleri de sunulacaktır. Tüm mesleklerdeki bilgisayar öğrenen kişiler tarafından anlaşılmasının hedeflenmesi nedeniyle, bu bölümdeki uygulamalar/problemler birçok farklı alana ait konulardan seçilecektir.

Örnek-6.1: Klavyeden bir kenar uzunluğu ve o kenara ait yüksekliği girilen üçgenin alanını hesaplayan program.



$$Alan = \frac{a.h_a}{2}$$

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



C

```
/* Üçgen alanı 1 */
#include <stdio.h>
#include <conio.h>
float a,h,alan;
main()
{   clrscr();printf("Kenar uzunluğunu giriniz (cm): ");scanf("%f",&a);
    printf("Yüksekliği giriniz (cm): "); scanf("%f",&h);
    alan=a*h/2; printf("\nÜçgenin alanı (cm2): %0.2f",alan);
    getch(); return 0; }
```

```
Kenar uzunluğunu giriniz (cm): 5
Yüksekliği giriniz (cm): 10

Üçgenin alanı (cm2): 25.00
```

C++

```
// Üçgen alanı 1
#include <iostream>
using namespace std;
float a,h,alan;
int main()
{   system("cls"); cout << "Kenar uzunluğunu giriniz (cm): "; cin >> a;
    cout << "Yüksekliği giriniz (cm): "; cin >> h;
    Alan=a*h/2; cout << "\nÜçgenin alanı (cm2): " << Alan << endl;
    system("pause"); return 0; }
```

C#

```
// Üçgen alanı 1
using System;
namespace ornek61
{ class Program
  { static void Main(string[] args)
    { double a, h, alan;
      Console.Clear();Console.Write("Kenar uzunluğunu giriniz (cm): ");
      a=Convert.ToDouble(Console.ReadLine());
      Console.Write("Yüksekliği giriniz (cm): ");
      h=Convert.ToDouble(Console.ReadLine()); alan=a*h/2;
      Console.WriteLine("\nÜçgenin alanı (cm2): {0:F2}",alan);
      Console.ReadLine();  }}}
```

JAVA

```
// Üçgen alanı 1
import java.util.Scanner;
public class ornek61 {
public static void main(String[] args)
{ float a,h,alan;
  Scanner klavye=new Scanner(System.in);
  System.out.print("Kenar uzunluğunu giriniz (cm): ");
  a=klavye.nextFloat();
  System.out.print("Yüksekliği giriniz (cm): ");h=klavye.nextFloat();
  alan=a*h/2; System.out.printf("\nÜçgenin alanı (cm2): %.2f",alan); }}
```

MATLAB

```
% Üçgen alanı 1
clear all; clc
a=input('Kenar uzunluğunu giriniz (cm): ');
h=input('Yüksekliği giriniz (cm): '); alan=a*h/2;
fprintf('\nÜçgenin alanı (cm2): %0.2f\n',alan);
```

PYTHON

```
# Üçgen alanı 1
a=float(input("Kenar uzunluğunu giriniz (cm): "))
h=float(input("Yüksekliği giriniz (cm): "))
alan=a*h/2
print("\nÜçgenin alanı (cm2): %0.2f\n"% (alan))
```

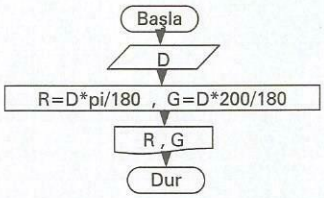
Not

Programlarınızda kullanıcıdan verileri, açıklama ifadeleriyle isteyiniz. Ayrıca açıklama ifadelerinde - varsa - birimleri de unutmayınız.

Örnek-6.2: Klavyeden '*Derece*' cinsinden girilen açıyı; '*Radyan*' ve '*Grad*' cinsine çeviren program.

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

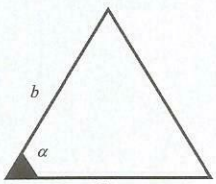
1. Başla
2. Açıyı (D) gir
3. R=D*pi/180
4. G=D*200/180
5. Yaz R,G
6. Dur



Not

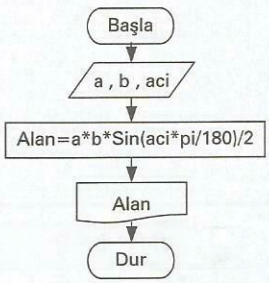
Programlama dillerindeki trigonometrik fonksiyonlar (komutlar), genel olarak "radyan" cinsinden çalışmaktadırlar.

Örnek-6.3: Klavyeden iki kenarı ve derece cinsinden aradaki açısı girilen üçgenin alanını hesaplayan program.



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

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



C

```
/* Üçgen alanı 2 */
#include <stdio.h>
#include <conio.h>
#include <math.h>
float a,b,aci,alan;
void main()
{  clrscr();printf("Birinci kenarı giriniz (cm): ");scanf("%f",&a);
   printf("İkinci kenarı giriniz (cm): ");scanf("%f",&b);
   printf("Aradaki açıyı giriniz (derece): ");scanf("%f",&aci);
   alan=a*b*sin(M_PI*aci/180)/2;
   printf("\nÜçgenin alanı (cm2): %0.2f",alan); getch(); }
```

C++

```
// Üçgen alanı 2
#include <iostream>
#include <cmath> // #include <math.h>
using namespace std;
float a,b,aci,alan;
int main()
{   system("cls"); cout << "Birinci kenarı giriniz (cm): "; cin >> a;
    cout << "İkinci kenarıı giriniz (cm): "; cin >> b;
    cout << "Aradaki açıyı giriniz (derece): "; cin >> aci;
    alan=a*b*sin(M_PI*aci/180)/ 2;
    cout << "\nÜçgenin alanı (cm2): " << alan << endl;
    system("pause"); return 0; }
```

```
Birinci kenarı giriniz (cm): 10
İkinci kenarı giriniz (cm): 20
Aradaki açıyı giriniz (derece): 50

Üçgenin alanı (cm2): 76.6044
```

C#

```
// Üçgen alanı 2
using System;
namespace ornek63
{ class Program
   { static void Main(string[] args)
      { double a, b, aci, alan;
        Console.Clear();Console.Write("Birinci kenarı giriniz (cm): ");
        a=Convert.ToDouble(Console.ReadLine());
        Console.Write("İkinci kenarı giriniz (cm): ");
        b=Convert.ToDouble(Console.ReadLine());
        Console.Write("Aradaki açıyı giriniz (derece): ");
        aci=Convert.ToDouble(Console.ReadLine());
        alan=a*b*Math.Sin(Math.PI*aci/180)/2;
        Console.WriteLine("\nÜçgenin alanı (cm2): {0:F2}", alan);
        Console.ReadLine(); } } }
```

JAVA

```
// Üçgen alanı 2
import java.util.Scanner;
public class ornek63 {
public static void main(String[] args)
{ double a,b,aci,alan;
  Scanner klavye=new Scanner(System.in);
  System.out.print("Birinci kenarı giriniz (cm): ");
  a=klavye.nextDouble();
  System.out.print("İkinci kenarı giriniz (cm): ");
  b=klavye.nextDouble();
  System.out.print("Aradaki açıyı giriniz (derece): ");
  aci=klavye.nextDouble();
  alan=a*b*Math.sin(Math.toRadians(aci))/2;
  System.out.printf("\nÜçgenin alanı (cm2): %.2f",alan); }}
```

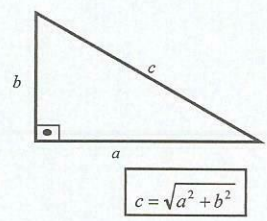
MATLAB

```
% Üçgen alanı 2
clear all; clc
a=input('Birinci kenarı giriniz (cm): ');
b=input('İkinci kenarı giriniz (cm): ');
aci=input('Aradaki açıyı giriniz (derece): ');
alan=a*b*sind(aci)/2;
fprintf('\nÜçgenin alanı (cm2): %0.2f\n',alan);
```

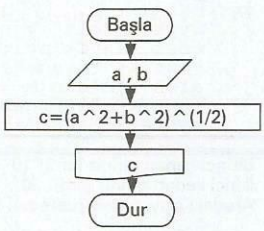
PYTHON

```
# Üçgen alanı 2
import math
a=eval(input("Birinci giriniz (cm): "))
b=eval(input("İkinci kenarı giriniz (cm): "))
aci=eval(input("Aradaki açıyı giriniz (derece): "))
alan=a*b*math.sin(math.radians(aci))/2
print("\nÜçgenin alanı (cm2): {:.2f}\n".format(alan))
```

Örnek-6.4: Klavyeden iki dik kenarı girilen dik üçgenin hipotenüsünü hesaplayan program (*Pisagor Teoremi*).



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



C

```
/* Hipotenüs */
#include <stdio.h>
#include <conio.h>
#include <math.h>
float a,b,c;
main()
{ clrscr();printf("Birinci dik kenarı giriniz (cm): ");scanf("%f",&a);
  printf("İkinci dik kenarı giriniz (cm): ");scanf("%f",&b);
  c=sqrt(a*a+b*b); printf("\nHipotenüs (cm): %0.2f",c);
  getch(); return 0; }
```

C++

```
// Hipotenüs
#include <iostream>
#include <cmath>
using namespace std;
float a,b,c;
int main()
{   system("cls");cout<<"Birinci dik kenarini giriniz (cm): "; cin>>a;
    cout << "Ikinci dik kenarini giriniz (cm): "; cin >> b;
    c=sqrt(a*a+b*b); cout << "\nHipotenus (cm): " << c << endl;
    system("pause"); return 0;  }
```

C#

```
// Hipotenüs
using System;
namespace ornek64
{ class Program
  { static void Main(string[] args)
    { double a, b, c;
    Console.Clear();Console.Write("Birinci dik kenarı giriniz (cm): ");
        a=Convert.ToDouble(Console.ReadLine());
        Console.Write("İkinci dik kenarı giriniz (cm): ");
        b=Convert.ToDouble(Console.ReadLine()); c=Math.Sqrt(a*a+b*b);
        Console.WriteLine("\nHipotenüs (cm): {0:F2}",c);
        Console.ReadLine();    }    }    }
```

```
Birinci dik kenarı giriniz (cm): 3
İkinci dik kenarı giriniz (cm): 4

Hipotenüs (cm): 5.00
```

JAVA

```
// Hipotenüs
import java.util.Scanner;
public class ornek64 {
public static void main(String[] args)
{ double a,b,c;
  Scanner klavye=new Scanner(System.in);
  System.out.print("Birinci dik kenarı giriniz (cm): ");
  a=klavye.nextDouble();
  System.out.print("İkinci dik kenarı giriniz (cm): ");
  b=klavye.nextDouble();
  c=Math.sqrt(Math.pow(a,2)+Math.pow(b,2));
  System.out.printf("\nHipotenüs (cm): %.2f",c); }}
```

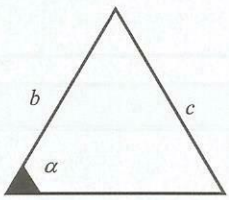
MATLAB

```
% Hipotenüs
clear all; clc ; a=input('Birinci dik kenarı giriniz (cm): ');
b=input('İkinci dik kenarı giriniz (cm): ');
c=(a^2+b^2)^0.5; fprintf('\nHipotenüs (cm): %0.2f\n',c);
```

PYTHON

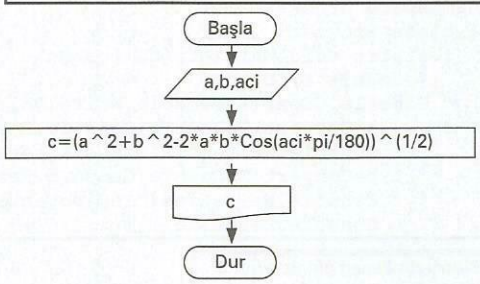
```
# Hipotenüs
a=eval(input("Birinci dik kenarı giriniz (cm): "))
b=eval(input("İkinci dik kenarı giriniz (cm): "))
c=(a*a+b*b)**0.5
print("\nHipotenüs (cm): {:.2f}\n".format(c))
```

Örnek-6.5: Klavyeden iki kenarı ve bunların arasındaki derece cinsinden açısı girilen üçgenin üçüncü kenarını hesaplayan program (*Cosinus bağıntısı*).

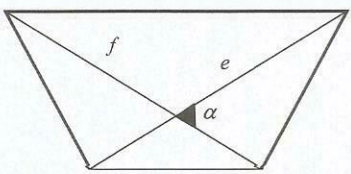


$$c^2 = a^2 + b^2 - 2abCos\alpha$$

1. Başla
2. Birinci kenarı (a) gir
3. İkinci kenarı (b) gir
4. Aradaki açıyı (aci) gir
5. c=(a^2+b^2-2*a*b*Cos(aci*pi/180))^(1/2)
6. Yaz c
7. Dur

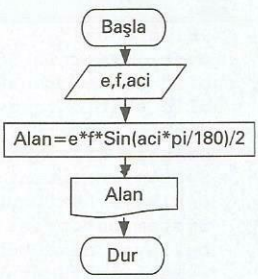


Örnek-6.6: Klavyeden köşegenleri ve arasındaki derece cinsinden açısı girilen dörtgenin alanını hesaplayan program.

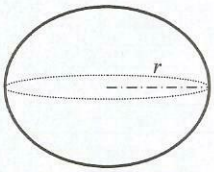


$$Alan = e.f.Sin(\alpha)/2$$

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

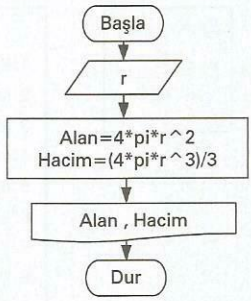


Örnek-6.7: Klavyeden yarıçapı girilen kürenin alanını ve hacmini hesaplayan program.

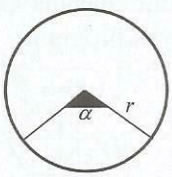


$$Alan = 4.\pi.r^2$$
$$Hacim = 4.\pi.r^3 / 3$$

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

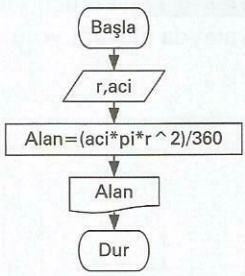


Örnek-6.8: Klavyeden yarıçapı ve derece cinsinden açısı girilen daire diliminin alanını hesaplayan program.

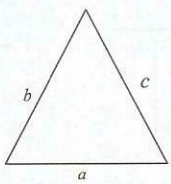


$$Alan = \alpha.\pi.r^2 / 360$$

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

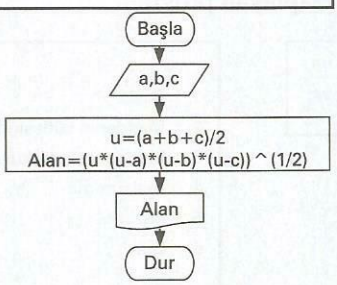


Örnek-6.9: Klavyeden tüm kenar uzunlukları girilen üçgenin alanını hesaplayan program (*Heron formülü*).



$$u = \frac{a+b+c}{2}$$
$$Alan = \sqrt{u(u-a)(u-b)(u-c)}$$

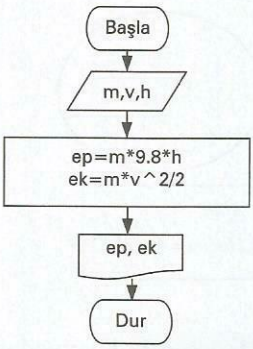
1. Başla
2. Birinci kenarı (a) gir
3. İkinci kenarı (b) gir
4. Üçüncü kenarı (c) gir
5. u=(a+b+c)/2
6. Alan=(u*(u-a)*(u-b)*(u-c)) ^ (1/2)
7. Yaz Alan
8. Dur



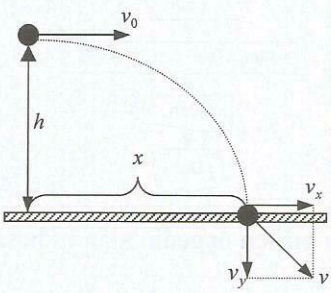
Örnek-6.10: Klavyeden kütlesi, hızı ve yerden yüksekliği girilen maddenin potansiyel ve kinetik enerjisini hesaplayan program.

$$Ep = m.g.h$$
$$Ek = \frac{1}{2}.m.v^2$$

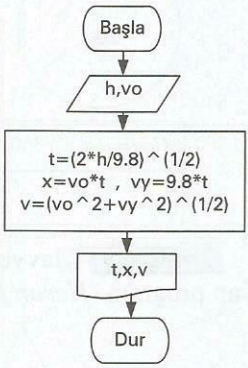
1. Başla
2. Maddenin kütlesini (m) gir
3. Maddenin hızını (v) gir
4. Maddenin yerden yüksekliğini (h) gir
5. ep=m*9.8*h
6. ek=m*v^2/2
7. Yaz ep,ek
8. Dur



Örnek-6.11: Yerden yüksekliği ve ilk hızı girilen cismin havada kalma süresini, yatayda alacağı yolu ve yere çarpma hızını hesaplayan program.



1. Başla
2. Yerden yüksekliği (h) gir
3. Hızı (vo) gir
4. t=(2*h/9.8)^(1/2)
5. x=vo*t
6. vy=9.8*t
7. v=(vo^2+vy^2)^(1/2)
8. Yaz t,x,v
9. Dur

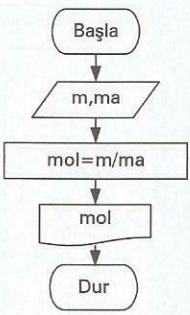


$$h = \frac{1}{2}gt^2 \quad , \quad x = v_0 t \quad , \quad v_x = v_0 \quad , \quad v_y = gt \quad , \quad v = \sqrt{v_x^2 + v_y^2}$$

Örnek-6.12: Klavyeden kütlesi ve molekül ağırlığı girilen maddenin mol sayısını hesaplayan program.

$$n = \frac{m}{m_A}$$

1. Başla
2. Maddenin kütlesini (m) gir
3. Maddenin molekül ağırlığını (ma) gir
4. mol=m/ma
5. Yaz mol
6. Dur

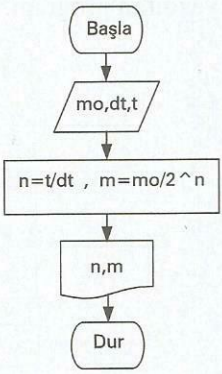


Örnek-6.13: Klavyeden başlangıç kütlesi ile yarılanma süresi girilen bir radyoaktif elementin belirtilen süre sonunda kaç yarılanma geçireceğini ve kütlesinin ne kadar kalacağını hesaplayan program.

Yarılanma sayısı: $n = \frac{t}{\Delta t}$

Kalan kütle: $m = \frac{m_0}{2^n}$

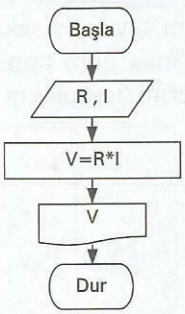
1. Başla
2. Başlangıç kütlesini (mo) gir
3. Yarılanma süresini (dt) gir
4. Süreyi (t) gir
5. n=t/dt
6. m=mo/2^n
7. Yaz n,m
8. Dur



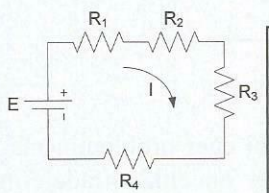
Örnek-6.14: Klavyeden bir iletkenin direnciyle içinden akan akımı girildiğinde, uçlarındaki gerilimi hesaplayan program (*Ohm Kanunu*).

$V = R.I$

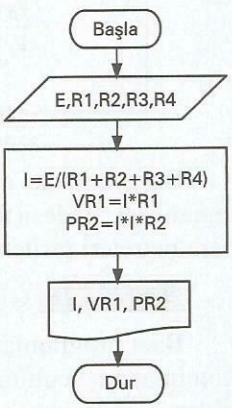
1. Başla
2. İletkenin direncini (R) gir
3. İletkenin içinden akan akımı (I) gir
4. V=R*I
5. Yaz V
6. Dur



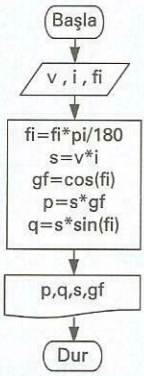
Örnek-6.15: Şekildeki devreye ait gerilim kaynağı ve direnç değerleri klavyeden girilmektedir. Buna göre devreden akan akımı (I), R1 direnci üzerindeki gerilim düşümünü ve R2 direncinde harcanan gücü hesaplayıp ekrana yazdıran program.



1. Başla
2. Gerilimi (E) ve dirençleri (R1, R2, R3, R4) gir
3. I=E/(R1+R2+R3+R4)
4. VR1=R1*I
5. PR2=I^2*R2
6. Yaz I, VR1, PR2
6. Dur

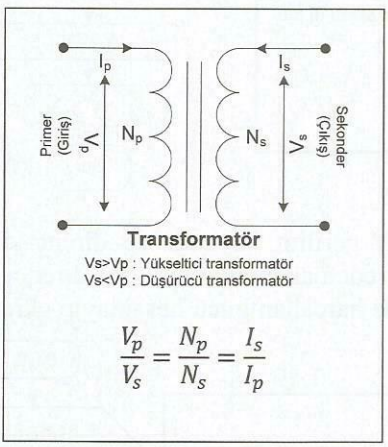


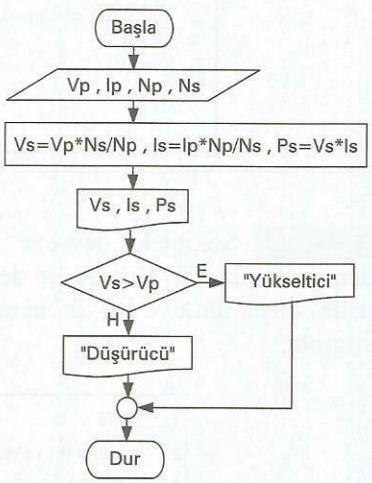
Örnek-6.16: Bir güç sistemine ait gerilim ve akımın "Volt" ile "Amper" cinsinden genlikleri ve "derece" cinsinden de faz açıları girilmektedir. Buna göre sistemdeki aktif-reaktif-görünür gücü ve güç faktörünü hesaplayıp ekrana yazdıran program.



- ✓ *Aktif güç:* $P = V.I.Cos(\varphi)$
- ✓ *Reaktif güç:* $Q = V.I.Sin(\varphi)$
- ✓ *Görünür güç:* $S = V.I$
- ✓ *Güç faktörü:* $pf = \frac{P}{S}$

Örnek-6.17: İdeal transformatörün primer (giriş) tarafına ait gerilim, akım ve sarım sayısıyla sekonder (çıkış) tarafına ait sarım sayısı klavyeden girilmektedir. Buna göre transformatörün sekonder tarafına ait gerilim, akım ve güç değerlerini hesaplayıp türünü tespit ederek sonuçları yazdıran program.



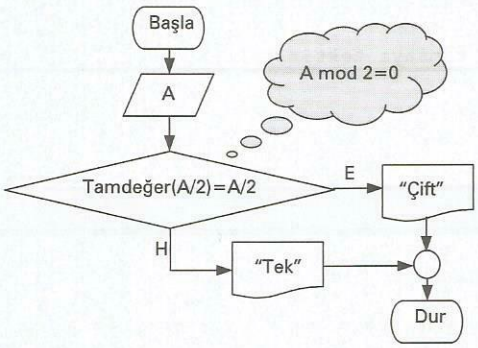


Buraya kadar anlatılan örneklerden de görüldüğü gibi eğer problemin matematiksel ifadesi(formülü)/ifadeleri verilirse; programda öncelikle ifadelerin parametreleri girilerek hesaplamalar yapılmakta ve sonuçlar yazdırılmaktadır.

Örnek-6.18: Girilen tamsayının tek mi çift mi olduğunu bulan program.

Bazı programlama dillerinde; bir tamsayının tek mi çift mi olduğunu veren komutlar mevcuttur. Ancak burada ilgili komutlar kullanılmayıp çalışma mantıkları sunulacaktır.

Bir tamsayının tek mi çift mi olduğunu bulmanın en kolay yolu; tamsayının 2'ye göre modunu almaktır (2'ye tam bölünüp bölünmediğini kontrol etmektir). Eğer 2'ye göre modu 0 ise, yani 2'ye tam bölünüyorsa tamsayı çifttir; aksi durumda tektir. Diğer bir yol da tamsayıyı, 2'ye bölüp bulunan sonucun tam kısmı (tamdeğer fonksiyonu) ile yine aynı tamsayının 2'ye bölümünü kıyaslamaktır. Çift olan tamsayılarda 2'ye bölümün tam kısmı ile tamsayının 2'ye bölümü aynıdır. Oysa tek tamsayılarda bu değerler farklıdır (Tablo 6.1).



Tablo 6.1: Örnek 6.18 için işlem adımları

Sayı	Tamdeğer (Sayı/2)	Sayı/2	Eşit mi?	Sayı mod 2	Tek/Çift
1	0	0,5	Hayır	1	Tek
2	1	1	Evet	0	Çift
3	1	1,5	Hayır	1	Tek
4	2	2	Evet	0	Çift

C

```
/* Tek-çift */
#include <stdio.h>
#include <conio.h>
main()
{ float a; clrscr();printf("Tamsayıyı giriniz: ");scanf("%f",&a);
  if (a%2==0) printf("\nGirilen tamsayı çifttir.");
  else printf("\nGirilen tamsayı tektir.");   getch(); return 0; }
```

C++

```
// Tek-Çift
#include <iostream>
using namespace std;
main()
{  float a; system("cls"); cout<<"Tamsayıyı giriniz: "; cin>>a;
   if(a%2==0) cout << "\nGirilen tamsayi cifttir." << endl;
   else cout << "\nGirilen tamsayi tektir." << endl;
   system("pause");return 0; }
```

C#

```
// Tek-çift
using System;
namespace ornek618
{class Program
  {static void Main(string[] args)
    {double a; Console.Clear(); Console.Write("Tamsayıyı giriniz: ");
     a=Convert.ToDouble(Console.ReadLine());
     if (a%2==0) Console.WriteLine("\nGirilen tamsayi cifttir.");
     else Console.WriteLine("\nGirilen tamsayi tektir.");
     Console.ReadLine();  }  }  }
```

JAVA

```
// Tek-çift
import java.util.Scanner;
public class ornek618 {
public static void main(String[] args)
{  float a; Scanner klavye=new Scanner(System.in);
   System.out.print("Tamsayıyı giriniz: "); a=klavye.nextInt();
   if(a%2==0)
      System.out.println("\nGirilen tamsayı çifttir.");
   else
      System.out.println("\nGirilen tamsayı tektir."); }}
```

```
Tamsayıyı giriniz: 99

Girilen tamsayı tektir.
```

MATLAB

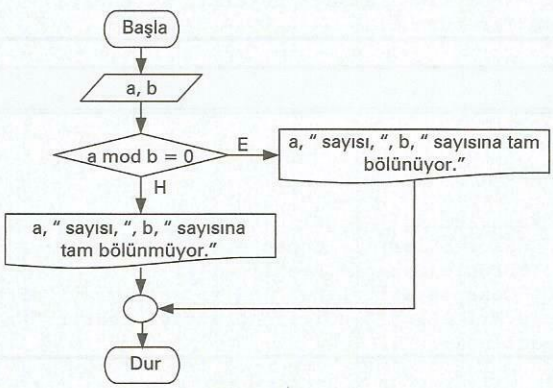
```
% Tek-çift
clear all; clc; a=input('Tamsayıyı giriniz: ');
if (mod(a,2)==0)
    fprintf('\nGirilen tamsayı çifttir.\n');
else
    fprintf('\nGirilen tamsayı tektir.\n');
end
```

PYTHON

```
# Tek-çift
a=int(input("Tamsayıyı giriniz: "))
if (a%2==0):
    print("\nGirilen tamsayı çifttir.\n")
else:
    print("\nGirilen tamsayı tektir.\n")
```

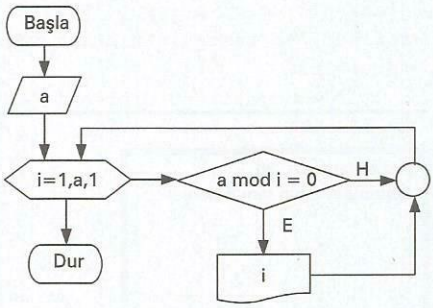
Örnek-6.19: Klavyeden girilen a sayısının, yine klavyeden girilen b sayısına tam bölünüp bölünmediğini veren program.

a sayısı b sayısına tam bölünüyorsa, kalan sıfırdır ($a \mod b = 0$).



Örnek-6.20: Klavyeden girilen pozitif bir a tamsayısının tam bölenlerini hesaplayıp listeleyen program.

a pozitif tamsayısının tam bölenlerini bulmak için $1-a$ arasında bir döngü (i) açılır ve a tamsayısının, döngü değişkenine tam bölünüp bölünmediği kontrol edilir. Eğer bölünüyorsa döngü değişkeni yazdırılır, bölünmüyorsa döngüye devam edilir.



C

```
/* Tam bölenler */
#include <stdio.h>
#include <conio.h>
void main()
{  int a,i; clrscr();printf("Pozitif tamsayı: ");scanf("%u",&a);
   printf("\n%u tamsayısının tam bölenleri:\n",a);
   for(i=1;i<=a;i++) if(a%i==0) printf("%u\n",i); getch(); }
```

C++

```
// Tam bölenler
#include <iostream>
using namespace std;
int main()
{   int a; system("cls"); cout << "Pozitif tamsayı: "; cin >> a;
    cout <<"\n"<<a<<" tamsayisinin tam bolenleri:"<<endl;
    for(int i=1;i<=a;i++)
       if(a%i==0) cout << i << endl;     system("pause");return 0; }
```

C#

```
// Tam bölenler
using System;
namespace ornek620
{ class Program
  {static void Main(string[] args)
   {int a; Console.Write("Pozitif tamsayı: ");
    a=Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("{0} tamsayısının tam bölenleri:",a);
    for (int i=1;i<=a;i++)
      if (a%i==0) Console.WriteLine("{0}", i); Console.ReadLine(); }}}
```

JAVA

```
// Tam bölenler
import java.util.Scanner;
public class ornek620 {
public static void main(String[] args)
{       int a; Scanner klavye=new Scanner(System.in);
        System.out.print("Pozitif tamsayı: ");a=klavye.nextInt();
        System.out.println(a+" tamsayısının tam bölenleri:");
        for(int i=1;i<=a;i++) if(a%i==0) System.out.println(i); }}
```

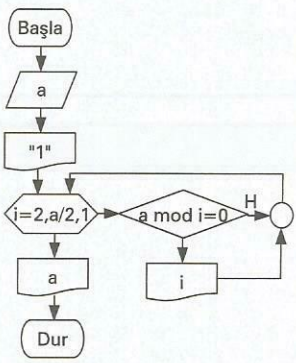
MATLAB

```
% Tam bölenler
clear all; clc; a=input('Pozitif tamsayı: ');
fprintf('%d tamsayısının tam bölenleri:\n',a);
for i=1:a
    if (mod(a,i)==0) fprintf('%d\n',i); end; end
```

```
Pozitif tamsayı: 30
30 tamsayısının tam bölenleri:
1
2
3
5
6
10
15
30
```

PYTHON

```
# Tam bölenler
a=int(input("Pozitif tamsayı: "))
print("%d tamsayısının tam bölenleri:"%a)
for i in range(1,a+1):
    if (a%i==0):
        print("%d"%i)
```



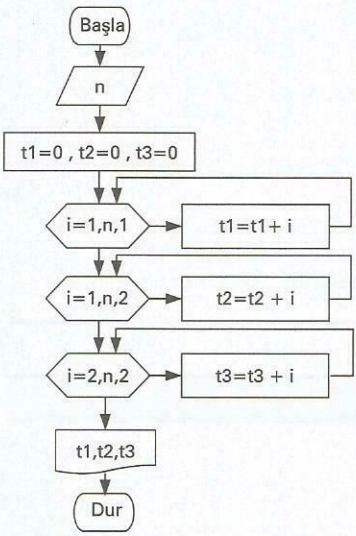
Yukarıdaki programlar, girilen tamsayının tam bölenlerini doğru bir şekilde hesaplayıp listelemektedir. Ancak bu programların daha hızlı çalışması (daha az işlem yapması) sağlanabilir. Örneğin klavyeden girilen tamsayı $a=30$ ise $1-30$ arasında açılan döngü ile 30 tane mod alma işlemi yapılmakta ve 0'a eşit olup olmadığı kontrol edilmektedir. Hâlbuki hiçbir tamsayı; kendi yarısından daha büyük bir tamsayıya (kendisi hariç), tam bölünememektedir. Ayrıca tüm tamsayılar, 1'e ve kendisine tam bölünmektedirler.

Bu durumda döngünün $2-15$ arasında açılması (dolayısıyla 14 tane mod alma işlemi ve 0'a eşit olup olmama kontrolü) ve 1 ile kendisinin doğrudan yazdırılması daha uygun olacaktır. Böylece programdaki işlem sayısı azalmakta ve çalışma hızı artmaktadır.

Not

Programlarda, gereksiz (sonuca etki etmeyecek) işlemlerden kaçınınız.

Örnek-6.21: Klavyeden girilen n sayısına göre; 1'den n'e kadar tamsayıların toplamını ($t1$), 1'den n'e kadar tek tamsayıların toplamını ($t2$) ve 2'den n'e kadar çift tamsayıların toplamını ($t3$) hesaplayan program.



Programda, toplanacak tamsayıları oluşturmak için döngülerin kullanılması (açılması) gerekmektedir. Bu durumda;

- $t1$ için 1'den n'e kadar birer birer artan bir döngü,
- $t2$ için 1'den n'e kadar ikişer ikişer artan bir döngü,
- $t3$ için de 2'den n'e kadar ikişer ikişer artan bir döngü

açılır ve ardışık toplama işlemleriyle istenilenler hesaplanır.

C

```
/* Toplamlar */
#include <stdio.h>
#include <conio.h>
int i,n; float t1=0,t2=0,t3=0;
void main()
{  clrscr();printf("Pozitif tamsayı giriniz: ");scanf("%u",&n);
for(i=1;i<=n;i++) t1+=i;
for(i=1;i<=n;i+=2) t2+=i;
for(i=2;i<=n;i+=2) t3+=i;
printf("\n1'den %d'e kadar sayıların toplamı: %0.2f\n",n,t1);
printf("1'den %d'e kadar tek sayıların toplamı: %0.2f\n",n,t2);
printf("1'den %d'e kadar çift sayıların toplamı: %0.2f",n,t3);
getch(); }
```

C++

```
// Toplamlar
#include <iostream>
using namespace std;
int i,n; float t1=0,t2=0,t3=0;
int main()
{  system("cls");cout << "Pozitif tamsayı giriniz: "; cin >> n;
   for(i=1;i<=n;i++) t1+=i;
   for(i=1;i<=n;i+=2) t2+=i;
   for(i=2;i<=n;i+=2) t3+=i;
   cout<<"\n1'den "<<n<<"'e kadar sayilarin toplami: "<<t1<<endl;
   cout<<"1'den "<<n<<"'e kadar tek sayilarin toplami: "<<t2<<endl;
   cout<<"2'den "<<n<<"'e kadar çift sayilarin toplami "<<t3<<endl;
   system("pause");return 0; }
```

C#

```
// Toplamlar
using System;
namespace ornek621
{ class Program
   { static void Main(string[] args)
      { int i, n; float t1=0,t2=0,t3=0;
        Console.Write("Pozitif tamsayı giriniz: ");
        n=Convert.ToInt32(Console.ReadLine());
        for(i=1;i<=n;i++) t1+=i;
        for(i=1;i<=n;i+=2) t2+=i;
        for(i=2;i<=n;i+=2) t3+=i;
Console.WriteLine("\n1'den {0}'e kadar sayilarin toplami: {1}",n,t1);
Console.WriteLine("1'den {0}'e kadar tek sayilarin toplami: {1}",n,t2);
Console.WriteLine("1'den {0}'e kadar cift sayilarin toplami:{1}",n,t3);
        Console.ReadLine(); }    }   }
```

JAVA

```
// Toplamlar
import java.util.Scanner;
public class ornek621 {
public static void main(String[] args)
{ int i,n; float t1=0,t2=0,t3=0;
  Scanner klavye=new Scanner(System.in);
  System.out.print("Pozitif tamsayı giriniz: "); n=klavye.nextInt();
  for(i=1;i<=n;i++) t1+=i;
  for(i=1;i<=n;i+=2) t2+=i;
  for(i=2;i<=n;i+=2) t3+=i;
  System.out.println("\n1 den "+n+"'e kadar sayıların toplamı: "+t1);
  System.out.println("1 den "+n+"'e kadar tek sayıların toplamı: "+t2);
 System.out.println("1 den "+n+"'e kadar çift sayıların toplamı: "+t3);
}}
```

MATLAB

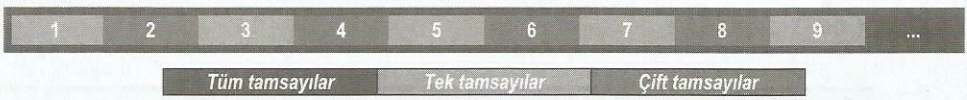
```
% Toplamlar
clear all; clc
n=input('Pozitif tamsayı giriniz: '); t1=0; t2=0; t3=0;
for i=1:n
    t1=t1+i;
end % Döngü yerine t1=sum(1:n);
for i=1:2:n
    t2=t2+i;
end % Döngü yerine t2=sum(1:2:n);
for i=2:2:n
    t3=t3+i;
end % Döngü yerine t3=sum(2:2:n);
fprintf('\n1%cden %d%ce kadar sayıların toplamı: %d\n',96,n,96,t1);
fprintf('1%cden %d%ce kadar tek sayıların toplamı: %d\n',96,n,96,t2);
fprintf('2%cden %d%ce kadar çift sayıların toplamı: %d\n',96,n,96,t3);
```

PYTHON

```
# Toplamlar
n=int(input("Pozitif tamsayı giriniz: "))
t1=t2=t3=0
for i in range(1,n+1):
    t1+=i
# Döngü yerine t1=sum(range(1,n+1,1))
for i in range(1,n+1,2):
    t2+=i
# Döngü yerine t2=sum(range(1,n+1,2))
for i in range(2,n+1,2):
    t3+=i
# Döngü yerine t3=sum(range(2,n+1,2))
print("\n1'den %d'ye kadar sayıların toplamı:"%n,t1)
print("1'den %d'ye kadar tek sayıların toplamı:"%n,t2)
print("2'den %d'ye kadar sayıların toplamı:"%n,t3)
```

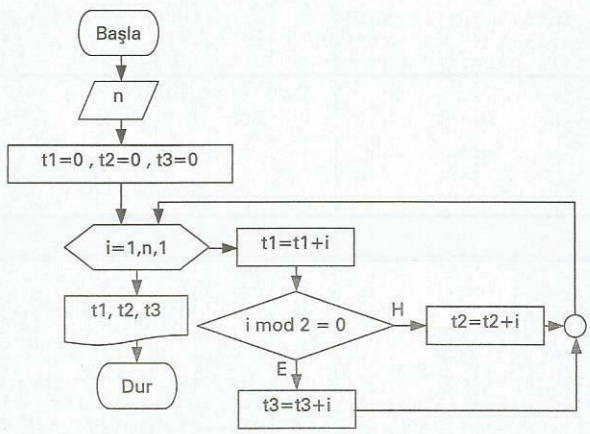
```
Pozitif tamsayı giriniz: 50

1'den 50 'ye kadar sayıların toplamı: 1275
1'den 50 'ye kadar tek sayıların toplamı: 625
2'den 50 'ye kadar çift sayıların toplamı: 650
```

Bu program, tek döngü kullanılarak da gerçekleştirilebilir. 1'den n'e kadar birer birer artan döngü aynı zamanda tek ve çift sayıları içermektedir. Bu nedenle aynı döngü içinde, döngü değişkeninin tek veya çift olduğu kontrol edilir ve duruma göre ilgili toplama işlemi yaptırılabilir.



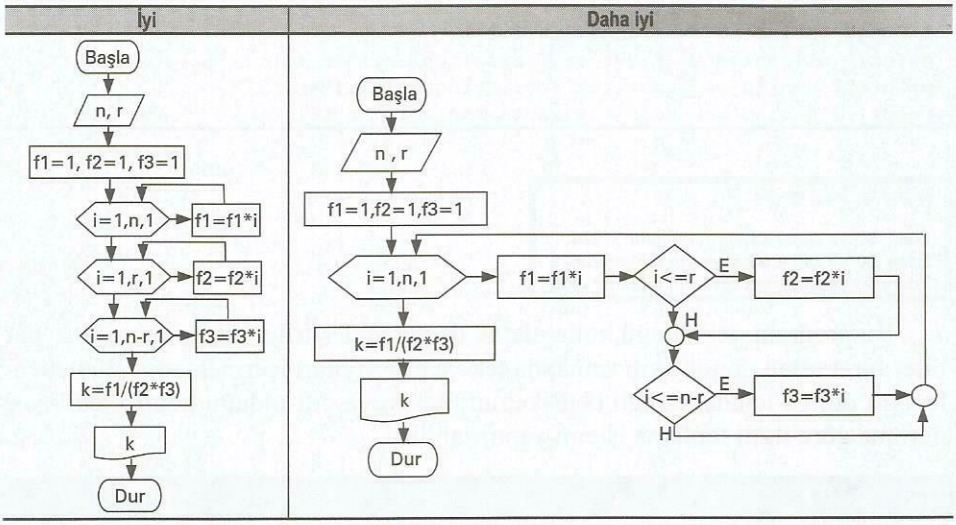
Not

Programlarda, aynı/benzer işlemleri tekrarlamaktan kaçınınız.



Örnek-6.22: Klavyeden eleman sayısı girilen bir kümenin belirtilen kombinasyonlarının sayısını hesaplayan program.

n elemanlı bir kümenin r'li kombinasyonlarının sayısı $C(n,r)=\binom{n}{r}=\frac{n!}{r!(n-r)!}$ ile hesaplanmaktadır. Problemi basitleştirmek için $f1=n!$, $f2=r!$, $f3=(n-r)!$ alt işlemlerine ayrıştırılır ve $k=\frac{f1}{f2.f3}$ ile sonuca ulaşılabilir.



C

```
/* Kombinasyon */
#include <stdio.h>
#include <conio.h>
int i,n,r; float f1=1,f2=1,f3=1,k;
void main()
{  clrscr();printf("n= ");scanf("%u",&n);printf("r= ");scanf("%u",&r);
   for(i=1;i<=n;i++) {  f1*=i; if (i<=r) f2*=i; if (i<=n-r) f3*=i; };
    k=f1/(f2*f3); printf("\nSonuç= %0.0f",k); getch();  }
```

```
n= 5
r= 3

Sonuç= 10
```

C++

```
// Kombinasyon
#include <iostream>
using namespace std;
int n,r; float f1=1,f2=1,f3=1,k;
int main()
{ system("cls"); cout << "n= "; cin >> n; cout << "r= "; cin >> r;
  for(int i=1;i<=n;i++) { f1*=i; if(i<=r) f2*=i; if(i<=n-r) f3*=i; }
  k=f1/(f2*f3); cout<<"\nSonuc= "<<k<<endl;system("pause");return 0; }
```

C#

```
// Kombinasyon
using System;
namespace ornek622
{ class Program
   { static void Main(string[] args)
      { int n,r; double f1=1, f2=1, f3=1, k;
        Console.Write("n= "); n = Convert.ToInt32(Console.ReadLine());
        Console.Write("r= "); r = Convert.ToInt32(Console.ReadLine());
        for(int i=1;i<=n;i++) {f1*=i;if(i<=r) f2*=i;if(i<=n-r) f3*=i;}
        k=f1/(f2*f3); Console.WriteLine("\nSonuc= {0:F0}",k);
        Console.ReadLine(); }    }   }
```

JAVA

```
// Kombinasyon
import java.util.Scanner;
public class ornek622 {
public static void main(String[] args)
{ float f1=1,f2=1,f3=1,k;
  int n,r; Scanner klavye=new Scanner(System.in);
  System.out.print("n= "); n=klavye.nextInt();
  System.out.print("r= "); r=klavye.nextInt();
  for(int i=1;i<=n;i++) {f1*=i; if(i<=r) f2*=i; if(i<=n-r) f3*=i; }
  k=f1/(f2*f3); System.out.println("\nSonuc= "+k);   }}
```

MATLAB

```
% Kombinasyon
clear all; clc; f1=1; f2=1;f3=1;
n=input('n= '); r=input('r= ');
for i=1:n
    f1=f1*i;    % f1=factorial(n); veya f1=prod(1:n);
    if (i<=r)
        f2=f2*i;    % f2=factorial(r); veya f2=prod(1:r);
    end
    if (i<=n-r)
        f3=f3*i;    % f3=factorial(n-r); veya f3=prod(1:n-r);
    end
end ; k=f1/(f2*f3); fprintf('\nSonuç= %d\n',k);
```

PYTHON

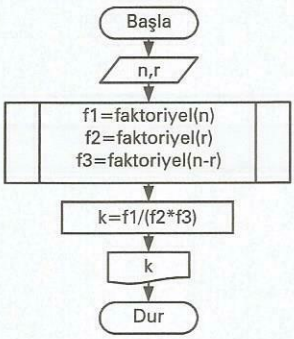
```
# Kombinasyon
f1=f2=f3=1; n=int(input("n: ")); r=int(input("r: "))
for i in range(1,n+1):
    f1*=i
    if (i<=r):
        f2*=i
    if (i<=n-r):
        f3*=i
# veya f1=math.factorial(n),f2=math.factorial(r),f3=math.factorial(n-r)
k=f1/(f2*f3)
print("\nSonuç=%d"%k)
```

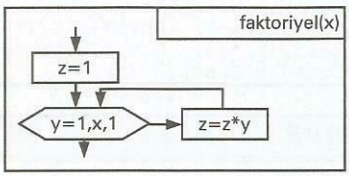
Not

Makine diline dönüşmüş (derlenmiş) çalıştırılabilir "amaç program"larınızın kodlarını görme imkânı olmadığını unutmayınız. O nedenle programlarınızda verileri isterken ve sonuçları listelerken, açıklayıcı ifadeleriniz yeterli olmalıdır.

Yukarıdaki kodlarda, kullanıcıdan n ve r istenmektedir. Ancak bu programın ne iş yaptığı, istenilenlerin ne olduğu ve sonuçta ne bulduğu; çalıştırılabi-

lir programa sahip olup kodları göremeyen veya kodları görüp de programlama bilgisine sahip olmayan kişiler için belli değildir. Bu nedenle yukarıdaki programın açıklamalar açısından daha uygun kodlamaları, aşağıda verilecektir. Ayrıca aynı problem, bir tane faktoriyel altprogramı/fonksiyonu yazılarak daha sade kodlanabilir.





Not

Programlarınızı, daha küçük alt programlara ayırarak kodlamanız; genel olarak sizlere büyük kolaylıklar (kod sadeliği, anlaşılırlığı, modülerliği; hata takibi vb.) sağlayacaktır.

C

```
/* Kombinasyon_1 */
#include <stdio.h>
#include <conio.h>
float faktoriyel(int x)
{ int y; float z=1; for(y=1;y<=x;y++) z*=y; return z; }
void main()
{ int n,r; float f1,f2,f3,k;
  clrscr();printf("Kombinasyon(n,r)\n\n");
  printf("n= ");scanf("%u",&n);printf("r= ");scanf("%u",&r);
  f1=faktoriyel(n); f2=faktoriyel(r); f3=faktoriyel(n-r);
  k=f1/(f2*f3); printf("\nKombinasyon(%d,%d)= %0.0f",n,r,k); getch(); }
```

C++

```
// Kombinasyon_1
#include <iostream>
using namespace std;
float faktoriyel(int x)
{ float z=1; for(int y=1;y<=x;y++) z*=y; return z; }
int main()
{   int n,r; float f1,f2,f3,k;
    system("cls"); cout << "Kombinasyon(n,r)" << endl<<endl;
    cout << "n= "; cin >> n; cout << "r= "; cin >> r;
    f1=faktoriyel(n); f2=faktoriyel(r); f3=faktoriyel(n-r);
    k=f1/(f2*f3); cout << "\nKombinasyon("<<n<<","<<r<<")="<<k<<endl;
    system("pause");return 0; }
```

```
Kombinasyon(n,r)

n= 5
r= 3

Kombinasyon(5,3)= 10
```

C#

```
// Kombinasyon 1
using System;
namespace ornek622
{ class Program
   { public static float faktoriyel(int x)
      { float z = 1; for (int y = 1; y <= x; y++) z *= y; return z; }
     static void Main(string[] args)
      { int n, r; float f1, f2, f3, k;
        Console.WriteLine("Kombinasyon(n,r)\n");
        Console.Write("n= "); n=Convert.ToInt32(Console.ReadLine());
        Console.Write("r= "); r=Convert.ToInt32(Console.ReadLine());
        f1=faktoriyel(n); f2=faktoriyel(r); f3=faktoriyel(n-r);
        k=f1/(f2*f3);
        Console.WriteLine("\nKombinasyon({0},{1})={2:F0}",n,r,k);
        Console.ReadLine(); } } }
```

JAVA

```
// Kombinasyon_1
import java.util.Scanner;
public class ornek622 {
public static float faktoriyel(int x) {
  float z=1; for(int y=1;y<=x;y++) z*=y; return z; }
public static void main(String[] args)
{ int n,r; float f1,f2,f3,k;
  Scanner klavye=new Scanner(System.in);
  System.out.println("Kombinasyon(n,r)\n");
  System.out.print("n= "); n=klavye.nextInt();
  System.out.print("r= "); r=klavye.nextInt();
  f1=faktoriyel(n); f2=faktoriyel(r); f3=faktoriyel(n-r);
  k=f1/(f2*f3); System.out.println("\nKombinasyon("+n+","+r+")="+k); }}
```

MATLAB

```
% Kombinasyon 1
clear all; clc
fprintf('Kombinasyon(n,r)\n\n'); n=input('n= '); r=input('r= ');
f1=factorial(n); f2=factorial(r); f3=factorial(n-r);
k=f1/(f2*f3); fprintf('\nKombinasyon(%d,%d)=%d\n',n,r,k);
```

PYTHON

```
# Kombinasyon 1
import math
print("Kombinasyon(n,r)\n"); n=int(input("n: ")); r=int(input("r: "))
f1=math.factorial(n); f2=math.factorial(r); f3=math.factorial(n-r)
k=f1/(f2*f3); print("\nKombinasyon(%d,%d)=%d"%(n,r,k))
```

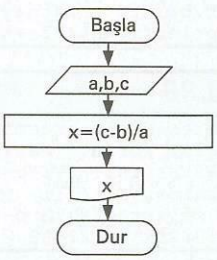
Örnek-6.23: Klavyeden katsayıları girilen birinci dereceden denklemin kökünü hesaplayan program.

$a \neq 0$ olmak üzere $ax + b = c$ şeklindeki birinci dereceden denklemin kökü, aşağıdaki gibi hesaplanır:

$$ax + b = c$$

$$ax = c - b$$

$$x = \frac{c-b}{a}$$



C

```
/* Birinci dereceden denklemin kökü */
#include <stdio.h>
#include <conio.h>
main()
{ float a,b,c,x;
  clrscr();printf("ax+b=c\n"); printf("a katsayısı: ");scanf("%f",&a);
  printf("b katsayısı: ");scanf("%f",&b);
  printf("c katsayısı: ");scanf("%f",&c); x=(c-b)/a;
  printf("\n%0.2fx+%0.2f=%0.2f denkleminin kökü:%0.3f",a,b,c,x);
  getch(); return 0; }
```

C++

```
// Birinci dereceden denklemin kökü
#include <iostream>
using namespace std;
main()
{ float a,b,c,x;
  system("cls"); cout << "ax+b=c\n"; cout << "a katsayısı: "; cin >> a;
  cout << "b katsayısı: "; cin >> b; cout << "c katsayısı: "; cin >> c;
  x=(c-b)/a;
  cout<<"\n"<<a<<"x+"<<b<<"="<<c<<" denkleminin kökü: "<<x<<endl;
  system("pause");return 0; }
```

C#

```
// Birinci dereceden denklemin kökü
using System;
namespace ornek623
{ class Program
   { static void Main(string[] args)
    { float a, b, c, x; Console.WriteLine("ax+b=c");
 Console.Write("a katsayısı: ");a=Convert.ToSingle(Console.ReadLine());
 Console.Write("b katsayısı: ");b=Convert.ToSingle(Console.ReadLine());
 Console.Write("c katsayısı: ");c=Convert.ToSingle(Console.ReadLine());
 x=(c-b)/a; Console.Write("\n{0:F2}x+{1:F2}={2:F2}
            denkleminin kökü:{3:F3}",a,b,c,x); Console.ReadLine();}}}
```

```
ax+b=c
a katsayısı: 2
b katsayısı: 5
c katsayısı: 11

2,00x+5,00=11,00 denkleminin kökü: 3,000
```

JAVA

```
// Birinci dereceden denklemin kökü
import java.util.Scanner;
public class ornek623 {
public static void main(String[] args)
{ float a,b,c,x; Scanner klavye=new Scanner(System.in);
  System.out.println("ax+b=c");
  System.out.print("a katsayısı: ");a=klavye.nextFloat();
  System.out.print("b katsayısı: ");b=klavye.nextFloat();
  System.out.print("c katsayısı: ");c=klavye.nextFloat(); x=(c-b)/a;
System.out.printf("\n%.2fx+%.2f=%.2f denkleminin kökü:%.3f",a,b,c,x);}}
```

MATLAB

```
% Birinci dereceden denklemin kökü
clear all; clc; fprintf('ax+b=c\n');
a=input('a katsayısı: '); b=input('b katsayısı: ');
c=input('c katsayısı: ');
x=(c-b)/a;   % x=roots([a b-c]);
fprintf('\n%0.2fx+%0.2f=%0.2f denkleminin kökü: %0.3f\n',a,b,c,x);
```

MATLAB

```
% Birinci dereceden denklemin kökü
clear all; clc;
syms x
fprintf('a*x+b=c\n'); fx=input('Denklem: ','s'); x=solve(fx);
fprintf('\n%s denkleminin kökü: %0.3f\n',fx,double(x));
```

PYTHON

```
# Birinci dereceden denklemin kökü
print("ax+b=c"); a=eval(input("a katsayısı: "))
b=eval(input("b katsayısı: ")); c=eval(input("c katsayısı: "))
x=(c-b)/a
print("\n%0.2fx+%0.2f=%0.2f denkleminin kökü: %0.3f\n"%(a,b,c,x))
```

Yukarıdaki program kodlarında, kullanıcının *a* katsayısını 0 (sıfır) girmesi durumunda sorun (sıfıra bölme) oluşacaktır. Bu nedenle problemin akış diyagramının yandaki gibi olması daha uygundur. Bu akış diyagramının örnek bir dildeki kodu ve ekran çıktısı, aşağıda verilmektedir.

MATLAB

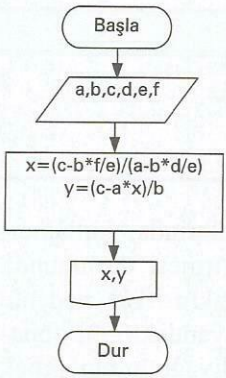
```
% Birinci dereceden denklemin kökü
clear all; clc; fprintf('ax+b=c\n\n'); a=input('a katsayısı: ');
while (a==0)
    fprintf('Sıfırdan farklı bir değer giriniz\n');
    a=input('a katsayısı: ');
end; b=input('b katsayısı: '); c=input('c katsayısı: '); x=(c-b)/a;
fprintf('\nKök: %0.3f\n',x);
```

```
ax+b=c

a katsayısı: 0
Sıfırdan farklı bir değer giriniz
a katsayısı: 2
b katsayısı: 3
c katsayısı: 7

Kök: 2.000
```

Not

Yazdığınız programlarınızı; hiç programlama bilmeyen veya programın işleviyle ilgili bilgiye sahip olmayan kişilerin de kullanabileceğini unutmayınız. Bu nedenle tüm ihtimalleri (geçersiz veriler, kısıtlamalar vb.) programlarınıza dâhil etmeye çalışınız.

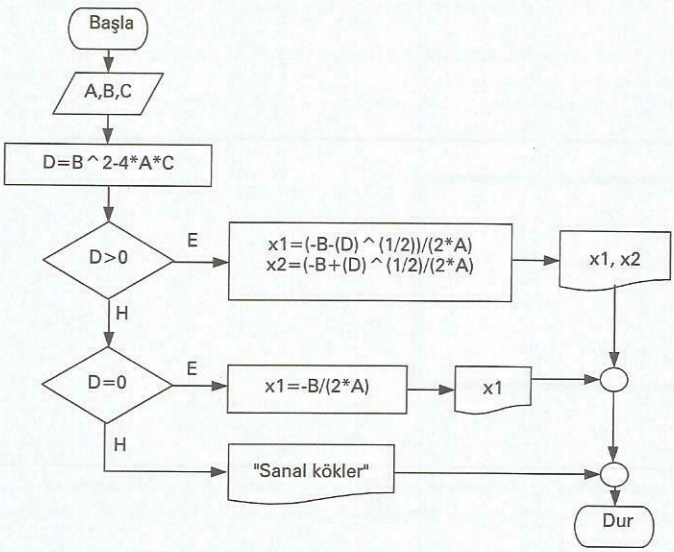
Örnek-6.24: Katsayıları klavyeden girilen $\left.\begin{matrix} ax+by=c \\ dx+ey=f \end{matrix}\right\}$ iki bilinmeyenli iki denklemden oluşan sistemi çözen program (Girilen denklem sisteminin çözümünün olduğunu varsayınız; yani hata denetimi yapmayınız, katsayıların uygun girildiğini kabul ediniz).



Örnek-6.25: $A \neq 0$ olmak üzere klavyeden katsayıları girilen ikinci dereceden denklemin köklerini hesaplayan program.

$Ax^2+Bx+C=0$ ikinci derece denkleminin köklerini hesaplamak için öncelikle $\Delta = B^2-4AC$ diskriminantı hesaplanır. Diskriminantın durumuna göre kökler yandaki gibi hesaplanmaktadır.

$$\begin{cases} \Delta > 0 \Rightarrow x_{1,2} = \dfrac{-B \pm \sqrt{\Delta}}{2A} \\ \Delta = 0 \Rightarrow x_1 = x_2 = \dfrac{-B}{2A} \\ \Delta < 0 \Rightarrow \textit{Sanal kökler} \end{cases}$$



C

```
/* 2. dereceden denklemin kökleri */
#include <stdio.h>
#include <conio.h>
#include <math.h>
float a,b,c,d,x1,x2;
main()
{ clrscr(); printf("Ax^2+Bx+C=0"); printf("A= ");scanf("%f",&a);
  printf("B= ");scanf("%f",&b);
  printf("C= ");scanf("%f",&c);printf("\n"); d=b*b-4*a*c;
  if (d>0)
   { printf("Gerçel kökler vardır:\n");
     x1=(-b-sqrt(d))/(2*a);x2=(-b+sqrt(d)/(2*a);
     printf("Kökler: %0.2f\t%0.2f",x1,x2); }
  else if (d==0)
   { printf("Katlı kök vardır:\n");
     x1=-b/(2*a); printf("Katlı kök: %0.2f",x1); }
  else printf("Sanal kökler vardır.");   getch(); return 0;   }
```

C++

```
// 2. dereceden denklemin kökleri
#include <iostream>
#include <cmath>
using namespace std;
float a,b,c,d,x1,x2;
int main()
{ system("cls"); cout << "Ax^2+Bx+C=0\n"; cout << "A= "; cin >> a;
  cout<<"B= ";cin>>b;cout<<"C= ";cin>>c;d=b*b-4*a*c; cout<<"\n";
  if(d>0)
    { cout << "Gerçel kökler vardır:" << endl;
      x1=(-b-sqrt(d))/(2*a);x2=(-b+sqrt(d))/(2*a);
      cout << "Kökler: " << x1 <<"\t"<< x2 << endl;  }
  else if (d==0)
    { cout <<"Katlı kök vardır:" << endl;
      x1=-b/(2*a);cout<<"Katlı kök: "<<x1<<endl; }
  else cout <<"Sanal kökler vardır."<<endl; system("pause");return 0; }
```

C#

```
// İkinci dereceden denklemin kökü
using System;
namespace ornek625
{ class Program
   { static void Main(string[] args)
    { double a,b,c,d,x1,x2;
      Console.WriteLine("Ax^2+Bx+C=0");
      Console.Write("A= ");a=Convert.ToDouble(Console.ReadLine());
      Console.Write("B= ");b=Convert.ToDouble(Console.ReadLine());
      Console.Write("C= ");c=Convert.ToDouble(Console.ReadLine());
      d=b*b-4*a*c;
      if(d>0)
        { Console.WriteLine("\nGerçel kökler vardır:");
          x1=(-b-Math.Sqrt(d))/(2*a);x2=(-b+Math.Sqrt(d))/(2*a);
          Console.WriteLine("Kökler: {0:F2}\t{1:F2}",x1,x2); }
      else if(d==0)
        { Console.WriteLine("\nKatlı kök vardır:"); x1=-b/(2*a);
          Console.WriteLine("Katlı kök: {0:F2}",x1); }
      else Console.WriteLine("\nSanal kökler vardır.");
      Console.ReadLine();   }   }   }
```

JAVA

```
// 2. dereceden denklemin kökleri
import java.util.Scanner;
public class ornek625 {
public static void main(String[] args)
{ double a,b,c,d,x1,x2;
  Scanner klavye=new Scanner(System.in);
  System.out.println("Ax^2+Bx+C=0");
  System.out.print("A= ");a=klavye.nextDouble();
  System.out.print("B= ");b=klavye.nextDouble();
  System.out.print("C= ");c=klavye.nextDouble();  d=b*b-4*a*c;
  if(d>0)
    { System.out.println("\nGerçel kökler vardır:");
      x1=(-b-Math.sqrt(d))/(2*a);x2=(-b+Math.sqrt(d))/(2*a);
      System.out.printf("Kökler: %.2f\t%.2f",x1,x2); }
  else if(d==0)
    { System.out.println("\nKatlı kök vardır:"); x1=-b/(2*a);
      System.out.printf("Katlı kök: %.2f",x1); }
  else System.out.println("\nSanal kökler vardır.");  }}
```

MATLAB

```
% 2. dereceden denklemin kökleri
clear all; clc ; fprintf('Ax^2+Bx+C=0\n');
a=input('A= '); b=input('B= '); c=input('C= ');
x=roots([a b c]); fprintf('\nKökler:\n');disp(x)
```

```
Ax^2+Bx+C=0
A= 1
B= 2
C= 3

Kökler:
 -1.0000 + 1.4142i
 -1.0000 - 1.4142i
```

PYTHON

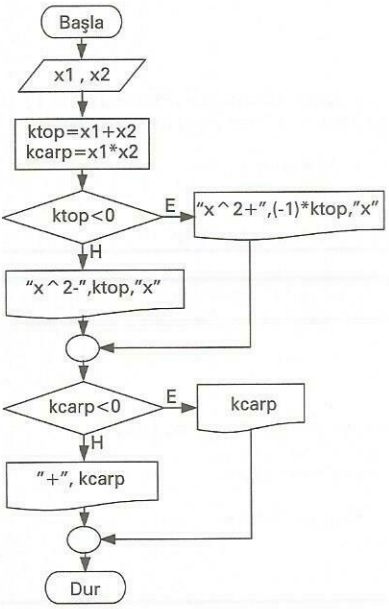
```
# İkinci dereceden denklemin kökleri
import math
import cmath
print("Ax^2+Bx+C=0"); a=eval(input("A= ")); b=eval(input("B= "))
c=eval(input("C= ")); d=b**2-4*a*c
if (d>=0):
    x1=(-b-math.sqrt(d))/(2*a); x2=(-b+math.sqrt(d))/(2*a)
else:
    x1=(-b-cmath.sqrt(d))/(2*a); x2=(-b+cmath.sqrt(d))/(2*a)
print("\nKökler:"); print(x1,x2,sep='\n')
```

```
Ax^2+Bx+C=0
A= 1
B= 2
C= 3

Kökler:
(-1-1.4142135623730951j)
(-1+1.4142135623730951j)
```

Örnek-6.26: "*Viète formülleri*", polinom katsayılarıyla bu polinomum kökleri arasındaki ilişkileri gösteren eşitliklerdir. $n \geq 1$ ve $a_n \neq 0$ olmak üzere $f(x)=a_n x^n+a_{n-1}x^{n-1}+a_{n-2}x^{n-2}+...+a_1x+a_0$ polinomunda katsayılar ile kökler arasında

$$\begin{cases} x_1+x_2+...+x_{n-1}+x_n=-\dfrac{a_{n-1}}{a_n} \\ (x_1x_2+x_1x_3+...+x_1x_n)+(x_2x_3+x_2x_4+...+x_2x_n)+...+x_{n-1}.x_n=\dfrac{a_{n-2}}{a_n} \\ \vdots \\ x_1.x_2.....x_{n-1}.x_n=(-1)^n\dfrac{a_0}{a_n} \end{cases}$$

ilişkileri mevcuttur. Buna göre klavyeden girilen x_1 ve x_2 kökleri için 2. dereceden polinomu oluşturup yazdıran program.



"*Viète formülleri*"ne göre kökleri x_1 ve x_2 olan 2. dereceden $a_2x^2+a_1x+a_0$ polinomunda $x_1+x_2=-\dfrac{a_1}{a_2}$ ve $x_1.x_2=\dfrac{a_0}{a_2}$ ilişkileri vardır. Yani $ax^2+bx+c=0$ 2. dereceden denkleminde $x_1+x_2=-\dfrac{b}{a}$ ve $x_1.x_2=\dfrac{c}{a}$ dir.

C

```
/* 2. dereceden polinom olusturma */
#include <stdio.h>
#include <conio.h>
void main()
{ float x1,x2,ktop,kcarp;
  clrscr();printf("1. kök: ");scanf("%f",&x1);
  printf("2. kök: ");scanf("%f",&x2); ktop=x1+x2; kcarp=x1*x2;
  if (ktop<0) printf("\nx^2+%0.3fx",-1*ktop);
  else printf("\nx^2-%0.3fx",ktop);
  if (kcarp<0) printf("%0.3f",kcarp); else printf("+%0.3f",kcarp);
  getch(); }
```

C++

```
// 2. dereceden polinom oluşturma
#include <iostream>
using namespace std;
int main()
{   float x1,x2,ktop,kcarp;
    system("cls"); cout << "1. kök: "; cin >> x1;
    cout << "2. kök: "; cin >> x2; ktop=x1+x2;kcarp=x1*x2;
    if (ktop<0) cout << "\nx^2+" << -1*ktop << "x";
    else cout << "\nx^2-" << ktop << "x";
    if (kcarp<0) cout << kcarp << endl;
    else cout << "+" << kcarp << endl;
    system("pause");return 0; }
```

C#

```
// İkinci dereceden polinom oluşturma
using System;
namespace ornek626
{ class Program
   {static void Main(string[] args)
    {double x1,x2,ktop,kcarp;
     Console.Write("1. kok: ");x1=Convert.ToDouble(Console.ReadLine());
     Console.Write("2. kok: ");x2=Convert.ToDouble(Console.ReadLine());
     ktop=x1+x2;kcarp=x1*x2;
     if (ktop<0) Console.Write("\nx^2+{0}x",(-1)*ktop);
     else Console.Write("\nx^2-{0}x",ktop);
     if (kcarp<0) Console.Write("{0}",kcarp);
     else Console.Write("+{0}",kcarp);  Console.ReadLine(); }    }   }
```

JAVA

```
// 2. dereceden polinom oluşturma
import java.util.Scanner;
public class ornek626 {
public static void main(String[] args)
{ float x1,x2,ktop,kcarp; Scanner klavye=new Scanner(System.in);
  System.out.print("1. kök: "); x1=klavye.nextFloat();
  System.out.print("2. kök: "); x2=klavye.nextFloat();
  ktop=x1+x2;kcarp=x1*x2;
  if (ktop<0) System.out.print("\nx^2+"+(-1)*ktop+"x");
  else System.out.print("\nx^2-"+ktop+"x");
  if (kcarp<0) System.out.print(kcarp);
  else System.out.print("+"+kcarp);     }}
```

MATLAB

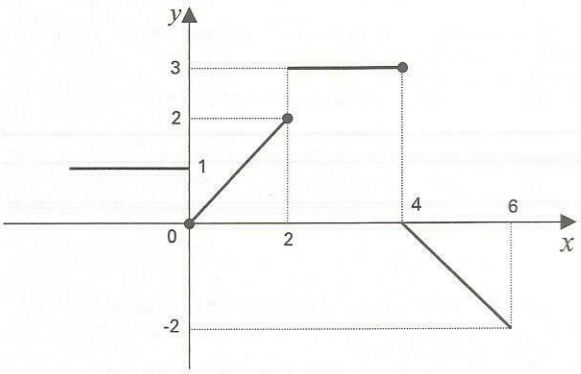
```
% 2. dereceden polinom oluşturma
clear all; clc
x1=input('1. kök: '); x2=input('2. kök: '); ktop=x1+x2; kcarp=x1*x2;
if (ktop<0)
    fprintf('\nx^2+%0.3fx',-1*ktop);
else
    fprintf('\nx^2-%0.3fx',ktop);
end
if (kcarp<0)
    fprintf('%0.3f\n',kcarp);
else
    fprintf('+%0.3f\n',kcarp);
end
```

PYTHON

```
# 2. dereceden polinom oluşturma
x1=eval(input("1. kök: ")); x2=eval(input("2. kök: "))
ktop=x1+x2; kcarp=x1*x2
if (ktop<0):
    print("\nx^2+%0.3fx"%-1*ktop,end='')
else:
    print("\nx^2-%0.3fx"%ktop,end='')
if (kcarp<0):
    print("%0.3f"%kcarp)
else:
    print("+%0.3f"%kcarp)
```

```
1. kök: -1
2. kök: 5

x^2-4.000x-5.000
```

Örnek-6.27: Bir $y = f(x)$ fonksiyonu, grafiksel olarak aşağıdaki şekilde verilmektedir. Buna göre klavyeden girilen x için y 'yi hesaplayan program.

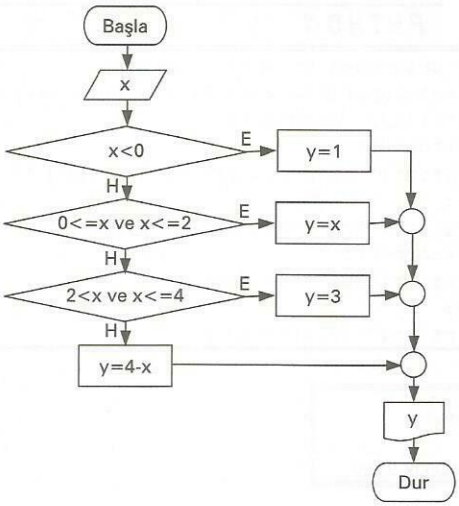


Verilen grafikte parçalı bir fonksiyon vardır. Öncelikle bu fonksiyonun matematiksel ifadesinin elde edilmesi gerekmektedir, yani hangi x değerlerinde hangi y değerleri elde edilmektedir. Dikkat edilirse; fonksiyon, dört parçadan oluşmaktadır:

- ✓ $x < 0$ olduğu yerlerde fonksiyonun değeri sabit olup 1'dir.
- ✓ $x \geq 0$ ve $x \leq 2$ olduğu aralıkta, fonksiyonun değerleri bir doğru denklemi şeklinde değişmektedir. (x_1, y_1) ve (x_2, y_2) gibi iki noktası bilinen bir doğrunun denklemi $\frac{x - x_1}{x_1 - x_2} = \frac{y - y_1}{y_1 - y_2}$ eşitliğiyle elde edilmektedir. Bu aralıktaki doğru da (0,0) ve (2,2) noktalarından geçtiğinden hesaplama sonucu doğru denklemi $y = x$ şeklinde bulunur.
- ✓ $x > 2$ ve $x \leq 4$ olduğu aralıkta, fonksiyonun değeri sabit olup 3 'e eşittir.
- ✓ $x > 4$ olması durumunda, fonksiyon değerleri yine doğru denklemi şeklinde değişmektedir. Bu bölgedeki doğrunun denklemi de $y = 4 - x$ 'tir.

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

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



C

```
/* Parçalı fonksiyon */
#include <stdio.h>
#include <conio.h>
main()
{ float x,y;clrscr(); printf("x noktasını giriniz: ");scanf("%f",&x);
  if (x<0) y=1; else if ((0<=x) && (x<=2)) y=x;
  else if ((2<x) && (x<=4)) y=3; else y=4-x;
  printf("Fonksiyonun x=%0.2f noktasındaki değeri=%0.2f",x,y);
  getch(); return 0; }
```

C++

```
/* Parçalı fonksiyon */
#include <iostream>
using namespace std;
int main()
{   float x,y;system("cls");cout<<"x noktasını giriniz: "; cin>>x;
    if(x<0) y=1; else if((0<=x) && (x<=2)) y=x;
    else if((2<=x) && (x<=4)) y=3; else y=4-x;
    cout<<"Fonksiyonun x="<<x<<" noktasındaki değeri= "<<y<<endl;
    system("pause");return 0; }
```

```
x noktasını giriniz: 5
Fonksiyonun x=5 noktasındaki değeri= -1
```

C#

```
// Parçalı fonksiyon
using System;
namespace ornek627
{ class Program
   { static void Main(string[] args)
    { float x,y; Console.Write("x noktasini giriniz: ");
      x=Convert.ToSingle(Console.ReadLine());
      if(x<0) y=1;  else if((0<=x) && (x<=2)) y=x;
      else if((2<x) && (x<=4)) y=3; else y=4-x;
  Console.Write("Fonksiyonun x={0:F2} noktasindaki degeri={1:F2}",x,y);
      Console.ReadLine();     }    }    }
```

JAVA

```
// Parçalı fonksiyon
import java.util.Scanner;
public class ornek627 {
public static void main(String[] args)
{  float x,y; Scanner klavye=new Scanner(System.in);
   System.out.print("x noktasını giriniz: "); x=klavye.nextFloat();
   if(x<0) y=1; else if((0<=x) && (x<=2)) y=x;
   else if((2<x) && (x<=4)) y=3; else y=4-x;
System.out.printf("Fonksiyonun x=%.2f noktasındaki değeri=%.2f",x,y);}}
```

MATLAB

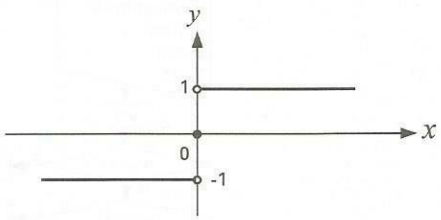
```
% Parçalı fonksiyon
clear all; clc ; x=input('x noktasını giriniz: ');
if (x<0) y=1;
elseif((0<=x) && (x<=2)) y=x;
elseif((2<x) && (x<=4)) y=3;
else y=4-x;
end; fprintf('Fonksiyonun x=%0.2f noktasındaki değeri=%0.2f\n',x,y);
```

PYTHON

```
# Parçalı fonksiyon
x=eval(input("x noktasını giriniz: "))
if (x<0):
    y=1
elif ((0<=x) & (x<=2)):
    y=x
elif ((2<x) & (x<=4)):
    y=3
else:
    y=4-x
print("Fonksiyonun x=%0.2f noktasındaki değeri=%0.2f\n"%(x,y))
```

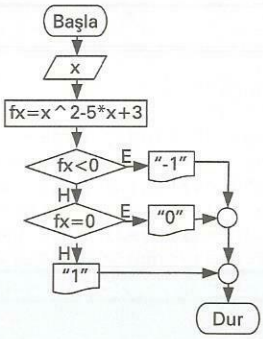
Örnek-6.28: *'İşaret (signum) fonksiyonu'*nun tanımı ve grafiksel gösterimi aşağıda verilmektedir. Tanıma göre klavyeden girilen x değeri için $f(x)=x^2-5x+3$ fonksiyonunun işaretini hesaplayan program.

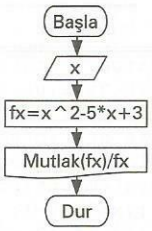
$$Sgn(f(x))=\begin{cases}-1 & , \quad f(x)<0\\ 0 & , \quad f(x)=0\\ 1 & , \quad f(x)>0\end{cases}$$



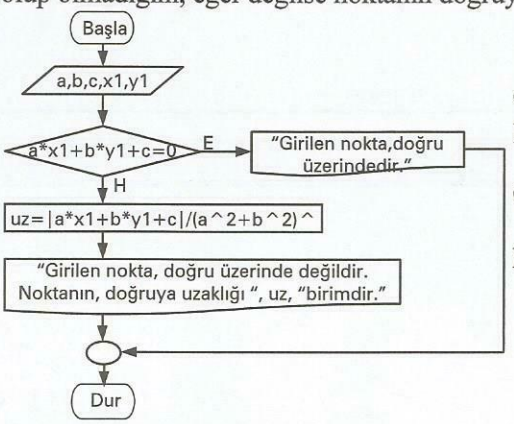
İşaret fonksiyonu; matematiksel olarak sonucun pozitif, negatif veya sıfır olup olmadığını verir. Programın yazımı için klavyeden girilen x değeri, $f(x)$ fonksiyonunda yerine yazılıp hesaplanır ve sonuç sıfır ile karşılaştırılır. Alternatif yol olarak da değerin mutlak değerinin, değere oranından da işaret elde edilebilir ($Sgn(f(x))=\dfrac{|f(x)|}{f(x)}$ veya $Sgn(f(x))=\dfrac{f(x)}{|f(x)|}$).

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



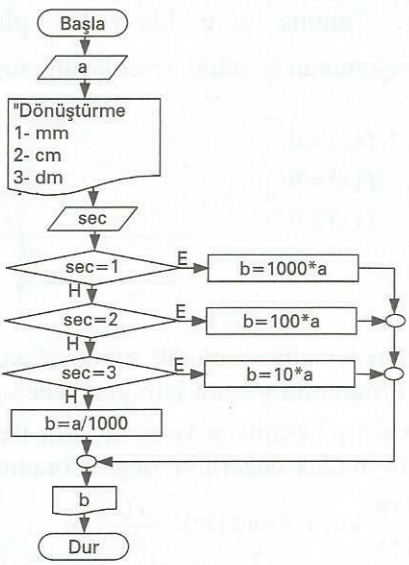


Örnek-6.29: Klavyeden girilen bir noktanın $ax+by+c=0$ doğrusu üzerinde olup olmadığını, eğer değilse noktanın doğruya olan uzaklığını veren program.



$P(x_1, y_1)$ noktası; doğru üzerinde ise $ax_1+by_1+c=0$ doğru denklemini sağlar, değilse de doğruya olan uzaklığı $uz=\dfrac{|ax_1+by_1+c|}{\sqrt{a^2+b^2}}$ ile hesaplanır.

Örnek-6.30: Klavyeden metre cinsinden girilen uzunluğu, seçilen birime dönüştüren program.



C

```
/* Birim donusturme */
#include <stdio.h>
#include <conio.h>
void main()
{ float a,b; int sec;
  clrscr(); printf("Uzunlugu giriniz (m): "); scanf("%f",&a);
  printf("\nDonusturme\n1- mm\n2- cm\n3- dm\n4- km\n");
  printf("\nSeciminiz: ");scanf("%d",&sec);
  switch (sec)
  { case 1: { b=1000*a; break; }
    case 2: { b=100*a; break; }
    case 3: { b=10*a; break; }
    case 4: { b=a/1000; break; }
    default: b=0; }  printf("\nSonuc: %0.5f\n",b); getch(); }
```

C++

```
// Birim donusturme
#include <iostream>
using namespace std;
int main()
{ float a,b; int sec;
  system("cls");cout<<"Uzunluğu giriniz (m): ";cin>>a;
  cout<<"\nDönüştürme\n1- mm\n2- cm\n3- dm\n4- km\n";
  cout<<"\nSeciminiz: "; cin>>sec;
  switch (sec)
  { case 1: { b=1000*a; break; }
    case 2: { b=100*a; break; }
    case 3: { b=10*a; break; }
    case 4: { b=a/1000; break; }
    default: b=0; }
  cout<<"\nSonuc: "<<b<<endl; system("pause"); return 0; }
```

C#

```
using System;
namespace ornek630
{class Program
  { static void Main(string[] args)
     { float b;
       Console.Clear(); Console.Write("Uzunlugu giriniz (m): ");
       float a = Convert.ToSingle(Console.ReadLine());
       Console.WriteLine("\nDönüştürme\n1- mm\n2- cm\n3- dm\n4- km\n");
       Console.Write("\nSeciminiz: ");
       int sec = Convert.ToInt32(Console.ReadLine());
       switch (sec)
         {  case 1: { b = 1000 * a; break; }
            case 2: { b = 100 * a; break; }
            case 3: { b = 10 * a; break; }
            case 4: { b = a / 1000; break; }
            default: { b = 0; break; }}
       Console.Write("Sonuc: {0:F5}", b); Console.ReadLine(); }}}
```

JAVA

```
// Birim donusturme
import java.util.Scanner;
public class ornek630 {
public static void main(String[] args)
{ float a,b; int sec;
  Scanner klavye=new Scanner(System.in);
  System.out.print("Uzunluğu giriniz (m): "); a=klavye.nextFloat();
  System.out.printf("\nDönüştürme\n1- mm\n2- cm\n3- dm\n4- km\n");
  System.out.print("\nSeciminiz: "); sec=klavye.nextByte();
  switch (sec)
  {  case 1: { b = 1000 * a; break; }
     case 2: { b = 100 * a; break; }
     case 3: { b = 10 * a; break; }
     case 4: { b = a / 1000; break; }
     default: { b = 0; break; }}
  System.out.printf("Sonuc: %.5f\n",b); }}
```

MATLAB

```
% Birim dönüştürme
clear all; clc; a=input('Uzunluğu giriniz (m): ');
fprintf('\nDönüştürme\n1- mm\n2- cm\n3- dm\n4- km\n\n');
sec=input('Seçiminiz: ');
switch (sec)
    case 1
        b=1000*a;
    case 2
        b=100*a;
    case 3
        b=10*a;
    case 4
        b=a/1000;
    otherwise
        b=0;
end; fprintf('\nSonuç: %0.5f\n',b);
```

PYTHON

```
# Birim dönüştürme
a=eval(input("Uzunluğu giriniz (m): "))
print("\nDönüştürme\n1- mm\n2- cm\n3- dm\n4- km\n")
sec=eval(input("Seçiminiz: "));
if (sec==1):
    b=1000*a
elif (sec==2):
    b=100*a
elif (sec==3):
    b=10*a
elif (sec==4):
    b=a/1000
else:
    b=0
print("\nSonuç: %0.5f"%b)
```

```
Uzunluğu giriniz (m): 5

Dönüştürme
1- mm
2- cm
3- dm
4- km

Seçiminiz: 2

Sonuç: 500.00000
```

Örnek-6.31: "*İdeal kilo*" (ideal beden/vücut ağırlığı) hesapları ve durumları değişik şekillerde yapılabilmektedir. Bu yöntemlerden bazıları aşağıda verilmektedir.

Beden yüzey alanı	$BYA = 0{,}20247.\{Boy\ (m)\}^{0{,}725}.\{Kilo\ (kg)\}^{0{,}425}$

Beden kütle/kitle indeksi	$BKI = \frac{Kilo\ (kg)}{Boy\ (m).Boy\ (m)}$

	Kadın	Erkek
Yöntem 1	$45{,}5 + 2{,}3.[\ Boy\ (inç) - 60\ ]$	$50{+}2{,}3.[\ Boy\ (inç) - 60\ ]$

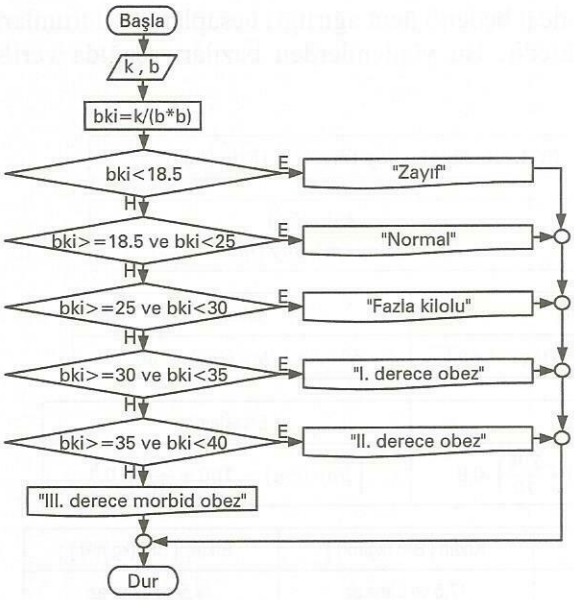
	Kadınlarda	Erkeklerde
Yöntem 2	$\left[Boy\ (cm) - 100 + \frac{yaş}{10}\right].0{,}8$	$\left[Boy\ (cm) - 100 + \frac{yaş}{10}\right].0{,}9$

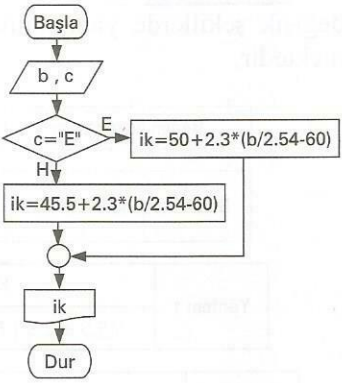
		Kadın [BKI (kg/m²)]	Erkek [BKI (kg/m²)]
Yöntem 3	Anoreksi, aşırı zayıf, yüksek risk	17,5 ve daha az	17,5 ve daha az
	Zayıf	17,5 – 19,1	17,5 – 20,7
	Normal kilolu	19,1 – 25,8	20,7 – 26,4
	Biraz fazla kilolu	25,8 – 27,3	26,4 – 27,8
	Fazla kilolu	27,3 – 32,3	27,8 – 31,1
	Çok fazla kilolu	32,3 - 34,9	31,1 - 34,9
	Sağlık açısından yüksek riskli kilolu	35 - 40	35 - 40
	Hastalıklı bir şekilde aşırı kilolu	40 - 50	40 - 50
	Süper aşırı kilolu	50 - 60	50 - 60

		BKI (kg/m²)
Yöntem 4	Zayıf	18,5 altında
	Normal	18,5 - 24,9
	Fazla kilolu	25 - 29,9
	I. derece obez	30 - 34,9
	II. derece obez	35 - 39,9
	III. derece morbid obez	40 ve üzerinde

	Kadın	Erkek
Yağsız beden ağırlığı	$1{,}07.Kilo(kg) - 148.\frac{\{Kilo(kg)\}^2}{\{100.Boy(m)\}^2}$	$1{,}1.Kilo(kg) - 128.\frac{\{Kilo(kg)\}^2}{\{100.Boy(m)\}^2}$

"Yöntem 4"e göre girilen kilo ve boy için ideal kiloya göre kişinin durumunu belirleyen programın akış diyagramıyla "Yöntem 1"e göre ideal kilo hesabı gerçekleştiren programın akış diyagramı ve kodları, aşağıda verilmektedir. Diğer yöntemlere göre ideal kilo hesabı ve durum tespiti yapan programların akış diyagramlarını çizip kodlayınız





C

```
/* İdeal kilo */
#include <stdio.h>
#include <conio.h>
void main()
{ float b,ik;  int c;
  clrscr(); printf("Boyunuzu giriniz (cm): "); scanf("%f",&b);
  printf("Cinsiyetiniz [1-Erkek / 2-Kadın]: "); scanf("%u",&c);
  if (c==1) ik=50+2.3*(b/2.54-60); else ik=45.5+2.3*(b/2.54-60);
  printf("\nİdeal kilonuz (kg): %0.2f\n",ik); getch(); }
```

C++

```
// İdeal kilo
#include <iostream>
using namespace std;
int main()
{   float b,ik; int c;
    system("cls"); cout << "Boyunuzu giriniz (cm): "; cin >> b;
    cout << "Cinsiyetiniz [1-Erkek / 2-Kadın]: "; cin >> c;
    if (c==1) ik=50+2.3*(b/2.54-60); else ik=45.5+2.3*(b/2.54-60);
    cout<<"\nİdeal kilonuz (kg): "<<ik<<endl;system("pause");return 0;}
```

C#

```
// İdeal kilo
using System;
namespace ornek631
{ class Program
   { static void Main(string[] args)
      { double b,ik;
        Console.Clear(); Console.Write("Boyunuzu giriniz (cm): ");
        b=Convert.ToDouble(Console.ReadLine());
        Console.Write("Cinsiyetiniz [1-Erkek / 2-Kadın]: ");
        int c=Convert.ToInt32(Console.ReadLine());
        if (c==1) ik=50+2.3*(b/2.54-60); else ik=45.5+2.3*(b/2.54-60);
        Console.Write("İdeal kilonuz (kg): {0:F2}", ik);
        Console.ReadLine();      }    }    }
```

JAVA

```
// İdeal kilo
import java.util.Scanner;
public class ornek631 {
public static void main(String[] args)
{ double b,ik; Scanner klavye=new Scanner(System.in);
  System.out.print("Boyunuzu giriniz (cm): "); b=klavye.nextDouble();
  System.out.print("Cinsiyetiniz [1-Erkek / 2-Kadın]: ");
  int c=klavye.nextInt();
  if (c==1) ik=50+2.3*(b/2.54-60); else ik=45.5+2.3*(b/2.54-60);
  System.out.printf("\nİdeal kilonuz (kg): %.2f\n",ik); }}
```

MATLAB

```
% İdeal kilo
clear all; clc ;
b=input('Boyunuzu giriniz (cm): ');
c=input('Cinsiyetiniz [1-Erkek / 2-Kadın]: ');
if (c==1)
    ik=50+2.3*(b/2.54-60);
else
    ik=45.5+2.3*(b/2.54-60);
end; fprintf('\nİdeal kilonuz (kg): %0.2f\n',ik);
```

```
Boyunuzu giriniz (cm): 177
Cinsiyetiniz [1-Erkek / 2-Kadın]: 1

İdeal kilonuz (kg): 72.28
```

PYTHON

```
# İdeal kilo
b=eval(input("Boyunuzu giriniz (cm): "))
c=int(input("Cinsiyetiniz [1-Erkek / 2-Kadın]: "))
if (c==1):
    ik=50+2.3*(b/2.54-60);
else:
    ik=45.5+2.3*(b/2.54-60);
print("\nİdeal kilonuz (kg): %0.2f"%ik);
```

Örnek-6.32: Bir kişinin bir gün (24 saat) içinde dinlenme durumunda iken (hiçbir aktivitede bulunmazken) tüm organları için gerekli olan enerji ihtiyacı, yani vücudun fizyolojik olaylarını (kalp atımı, nefes alma, beyin mesajların iletimi, vucüt ısısını dengeleme, vücut içindeki birçok kimyasal reaksiyonlar vb.) sürdürmesi (vücudun canlılığını sürdürebilmesi) için harcadığı toplam enerji; "*bazal metabolizma hızı*" (BMH) olarak tanımlanmaktadır. Normal koşullarda BMH, belirli bir süre içinde oksijen tüketimi temelli kalorimetre yardımıyla ölçülebilmektedir. Bunun yerine - gerçek sonuçlara yakın değerler verdiklerinden - bazı matematiksel hesaplama yöntemleri de kullanılmaktadır. Bunlardan bazıları aşağıda verilmektedir.

Harris-Benedict denklemi	Kadın	Erkek
	$655{,}0955 + 9{,}5634.Kilo(kg)$ $+1{,}8496.Boy(cm) - 4{,}6756.Yaş(yil)$	$66{,}473 + 13{,}7516.Kilo(kg)$ $+5{,}0033.Boy(cm) - 6{,}775.Yaş(yil)$

Schofield denklemi	Yaş	Kadın	Erkek
	10-17	$692{,}6 + 13{,}384.Kilo(kg)$	$658{,}2 + 17{,}686.Kilo(kg)$
	18-29	$486{,}6 + 14{,}818.Kilo(kg)$	$692{,}2 + 15{,}057.Kilo(kg)$
	30-59	$845{,}6 + 8{,}126.Kilo(kg)$	$873{,}1 + 11{,}472.Kilo(kg)$
	60-	$658{,}5 + 9{,}082.Kilo(kg)$	$587{,}7 + 11{,}711.Kilo(kg)$

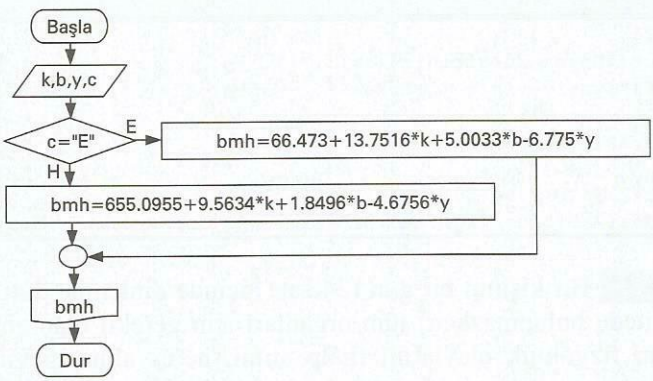
Owen denklemi	Kadın	Erkek
	$795 + 7{,}18.Kilo(kg)$	$879 + 10{,}2.Kilo(kg)$

Mifflin-St Jeor denklemi	Kadın	Erkek
	$9{,}99.Kilo(kg) + 6{,}25.Boy(cm)$ $-4{,}92.Yaş(yil) - 161$	$9{,}99.Kilo(kg) + 6{,}25.Boy(cm)$ $-4{,}92.Yaş(yil) + 5$

Dünya Sağlık Örgütü Pratik hesaplama formülü	Kadın	Erkek
	$22{,}8.Kilo(kg)$	$24.Kilo(kg)$

Dünya Sağlık Örgütü (WHO/FAO/UNU)	Yaş	Kadın	Erkek
	18-30	$496 + 14{,}7.Kilo(kg)$	$679 + 15{,}3.Kilo(kg)$
	31-60	$829 + 8{,}7.Kilo(kg)$	$879 + 11{,}6.Kilo(kg)$
	61-	$596 + 10{,}5.Kilo(kg)$	$487 + 13{,}5.Kilo(kg)$
	18-30	$35 + 13{,}3.Kilo(kg) + 334.Boy(m)$	$717 + 15{,}4.Kilo(kg) - 27.Boy(m)$
	31-60	$865 + 8{,}7.Kilo(kg) - 25.Boy(m)$	$901 + 11{,}3.Kilo(kg) + 16.Boy(m)$
	61-	$-302 + 9{,}2.Kilo(kg) + 637.Boy(m)$	$-1071 + 8{,}8.Kilo(kg) + 1128.Boy(m)$

Sıklıkla kullanılan Harris-Benedict denklemine göre çizilmiş akış diyagramı aşağıda verilmektedir. Diğer denklemlere göre akış diyagramları çizip kodlayınız.



Örnek-6.33: Klavyeden girilen yılın, "*artık yıl*" olup olmadığını test eden program.

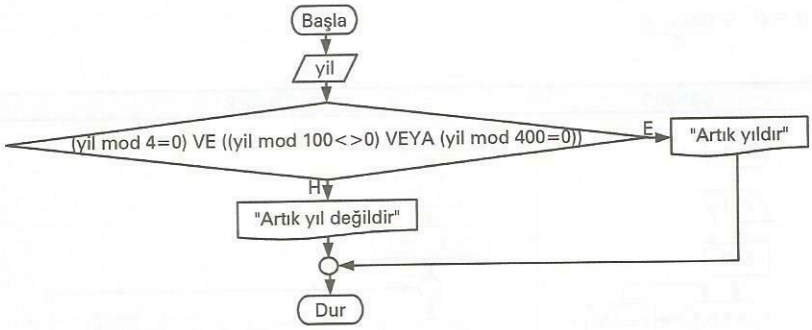
Bir yıl, 365 gün ve 6 saattir. 4 yılda 6 saatten 24 saat (1 gün) artmış olmaktadır. Dört yılda bir oluşan bu durumun nedeni de Dünya'nın Güneş etrafında dönme süresinin(astronomik yılın), günün (Güneş'in aynı meridyenden iki kez geçişi arasındaki ortalama sürenin) tam katı olmamasıdır (1 astronomik yıl =

365.242 gün). "Artık yıl"; Miladi (Gregoryen)) takvime göre yılın 366 gün, yani 29 şubatın da olduğu yıldır. Artık yıllar, 4'ün katı olan yıllardır. Ancak iki istisnai durum mevcuttur:

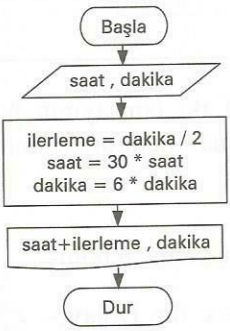
i. 100'ün katı olan yıllardan sadece 400'e tam bölünebilenler artık yıldır

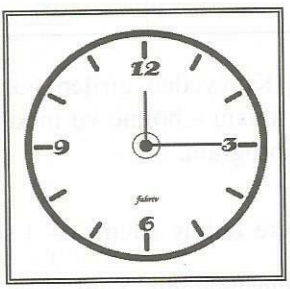
ii. 4000, 16000, 24000 vb. yıllar, 400'e tam bölünebilmesine rağmen artık yıl değildir.

Buna göre girilen yılın (yıl<4000), artık yıl olup olmadığını test eden program.

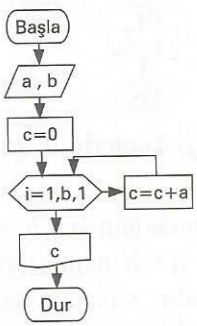


Örnek-6.34: Klavyeden ayrı ayrı girilen saat (0-11) ve dakika (0-59) bilgisine göre akrep ile yelkovanın gösterdikleri açıları hesaplayıp yazdıran program.





Örnek-6.35: Klavyeden girilen iki pozitif tamsayıyı - çarpma operatörü kullanmadan - çarpan program.



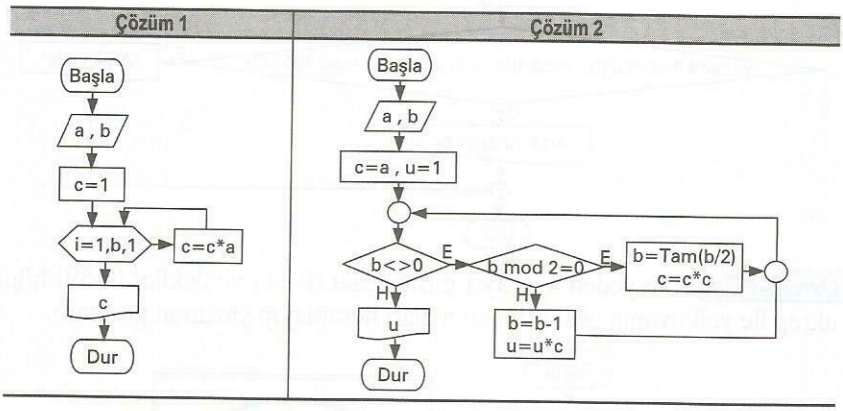
İki pozitif tamsayının çarpımı; birisinin kendisiyle, diğeri kadar toplamıdır.

$$c = a.b = \underbrace{a + a + \ldots + a}_{b}$$

Örnek-6.36: Klavyeden girilen bir tamsayının, yine klavyeden girilen pozitif tamsayı üssünü (kuvvetini) - üs alma operatörü veya hazır komut/fonksiyon kullanmadan - hesaplayan program.

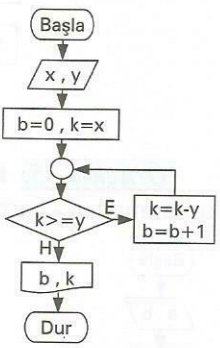
Herhangi bir sayının pozitif tamsayı üssü; sayının, kendisiyle belirtilen üs kadar çarpımıdır. Aşağıda iki alternatif çözüm/algoritma verilmektedir. Birinci algoritmada, döngünün ters açılması durumunda (`i=b,1,-1`) da sonucu hesaplayabileceğini gözden kaçırmayınız. Bu algoritmalardan hangisi daha hızlıdır veya daha az işlemle sonuca ulaşmaktadır?

$$u = a^b = \underbrace{a.a....a}_{b}$$



Örnek-6.37: Klavyeden girilen pozitif iki tamsayının bölünmesiyle elde edilen bölüm ve kalanı - bölme ve mod alma operatörleri/komutları kullanmadan - hesaplayan program.

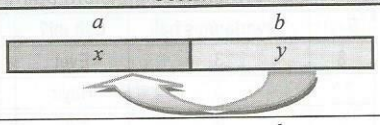
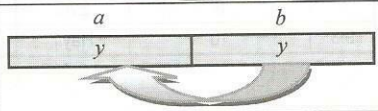
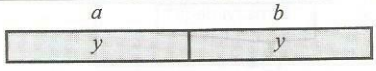
Bilindiği üzere bölme işlemi ($\frac{x}{y}$) sonucunda b bölümü ile k kalanı oluşmaktadır. Yani $x = b.y + k$ dır. Dolayısıyla kalan, bölenden büyük veya eşit olduğu sürece ($k \geq y$); kalan değeri bölen kadar azaltılır ($k = k - y$) ve bölüm birer arttırılırsa ($b = b + 1$) istenilen sonuca ulaşılır.



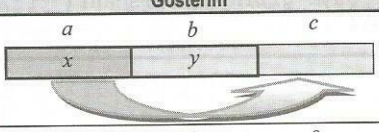
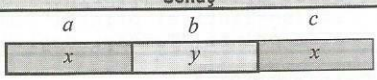
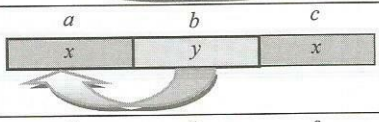
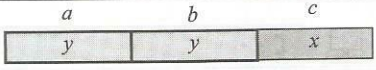
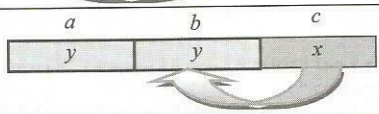
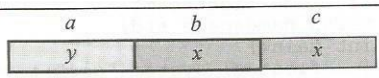
Örnek-6.38: Klavyeden girilen iki sayının aktarıldığı değişkenlerin içeriklerini karşılıklı olarak değiştiren program.

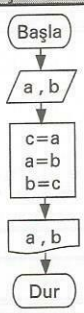
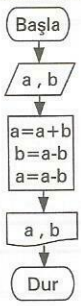
İki değişkenin (a, b) içeriklerini karşılıklı olarak değiştirmek için $a = b$ ve $b = a$ yazıldığında doğru sonuç elde edilmeyecektir. Çünkü $a = b$ atamasıyla a değişkeninin içeriği silinerek (üstüne yazılarak) b olmaktadır, yani kaybol-

maktadır. Dolayısıyla takip eden adımdaki $b = a$ atamasıyla aslında $b = b$ işlemi gerçekleşmektedir.

İşlem adımı	Gösterim	Sonuç
$a = b$		
$b = a$		

Çözüm için üçüncü bir c değişkeni kullanılabilir ve $a = b$ atamasından önce $c = a$ ile birinci değişkenin içeriği saklanabilir. Böylece işlem adımları $c = a \rightarrow a = b \rightarrow b = c$ sırasını takip ederek, değişkenlerin içeriklerini karşılıklı olarak değiştirir.

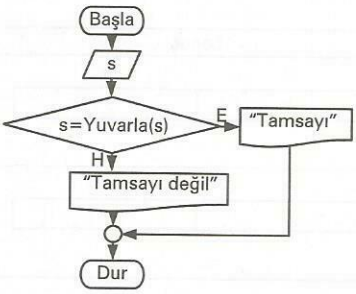
İşlem adımı	Gösterim	Sonuç
$c = a$		
$a = b$		
$b = c$		

Çözüm 1	Çözüm 2
	

Örnek-6.39: Klavyeden girilen pozitif bir sayının, tamsayı olup olmadığını test eden program.

Herhangi bir sayının, tamsayı olup olmadığını tespit etmenin en kolay yollarından birisi sayı ile sayının tamsayıya yuvarlanmış halini karşılaştırmaktır. Eğer ikisi de aynı ise sayı tamsayıdır (bknz. Tablo 6.2). Bir diğer yol da sayıyı

kelime gibi işleyip içinde nokta veya virgül karakterini aramaktır. Eğer yoksa tamsayı, varsa ondalıklı sayıdır.



Tablo 6.2: Örnek 6.39 için işlem adımı örnekleri

Sayı	Yuvarlanmış hali	Eşit mi?	Sonuç
3	3	Evet	Tamsayıdır
3.3	3	Hayır	Tamsayı değildir
9	9	Evet	Tamsayıdır
9.8	10	Hayır	Tamsayı değildir

C

```
/* Tamsayı testi */
#include <stdio.h>
#include <conio.h>
void main()
{ float s; clrscr();printf("Bir sayı giriniz: ");scanf("%f",&s);
  if (s==(int) s) printf("Tamsayı");
  else printf("Tamsayı değil");          getch();    }
```

```
Bir sayı giriniz: 99
Tamsayı
```

C++

```
// Tamsayı testi
#include <iostream>
using namespace std;
int main()
{   float s; system("cls"); cout << "Bir sayı giriniz: "; cin >> s;
    if (s==(int) s) cout << "Tamsayı";
    else cout << "Tamsayı değil";      system("pause");return 0; }
```

C#

```
// Tamsayı testi
using System;
namespace ornek639
{ class Program
   { static void Main(string[] args)
      { Console.Write("Bir sayi giriniz: ");
        double s=Convert.ToDouble(Console.ReadLine());
        if (s==(int)s) Console.WriteLine("Tamsayı");
        else Console.WriteLine("Tamsayı değil"); Console.ReadLine();}}}
```

JAVA

```
// Tamsayı testi
import java.util.Scanner;
public class ornek639 {
public static void main(String[] args)
{  float s; Scanner klavye=new Scanner(System.in);
   System.out.print("Bir sayı giriniz: "); s=klavye.nextFloat();
   if (s==(int)s) System.out.print("Tamsayı");
   else System.out.print("Tamsayı değil"); }}
```

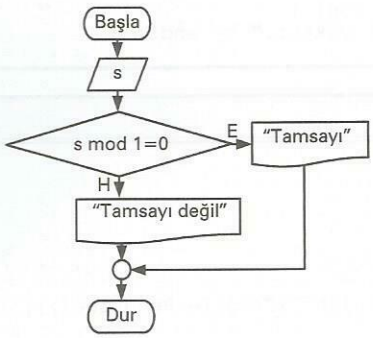
MATLAB

```
% Tamsayı testi
clear all; clc ; s=input('Bir sayı giriniz: ');
if (s==round(s))
    fprintf('Tamsayı\n');
else
    fprintf('Tamsayı değil\n');
end
```

PYTHON

```
# Tamsayı testi
s=eval(input("Bir sayı giriniz: "))
if (s==round(s)):
    print("Tamsayı")
else:
    print("Tamsayı değil")
```

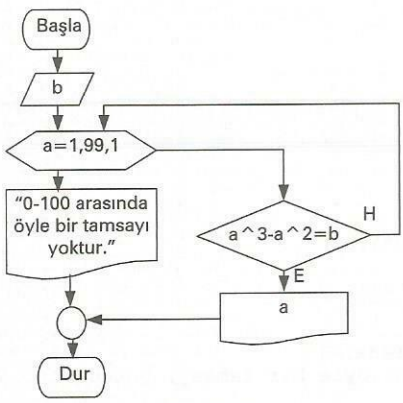
Alternatif diğer bir yol da mod işlemi kullanmaktır. Herhangi bir pozitif sayının, 1'e göre modu 0 ise tamsayıdır (bknz. Tablo 6.3).



Tablo 6.3: Örnek 6.39 için işlem adımı örnekleri

Sayı	Sayı mod 1	=0	Sonuç
3	0	Evet	Tamsayıdır
3.3	0.3	Hayır	Tamsayı değildir
9	0	Evet	Tamsayıdır
9.8	0.8	Hayır	Tamsayı değildir

Örnek-6.40: Klavyeden girilen b tamsayısına göre $a^3 - a^2 = b$ şartını sağlayan $0 < a < 100$ tamsayısını bulan program.



Eğer problemlerin (denklemlerin) belirli bir çözüm formülasyonu (örneğin birinci veya ikinci dereceden denklemlerdeki gibi) yoksa; bilgisayarların hızlı işlem yapma özelliğinden faydalanılarak, tek tek seçenekleri deneme yoluna gidilebilir. Bu nedenle örnek problemin çözümü için yürütülebilecek en basit mantık, a değişkenine 0-100 arasında sırasıyla değer vermek ve her değerde $a^3 - a^2 = b$ şartını kontrol etmektir.

C

```
/* Denklem 3 */
#include <stdio.h>
#include <conio.h>
int a,b;
main()
{ clrscr();printf("a^3-a^2=b\n");
  printf("b= ");scanf("%d",&b);printf("\n");
  for(a=1;a<=99;a++)
    if ((a*a*a-a*a)==b) { printf("a= %u",a);goto son; }
  printf("0-100 arasında öyle bir tamsayı yoktur.");
  son: getch(); return 0; }
```

C++

```
// Denklem 3
#include <iostream>
using namespace std;
int b;
int main()
{   system("cls"); cout << "a^3-a^2=b" << endl;
    cout << "b= "; cin >> b; cout<<"\n";
    for(int a=1;a<=99;a++)
       if((a*a*a-a*a)==b)
          { cout << "a= " << a << endl; goto son; }
    cout << "0-100 arasında öyle bir tamsayı yoktur." << endl;
    son: system("pause");return 0;  }
```

C#

```
// Denklem3
using System;
namespace ornek640
{ class Program
   { static void Main(string[] args)
        {   int b;
            Console.WriteLine("a^3-a^2=b");
            Console.Write("b= ");b=Convert.ToInt32(Console.ReadLine());
            for(int a=1;a<=99;a++)
                if((a*a*a-a*a)==b)
                  { Console.Write("\na={0}",a); goto son; }
            Console.Write("0-100 arasında öyle bir tamsayı yoktur");
son:        Console.ReadLine();    }    }    }
```

```
a^3-a^2=b
b= 18

a= 3
```

JAVA

```
// Denklem 3
import java.util.Scanner;
public class ornek640 {
public static void main(String[] args)
{ int b,k=0; System.out.println("a^3-a^2=b");
  Scanner klavye=new Scanner(System.in);
  System.out.print("b= "); b=klavye.nextInt();
  for(int a=1;a<=99;a++)
   if((a*a*a-a*a)==b)
    { System.out.println("\na= "+a);k=1;break; }
if(k==0) System.out.println("0-100 arasında öyle bir tamsayı yoktur.");
   }}
```

MATLAB

```
% Denklem 3
clear all; clc ; k=0; fprintf('a^3-a^2=b\n'); b=input('b= ');
for a=1:99
    if (a^3-a^2==b)
        fprintf('\na=%d\n',a); k=1; break;
    end
end
if (k==0)
    fprintf('0-100 arasında öyle bir tamsayı yoktur\n');
end
```

PYTHON

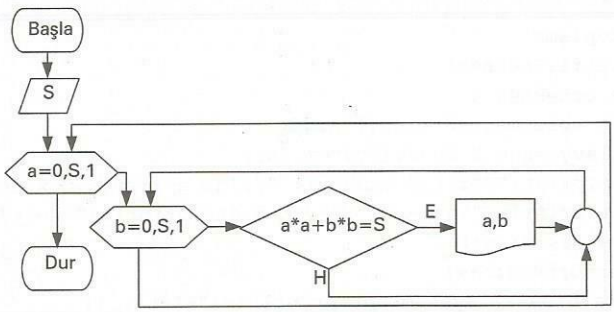
```
# Denklem 3
print("a^3-a^2=b"); k=0; b=eval(input("b= "))
for a in range(1,100):
    if (a**3-a**2==b):
        print("\na=%d"%a); k=1; break
if (k==0):
    print("0-100 arasında öyle bir tamsayı yoktur")
```

Not

Bilgisayar biliminde; problemin çözümüne aday tüm değerleri üretip test etme (deneme) işlemi "brute-force search" veya "exhaustive search" olarak adlandırılır.

Örnek-6.41: Girilen pozitif bir tamsayının, iki sayının kareleri toplamı şeklinde yazılıp yazılamayacağını hesaplayan program.

Problemin çözümü için iki tane döngü açılır ve döngü değişkenlerinin kareleri toplanarak girilen pozitif tamsayıya eşit olup olmadığı kontrol edilir.



C

```
/* Kareler toplamı */
#include <stdio.h>
#include <conio.h>
main()
{ int a,b,s; clrscr();printf("Pozitif tamsayı: ");scanf("%u",&s);
  printf("\na - b\n");printf("=====\n");
  for(a=0;a<=s;a++)
   for(b=0;b<=s;b++)
    if (a*a+b*b==s) printf("%u-%u\n",a,b); getch();return 0; }
```

C++

```
// Kareler toplamı
#include <iostream>
using namespace std;
int main()
{  int s; system("cls"); cout << "Pozitif tamsayı: "; cin >> s;
   cout << "\na  -  b\n"; cout << "=====\n";
   for(int a=0; a<=s;a++)
      for(int b=0; b<=s;b++)
         if(a*a+b*b==s) cout << a <<" - " << b << endl;
  system("pause");return 0; }
```

C#

```
// Kareler toplamı
using System;
namespace ornek641
{ class Program
   { static void Main(string[] args)
      { Console.Write("Pozitif tamsayi: ");
        int s=Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("\na - b ");Console.WriteLine("=====");
        for(int a=0;a<=s;a++)
      for(int b=0;b<=s;b++)
        if(a*a+b*b==s) Console.WriteLine("{0}-{1}",a,b);
        Console.ReadLine();      }   }   }
```

JAVA

```
// Kareler toplamı
import java.util.Scanner;
public class ornek641 {
public static void main(String[] args)
{  Scanner klavye=new Scanner(System.in);
   System.out.print("Pozitif tamsayı: ");int s=klavye.nextInt();
   System.out.println("\na - b ");System.out.println("=====");
   for(int a=0;a<=s;a++)
    for(int b=0;b<=s;b++)
       if(a*a+b*b==s) System.out.println(a+"-"+b); }}
```

MATLAB

```
% Kareler toplamı
clear all; clc ; s=input('Pozitif tamsayı: ');
fprintf('a - b \n=====\n');
for a=0:s
    for b=0:s
        if (a^2+b^2==s) fprintf('%d - %d\n',a,b); end; end; end
```

PYTHON

```
# Kareler toplamı
s=eval(input("Pozitif tamsayı: "))
print("\na - b\n=====")
for a in range(0,s+1):
    for b in range(0,s+1):
        if (a**2+b**2==s):
            print("%d-%d"%(a,b))
```

```
Pozitif tamsayı: 100

a - b
=====
0-10
6-8
8-6
10-0
```

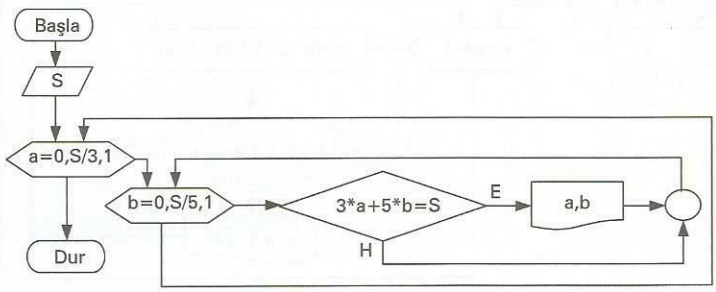
Yukarıdaki akış diyagramı ve kodlar, problemin çözümünü gerçekleştirmektedirler. Ancak bilgisayar, fazladan (gereksiz) birçok denemeler yapmakta ve bu durum işlem hızını düşürmektedir. $a^2+b^2=s$ denkleminde a veya b nin alabileceği en büyük değer $\sqrt{s}$ (diğerine en küçük değer olan sıfır verildiğinde) dir. Örnek çıktıdaki gibi 100 tamsayısı için a ve b döngüleri 101'er defa dönmekte, böylece toplamda 101x101=10201 çevrim yapmaktadır. Ancak bu döngüler $\sqrt{100}=10$ 'a kadar açılmış olsaydı, aynı sonuca 11x11=121 çevrimde ulaşılacaktı. Dolayısıyla yukarıdaki akış diyagramı ve kodlarda, döngülerin bitiş değerlerinin $\sqrt{s}$ olması daha uygundur.

Not

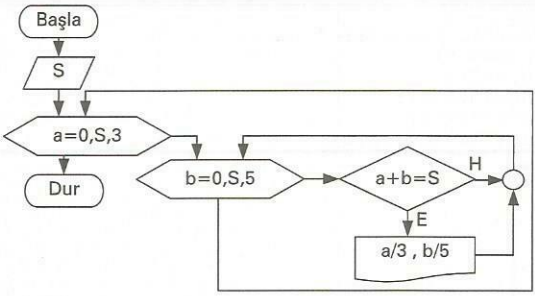
Kodlarınızdaki döngü sınırlarını en uygun (optimum) seçerek, programınızın hızını arttırabilirsiniz (işlem sayısını azaltabilirsiniz, cevap verme süresini düşürebilirsiniz).

Örnek-6.42: 7'den büyük tamsayılar ($S>7$), 3 ve 5'in katlarının toplamı ($3.a+5.b=S$) olarak yazılabilir. Girilen tamsayı için 3 (a) ve 5'in (b) katlarını bulan program.

Bu problemin çözümünde a ile b için birer döngü açılır ve eşitlik koşulu kontrol edilir. Bir önceki örnekten hatırlanacağı gibi işlemleri azaltmak, dolayısıyla programı hızlandırmak için a döngüsü $S/3$ ve b döngüsü de $S/5$ 'e kadar açılmalıdır.



Alternatif bir çözüm olarak da aşağıdaki akış diyagramı çizilebilir.



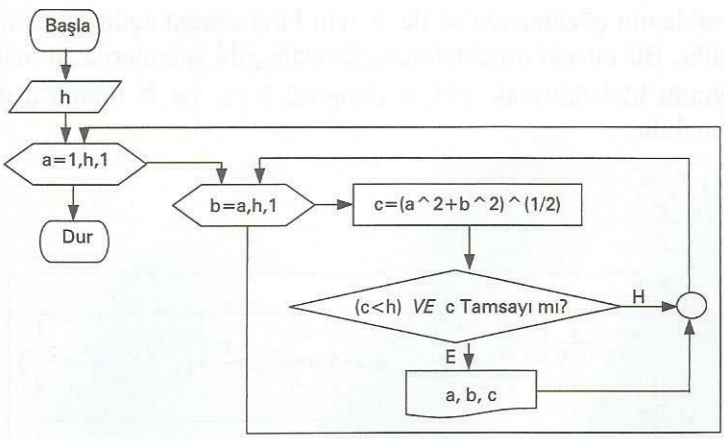
Örnek-6.43: Klavyeden girilen hipotenüs uzunluğuna göre tüm kenarları tamsayı ve hipotenüsü belirtilen değerden küçük tüm dik üçgenleri bulan program.

Dik üçgende; hipotenüsün uzunluğu, dik kenarların karelerinin toplamının karekökünе eşittir. Programın üreteceği sonuçlar iki koşulu sağlamalıdır:

- Dik üçgeni tanımlayan $c = \sqrt{a^2 + b^2}$ Pisagor bağıntısını,
- Tüm kenarların tamsayı olmasını.

Bunun için dik kenarlara tamsayı değerler verilerek koşullar kontrol edilir. Yani a ve b döngüleri açılarak $c = \sqrt{a^2 + b^2}$ hesaplatılır ve yukarıdaki iki koşulu sağlayıp sağlamadığı kontrol edilir. Burada dikkat edilmesi gereken bir nokta da, program; örneğin 3-4-5 dik üçgenini bulup ekrana yazdırdığında sonraki adımlarda 4-3-5 aynı üçgeni yazdırmamasıdır. Bu da ikinci (iç) döngü (yani b), birinci (dış) döngünün o anki değeriyle başlatılarak sağlanır. Döngülerin bitişi olarak da istenilen hipotenüs uzunluğu verilebilir.

$$\left.\begin{matrix} a = h \\ b = h \end{matrix}\right\} \Rightarrow c = \sqrt{a^2 + b^2} = \sqrt{h^2 + h^2} = \sqrt{2h^2} = h\sqrt{2} > h$$



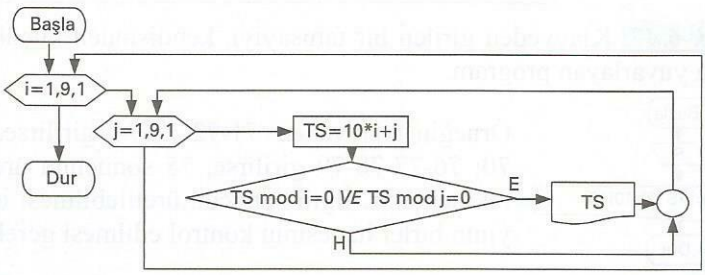
Örneğin hipotenüsü 6'dan küçük dik üçgenleri bulan programın çalışması Tablo 6.4'teki gibi olacaktır.

Tablo 6.4: Örnek 6.43 için uygulama sonuçları

a	b	$c=\sqrt{a^2+b^2}$	*c tamsayı mı?*	$c<6$?	Ekrana yaz
1	1	*1,414214*	*Hayır*	*Evet*	-
1	2	*2,236068*	*Hayır*	*Evet*	-
1	3	*3,162278*	*Hayır*	*Evet*	-
1	4	*4,123106*	*Hayır*	*Evet*	-
1	5	*5,09902*	*Hayır*	*Evet*	-
1	6	*6,082763*	*Hayır*	*Evet*	-
2	2	*2,828427*	*Hayır*	*Evet*	-
2	3	*3,605551*	*Hayır*	*Evet*	-
2	4	*4,472136*	*Hayır*	*Evet*	-
2	5	*5,385165*	*Hayır*	*Evet*	-
2	6	*6,324555*	*Hayır*	*Hayır*	-
3	3	*4,242641*	*Hayır*	*Evet*	-
3	**4**	***5***	***Evet***	***Evet***	**3-4-5**
3	5	*5,830952*	*Hayır*	*Evet*	-
3	6	*6,708204*	*Hayır*	*Hayır*	-
4	4	*5,656854*	*Hayır*	*Evet*	-
4	5	*6,403124*	*Hayır*	*Hayır*	-
4	6	*7,211103*	*Hayır*	*Hayır*	-
5	5	*7,071068*	*Hayır*	*Hayır*	-
5	6	*7,81025*	*Hayır*	*Hayır*	-
6	6	*8,485281*	*Hayır*	*Hayır*	-

Örnek-6.44: Kendi hanelerine tam bölünen 11-99 arasındaki tamsayıları listeleyen program.

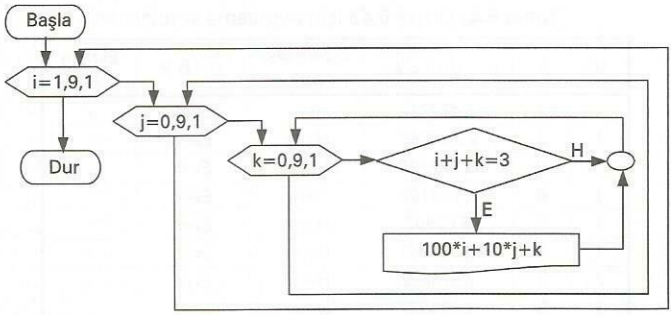
10-99 arasındaki iki haneli tamsayıları oluşturmak için iki döngü (i ve j) gereklidir. Açılan iki döngünün oluşturacağı tamsayı $10i+j$ olup bu sayının hem i hem de j ye tam bölünmesi gerekir.



Örnek-6.45: Basamaklarının toplamı, basamak sayısına eşit olan 100-999 arasındaki tamsayıları bulan program.

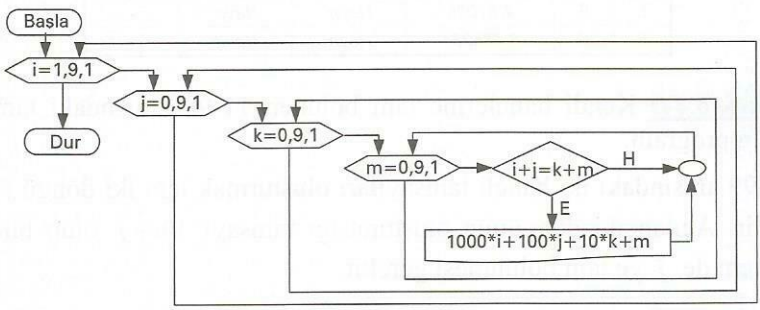
Örneğin 102 tamsayısında; basamaklarının toplamı 1+0+2=3 tür. Program, bu tür tamsayıları üretmelidir. Üç haneli tamsayıları oluşturmak için üç döngü (

i, j, k) gerekmekte olup oluşan sayı $100i+10j+k$ dır. İstenen tamsayıların seçilmesi için de $i+j+k=3$ koşulu ile yapılır.

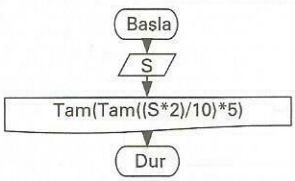


Örnek-6.46: İlk N basamağındaki rakamların toplamı, son N basamağındaki rakamların toplamına eşit olan $2N$ basamaklı tamsayılar elde edilmeye çalışılmaktadır. Örneğin 1230 tamsayısı; 4 basamaktan oluşmakta ve ilk 2 basamağının toplamı (1+2=3), son iki basamağının toplamına (3+0) eşittir. Bu özelliğe sahip 1000-9999 arasındaki tamsayıların listesini veren program.

1000-9999 arasındaki sayılar gerektiğinden, 4 döngü (i, j, k, m) ile $1000i+100j+10k+m$ değerindeki tamsayılar oluşturulur. Üretilen bu sayılardan istenilenleri elemek için $i+j=k+m$ koşulu kullanılır.



Örnek-6.47: Klavyeden girilen bir tamsayıyı, kendisinden küçük en yakın 5'in katına yuvarlayan program.



Örneğin klavyeden 71-72-73-74 girilirse, program 70; 76-77-78-79 girilirse, 75 sonucunu üretmesi gerekmektedir. İlgili sonucun üretilebilmesi için tamsayının birler hanesinin kontrol edilmesi gerekir.

C

```
/* 5 in katı */
#include <stdio.h>
#include <conio.h>
main()
{ int s; clrscr();printf("Tamsayı: ");scanf("%d",&s);
  printf("\n%d",(int) (((int) ((s*2)/10))*5));  getch();return 0; }
```

C++

```
// 5'in kati
#include <iostream>
using namespace std;
int main()
{   int s; system("CLS"); cout << "Tamsayı: "; cin >> s;
    cout << "\n" << (int) (((int) ((s*2)/10))*5) << endl;
    system("PAUSE");return 0; }
```

C#

```
// 5'in katı
using System;
namespace ornek647
{ class Program
   { static void Main(string[] args)
     { Console.Write("Tamsayı: ");
       int s = Convert.ToInt32(Console.ReadLine());
       Console.WriteLine("{0}",(int)(((int)((s*2)/10))*5));
       Console.ReadLine();      }      }      }
```

```
Tamsayı: 77

75
```

JAVA

```
// 5'in katı
import java.util.Scanner;
public class ornek647 {
public static void main(String[] args)
{  int s; Scanner klavye=new Scanner(System.in);
   System.out.print("Tamsayı: ");s=klavye.nextInt();
   System.out.println((int)(((int)((s*2)/10))*5)); }}
```

MATLAB

```
% 5'in katı
clear all; clc ; s=input('Tamsayı: ');
fprintf('%d\n',floor(floor((s*2)/10)*5));
```

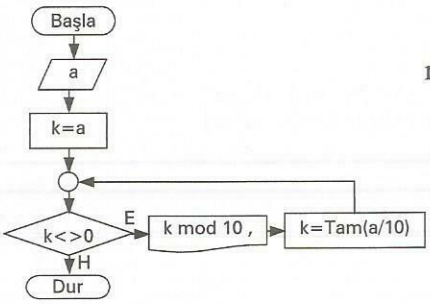
PYTHON

```
# 5'in katı
import math
s=eval(input("Tamsayı: ")); print(math.floor(math.floor((s*2)/10)*5));
```

Örnek-6.48: Klavyeden girilen pozitif tamsayıyı tersten yazdıran program.

Herhangi bir tamsayının birler basamağını elde etmek için kullanılabilecek en basit yol, 10'a göre mod işlemi uygulamaktır. Bu problemde de birler basamağı mod işlemiyle elde edilip sayının basamak sayısının azaltılması gerekmektedir. Basamak sayısını kademeli olarak indirgemek için de 10'a bölme işlemi uygulanıp kalan kısım atılabilir. Böylece sayı, 0 oluncaya kadar

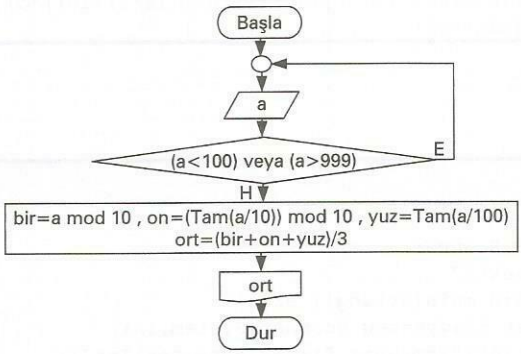
mod işlemiyle birler basamağı yazdırılır ve bölme işlemiyle de bir basamak indirgenir.



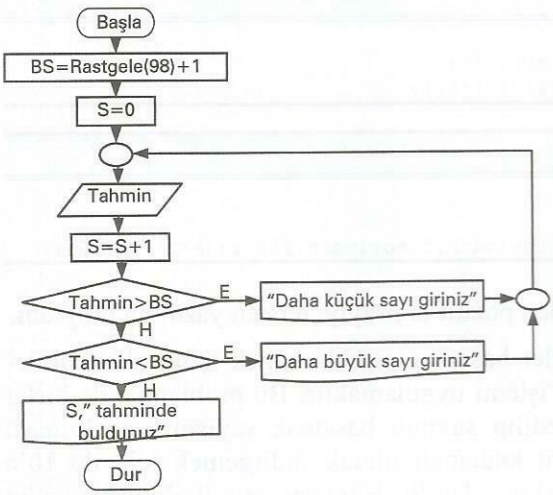
Örneğin 125 tamsayısını tersten yazdırırken;

- 125 mod 10 = *5* ve Tam(125/10) = 12
- 12 mod 10 = *2* ve Tam(12/10) = 1
- 1 mod 10 = *1* Tam(1/10) = 0

Örnek-6.49: Klavyeden girilen üç basamaklı pozitif tamsayının rakamlarının (basamaklarının) aritmetik ortalamasını hesaplayıp ekrana yazdıran program.



Örnek-6.50: Bilgisayarın ürettiği 1-99 arası bir tamsayının tahmin edilme oyunu ("*sayı bulma oyunu*") programı.



Programda öncelikle belirtilen sınırlar arasında rastgele tamsayı (BS) üretilip bir değişkende tutulur. Kullanıcı tahmin ettiği tamsayıyı (Tahmin) girdiğinde, bilgisayarın ürettiği tamsayı ile karşılaştırılır: Eğer büyük tamsayı girildiyse, daha küçük; küçük tamsayı girildiyse daha büyük girilmesi gerektiği mesajı yazdırılarak çevrim, tamsayı bulununcaya kadar devam eder.

C

```
/* Sayı tahmin oyunu */
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int BS,Tahmin,s=0;
main()
{  clrscr();randomize();BS=random(99)+1;
   do{ printf("Tahmininiz: ");scanf("%u",&Tahmin);s++;
      if (Tahmin>BS) printf("Daha küçük sayı giriniz...\n");
      if (Tahmin<BS) printf("Daha büyük sayı giriniz...\n");
      printf("\n");
    } while(Tahmin!=BS);
   printf("%u tahminde buldunuz...",s); getch();return 0; }
```

C++

```
// Sayi tahmin oyunu
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int BS,Tahmin,s=0;
int main()
{   system("cls"); srand(time(NULL)); BS=rand()%99+1;
    do { cout << "Tahmininiz: "; cin >> Tahmin; s++;
         if(Tahmin>BS) cout << "Daha küçük sayı giriniz...\n";
         if(Tahmin<BS) cout << "Daha büyük sayı giriniz...\n";
         cout << "\n";
       } while(Tahmin!=BS);
    cout << s <<" tahminde buldunuz \n"; system("pause");return 0; }
```

C#

```
// Sayı tahmin oyunu
using System;
namespace ornek650
{ class Program
    { static void Main(string[] args)
       { int BS,Tahmin,s=0;
         Random rnd=new Random(); BS=rnd.Next(1,100);
      do { Console.Write("Tahminimiz: ");
           Tahmin=Convert.ToInt32(Console.ReadLine());s++;
       if(Tahmin>BS) Console.WriteLine("Daha küçük sayı giriniz... ");
       if(Tahmin<BS) Console.WriteLine("Daha büyük sayı giriniz... ");
           Console.WriteLine("");
       }while(Tahmin!=BS);
Console.WriteLine("{0} tahminde buldunuz... ",s);Console.ReadLine();}}}
```

JAVA

```
// Sayı tahmin oyunu
import java.util.Scanner;
public class ornek650 {
public static void main(String[] args)
{  int BS,Tahmin,s=0;
   Scanner klavye=new Scanner(System.in); BS=(int)(Math.random()*98)+1;
   do{ System.out.print("Tahminimiz: "); Tahmin=klavye.nextInt(); s++;
   if(Tahmin>BS) System.out.println("Daha küçük sayı giriniz... ");
   if(Tahmin<BS) System.out.println("Daha büyük sayı giriniz... ");
   System.out.println("");
   }while(Tahmin!=BS);System.out.println(s+" tahminde buldunuz...");}}
```

```
Tahmininiz: 90
Daha küçük sayı giriniz...

Tahmininiz: 77

2 tahminde buldunuz...
```

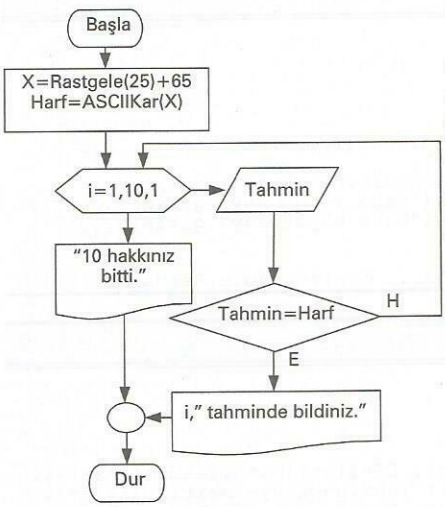
MATLAB

```
% Sayı bulma oyunu
clear all; clc ; tahmin=-1; s=0; bs=floor(98*rand+1);
while (tahmin~=bs)
    tahmin=input('Tahmininiz: '); s=s+1;
    if (tahmin>bs) fprintf('Daha küçük sayı giriniz...\n\n');end;
    if (tahmin<bs) fprintf('Daha büyük sayı giriniz...\n\n');end;
end
fprintf('\n%d tahminde buldunuz...\n',s);
```

PYTHON

```
# Sayı bulma oyunu
import random
tahmin=-1; s=0; bs=random.randint(1,99)
while (tahmin!=bs):
    tahmin=int(input("Tahmininiz: "))
    s=s+1
    if (tahmin>bs):
        print("Daha küçük sayı giriniz...\n")
    if (tahmin<bs):
        print("Daha büyük sayı giriniz...\n")
print("\n%d tahminde buldunuz..."%s)
```

Örnek-6.51: Bilgisayarın ürettiği rastgele büyük harfin, en fazla 10 denemede tahmin edilme oyunu programı.



Bilgisayarın rastgele büyük harf oluşturabilmesi için 65-90 arası (sınırlar dâhil) rastgele bir tamsayı üretmesi gerekir. Çünkü "A" harfinin ASCII kodu 65 ve "Z" harfinin ASCII kodu 90'dır. Bu aralıkta üretilen tamsayı, karaktere (harfe) dönüştürülür ve 1-10 (en fazla 10 deneme hakkı olduğu için) arası bir döngü içinde girilen harfle karşılaştırılır.

Örnek-6.52: $Cos(x)$ fonksiyonu, Maclaurin serisine aşağıdaki gibi açılmaktadır. Buna göre klavyeden girilen x değerinin kosinüs'ünü belirtilen terim sayısına kadar seriye açarak hesaplayan program.

$$Cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \ldots$$

Türetilebilir herhangi bir $f(x)$ fonksiyonu a noktası civarında

$$f(x)=f(a)+\frac{f'(a)}{1!}(x-a)+\ldots+\frac{f^{(n)}(a)}{n!}(x-a)^n+\ldots=\sum_{k=0}^{\infty}\frac{f^{(k)}(a)}{k!}(x-a)^k$$

şeklinde "*Taylor serisi*"ne açılabilmektedir. $a=0$ için

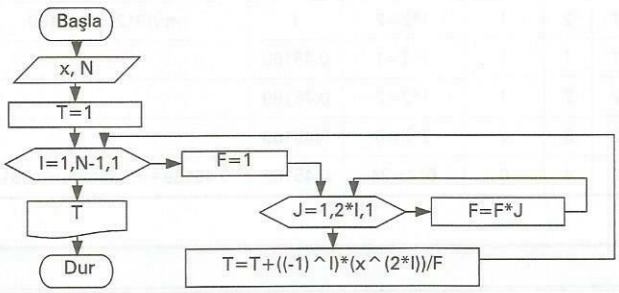
$$f(x)=f(0)+\frac{f'(0)}{1!}x+\frac{f''(0)}{2!}x^2+\ldots+\frac{f^{(n)}(0)}{n!}x^n+\ldots=\sum_{k=0}^{\infty}\frac{f^{(k)}(0)}{k!}x^k$$

"*Maclaurin serisi*" elde edilir. Böylece fonksiyon değerleri, sadece dört temel işlem (toplama, çıkarma, çarpma ve bölme) kullanılarak hesaplanabilmektedir.

Seri açılımı şeklinde verilen ifadelerde, öncelikle genel teriminin bulunması gerekmektedir. Yani seri elemanlarının değişimi gözlemlenip genel bir ifade ile tanımlanması gerekir. Seri terimlerinin pay kısmında; açının, düzgün sırada artan çift kuvvetleri, paydada da aynı kuvvetlerin faktöriyelleri yer almaktadır. Ayrıca terimler ardışıl olarak toplanırken işaret, her adımda artı-eksi olarak değişmektedir. Buna göre seri açılımının genel ifadesi

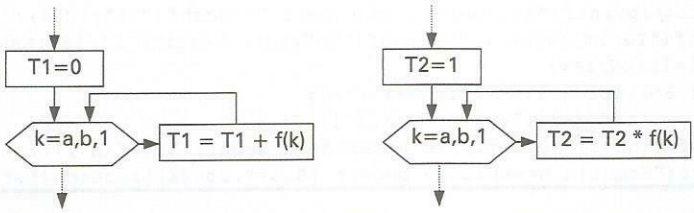
$$Cos(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-\ldots=\sum_{k=0}^{\infty}\frac{(-1)^k.x^{2k}}{(2k)!}$$

şeklinde ortaya çıkar.



Not

$T_1=\sum_{k=a}^{b}f(k)$ veya $T_2=\prod_{k=a}^{b}f(k)$ şeklindeki seri açılımlarının akış diyagramı olarak karşılıkları aşağıdaki gibidir:



Not

k değişkenine bağlı olarak tek-çift-tüm sayıları gösterme/oluşturma:

- ✓ Tek sayılar için: $2k+1$ veya $2k-1$
- ✓ Çift sayılar için: $2k$
- ✓ Bir önceki sayı: $k-1$
- ✓ Şu anki sayı: k
- ✓ Bir sonraki sayı: $k+1$

Not

Ardışık işaret değişimi $(-1)^k$ ile veya döngü içinde (-1) ile çarpılarak sağlanır.

Not

Trigonometrik seri açılımlarındaki açı değeri, radyan cinsindendir.

Örneğin $Cos(60^\circ)$ hesaplamak için $x=\pi/3$ ve $N=3$ alınırsa programın çalışması Tablo 6.5'deki gibi olacaktır.

Tablo 6.5: Örnek-6.52 için çalıştırma adımları

I	J	Eski F	Yeni F	Eski T	Yeni T
1	1	1	1*1=1	1	---
1	2	1	1*2=2	1	$1-\pi^2/(9*2)=0.45169$
1	1	1	1*1=1	0.45169	---
2	2	1	1*2=2	0.45169	---
2	3	2	2*3=6	0.45169	---
2	4	6	6*4=24	0.45169	$0.45169+\pi^4/(81*24)=0.5018$

C

```
/* Cosinus */
#include <stdio.h>
#include <conio.h>
#include <math.h>
int i,j,N,is; float aci,X,F,T=1;
main()
{ clrscr();printf("Açı değeri (derece): ");scanf("%f",&aci);
  printf("Terim sayısı: ");scanf("%u",&N); X=aci*M_PI/180;is=1;
  for(i=1;i<N;i++)
      { F=1;for(j=1;j<=2*i;j++) F*=j;
          is*=(-1);T+=is*pow(X,2*i)/F; }
  printf("\nSeri açılımı ile hesaplanan değer: %0.5f\n",T);
  printf("Komutla hesaplanan değer: %0.5f",cos(X)); getch();return 0; }
```

C++

```
// Cosinus
#include <iostream>
#include <math.h>
using namespace std;
int N,is; float aci,X,F,T=1;
int main()
{   system("cls");cout << "Açı değeri (derece): "; cin >> aci;
    cout << "Terim sayısı: "; cin >> N; X=aci*M_PI/180; is=1;
    for(int i=1;i<N;i++)
      { F=1; for(int j=1;j<=2*i;j++) F*=j;
        is*=(-1); T+=is*pow(X,2*i)/F; }
    cout<<"\nSeri acilimi ile hesaplanan deger: "<<T<<endl;
    cout << "Komutla hesaplanan deger: " << cos(X) << endl;
    system("pause");return 0; }
```

C#

```
// Cosinus
using System;
namespace ornek652
{ class Program
  { static void Main(string[] args)
    { int N,s; double aci,X,F,T=1;
      Console.Write("Açı değeri (derece): ");
      aci=Convert.ToDouble(Console.ReadLine());
      Console.Write("Terim sayısı: ");
      N=Convert.ToInt32(Console.ReadLine()); X=Math.PI*aci/180.0;s=1;
      for(int i=1;i<=N-1;i++)
    { F=1;for(int j=1;j<=2*i;j++) F*=j;
      s*=(-1);T+=s*Math.Pow(X, 2*i)/F;}
    Console.WriteLine("\nSeri açılımı ile hesaplanan değer: {0:F5}",T);
    Console.WriteLine("Komutla hesaplanan değer: {0:F5}",Math.Cos(X));
      Console.ReadLine();      } } }
```

JAVA

```
// Cosinus
import java.util.Scanner;
public class ornek652 {
public static void main(String[] args)
{  int N,is; double aci,X,F,T=1;
   Scanner klavye=new Scanner(System.in);
   System.out.print("Açı değeri (derece): ");aci=klavye.nextDouble();
   System.out.print("Terim sayısı: ");N=klavye.nextInt();
   X=Math.toRadians(aci);is=1;
   for(int i=1;i<=N-1;i++)
       { F=1;for(int j=1;j<=2*i;j++) F*=j;
         is*=(-1);T+=is*Math.pow(X, 2*i)/F;}
   System.out.printf("\nSeri açılımı ile hesaplanan değer: %.5f\n",T);
   System.out.printf("Komutla hesaplanan değer: %.5f\n",Math.cos(X));}}
```

MATLAB

```
% Cosinus
clear all; clc ; T=1; aci=input('Açı değeri (derece): ');
x=degtorad(aci); N=input('Terim sayısı: ');
for i=1:N-1
    T=T+(-1)^i*x^(2*i)/factorial(2*i);
End; fprintf('\nSeri açılımı ile hesaplanan değer: %0.5f\n',T);
fprintf('Komutla hesaplanan değer: %0.5f\n',cosd(aci));
```

PYTHON

```
# Cosinus
import math
aci=eval(input("Açı değeri (derece): "))
N=int(input("Terim sayısı: "))
T=1; x=math.radians(aci)
for i in range(1,N):
   T+=(-1)**i*x**(2*i)/math.factorial(2*i)
print("\nSeri açılımı ile hesaplanan değer: %0.5f"%T)
print("Komutla hesaplanan değer: %0.5f"%math.cos(x))
```

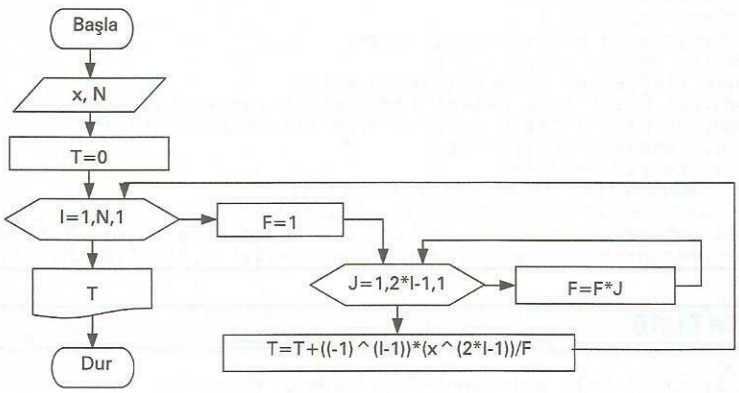
```
Açı değeri (derece): 30
Terim sayısı: 10

Seri açılımı ile hesaplanan değer: 0.86603
Komutla hesaplanan değer: 0.86603
```

Örnek-6.53: $Sin(x)$ fonksiyonu, Maclaurin serisine aşağıdaki gibi açılmaktadır. Buna göre klavyeden girilen x açısının sinüs değerini, belirtilen terim sayısına kadar açarak hesaplayan program.

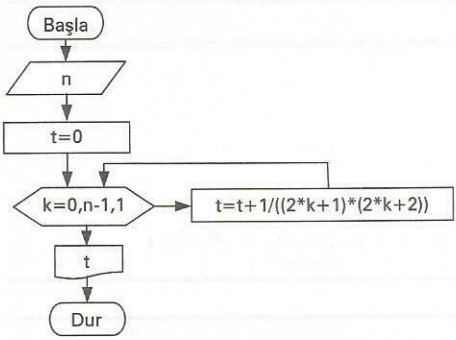
$$Sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \ldots$$

Bir önceki örnekte olduğu gibi burada da benzer değişimler görülmektedir: pay kısmında açının tek kuvvetleri, paydada bu tek sayıların faktöriyelleri yer almakta ve terimler artı-eksi şeklinde gitmektedir.

$$Sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \ldots = \sum_{k=1}^{\infty} \frac{(-1)^{k-1}.x^{2k-1}}{(2k-1)!}$$



Örnek-6.54: $Ln(2) = \frac{1}{2} + \frac{1}{12} + \frac{1}{30} + \frac{1}{56} + \frac{1}{90} + \ldots$ şeklinde seriye açılarak yaklaşık olarak hesaplanabilmektedir. Girilen terim sayısına göre seri açılımıyla $Ln(2)$ değerini hesaplayıp yazdıran program.



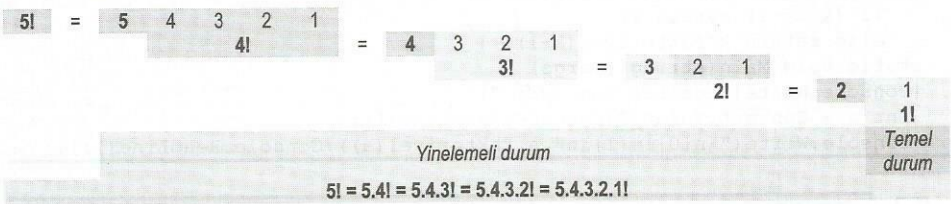
Seri toplamına dikkat edilirse, paylar sürekli 1 ve paydalar da ardışık sayıların (2=1.2, 12=3.4, 30=5.6, vb.) çarpımından oluşmaktadır.

Örnek-6.55: Faktoriyel işlemini "yinelemeli (rekürsif) fonksiyon"la gerçekleştiren program.

En genel anlamda yinelemeli (kendi kendini çağıran) fonksiyonlar iki kısımdan oluşur:

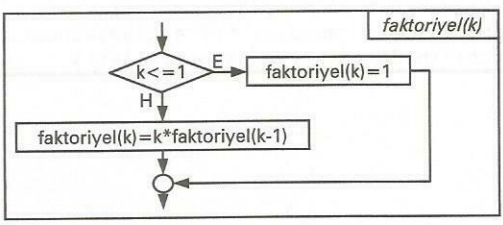
- ✓ *Temel durum*: Problemin en basit çözümüdür.
- ✓ *Genel (yineleme) durum*: Temel (basit) çözüme yaklaştıran yineleme çözümüdür.

Matematiksel olarak 0!=1 ve 1!=1 olduğunu bilinmektedir. Yani faktoriyeli hesaplanacak sayı, 1'e eşit veya küçükse çözüm daima 1'dir. İşte bu, temel durumu belirtmektedir. 2'nin faktoriyeli hesaplanırken 2!=2.1! işlemi gerçekleştrilmektedir. Yani temel durumdaki çözüm ile 2 çarpılmaktadır. Benzer şekilde 3!=3.2!=3.2.1! veya 4!=4.3!=4.3.2!=4.3.2.1! dir.



Görüldüğü gibi her aşamada, fonksiyon kendi kendini çağırarak temel duruma ulaşmaktadır. O zaman fonksiyon tanımlaması şu şekilde yapılabilir:

$$faktoriyel(k) = \begin{cases} sonuç = 1 & , \ k \le 1 \rightarrow \textit{Temel durum} \\ sonuç = k * faktoriyel(k-1) & , \ k > 1 \rightarrow \textit{Genel durum} \end{cases}$$



C

```
/* Faktoriyel */
#include <stdio.h>
#include <conio.h>
int faktoriyel(int k)
{ if (k<=1) return 1;
  else return k*faktoriyel(k-1); }
void main()
{ int a; clrscr();printf("Pozitif tamsayı: "); scanf("%u",&a);
  printf("\n%u!=%u\n",a,faktoriyel(a));       getch();    }
```

C++

```
// Faktoriyel
#include <iostream>
using namespace std;
int faktoriyel(int k)
{   if (k<=1) return 1;
    else return k*faktoriyel(k-1);  }
int main()
{   int a; system("cls"); cout << "Pozitif tamsayı: "; cin >> a;
    cout<<"\n"<<a<<"!="<<faktoriyel(a)<<"\n";
    system("pause");return 0; }
```

C#

```
// Faktoriyel
using System;
namespace ornek655
{class Program
 {public static int faktoriyel(int k)
    if (k <= 1) return 1;
    else return k*faktoriyel(k-1); }
 static void Main(string[] args)
 {Console.Write("Pozitif tamsayı: ");
  int a = Convert.ToInt32(Console.ReadLine());
  Console.Write("\n{0}!={1}\n",a,faktoriyel(a));Console.ReadLine();}}}
```

JAVA

```
// Faktoriyel
import java.util.Scanner;
public class ornek655 {
public static int faktoriyel(int k){
  if (k<=1) return 1;
  else return k*faktoriyel(k-1); }
public static void main(String[] args) {
  int a; Scanner klavye=new Scanner(System.in);
  System.out.print("Pozitif tamsayı: "); a=klavye.nextInt();
  System.out.printf("\n%d!=%d\n",a,faktoriyel(a));     }}
```

```
Pozitif tamsayı: 6

6!=720
```

MATLAB

```
function f=aktoriyel(k)
if (k<=1)
    f=1;
else
    f=k*faktoriyel(k-1);
end
```

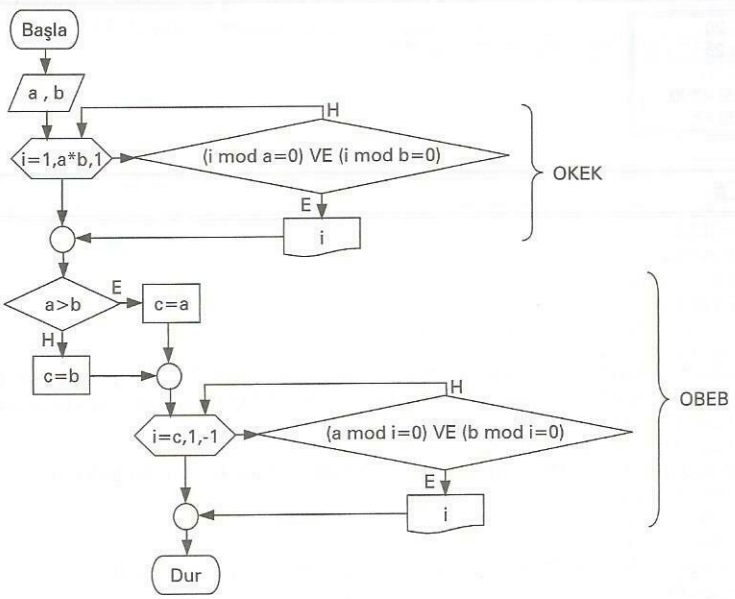
```
% Faktoriyel
clear all; clc
a=input('Pozitif tamsayı: ');fprintf('\n%d!=%d\n',a,faktoriyel(a));
```

PYTHON

```
# Faktoriyel
def faktoriyel(k):
   if (k<=1):
      return 1
   else:
      return k*faktoriyel(k-1)
a=int(input("Pozitif tamsayı: ")); print("\n%d!=%d"%(a,faktoriyel(a)))
```

Örnek-6.56: Klavyeden girilen iki pozitif tamsayının ortak katlarının en küçüğü (OKEK) ile ortak bölenlerinin en büyüğünü (OBEB) hesaplayan program.

İki tamsayının OKEK değeri en fazla bu ikisinin çarpımı, OBEB'i de bu tamsayılardan büyük olanının değeri kadar olabilir. Bu nedenle OKEK için birden iki tamsayının çarpımına kadar; OBEB için de önce iki tamsayıdan büyük olan bulunarak birden büyük tamsayıya kadar döngüler açılır ve tam bölünebilme koşulları kontrol edilerek istenilen sonuçlara ulaşılabilir.



C

```
/* OKEK-OBEB */
#include <stdio.h>
#include <conio.h>
void main()
{ int a,b,c,i;
  clrscr(); printf("1. tamsayı: "); scanf("%d",&a);
  printf("2. tamsayı: "); scanf("%d",&b);
  for(i=1;i<=a*b;i++)
       if ((i%a==0) && (i%b==0))
       {      printf("\nOKEK(%d,%d)=%d\n",a,b,i); break; }
  (a>b) ? c=a:c=b;
  for(i=c;i>0;i--)
       if ((a%i==0) && (b%i==0))
       {      printf("OBEB(%d,%d)=%d\n",a,b,i); break; }  getch();}
```

C++

```
// OKEK-OBEB
#include <iostream>
using namespace std;
int main()
{   int a,b,c,i;
    system("cls"); cout << "1. tamsayı: "; cin >> a;
    cout << "2. tamsayı: "; cin >> b;
    for(i=1;i<=a*b;i++)
       if ((i%a==0) && (i%b==0))
       {      cout<<"\nOKEK("<<a<<","<<b<<")="<<i<<"\n"; break; }
  (a>b) ? c=a:c=b;
  for(i=c;i>0;i--)
       if ((a%i==0) && (b%i==0))
       {      cout<<"OBEB("<<a<<","<<b<<")="<<i<<"\n"; break; }
    system("pause");return 0; }
```

```
1. tamsayı: 20
2. tamsayı: 25

OKEK(20,25)=100
OBEB(20,25)=5
```

C#

```
// OKEK-OBEB
using System;
namespace ornek656
{class Program
 {static void Main(string[] args)
  {int a, b, c, i;
   Console.Write("1. tasmayı: ");a=Convert.ToInt32(Console.ReadLine());
   Console.Write("2. tasmayı: ");b=Convert.ToInt32(Console.ReadLine());
   for (i=1;i<=a*b;i++)
    if ((i%a==0) && (i%b==0))
     { Console.Write("\nOKEK({0},{1})={2}\n",a,b,i); break; }
   c=(a>b) ? a:b;
   for (i=c;i>0;i--)
    if ((a%i==0) && (b%i==0))
     { Console.Write("OBEB({0},{1})={2}\n",a,b,i); break; }
   Console.ReadLine();      }    }   }
```

JAVA

```
// OKEK-OBEB
import java.util.Scanner;
public class ornek656 {
public static void main(String[] args) {
 int a,b,c,i;
 Scanner klavye=new Scanner(System.in);
 System.out.print("1. tamsayı: "); a=klavye.nextInt();
 System.out.print("2. tamsayı: "); b=klavye.nextInt();
 for (i=1;i<=a*b;i++)
    if ((i%a==0) && (i%b==0))
     { System.out.printf("\nOKEK(%d,%d)=%d\n",a,b,i); break; }
   c=(a>b) ? a:b;
   for (i=c;i>0;i--)
    if ((a%i==0) && (b%i==0))
     { System.out.printf("OBEB(%d,%d)=%d\n",a,b,i); break; } }}
```

MATLAB

```
% OKEK-OBEB
clear all; clc ;
a=input('1. tamsayı: '); b=input('2. tamsayı: ');
for i=1:a*b
  if ((mod(i,a)==0) && (mod(i,b)==0))
    fprintf('\nOKEK(%d,%d)=%d\n',a,b,i); break; end;end;
c=max([a b]);
for i=c:-1:1
  if ((mod(a,i)==0) && (mod(b,i)==0))
    fprintf('OBEB(%d,%d)=%d\n',a,b,i); break; end;end
--- Hazır komut/fonksiyonlarla ---
% OKEK-OBEB
clear all; clc ; a=input('1. tamsayı: '); b=input('2. tamsayı: ');
fprintf('\nOKEK(%d,%d)=%d\n',a,b,lcm(a,b));
fprintf('OBEB(%d,%d)=%d\n',a,b,gcd(a,b));
```

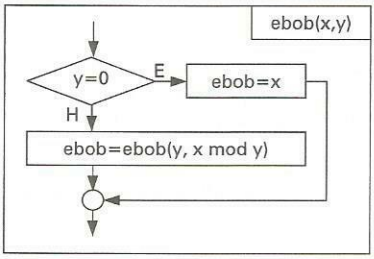
PYTHON

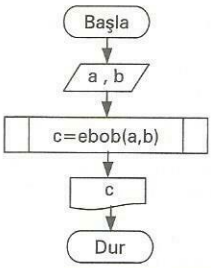
```
# OKEK-OBEB
a=int(input("1. tamsayı: ")); b=int(input("2. tamsayı: "))
for i in range(1,a*b+1):
   if ((i%a==0) & (i%b==0)):
      print("\nOKEK(%d,%d)=%d\n"%(a,b,i))
      break
c=max(a,b)
for i in range(c,1,-1):
   if ((a%i==0) & (b%i==0)):
      print("OBEB(%d,%d)=%d\n"%(a,b,i))
      break
--- --- Hazır komut/fonksiyonlarla ---
OBEB için: fractions.gcd(a,b)
```

Örnek-6.57: İki tamsayının en büyük ortak bölenini (EBOB) (ortak bölenlerin en büyüğü - OBEB) hesaplamakta kullanılan *Öklid algoritması*, $x \geq y$ olmak üzere

$$ebob(x,y) = \begin{cases} x & , \ y = 0 \\ ebob(y, kalan(x,y)) & , \ diğer \end{cases}$$

ünlü yinelemeli fonksiyonuyla tanımlanmaktadır. Buna göre klavyeden girilen iki tamsayının en büyük ortak bölenini, "ebob" yinelemeli fonksiyonuyla hesaplayıp ekrana yazdıran program.





C

```
/* EBOB */
#include <stdio.h>
#include <conio.h>
int ebob(int x,int y)
{ if (y==0) return x; else return ebob(y,x%y); }
void main()
{  int a,b; clrscr();printf("1. tamsayı: ");scanf("%u",&a);
   printf("2. tamsayı: ");scanf("%u",&b);
   printf("\nEBOB(%u,%u)=%u\n",a,b,ebob(a,b));   getch(); }
```

C++

```
// EBOB
#include <iostream>
using namespace std;
int ebob(int x,int y)
{ if (y==0) return x; else return ebob(y,x%y); }
int main()
{   int a,b; system("cls"); cout << "1. tamsayı: "; cin >> a;
    cout << "2. tamsayı: "; cin >> b;
    cout<<"\n"<<"EBOB("<<a<<","<<b<<")="<<ebob(a,b)<<endl;
    system("pause");return 0; }
```

C#

```
// EBOB
using System;
namespace ornek657
{ class Program
  { public static int ebob(int x, int y)
    { if (y == 0) return x; else return ebob(y, x % y); }
     static void Main(string[] args)
      { int a,b;
        Console.Write("1. tamsayı: ");
        a=Convert.ToInt32(Console.ReadLine());
        Console.Write("2. tamsayı: ");
        b=Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("\nEBOB({0},{1})={2}",a,b,ebob(a,b));
        Console.ReadLine(); }}}
```

```
1. tamsayı: 99
2. tamsayı: 77

EBOB(99,77)=11
```

JAVA

```
// EBOB
import java.util.Scanner;
public class ornek657 {
public static int ebob(int x, int y){
  if (y == 0) return x; else return ebob(y, x % y); }
public static void main(String[] args)
{  int a,b; Scanner klavye=new Scanner(System.in);
   System.out.print("1. tamsayı: "); a=klavye.nextInt();
   System.out.print("2. tamsayı: "); b=klavye.nextInt();
   System.out.printf("\nEBOB(%d,%d)=%d\n",a,b,ebob(a,b)); }}
```

MATLAB

```
function s=ebob(x,y)
if (y==0)
  s=x;
else
  s=ebob(y,mod(x,y));
end
```

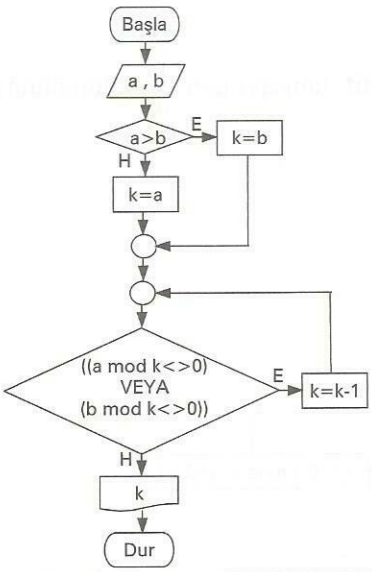
```
% EBOB
clear all; clc;
a=input('1. tamsayı: '); b=input('2. tamsayı: ');
fprintf('\nEBOB(%d,%d)=%d\n',a,b,ebob(a,b));
```

PYTHON

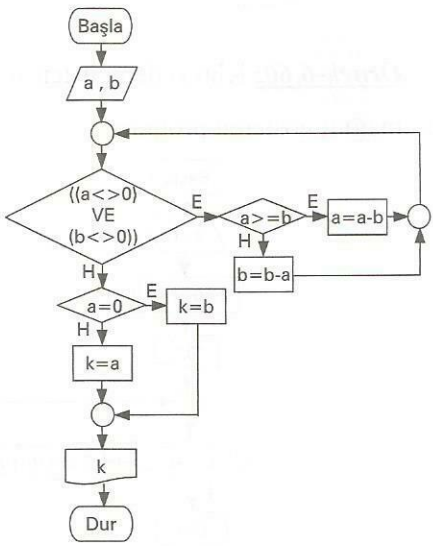
```
# EBOB
def ebob(x,y):
   if (y==0):
      return x
   else:
      return ebob(y,x%y)
a=int(input("1. tamsayı: ")); b=int(input("2. tamsayı: "))
print("\nEBOB(%d,%d)=%d"%(a,b,ebob(a,b)))
```

Alternatif çözümler

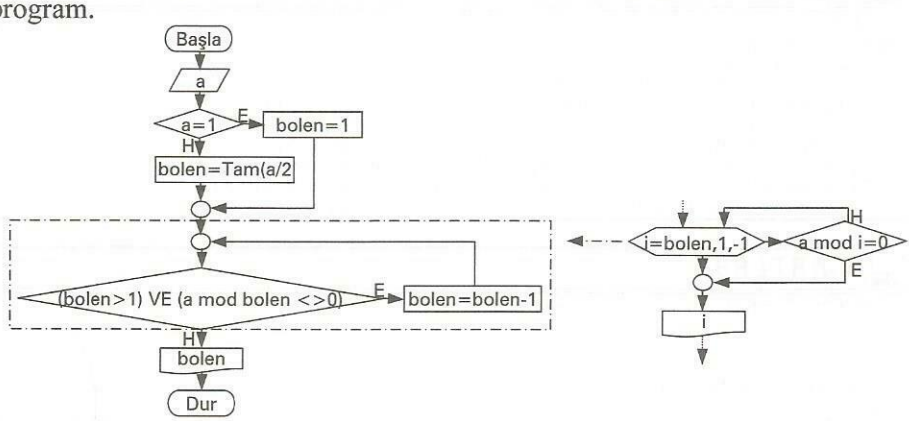
Çözüm 1



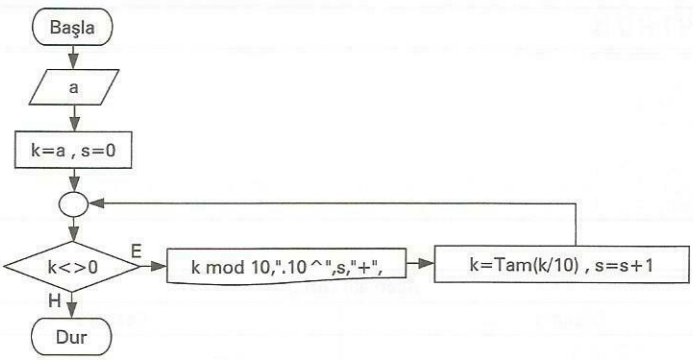
Çözüm 2



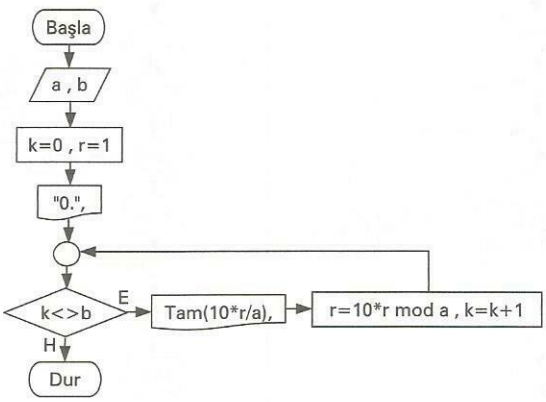
Örnek-6.58: Klavyeden girilen pozitif tamsayının en büyük bölenini bulan program.



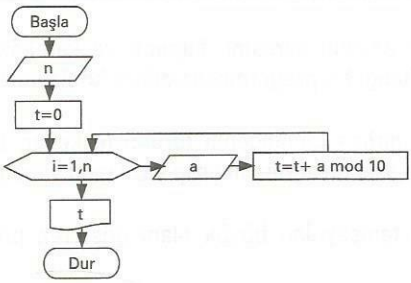
Örnek-6.59: Klavyeden girilen pozitif tamsayının, 10 tabanındaki gösterimini ekrana yazdıran program.



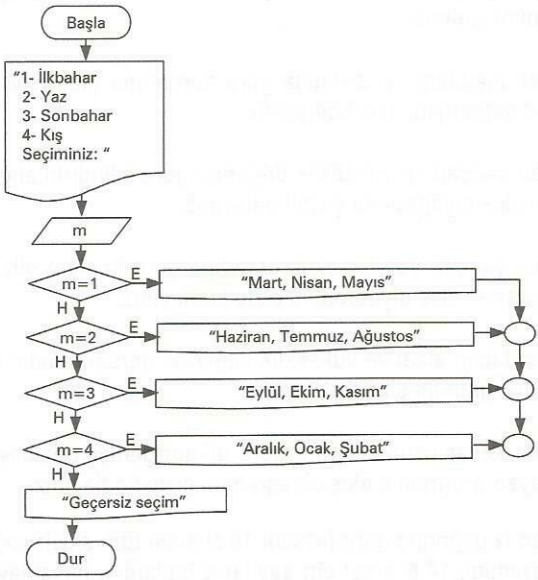
Örnek-6.60: Klavyeden girilen a pozitif tamsayı için $\frac{1}{a}$ bölümünün ilk b basamağını gösteren program.



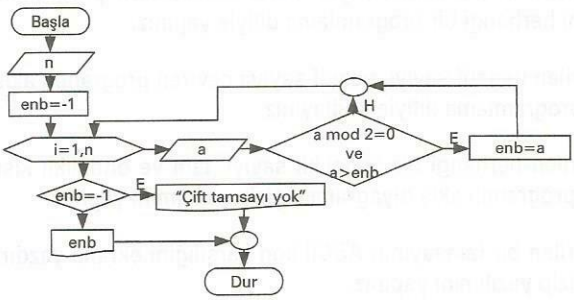
Örnek-6.61: Klavyeden girilen n adet pozitif tamsayının sadece birler basamaklarını toplayan program.



Örnek-6.62: Seçilen mevsime göre ay isimlerini listeleyen program.



Örnek-6.63: Klavyeden girilen n tane pozitif tamsayıdan en büyük çift tamsayıyı bulan program.



Not

Bölüm 11'deki özel sayı ve şekilleri inceleyiniz.

SORULAR

1. Klavyeden girilen bir sayının karesini, küpünü ve karekökünü alan programın akış diyagramını çizip herhangi bir programlama diliyle kodlayınız.

2. Klavyeden üç kenarı girilen bir üçgenin türünü (eşkenar, ikizkenar veya çeşitkenar) tespit eden programın akış diyagramını çizip yazılımını yapınız.

3. Klavyeden girilen iki tamsayıdan büyük olanı gösteren programın akış diyagramını çizip kodlayınız.

4. Klavyeden girilen üç sayıyı büyükten küçüğe doğru sıralayan programın akış diyagramını çizip yazılımını yapınız.

5. Klavyeden girilen maddenin sıcaklığına göre durumunu (katı, sıvı veya gaz) gösteren programın akış diyagramını çizip kodlayınız.

6. Klavyeden girilen yarıçap ve yükseklik değerine göre silindirin alan ve hacmini hesaplayan programın akış diyagramını çizip kodlayınız.

7. Klavyeden girilen yarıçap ve yükseklik değerine göre koninin yüzey alanı ve hacmini hesaplayan programın akış diyagramını çizip kodlayınız.

8. Klavyeden girilen taban alanı ve yükseklik değerine göre piramitin hacmini hesaplayan programın akış diyagramını çizip kodlayınız.

9. Klavyeden girilen kenar uzunluklarına göre dikdörtgenler prizmasının yüzey alanı ve hacmini hesaplayan programın akış diyagramını çizip kodlayınız.

10. Klavyeden girilen N değerine göre (N>20); 10-N arası tüm sayıların toplamını, 5-N arası tek sayıların çarpımını, 14-N arası çift sayıların toplamını hesaplayıp yazdıran programın akış diyagramını çizip yazılımını gerçekleyiniz.

11. Klavyeden girilen küçük harfi, büyük harfe dönüştüren programın akış diyagramını çizip yazılımını herhangi bir programlama diliyle yapınız.

12. Klavyeden girilen negatif sayıyı, pozitif sayıya çeviren programın akış diyagramını çizip herhangi bir programlama diliyle kodlayınız.

13. Klavyeden girilen herhangi bir ondalıklı sayıyı, tam ve ondalıklı kısımlarını ayırıp ayrı ayrı yazdıran programın akış diyagramını çizip yazılımını yapınız.

14. 0-255 arası girilen bir tamsayının ASCII kod karşılığını ekrana yazdıran programın akış diyagramını çizip yazılımını yapınız.

15. 1-99 arasında haneleri toplamı tek olan tamsayıların listesini veren programın akış diyagramını çizip yazılımını yapınız.

16. Klavyeden girilen iki doğrunun birbirlerine göre durumunu (birbirine dik, paralel veya birbirini kesiyor) bulup yazdıran programın akış diyagramını çizip yazılımını yapınız.

17. $z = a + jb$ **şeklindeki karmaşık sayısı**

$$\left.\begin{array}{l} r = \sqrt{a^2 + b^2} \\ \theta = Arctg\left(\dfrac{b}{a}\right) \end{array}\right\} \Rightarrow \left\{\begin{array}{ll} Trigonometrik\ biçim & \rightarrow \quad z = r\{Cos(\theta) + jSin(\theta)\} \\ Üstel\ biçim\ (Euler\ formülü) & \rightarrow \quad z = r.e^{j\theta} \end{array}\right.$$

şeklinde ifade edilebilir. Buna göre klavyeden girilen karmaşık sayıyı her iki biçimde de ekrana yazdıran programın akış diyagramını çizip yazılımını herhangi bir programlama diliyle yapınız.

18. Klavyeden girilen bir tamsayının haneleri toplamını veren program akış diyagramını çizip yazılımını herhangi bir programlama diliyle kodlayınız.

19. Klavyeden girilen N üst sınırına göre

$$T1 = \frac{5}{3} + \frac{6}{4} + \frac{7}{5} + \frac{8}{6} + \ldots \quad , \quad T2 = \frac{2}{3} - \frac{3}{5} + \frac{4}{7} - \frac{5}{9} + \ldots \quad , \quad T3 = \frac{1}{2} + \frac{2}{5} + \frac{3}{10} + \frac{4}{17} + \frac{5}{26} + \ldots \quad ,$$

$$T4 = \frac{2}{3} - \frac{3}{8} + \frac{4}{15} - \frac{5}{24} + \ldots \quad , \quad T5 = \frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{5} + \ldots \quad , \quad T6 = \frac{1}{2} + \frac{2}{4} + \frac{3}{6} + \frac{4}{8} + \ldots$$

seri toplamlarını ayrı ayrı bulan programın akış diyagramını çizip kodlayınız.

20. Aşağıda verilen seri açılımlarını, klavyeden girilen terim sayısı kadar açarak belirtilen değerleri hesaplayan programların akış diyagramlarını ayrı ayrı çizerek kodlayınız.

$$Sinh(x) = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \ldots \quad , \quad Cosh(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \ldots$$

$|x| < 1$ **için** $\quad Ln(1-x) = -x - \dfrac{x^2}{2} - \dfrac{x^3}{3} - \dfrac{x^4}{4} - \ldots$

$|x| < 1$ **için** $\quad Ln(1+x) = x - \dfrac{x^2}{2} + \dfrac{x^3}{3} - \dfrac{x^4}{4} + \ldots$

$|x-1| \le 1$ **için** $Ln(x) = (x-1) - \dfrac{(x-1)^2}{2} + \dfrac{(x-1)^3}{3} - \ldots$

$|x| \le 1$ **için** $\quad ArcSin(x) = \displaystyle\sum_{k=0}^{\infty} \frac{(2k)!}{4^k.(k!)^2.(2k+1)} x^{2k+1}$

$|x| \le 1$ **için** $\quad ArcTan(x) = x - \dfrac{x^3}{3} + \dfrac{x^5}{5} - \dfrac{x^7}{7} + \ldots$

$|x| \le 1$ **için** $\quad \sqrt{x+1} = 1 + \dfrac{1}{2}x - \dfrac{1}{8}x^2 + \dfrac{1}{16}x^3 - \dfrac{5}{128}x^4 + \ldots = \displaystyle\sum_{k=0}^{\infty} \frac{(-1)^k (2k)!}{(1-2k)(k!)^2 4^k} x^k$

$|x| \le 1$ için $ArcSinh(x) = \sum_{k=0}^{\infty} \frac{(-1)^k.(2k)!}{4^k.(k!)^2.(2k+1)} x^{2k+1}$

$|x| \le 1$ için $ArcTanh(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots$

21. Klavyeden girilen terim sayısına göre x değeri için aşağıdaki işlemleri Maclaurin serisine açarak gerçekleştiren programların akış diyagramlarını ayrı ayrı çizerek kodlayınız.

$f(x) = \frac{e^x}{Cos(x)}$, $f(x) = 2.Cos(x) + 3.Sin(x) + 4.e^x$, $f(x) = \frac{Cos(x)}{Sin(x)}$

22. Klavyeden iki noktası $(x_1, y_1) - (x_2, y_2)$ girilen doğrunun denklemini elde eden programın akış diyagramını çizip kodlayınız.

23. "*Binom katsayıları*"

a) **Faktöriyel formülü ile:** $0 \le r \le n$ **için** $\binom{n}{r} = \frac{n!}{r!(n-r)!}$

b) **Çarpım formülü ile:** $\binom{n}{r} = \prod_{i=1}^{r} \frac{n-r+i}{i}$

c) **Yinelemeli (rekürsif) formül ile: Tüm** $n, r > 0$ **tamsayıları ve** $\binom{n}{0} = 1$ **ile** $\binom{0}{r} = 0$ **başlangıç koşulları için** $\binom{n}{r} = \binom{n-1}{r-1} + \binom{n-1}{r}$

ile hesaplanabilmektedir. Buna göre klavyeden belirtilen Binom katsayılarını, her bir förmülle hesaplayıp yazdıran programın akış diyagramlarını ayrı ayrı çizip kodlayınız.

24. Klavyeden girilen ondalıklı sayıyı, sadeleşemeyecek şekilde rasyonel sayı olarak yazdıran programın akış diyagramını çizip kodlayınız. Örneğin klavyeden "3.5" girilirse program "7/2" üretmelidir.

25. Herhangi bir n sayısının "üstel faktöriyeli", bu sayının $(n-1)$'den itibaren 1'e kadar sırasıyla kuvvetlerinin/üslerinin alınmasıyla elde edilir. Örneğin 4'ün faktöriyeli $4! = 4.3.2.1 = 24$ iken 4'ün üstel faktöriyeli de $4^{3^{2^1}} = 262144$ dir. Buna göre klavyeden girilen küçük pozitif tamsayının üstel faktöriyelini hesaplayan programın akış diyagramını çizip kodlayınız.

26. Klavyeden girilen iki pozitif tamsayı için "*ortak katlarının en küçüğünü (OKEK)*" hesaplayıp yazdıran programın akış diyagramını çiziniz ve herhangi bir programlama dilinde kodlayınız. Not: OBEB ile OKEK arasında $OKEK(a,b) = \frac{|a.b|}{OBEB(a,b)}$ **ilişkisi de mevcuttur.**

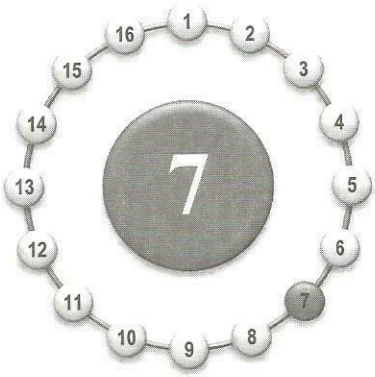
27. Klavyeden girilen n tane direncin, bobinin veya kondansatörün seri ve paralel bağlanması durumları için ayrı ayrı eşdeğer direnci hesaplayan programın akış diyagramını çiziniz ve kodlayınız.
28. Bilgisayarın ürettiği rastgele 5 harfli anlamlı veya anlamsız kelimenin tahmin edilme oyununun akış diyagramını çizip kodlayınız.
29. Klavyeden gün-ay-yıl olarak girilen tarihin geçerli olup olmadığını denetleyen programın akış diyagramını çizip kodlayınız.
30. Klavyeden gün-ay-yıl olarak girilen doğum tarihine göre yaşı ve bugüne kadar geçen gün sayısını hesaplayan programın akış diyagramını çizip kodlayınız.
31. "Hanoi Kuleleri" oyununu araştırarak (yinelemeli fonksiyon oluşturma) akış diyagramını çiziniz ve kodlayınız.
32. Uzunluk birimleri arasında dönüşüm yapabilen programın akış diyagramını çizip kodlayınız.
33. Alan birimleri arasında dönüşüm yapabilen programın akış diyagramını çizip kodlayınız.
34. Hacim birimleri arasında dönüşüm yapabilen programın akış diyagramını çizip kodlayınız.
35. Ağırlık birimleri arasında dönüşüm yapabilen programın akış diyagramını çizip kodlayınız.
36. Sıcaklık birimleri arasında dönüşüm yapabilen programın akış diyagramını çizip kodlayınız.
37. Güç birimleri arasında dönüşüm yapabilen programın akış diyagramını çizip kodlayınız.
38. Klavyeden herhangi bir tuşa basılıncaya kadar birer saniye aralıklarla üretilen 0-100 arası tamsayılardan çift olanları kırmızı ve tek olanları beyaz renkle yazdıran programın akış diyagramını çiziniz ve kodlayınız.
39. Klavyeden girilen satır sayısına göre aşağıdaki üçgeni oluşturan programın akış diyagramını çiziniz ve kodlayınız.

```
1
2     1
3     2     1
4     3     2     1
...   ...   ...   ...   ...
```

40. Klavyeden girilen satır sayısına göre ekranda "*" karakterlerinden eşkenar üçgen oluşturan programın akış diyagramını çiziniz ve kodlayınız.

BÖLÜM 7

BİR BOYUTLU DİZİ UYGULAMALARI



Bu bölümde

- Bir boyutlu diziler hakkındaki temel bilgileri,
- Bir boyutlu diziler üzerinde birçok işlem gerçekleştirmeyi,
- Alfasayısal verileri, dizi olarak işlemeyi öğreneceksiniz.

Bölüm sonunda

- Çoklu verileri dizi olarak işlemeyi öğrenecek,
- Programlarda dizileri kullanabilecek,
- Sayısal ve alfasayısal diziler üzerinde birçok işlem gerçekleştirebileceksiniz.

Genel olarak bilgisayarda işlenen verilerin sayısı fazladır. Çok sayıda verinin bilgisayara girilmesi/aktarılması, hızlı ve doğru şekilde işlenebilmesi, uygun şekilde saklanabilmesi için belirli bir düzende(sistematikte) olması gerekmektedir. Belirli bir düzende olan verilerin işlenmesi hem daha kolay hem de daha pratiktir. Bu nedenle bilgisayar programlarında; çoklu verileri işlemek için "*dizi*" olarak adlandırılan sıralı (ardışık) veri/bilgi alanları kullanılır. Programlarda tek isimle/tanımlayıcıyla adlandırılan bu veri alanları, dizi ismi yanındaki indis (yer/konum) numarasıyla çağırılıp işlenirler.

En temel çoklu veri alanı olan diziler, üç başlık altında incelenebilir:

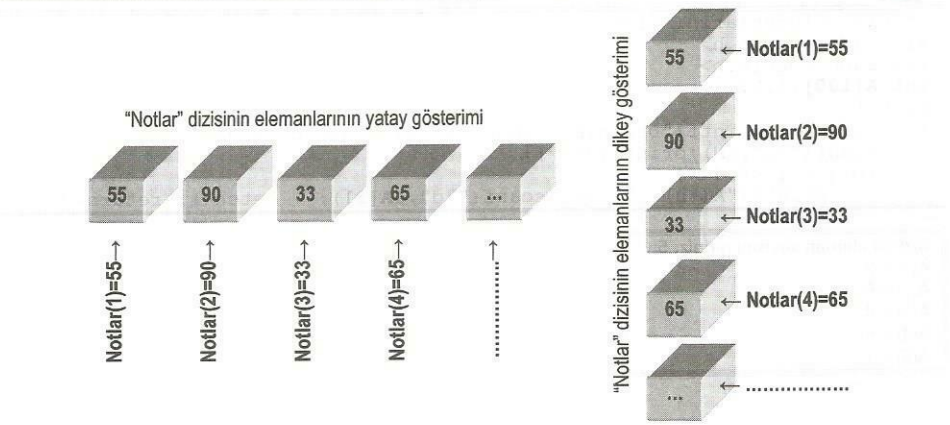
- *Bir boyutlu diziler*: "*Vektör*" olarak da adlandırılan ve sadece satır veya sütundan, yani tek sıradan oluşan ardışık veri alanıdır.
- *İki boyutlu diziler*: "*Matris*" olarak da adlandırılan ve satır ile sütundan oluşan (tablo – diziler dizisi) veri alanıdır.
- *Çok boyutlu diziler*: Üç veya daha fazla boyutlu veri alanlarıdır. Örneğin "*uzay*", genişlik-uzunluk-yükseklikten oluşan üç boyutlu veri alanıdır.

İster bir ister iki boyutlu olsun, veriler belleğe ardışık olarak (yani bir boyutlu dizi şeklinde) yerleşmektedirler. Ayrıca alfasayısal veriler de dizi şeklindedir ve kelimenin/cümlenin herbir karakteri/harfi bir dizi elemanı gibi davranmaktadır.

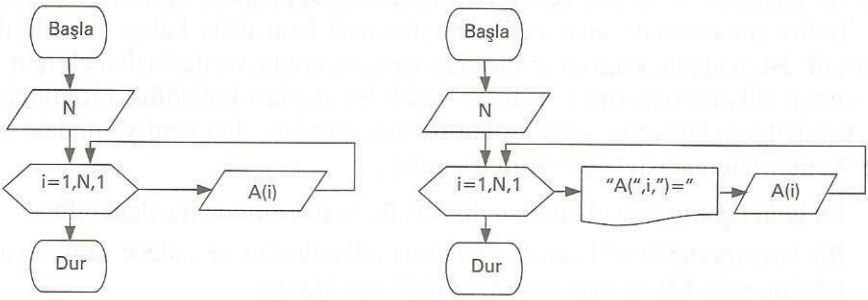
Örneğin 100 kişilik bir sınıfın "Algoritma Geliştirme" dersinden aldığı yılsonu notlarını; tek tek değişkenlere aktarmak yerine (100 tane değişken adı gerekli), bunları 'Notlar' isimli bir diziye (tek değişken adı gerekli) aktarılabilir. Bu şekilde birçok değişken adı kullanılmaz; bilgiler tek isim altında, belirli bir düzende tutulur ve hızlı bir şekilde işlenirler.

"Notlar" dizisi

Not →	**55**	**90**	**33**	**65**	...
İndis →	1.	2.	3.	4.	...
Dizi →	Notlar(1)	Notlar(2)	Notlar(3)	Notlar(4)	...



Örnek-7.1: N tane verinin klavyeden girilmesi ve A dizisine $A=(a_1,a_2,a_3,.....,a_N)$ şeklinde yerleştirilmesi.



$N=3$ olursa programın çalışması Tablo 7.1'deki gibi olacaktır.

Tablo 7.1: Örnek-7.1'in işlem adımları

Dizi indisi -> i	Girilen eleman –> A(i)
1	A(1)
2	A(2)
3	A(3)

Not

"N" elemanlı bir dizide;

- ✓ MATLAB'ta ilk elemanın indisi "1" ve son elemanın indisi "N"dir.
- ✓ C, C++, C#, Java ve Python'da ilk elemanın indisi "0" ve son elemanın indisi "N-1"dir.

Not

Kitaptaki uygulamaların akış diyagramlarında, matematiksel notasyona uygun olması için 1-N arası indis kullanılmıştır. Döngü değişkenleri 0'dan başlatılıp N-1'e kadar da açılabilir. Program yazanlar, bu iki yoldan birini tercih edebilir. Ayrıca bazı programlama dillerinde indislerin başlangıç değeri de ayarlanabilmektedir.

C

```
/* Dizi elemanlarını girme */
#include <stdio.h>
#include <conio.h>
int A[100],i,N;
main()
{   clrscr();printf("Dizinin eleman sayısını giriniz: ");
    scanf("%u",&N);printf("\n");
    for(i=0;i<N;i++)
      { printf("A(%u)= ",i+1);scanf("%d",&A[i]); } getch(); return 0; }
```

```
Dizinin eleman sayısını giriniz: 5
A(1)= 2
A(2)= 6
A(3)= 4
A(4)= 8
A(5)= 1
```

C++

```
// Dizi elemanlarını girme
#include <iostream>
using namespace std;
int N;
main()
{    system("cls"); cout<<"Dizinin eleman sayısını giriniz: "; cin>>N;
     int A[N]; cout << "\n";
     for(int i=0;i<N;i++)
       { cout << "A(" << i+1 <<")= "; cin >> A[i]; } system("pause"); }
```

C#

```
// Dizi elemanlarını girme
using System;
namespace ornek71
{ class Program
  { static void Main(string[] args)
     { Console.Write("Dizinin eleman sayısını giriniz: ");
       int N = Convert.ToInt32(Console.ReadLine());
       int[] A = new int[N];
       for (int i = 0; i < = N; i++)
        { Console.Write("A({0})= ", i+1);
          A[i] = Convert.ToInt32(Console.ReadLine()); }    }}}
```

JAVA

```
// Dizi elemanlarını girme
import java.util.Scanner;
public class ornek71 {
public static void main(String[] args)
{  Scanner klavye=new Scanner(System.in);
   System.out.print("Dizinin eleman sayısını giriniz: ");
   int N=klavye.nextInt();
   int A[]=new int[N];  // int[] A=new int[N];
   for(int i=0;i<N;i++)
    { System.out.print("A("+(i+1)+")= "); A[i]=klavye.nextInt(); }}}
```

MATLAB

```
% Dizi elemanlarını girme
clear all; clc; N=input('Dizinin eleman sayısını giriniz: ');
for i=1:N
    fprintf('A(%d)= ',i); A(i)=input('');
end
```

MATLAB

```
% Dizi elemanlarını girme
clear all; clc; A=input('Diziyi [a b c d] şeklinde giriniz !\n');
```

```
Diziyi [a b c d] şeklinde giriniz
!
[1 2 4 3 5]
```

PYTHON

```
# Dizi elemanlarını girme
A=[]
N=int(input("Dizinin eleman sayısını giriniz: "))
for i in range(0,N):
   print("A(%d)= "%(i+1),end='')
   A.append(int(input()))
```

PYTHON

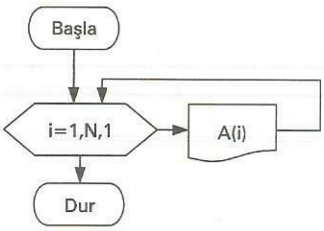
```
# Dizi elemanlarını girme
print("Aralarına boşluk bırakarak dizi elemanlarını giriniz!")
A=[int(i) for i in input().split()]
```

```
Aralarına boşluk bırakarak dizi elemanlarını
giriniz!
1 2 6 8 7
```

Yukarıdaki program kodlarından (C, C++, C#, Java) da görüldüğü gibi dizileri kullanmak için, öncelikle bellekte yer ayırmak gerekmektedir. Bellek ayırma işlemi program başında sabit (statik) olarak yapılabileceği gibi, program içinde dinamik olarak da gerçekleştirilebilir.

Örnek-7.2: N elemanlı bir dizinin elemanlarını yazdırmak.

$N = 5$ için programın çalışması Tablo 7.2'deki gibi olmaktadır.



Tablo 7.2: Örnek-7.2 için işlem adımları

Dizi indisi -> i	Yazdırılan eleman -> A(i)
1	A(1)
2	A(2)
3	A(3)
4	A(4)
5	A(5)

C

```
/* Dizi yazdırma */
#include <stdio.h>
#include <conio.h>
int i,N,A[100];
main()
{ clrscr();
  for(i=0;i<N;i++) printf("A(%u)= %d\n",i,A[i]);getch(); return 0; }
```

C++

```
// Dizi yazdırma
#include <iostream>
using namespace std;
int N,A[100];
main()
{ system("cls");for(int i=0;i<N;i++) cout<<"A("<<i<<")= "<<A[i]<<endl;
  system("pause"); }
```

C#

```
// Dizi yazdırma
using System;
namespace ornek72
{ class Program
  { static void Main(string[] args)
    { int N; int[] A = new int[N];
      for (int i=0;i<N;i++) Console.WriteLine("A({0})={1}",i,A[i]);
      Console.ReadLine(); }}}
```

JAVA

```
// Dizi yazdırma
import java.util.Scanner;
public class ornek72 {
public static void main(String[] args)
{  int N; int A[]=new int[100];
   for (int i=0;i<N;i++) System.out.println("A("+i+")="+A[i]);}}
```

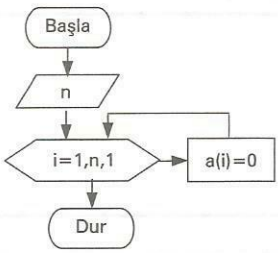
MATLAB

```
% Dizi elemanlarını yazdırma
clear all; clc;
for i=1:N fprintf('A(%d)=%d\n',i,A(i)); end
```

PYTHON

```
# Dizi elemanlarını yazdırma
for i in range(0,N):
   print("A(%d)=%d"%(i+1,A[i]))
```

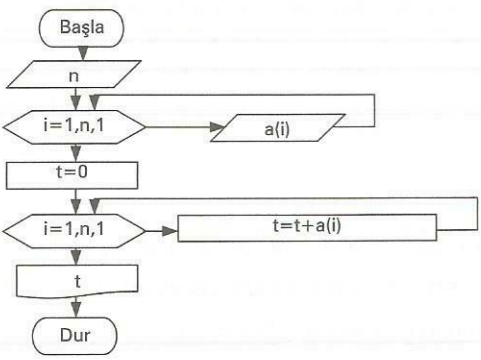
Örnek-7.3: Tüm elemanları 0 olan n elemanlı bir A dizisi oluşturan program.



Örnek-7.4: Klavyeden girilen n elemanlı bir A dizisinin tüm elemanlarını toplayan program.

$$A=(a_1,a_2,a_3,.....,a_n) \Rightarrow Toplam = a_1+a_2+a_3+.....a_N$$

Dizilerle ilgili problemlerde, öncelikle dizi indislerinin nasıl değiştiğini tespit etmek gerekmektedir. İndisler arasındaki ilişki ortaya çıktıktan sonra, döngü açılarak işlemler kolaylıkla yaptırılabilir.



$N=3$ için programın çalışması Tablo 7.3'teki gibi olmaktadır.

Tablo 7.3: Örnek-7.4 'ün çalışma prensibi

I	Eski t	Yeni t
1	0	0+a(1)
2	a(1)	a(1)+a(2)
3	a(1)+a(2)	a(1)+a(2)+a(3)

C

```
/* Dizi elemanları toplamı */
#include <stdio.h>
#include <conio.h>
int i,n,a[100],t=0;
main()
{  clrscr();printf("Dizinin eleman sayısını giriniz: ");
   scanf("%u",&n);printf("\n");
   for(i=0;i<n;i++)
       { printf("A(%u)=",i);scanf("%d",&a[i]); }
   for(i=0;i<n;i++) t+=a[i];
   printf("\nDizinin elemanları toplamı = %d",t); getch(); return 0; }
```

C++

```
// Dizi elemanları toplamı
#include <iostream>
using namespace std;
int i,n,t=0;
int main()
{  system("cls");cout << "Dizinin eleman sayısını giriniz: "; cin >> n;
   int a[n]; cout << "\n";
   for(i=0;i<n;i++)
     { cout << "A(" << i+1 << ")="; cin >> a[i]; }
   for(i=0;i<n;i++) t+=a[i];
   cout<<"\nDizinin elemanları toplamı = "<<t<<endl;
   system("pause"); return 0; }
```

```
Dizinin eleman sayısını giriniz: 3

A(1)= 3
A(2)= 8
A(3)= 4

Dizinin elemanları toplamı = 15
```

C#

```
// Dizi elemanları toplamı
using System;
namespace ornek74
{ class Program
  { static void Main(string[] args)
    { int i,n,t=0; Console.Write("Dizinin eleman sayısını giriniz: ");
      n = Convert.ToInt32(Console.ReadLine());
      int[] a = new int[n]; Console.WriteLine();
      for (i = 0; i < n; i++)
        { Console.Write("A({0})= ", i+1);
          a[i]=Convert.ToInt32(Console.ReadLine()); }
      for (i = 0; i < n; i++) t+= a[i];
      Console.WriteLine("\nDizi elemanları toplamı= {0}\n", t);
      Console.ReadLine();       }     }     }
```

JAVA

```
// Dizi elemanları toplamı
import java.util.Scanner;
public class ornek74 {
public static void main(String[] args)
{  int i,n,t=0; Scanner klavye=new Scanner(System.in);
   System.out.print("Dizinin eleman sayısını giriniz: ");
   n=klavye.nextInt();int a[]=new int[n]; System.out.println("");
   for(i=0;i<n;i++)
       { System.out.print("A("+(i+1)+")= ");a[i]=klavye.nextInt();}
   for(i=0;i<n;i++) t+=a[i];
   System.out.print("\nDizinin elemanları toplamı= "+t); }}
```

MATLAB

```
% Dizi elemanları toplamı
clear all; clc;n=input('Dizinin eleman sayısı: ');disp(' ');
for i=1:n
    fprintf('A(%d)= ',i);a(i)=input('');
end; fprintf('\nDizinin elemanları toplamı: %d\n',sum(a));
```

PYTHON

```
# Dizi elemanlarını toplama
a=[]; n=int(input("Dizinin eleman sayısı: "))
for i in range(0,n):
   print("A(%d)= "%i,end=''); a.append(int(input()))
print("\nDizinin elemanları toplamı: %d"%sum(a))
```

Yukarıdaki örnekte iki döngü kullanılmıştır. Ancak iyi bir programcı burada tek döngü kullanarak sonuca ulaşır: açılan döngüde sırasıyla eleman değeri girilir ve anında toplanır. Yukarıdaki akış diyagramını ve kodlarını, tek döngüye göre düzenleyiniz.

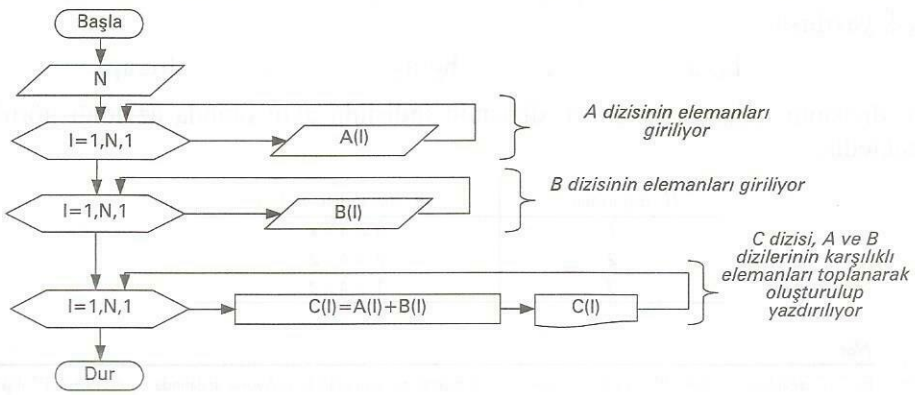
Örnek-7.5: Klavyeden girilen N elemanlı A ve B dizilerini toplayarak C dizisini oluşturan ve yazdıran program.

$$\left.\begin{array}{l} A=(a_1,a_2,...,a_N) \\ B=(b_1,b_2,...,b_N) \end{array}\right\} \Rightarrow C=A+B=(c_1,c_2,...,c_N)=(a_1+b_1,a_2+b_2,...,a_N+b_N)$$

Dizilerle (indisli değişkenlerle) ilgili işlemlerde, dizi indislerinin değişimi arasındaki ilişki gözlenerek uygun ifadeler elde edilmelidir. Bunun için en azından 3 elemanlı diziler alınarak, her bir işlem açıkça yazılmalıdır. Böylece işlemin/işlemlerin gerektirdiği ifade/ifadeler bulunarak programlarda kodlanır. İki dizinin toplamını gerçekleştirecek ifadeyi elde etmek için 3 eleman alınarak işlem açık yazılırsa

$c_1=a_1+b_1$, $c_2=a_2+b_2$, $c_3=a_3+b_3$

indislerin aynı şekilde değiştiği görülür. Dizinin ilk elemanından son elemanına kadar " i " döngüsü açılıp " c(i)=a(i)+b(i) " ifadesiyle toplama gerçekleştirilir.



$N=3$ için programın çalışması Tablo 7.4'teki gibi olmaktadır.

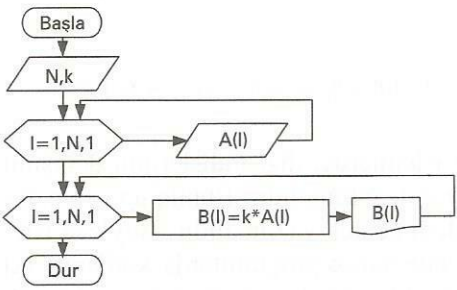
Tablo 7.4: Örnek-7.5'in işlem adımları

I	A(I)	B(I)	C(I)
1	A(1)	B(1)	C(1)=A(1)+B(1)
2	A(2)	B(2)	C(2)=A(2)+B(2)
3	A(3)	B(3)	C(3)=A(3)+B(3)

İki dizinin karşılıklı elemanları arasında temel dört matematiksel işlemleri uygulamak için döngüdeki "`C(I)=A(I)+B(I)`" yerine; çıkarma işlemi için "`C(I)=A(I)-B(I)`", çarpma işlemi için "`C(I)=A(I)*B(I)`" ve bölme işlemi için "`C(I)=A(I)/B(I)`" yazılır. Ayrıca tüm bu programlar tek döngü içinde de yapılabilir. Açılan döngünün içine A ve B dizilerinin sıradaki elemanlarının girilmesi ve ilgili işlemi yazılır.

Örnek-7.6: Klavyeden girilen N elemanlı A dizisini bir 'k' katsayısı ile çarpıp B dizisini oluşturan ve yazdıran program.

$$A=(a_1,a_2,...,a_N) \Rightarrow B=k.A=(b_1,b_2,...,b_N)=(k.a_1,k.a_2,k.a_3,.....,k.a_N)$$



$N=3$ için programın çalışması Tablo 7.5'teki gibi olmaktadır.

Tablo 7.5: Örnek-7.6'ın işlem adımları

I	A(I)	B(I)
1	A(1)	B(1)=k*A(1)
2	A(2)	B(2)=k*A(2)
3	A(3)	B(3)=k*A(3)

Örnek-7.7: Klavyeden girilen N elemanlı A dizisinin elemanlarını, ters sırada B dizisine yerleştiren ve yazdıran program.

$$A=(a_1,a_2,...,a_N) \Rightarrow B=(b_1,b_2,...,b_N)=(a_N,...,a_2,a_1)$$

İndisler arasındaki ilişkiyi görmek amacıyla 3 elemanlı dizideki işlemler açık yazılırsa

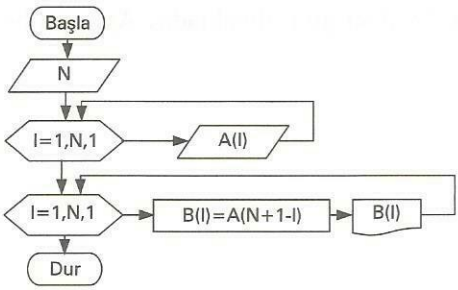
$$b_1=a_3 \quad , \quad b_2=a_2 \quad , \quad b_3=a_1$$

B dizisinin indisi artarken A dizisinin indisinin aynı oranda azaldığı görülmektedir.

B dizi indisi -> i	A dizi indisi -> 4 - i
1	3 = 4 - *1*
2	2 = 4 - *2*
3	1 = 4 - *3*

Not

Bir " i " değişkeni "1"den "N"e kadar artarken, " j " değişkeni de aynı oranda azalıyorsa aralarında " j = N+1-i " ilişkisi vardır.

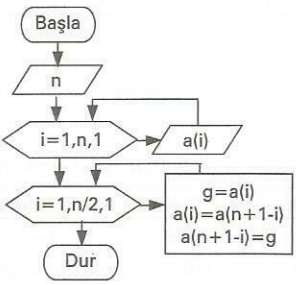


$N = 3$ için programın çalışması Tablo 7.6'daki gibi olmaktadır.

Tablo 7.6: Örnek-7.7 için işlem adımları

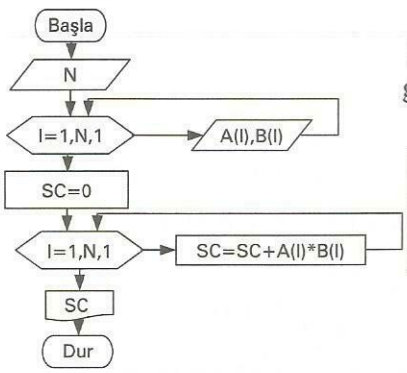
I	A(I)	B(I)=A(N+1-I)
1	A(1)	B(1)=A(3)
2	A(2)	B(2)=A(2)
3	A(3)	B(3)=A(1)

Örnek-7.8: Klavyeden girilen N elemanlı A dizisinin elemanlarını, başka bir dizi kullanmadan tersten yerleştiren program.



Örnek-7.9: Klavyeden girilen N elemanlı A ve B dizilerinin "*skaler çarpım*"ını hesaplayan program.

$$\left.\begin{array}{l} A = (a_1, a_2, ..., a_N) \\ B = (b_1, b_2, ..., b_N) \end{array}\right\} \Rightarrow SC = A.B = (a_1.b_1 + a_2.b_2 + ... + a_N.b_N)$$



$N = 3$ için programın çalışması Tablo 7.7'deki gibi olmaktadır.

Tablo 7.7: Örnek-7.9'un işlem adımları

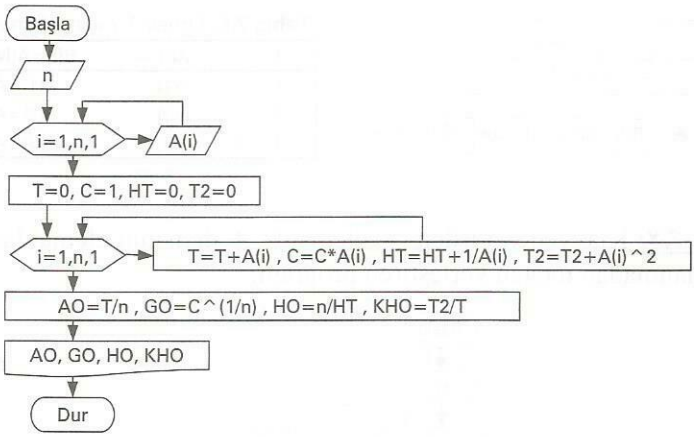
I	A(I)	B(I)	Eski SC	Yeni SC
1	A(1)	B(1)	0	0+A(1)*B(1)
2	A(2)	B(2)	A(1)*B(1)	A(1)*B(1)+A(1)*B(2)
3	A(3)	B(3)	A(1)*B(1)+A(2)*B(2)	A(1)*B(1)+A(2)*B(2)+A(3)*B(3)

Örnek-7.10: Klavyeden girilen n elemanlı A dizisinin aritmetik, geometrik, harmonik ve kontraharmonik ortalamasını hesaplayan program.

$$Aritmetik \Rightarrow AO = \frac{a_1 + a_2 + a_3 + ... + a_n}{n} \quad , \quad Geometrik \Rightarrow GO = \sqrt[n]{a_1.a_2.a_3.....a_n}$$

$$Harmonik \Rightarrow HO = \frac{n}{\frac{1}{a_1} + \frac{1}{a_2} + \frac{1}{a_3} + ... + \frac{1}{a_n}} \quad , \quad Kontraharmonik \Rightarrow KHO = \frac{a_1^2 + a_2^2 + a_3^2 + ... + a_n^2}{a_1 + a_2 + a_3 + ... + a_n}$$

$n = 2$ için programın çalışması Tablo 7.8'deki gibi olmaktadır. Aynı problemi tek döngüyle çözünüz.



Tablo 7.8: Örnek-7.10 için çalışma adımları

I	Eski T	Eski C	Eski HT	Eski T2	Yeni T	Yeni C	Yeni HT	Yeni T2
1	0	1	0	0	0+A(1)	1*A(1)	0+1/A(1)	0+A(1)^2
2	A(1)	A(1)	1/A(1)	A(1)^2	A(1)+A(2)	A(1)*A(2)	1/A(1)+1/A(2)	A(1)^2+A(2)^2
Aritmetik ortalama: AO=T/n=(A(1)+A(2))/n								
*Geometrik ortalama: GO=C^(1/n)=(A(1)*A(2))^(1/n)*								
Harmonik ortalama: HO=n/HT=n/(1/A(1)+1/A(2))								
Kontraharmonik ortalama: KHO=(A(1)^2+A(2)^2)/(A(1)+A(2))								

C

```
/* Dizi ortalamaları */
#include <stdio.h>
#include <conio.h>
#include <math.h>
int i,N; float A[100],T=0,C=1,HT=0,T2=0,AO,HO,GO,KHO;
void main()
{  clrscr();printf("Dizinin eleman sayısını giriniz: ");
   scanf("%u",&N);printf("\n");
   for(i=0;i<N;i++) {printf("A(%u)= ",i+1);scanf("%f",&A[i]);}
   for(i=0;i<N;i++)
      {T+=A[i];C*=A[i];HT+=1/A[i];T2+=A[i]*A[i];}
   AO=T/N;GO=pow(C,1.0/N);HO=N/HT;KHO=T2/T;
   printf("\nDizinin aritmetik ortalaması = %0.3f\n",AO);
   printf("Dizinin geometrik ortalaması = %0.3f\n",GO);
   printf("Dizinin harmonik ortalaması = %0.3f\n",HO);
   printf("Dizinin kontraharmonik ortalaması = %0.3f\n",KHO); getch();}
```

C++

```
// Dizi ortalamaları
#include <iostream>
#include <math.h>
using namespace std;
int i,N; float T=0,C=1,HT=0,T2=0,AO,GO,HO,KHO;
int main()
{   system("cls");cout<<"Dizinin eleman sayısını giriniz: "; cin>>N;
    float A[N]; cout << "\n";
```

```
    for(i=0;i<N;i++) { cout << "A(" << i+1 << ")="; cin >> A[i]; }
    for(i=0;i<N;i++)
       { T+=A[i];C*=A[i];HT+=1/A[i];T2+=A[i]*A[i]; }
    AO=T/N;GO=pow(C,(1.0/N));HO=N/HT;KHO=T2/T;
    cout <<"\nDizinin aritmetik ortalaması ="<<AO << endl;
    cout << "Dizinin geometrik ortalaması =" << GO << endl;
    cout << "Dizinin harmonik ortalaması =" << HO << endl;
    cout << "Dizinin kontraharmonik ortalaması =" << KHO << endl;
    system("pause");return 0; }
```

C#

```
// Dizi ortalamaları
using System;
namespace ornek710
{ class Program
   { static void Main(string[] args)
     { int i,N; double T = 0, C = 1, HT = 0, T2=0, AO, GO, HO, KHO;
       Console.Write("Dizinin eleman sayısını giriniz: ");
       N = Convert.ToInt32(Console.ReadLine()); Console.WriteLine();
       float[] A=new float[N];
       for (i = 0; i < N; i++)
        { Console.Write("A({0})= ",i+1);
          A[i]=Convert.ToSingle(Console.ReadLine());}
       for (i=0;i<N;i++)
           {T+=A[i]; C*=A[i]; HT+=1/A[i];T2+=A[i]*A[i];}
       AO=T/N; GO=Math.Pow(C,(1.0/N)); HO=N/HT; KHO=T2/T;
       Console.WriteLine("\nDizinin aritmetik ortalaması= {0:F3}", AO);
       Console.WriteLine("Dizinin geometrik ortalaması= {0:F3}", GO);
       Console.WriteLine("Dizinin harmonik ortalaması= {0:F3}", HO);
   Console.WriteLine("Dizinin kontraharmonik ortalaması= {0:F3}", KHO);
       Console.ReadLine();      }     }     }
```

JAVA

```
// Dizi ortalamaları
import java.util.Scanner;
public class ornek710 {
public static void main(String[] args)
{  int i,N; double T=0,C=1,HT=0,T2=0,AO,GO,HO,KHO;
   Scanner klavye=new Scanner(System.in);
   System.out.print("Dizinin eleman sayısını giriniz: ");
   N=klavye.nextInt();System.out.println("");
   float A[]=new float[N];
   for(i=0;i<N;i++)
        { System.out.print("A("+(i+1)+")= ");A[i]=klavye.nextFloat();}
   for(i=0;i<N;i++)
      { T+=A[i]; C*=A[i]; HT+=1/A[i]; T2+=A[i]*A[i]; }
   AO=T/N;GO=Math.pow(C,(1.0/N));HO=N/HT;KHO=T2/T;
  System.out.printf("\nDizinin aritmetik ortalaması= %.3f\n",AO);
  System.out.printf("Dizinin geometrik ortalaması= %.3f\n",GO);
  System.out.printf("Dizinin harmonik ortalaması= %.3f\n",HO);
  System.out.printf("Dizinin kontraharmonik ortalaması= %.3f\n",KHO);}}
```

```
Dizinin eleman sayısını giriniz: 3

A(1)= 1
A(2)= 2
A(3)= 3

Dizinin aritmetik ortalaması = 2.000
Dizinin geometrik ortalaması = 1.817
Dizinin harmonik ortalaması = 1.636
Dizinin kontraharmonik ortalaması = 2.333
```

MATLAB

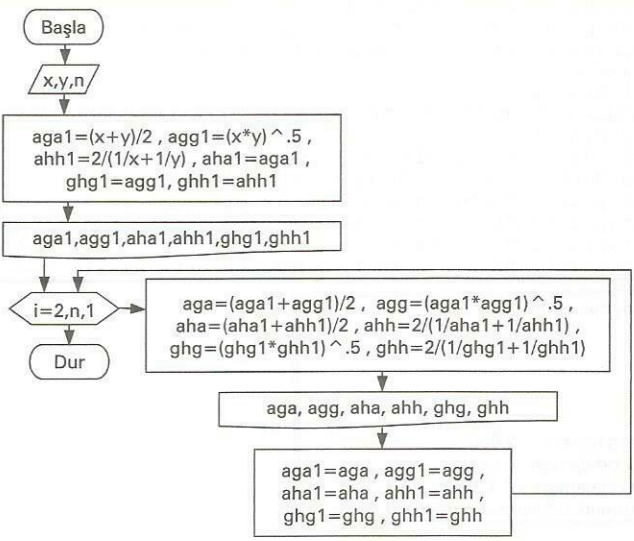
```
% Dizi ortalamaları
clear all; clc;N=input('Dizinin eleman sayısı: ');disp(' ');
for i=1:N
    fprintf('A(%d)= ',i);A(i)=input('');
end; fprintf('\nDizinin aritmetik ortalaması: %0.3f\n',mean(A));
fprintf('Dizinin geometrik ortalaması: %0.3f\n',(prod(A))^(1/N));
fprintf('Dizinin harmonik ortalaması: %0.3f\n',N/sum(1./A));
fprintf('Dizinin kontraharmonik ortalaması: %0.3f\n',sum(A.^2)/sum(A));
```

PYTHON

```
# Dizi ortalamaları
import statistics
A=[]; C=1; T2=0
N=int(input("Dizinin eleman sayısı: "))
for i in range(0,N):
   print("A(%d)="%(i+1),end=''); A.append(int(input()))
print("\nDizinin aritmetik ortalaması: %0.3f"%statistics.mean(A))
for i in range(N):
   C*=A[i]; T2+=A[i]**2
print("Dizinin geometrik ortalaması: %0.3f"%(C**(1/N)))
print("Dizinin harmonik ortalaması: %0.3f"%statistics.harmonic_mean(A))
print("Dizinin kontraharmonik ortalaması: %0.3f"%(T2/sum(A)))
```

Örnek-7.11: Pozitif iki x, y sayısının aritmetik-geometrik, aritmetik-harmonik ve geometrik-harmonik ortalamaları aşağıdaki iterasyonlarla tanımlanmaktadır. Buna göre klavyeden girilen sayıların belirtilen iterasyon sayısınca bu ortalamalarını listeleyen program.

Aritmetik-geometrik ortalama	Aritmetik-harmonik ortalama	Geometrik-harmonik ortalama
$a_1 = \frac{x+y}{2}$, $g_1 = \sqrt{x.y}$	$a_1 = \frac{x+y}{2}$, $h_1 = \frac{2}{\frac{1}{x}+\frac{1}{y}}$	$g_1 = \sqrt{x.y}$, $h_1 = \frac{2}{\frac{1}{x}+\frac{1}{y}}$
$a_{k+1} = \frac{a_k+g_k}{2}$, $g_{k+1} = \sqrt{a_k.g_k}$	$a_{k+1} = \frac{a_k+h_k}{2}$, $h_{k+1} = \frac{2}{\frac{1}{a_k}+\frac{1}{h_k}}$	$g_{k+1} = \sqrt{g_k.h_k}$, $h_{k+1} = \frac{2}{\frac{1}{g_k}+\frac{1}{h_k}}$



C

```
/* Iki sayının ortalamaları */
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{ int i, n;
  float x,y,aga1,agg1,aha1,ahh1,ghg1,ghh1,aga,agg,aha,ahh,ghg,ghh;
  clrscr();printf("1. pozitif sayı: ");scanf("%f",&x);
  printf("2. pozitif sayı: ");scanf("%f",&y);
  printf("Iterasyon sayısı: ");scanf("%u",&n);
  printf("\nIterasyon\tAritmetik-geometrik ort.\tAritmetik-harmonik
ort.\tGeometrik-harmonik ort.\n");
  printf("---------\t----------------------\t----------------------
\t----------------------\n");
  aga1=(x+y)/2; agg1=sqrt(x*y); ahh1=2/(1/x+1/y);
  aha1=aga1; ghg1=agg1; ghh1=ahh1;
  printf("\t1\t\t\t%.3f   %.3f\t\t\t\t%.3f   %.3f\t\t\t%.3f
%.3f\n",aga1,agg1,aha1,ahh1,ghg1,ghh1);
  for(i=2;i<=n;i++)
  { aga=(aga1+agg1)/2; agg=sqrt(aga1*agg1);
    aha=(aha1+ahh1)/2; ahh=2/(1/aha1+1/ahh1);
    ghg=sqrt(ghg1*ghh1); ghh=2/(1/ghg1+1/ghh1);
    printf("\t%d\t\t\t%.3f   %.3f\t\t\t\t%.3f   %.3f\t\t\t%.3f
%.3f\n",i,aga,agg,aha,ahh,ghg,ghh);
    aga1=aga; agg1=agg; aha1=aha; ahh1=ahh; ghg1=ghg; ghh1=ghh; }
  getch(); }
```

C++

```
// Iki sayının ortalamaları
#include <iostream>
#include <math.h>
using namespace std;
int main()
{ int i, n;
  float x,y,aga1,agg1,aha1,ahh1,ghg1,ghh1,aga,agg,aha,ahh,ghg,ghh;
  system("cls");cout<<"1. pozitif sayı: "; cin>>x;
  cout<<"2. pozitif sayı: ";cin>>y; cout<<"İterasyon sayısı: ";cin>>n;
  cout<<"\nİterasyon\tAritmetik-geometrik ort.\tAritmetik-harmonik
ort.\tGeometrik-harmonik ort.\n";
  cout<<"---------\t----------------------\t----------------------
\t----------------------\n";
  aga1=(x+y)/2; agg1=sqrt(x*y); ahh1=2/(1/x+1/y);
  aha1=aga1; ghg1=agg1; ghh1=ahh1;
  cout<<"\t1\t\t"<<aga1<<"   "<<agg1<<"\t\t"<<aha1<<"
"<<ahh1<<"\t\t"<<ghg1<<"   "<<ghh1<<"\n";
  for(i=2;i<=n;i++)
  {   aga=(aga1+agg1)/2; agg=sqrt(aga1*agg1);
      aha=(aha1+ahh1)/2; ahh=2/(1/aha1+1/ahh1);
      ghg=sqrt(ghg1*ghh1); ghh=2/(1/ghg1+1/ghh1);
      cout<<"\t"<<i<<"\t\t"<<aga<<"   "<<agg<<"\t\t"<<aha<<"
"<<ahh<<"\t\t"<<ghg<<"   "<<ghh<<"\n";
       aga1=aga; agg1=agg; aha1=aha; ahh1=ahh; ghg1=ghg; ghh1=ghh;   }
  system("pause");return 0; }
```

C#

```
// Iki sayının ortalamaları
using System;
namespace ornek711
{class Program
  {static void Main(string[] args)
    {int i, n;
     double x,y,aga1,agg1,aha1,ahh1,ghg1,ghh1,aga,agg,aha,ahh,ghg,ghh;
     Console.Write("1. pozitif sayı: ");
     x=Convert.ToDouble(Console.ReadLine());
     Console.Write("2. pozitif sayı: ");
     y=Convert.ToDouble(Console.ReadLine());
```

```
      Console.Write("İterasyon sayısı: ");
      n=Convert.ToInt32(Console.ReadLine());
      Console.WriteLine("\nİterasyon\tAritmetik-geometrik
ort.\tAritmetik-harmonik ort.\tGeometrik-harmonik ort.");
      Console.WriteLine("---------\t-----------------------\t----------
-------------\t-----------------------");
      aga1=(x+y)/2; agg1=Math.Sqrt(x*y); ahh1=2/(1/x+1/y);
      aha1=aga1; ghg1=agg1; ghh1=ahh1;
      Console.WriteLine("\t1\t\t\t{0:F3}   {0:F3}\t\t\t\t{0:F3}
{0:F3}\t\t\t{0:F3}   {0:F3}", aga1, agg1, aha1, ahh1, ghg1, ghh1);
      for (i = 2; i <= n; i++)
       { aga=(aga1+agg1)/2; agg=Math.Sqrt(aga1*agg1);
         aha=(aha1+ahh1)/2; ahh=2/(1/aha1+1/ahh1);
         ghg=Math.Sqrt(ghg1*ghh1); ghh=2/(1/ghg1+1/ghh1);
         Console.WriteLine("\t{0}\t\t\t{0:F3}   {0:F3}\t\t\t\t{0:F3}
{0:F3}\t\t\t{0:F3}   {0:F3}", i, aga, agg, aha, ahh, ghg, ghh);
         aga1=aga; agg1=agg; aha1=aha; ahh1=ahh; ghg1=ghg; ghh1=ghh;}
      Console.ReadLine();       }    }    }
```

JAVA

```
// İki sayının ortalamaları
import java.util.Scanner;
public class ornek711 {
public static void main(String[] args) {
 int i,n;
 double x,y,aga1,agg1,aha1,ahh1,ghg1,ghh1,aga,agg,aha,ahh,ghg,ghh;
 Scanner klavye=new Scanner(System.in);
 System.out.print("1. poizitif sayı: ");x=klavye.nextDouble();
 System.out.print("2. poizitif sayı: ");y=klavye.nextDouble();
 System.out.print("İterasyon sayısı: ");n=klavye.nextInt();
 System.out.println("\nİterasyon\tAritmetik-geometrik ort.\tAritmetik-
harmonik ort.\tGeometrik-harmonik ort.");
 System.out.println("---------\t-----------------------\t-------------
----------\t-----------------------");
 aga1=(x+y)/2;agg1=Math.sqrt(x*y);ahh1=2/(1/x+1/y);
 aha1=aga1;ghg1=agg1;ghh1=ahh1;
 System.out.printf("\t1\t\t\t%.3f   %.3f\t\t\t\t%.3f   %.3f\t\t\t%.3f
%.3f\n",aga1,agg1,aha1,ahh1,ghg1,ghh1);
 for(i=2;i<=n;i++)
 { aga=(aga1+agg1)/2; agg=Math.sqrt(aga1*agg1);
   aha=(aha1+ahh1)/2;ahh=2/(1/aha1+1/ahh1);
   ghg=Math.sqrt(ghg1*ghh1); ghh=2/(1/ghg1+1/ghh1);
   System.out.printf("\t%d\t\t\t%.3f   %.3f\t\t\t\t%.3f
%.3f\t\t\t%.3f   %.3f\n",i,aga,agg,aha,ahh,ghg,ghh);
   aga1=aga; agg1=agg; aha1=aha; ahh1=ahh; ghg1=ghg; ghh1=ghh; }   }  }
```

MATLAB

```
% İki sayının ortalamaları
clear all; clc;
x=input('1. pozitif sayı: ');y=input('2. pozitif sayı: ');
n=input('İterasyon sayısı: ');
fprintf('\nİterasyon\tAritmetik-geometrik ort.\tAritmetik-harmonik
ort.\tGeometrik-harmonik ort.\n')
fprintf('---------\t-----------------------\t----------------------
\t-----------------------\n');
aga1=(x+y)/2;agg1=(x*y)^.5;ahh1=2/(1/x+1/y);aha1=aga1;
ghg1=agg1;ghh1=ahh1;
fprintf('\t1\t\t\t%0.3f   %0.3f\t\t\t\t%0.3f   %0.3f\t\t\t%0.3f
%0.3f\n',aga1,agg1,aha1,ahh1,ghg1,ghh1);
for i=2:n
    aga=(aga1+agg1)/2; agg=(aga1*agg1)^.5;
    aha=(aha1+ahh1)/2;ahh=2/(1/aha1+1/ahh1);
    ghg=(ghg1*ghh1)^.5; ghh=2/(1/ghg1+1/ghh1);
    fprintf('\t%d\t\t\t%0.3f   %0.3f\t\t\t\t%0.3f   %0.3f\t\t\t%0.3f
%0.3f\n',i,aga,agg,aha,ahh,ghg,ghh);
    aga1=aga; agg1=agg; aha1=aha; ahh1=ahh; ghg1=ghg; ghh1=ghh;
end;
```

PYTHON

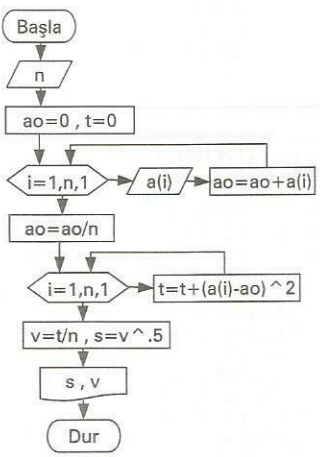
```
# İki sayının ortalamaları
x=eval(input("1. pozitif sayı: ")); y=eval(input("2. pozitif sayı: "))
n=int(input("İterasyon sayısı: "))
print("\nİterasyon\tAritmetik-geometrik ort.\tAritmetik-harmonik
ort.\tGeometrik-harmonik ort.")
print("---------\t------------------------\t-----------------------\t--
--------------------")
aga1=(x+y)/2; agg1=(x*y)**.5; ahh1=2/(1/x+1/y)
aha1=aga1; ghg1=agg1; ghh1=ahh1
print("1\t\t%0.3f    %0.3f\t\t\t%0.3f    %0.3f\t\t%0.3f
%0.3f"%(aga1,agg1,aha1,ahh1,ghg1,ghh1))
for i in range(2,n+1):
    aga=(aga1+agg1)/2; agg=(aga1*agg1)**.5; aha=(aha1+ahh1)/2
    ahh=2/(1/aha1+1/ahh1); ghg=(ghg1*ghh1)**.5; ghh=2/(1/ghg1+1/ghh1)
    print("%d\t\t%0.3f    %0.3f\t\t\t%0.3f    %0.3f\t\t%0.3f
%0.3f"%(i,aga,agg,aha,ahh,ghg,ghh))
    aga1=aga; agg1=agg; aha1=aha; ahh1=ahh; ghg1=ghg; ghh1=ghh
```

```
1. pozitif sayı: 24
2. pozitif sayı: 6
İterasyon sayısı: 10

İterasyon  Aritmetik-geometrik ort.  Aritmetik-harmonik ort.  Geometrik-harmonik ort.
---------  ------------------------  -----------------------  -----------------------
1          15.000    12.000          15.000    9.600          12.000    9.600
2          13.500    13.416          12.300    11.707         10.733    10.667
3          13.458    13.458          12.004    11.996         10.700    10.700
4          13.458    13.458          12.000    12.000         10.700    10.700
5          13.458    13.458          12.000    12.000         10.700    10.700
6          13.458    13.458          12.000    12.000         10.700    10.700
7          13.458    13.458          12.000    12.000         10.700    10.700
8          13.458    13.458          12.000    12.000         10.700    10.700
9          13.458    13.458          12.000    12.000         10.700    10.700
10         13.458    13.458          12.000    12.000         10.700    10.700
```

Örnek-7.12: n elemanlı bir a dizisinin standart sapması ve varyansı için bir tanımlama (başka bir tanımlamada $\frac{1}{n}$ çarpanı yerine $\frac{1}{n-1}$ kullanılmaktadır) aşağıda verilmektedir. Buna göre klavyeden girilen dizinin standart sapmasıyla varyansını hesaplayan program.

Standart sapma	Varyans
$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(a_i - \bar{a})^2}$	$Var(a) = \frac{1}{n}\sum_{i=1}^{n}(a_i - \bar{a})^2$

Eşitliklere dikkat edilirse önce dizinin aritmetik ortalaması hesaplanmalıdır. Daha sonra varyans hesaplanır ve varyansın karekökü alınarak da standart sapma bulunabilir.



C

```
/* Standart sapma ve varyans */
#include <stdio.h>
#include <conio.h>
#include <math.h>
int i,n; float a[100],ao=0,t=0,s,v;
void main()
{  clrscr();printf("Dizinin eleman sayısını giriniz: ");
   scanf("%u",&n);printf("\n");
   for(i=0;i<n;i++)
     {printf("a(%u)= ",i); scanf("%f",&a[i]); ao+=a[i];}
   ao=ao/n; for(i=0;i<n;i++) t+=pow((a[i]-ao),2); v=t/n; s=sqrt(v);
   printf("\nDizinin standart sapması: %0.3f\n",s);
   printf("Dizinin varyansı: %0.3f\n",v);   getch(); }
```

C++

```
// Standart sapma ve varyans
#include <iostream>
#include <math.h>
using namespace std;
int i,n; float ao=0,t=0,s,v;
int main()
{   system("cls");cout<<"Dizinin eleman sayisini giriniz: "; cin>>n;
    float a[n]; cout << "\n";
    for(i=0;i<n;i++) { cout<<"a("<<i<<")="; cin>>a[i]; ao+=a[i]; }
    ao=ao/n; for(i=0;i<n;i++) t+=pow((a[i]-ao),2); v=t/n; s=sqrt(v);
    cout <<"\nDizinin standart sapması: "<< s << endl;
    cout<<"Dizinin varyansı: "<<v<<endl; system("pause");return 0; }
```

C#

```
// Standart sapma ve varyans
using System;
namespace ornek712
{ class Program
   { static void Main(string[] args)
     { int i,n; double ao=0, t = 0, s, v;
       Console.Write("Dizinin eleman sayısını giriniz: ");
       n = Convert.ToInt32(Console.ReadLine()); Console.WriteLine();
       float[] a=new float[n];
       for (i = 0; i < n; i++)
        { Console.Write("a({0})= ",i);
          a[i]=Convert.ToSingle(Console.ReadLine()); ao+=a[i]; }
       ao=ao/n; for (i=0;i<n;i++) t+=Math.Pow((a[i]-ao),2);
       v=t/n; s=Math.Sqrt(v);
       Console.WriteLine("\nDizinin standart sapması: {0:F3}", s);
       Console.WriteLine("Dizinin varyansı: {0:F3}", v);
       Console.ReadLine();      }     }     }
```

JAVA

```
// Standart sapma ve varyans
import java.util.Scanner;
public class ornek712 {
public static void main(String[] args)
{ int i,n; double ao=0,t=0,s,v;
  Scanner klavye=new Scanner(System.in);
  System.out.print("Dizinin eleman sayısını giriniz: ");
  n=klavye.nextInt();System.out.println(""); float a[]=new float[n];
  for(i=0;i<n;i++)
   {System.out.print("a("+i+")= ");a[i]=klavye.nextFloat();ao+=a[i];}
  ao=ao/n;for(i=0;i<n;i++) t+=Math.pow((a[i]-ao),2);
  v=t/n;s=Math.sqrt(v);
  System.out.printf("\nDizinin standart sapması: %.3f\n",s);
  System.out.printf("Dizinin varyansı: %.3f\n",v); } }
```

```
Dizinin eleman sayısını giriniz: 5

a(1)= 1
a(2)= 2
a(3)= 4
a(4)= 6
a(5)= 9

Dizinin standart sapması: 2.871
Dizinin varyansı: 8.240
```

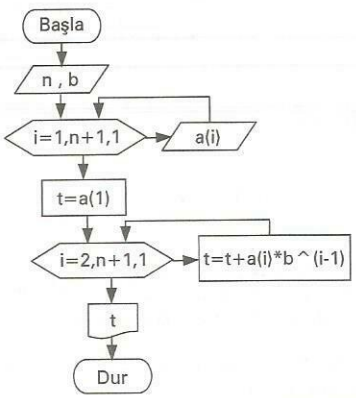
MATLAB

```
% Standart sapma ve varyans
clear all; clc; n=input('Dizinin eleman sayısını giriniz: ');disp(' ');
for i=1:n
    fprintf('a(%d)= ',i);a(i)=input('');
end; fprintf('\nDizinin standart sapması: %0.3f',std(a,1));
fprintf('\nDizinin varyansı: %0.3f\n',var(a,1));
```

PYTHON

```
# Standart sapma ve varyans
import statistics
a=[]; n=int(input("Dizinin eleman sayısını giriniz: "))
for i in range(n):
   print("a(%d)= "%(i+1),end=''); a.append(int(input()))
print("\nDizinin standart sapması: %0.3f"%statistics.pstdev(a))
print("Dizinin varyansı: %0.3f"%statistics.pvariance(a))
```

Örnek-7.13: Katsayıları klavyeden girilen n. dereceden $y(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_1 x + a_0$ polinomunun $x = b$ için değerini ($y(b)$) hesaplayan program.



C

```
/* Polinom değeri hesaplama */
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{ int i,n; float b, a[100],t;
  clrscr(); printf("y(x) polinomunun derecesi: "); scanf("%u",&n);
  for(i=0;i<=n;i++)
    {  printf("x^%d nin katsayısı: ",i); scanf("%f",&a[i]); }
  printf("\nHesaplanacak x degeri: "); scanf("%f",&b); t=a[0];
  for(i=1;i<=n;i++) t+=a[i]*pow(b,i);
  printf("\ny(%0.3f)=%0.5f\n",b,t);    getch(); }
```

C++

```
// Polinom değeri hesaplama
#include <iostream>
#include <math.h>
using namespace std;
int i,n; float b,t;
int main()
{ system("cls");cout<<"y(x) polinomunun derecesi: ";cin>>n;
  float a[n+1];
  for(i=0;i<=n;i++)
     { cout << "x^" << i << " nin katsayısı: "; cin >> a[i]; }
  cout << "\nHesaplanacak x değeri: "; cin >> b;
  t=a[0]; for(i=1;i<=n;i++) t+=a[i]*pow(b,i);
  cout<<"\ny("<<b<<")="<<t<<endl; system("pause"); return 0; }
```

C#

```
// Polinom değeri hesaplama
using System;
namespace ornek713
{ class Program
   { static void Main(string[] args)
      { int i, n; double b, t;
        Console.Write("y(x) polinomunun derecesi: ");
        n = Convert.ToInt32(Console.ReadLine());
        double[] a=new double[n+1];
        for (i = 0; i <= n; i++)
          { Console.Write("x^{0} nin katsayısı: ", i);
            a[i]=Convert.ToDouble(Console.ReadLine()); }
        Console.Write("\nHesaplanacak x değeri: ");
        b = Convert.ToDouble(Console.ReadLine());
        t = a[0]; for (i = 1; i <= n; i++) t += a[i] * Math.Pow(b, i);
        Console.WriteLine("\ny({0:F3})={1:F5}", b, t);
        Console.ReadLine();        }    }    }
```

JAVA

```
// Polinom değeri hesaplama
import java.util.Scanner;
public class ornek713 {
public static void main(String[] args)
{  int i,n;
    double b,t;
   Scanner klavye=new Scanner(System.in);
   System.out.print("y(x) polinomunun derecesi: ");
   n=klavye.nextInt();double a[]=new double[n+1];
   for(i=0;i<=n;i++)
    { System.out.print("x^"+i+" nin katsayısı: ");
      a[i]=klavye.nextDouble();}
   System.out.print("\nHesaplanacak x değeri: ");b=klavye.nextDouble();
   t=a[0]; for(i=1;i<=n;i++) t+=a[i]*Math.pow(b, i);
   System.out.println("\ny("+b+")="+t); }}
```

MATLAB

```
% Polinom değeri hesaplama
clear all; clc ; n=input('y(x) polinomunun derecesi: ');
for i=1:n+1
    fprintf('x^%d nin katsayısı: ',i-1); a(n+2-i)=input('');
end; fprintf('\n'); b=input('Hesaplanacak x değeri: ');
fprintf('\ny(%0.3f)=%0.5f\n',b,polyval(a,b));
```

```
y(x) polinomunun derecesi: 4
x^0 nin katsayısı: 1
x^1 nin katsayısı: 2
x^2 nin katsayısı: 3
x^3 nin katsayısı: 4
x^4 nin katsayısı: 5

Hesaplanacak x değeri: 2

y(2.000)=129.00000
```

MATLAB

```
% Polinom değeri hesaplama
clear all; clc ;
fprintf('Polinom katsayılarını [] içinde birer boşluk bırakarak en yüksek dereceden itibaren giriniz!\n');
a=input('');
b=input('Hesaplanacak x değeri: ');
fprintf('\ny(%0.3f)=%0.5f\n',b,polyval(a,b));
```

```
Polinom katsayılarını [] içinde birer boşluk bırakarak en yüksek dereceden itibaren giriniz!
[5 4 3 2 1]
Hesaplanacak x değeri: 2

y(2.000)=129.00000
```

PYTHON

```
# Polinom değeri hesaplama
import numpy as np
a=[]; n=int(input("y(x) polinomunun derecesi: "))
for i in range(n+1):
   print("x^%d nin katsayısı: "%i,end=''); a.append(eval(input()))
a.reverse(); b=eval(input("Hesaplanacak x değeri: "))
print("\ny(%0.3f)=%0.5f"%(b,np.polyval(a,b)))
```

PYTHON

```
# Polinom değeri hesaplama
import numpy as np
print("Polinom katsayılarını birer boşluk bırakarak en yüksek dereceden itibaren giriniz!")
a=[eval(x) for x in input().split()]
b=eval(input("Hesaplanacak x değeri: "))
print("\ny(%0.3f)=%0.5f"%(b,np.polyval(a,b)))
```

Örnek-7.14: Katsayıları klavyeden girilen $n.$ dereceden $y(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$ polinomunun $x = b$ için değerini ($y(b)$) "*Horner yöntemi*"yle hesaplayan program.

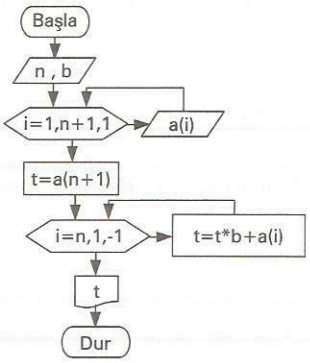
Herhangi bir $y(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x + a_0$ polinomu

$$y(x) = a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_2 x^2 + a_1 x + a_0$$
$$= a_0 + x\{a_1 + a_2 x + \ldots + a_{n-2}x^{n-3} + a_{n-1}x^{n-2} + a_n x^{n-1}\}$$
$$= a_0 + x\{a_1 + x\{a_2 + \ldots + a_{n-2}x^{n-4} + a_{n-1}x^{n-3} + a_n x^{n-2}\}\}$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$= a_0 + x\{a_1 + x\{a_2 + x\{\ldots\}\}\}$$

şeklinde ifade edilerek hesaplanabilir. Örneğin

$$y(x) = 2x^3 - 3x^2 + x + 4 = 4 + x\{1 - 3x + 2x^2\} = 4 + x\{1 + x\{-3 + 2x\}\}$$

şeklinde ifade edilebilmektedir. Dikkat edilirse bu ifadede üs alma işlemi yer almamaktadır.

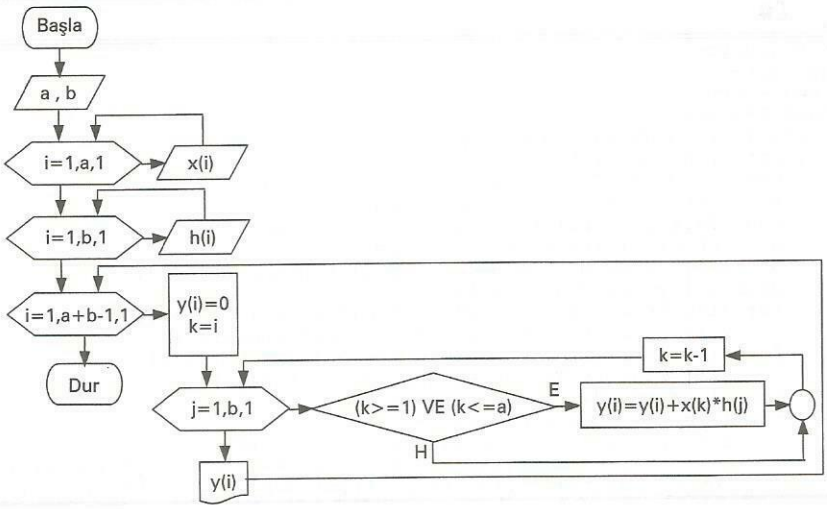


Örnek-7.15: Impulse (birim dürtü) cevabı $h(t)$ olan doğrusal zamanla değişmeyen sistemin girişine herhangi bir $x(t)$ işareti (sinyali) uygulandığında çıkışı, *konvolüsyon* işlemi $y(t) = x(t) * h(t) = \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau$ ile hesaplanmaktadır. Bu işlem ayrık zamanda $y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k].h[n-k]$ şeklinde gerçekleştirilmektedir. Buna göre; klavyeden birim dürtü ve giriş işareti girilen ayrık bir sistemin çıkışını hesaplayan program (1 boyutlu konvolüsyon işlemi).

Öncelikle konvolüsyon işlemini açık olarak $x[n] = \{5,4,3\}$ ve $h[n] = \{1,2\}$ için gerçekleştirelim. Toplam sembolünden de görüldüğü gibi dizinin biri sabit tutulurken, diğeri ters çevrilmekte ve sabit tutulan dizinin elemanlarıyla çarpılıp toplanarak adım adım geçirilmektedir.

x[n]		5	4	3		y[n]
h[n-k]	2	1			→	5.1=5
		2	1		→	5.2+4.1=14
			2	1	→	4.2+3.1=11
				2	1 →	3.2=6

$$\left.\begin{array}{l} x[n] = \{5,4,3\} \\ h[n] = \{1,2\} \end{array}\right\} \Rightarrow y[n] = x[n] * h[n] = \{5,14,11,6\}$$



C

```
/* Konvolüsyon */
#include <stdio.h>
#include <conio.h>
void main()
{ int i,j,k,a,b; float x[100],h[100],y[100];
  clrscr(); printf("Giriş işaretinin uzunluğu: "); scanf("%u",&a);
  printf("Birim dürtü cevabının uzunluğu: "); scanf("%u",&b);
  for(i=0;i<a;i++) {  printf("x[%d]: ",i+1); scanf("%f",&x[i]); }
  for(i=0;i<b;i++) {  printf("h[%d]: ",i+1); scanf("%f",&h[i]); }
  printf("\nÇıkış işareti: ");
  for(i=0;i<a+b-1;i++)
    { k=i;y[i]=0;for(j=0;j<b;j++)
                  { if ((k>=0) && (k<a)) y[i]+=x[k]*h[j]; k--; }
      printf("%0.3f\t",y[i]); }    getch();   }
```

C++

```
// Konvolüsyon
#include <iostream>
using namespace std;
int i,j,k,a,b;
int main()
{  system("cls");cout << "Giriş işaretinin uzunluğu: "; cin >> a;
   cout << "Birim dürtü cevabının uzunluğu: "; cin >> b;
   float x[a],h[b],y[a+b-1];
   for(i=0;i<a;i++) { cout << "x[" << i+1 << "]: "; cin >> x[i]; }
   for(i=0;i<b;i++) { cout << "h[" << i+1 << "]: "; cin >> h[i]; }
   cout << "\nÇıkış işareti: ";
   for(i=0;i<a+b-1;i++)
    { k=i;y[i]=0;for(j=0;j<b;j++)
                  { if ((k>=0) && (k<a)) y[i]+=x[k]*h[j]; k--; }
      cout<<y[i]<<"\t"; }    system("pause"); return 0; }
```

C#

```
// Konvolüsyon
using System;
namespace ornek715
{ class Program
   { static void Main(string[] args)
     { int i, j, k, a, b;
       Console.Write("Giriş işaretinin uzunluğu: ");
       a = Convert.ToInt32(Console.ReadLine());
       Console.Write("Birim dürtü cevabının uzunluğu: ");
       b = Convert.ToInt32(Console.ReadLine());
       double[] x=new double[a]; double[] h=new double[b];
       double[] y=new double[a+b-1];
       for (i=0;i<a;i++) { Console.Write("x[{0}]: ", i+1);
           x[i]=Convert.ToDouble(Console.ReadLine()); }
       for (i=0;i<b;i++) { Console.Write("h[{0}]: ", i+1);
           h[i]=Convert.ToDouble(Console.ReadLine()); }
       Console.Write("\nÇıkış işareti: ");
       for (i = 0; i < a + b - 1; i++)
         { k = i; y[i] = 0; for (j = 0; j < b; j++)
             { if ((k >= 0) && (k < a)) y[i] += x[k] * h[j]; k--; }
           Console.Write("{0:F3}\t",y[i]); } Console.ReadLine(); }}}
```

JAVA

```
// Konvolüsyon
import java.util.Scanner;
public class ornek715 {
public static void main(String[] args)
{  int i,j,k,a,b;
   Scanner klavye=new Scanner(System.in);
   System.out.print("Giriş işaretinin uzunluğu: ");
   a=klavye.nextInt();double x[]=new double[a];
   System.out.print("Birim dürtü cevabının uzunluğu: ");
   b=klavye.nextInt();double h[]=new double[b];
   double y[]=new double[a+b-1];
   for(i=0;i<a;i++)
        { System.out.print("x["+(i+1)+"]: ");x[i]=klavye.nextDouble();}
   for(i=0;i<b;i++)
        { System.out.print("h["+(i+1)+"]: ");h[i]=klavye.nextDouble();}
   System.out.print("\nÇıkış işareti: ");
   for(i=0;i<a+b-1;i++)
        { k=i;y[i]=0;for(j=0;j<b;j++)
                    { if ((k>=0) && (k<a)) y[i]+=x[k]*h[j]; k--; }
          System.out.printf("%.3f\t",y[i]); }}}
```

MATLAB

```
% Konvolüsyon
clear all; clc ; fprintf('Giriş işaretini [] içinde birer boşluk bıra-
karak giriniz!\n'); x=input('');
fprintf('Birim dürtü cevabını [] içinde birer boşluk bırakarak giri-
niz!\n'); h=input(''); fprintf('\nÇıkış işareti:'); disp(conv(x,h))
```

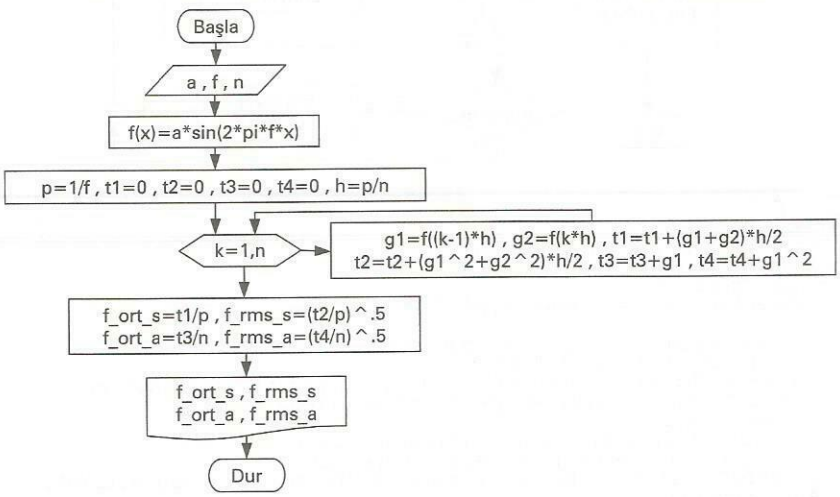
PYTHON

```
# Konvolüsyon
import numpy as np
print("Giriş işaretini birer boşluk bırakarak giriniz!")
x=[eval(i) for i in input().split()]
print("Birim dürtü cevabını birer boşluk bırakarak giriniz!")
h=[eval(i) for i in input().split()]
print("\nÇıkış işareti: ",end='')
print(np.convolve(x,h))
```

```
Giriş işaretini birer boşluk bırakarak giriniz!
5 4 3
Birim dürtü cevabını birer boşluk bırakarak giriniz!
1 2

Çıkış işareti: [ 5 14 11  6]
```

Örnek-7.16: Herhangi bir $f(t)$ fonksiyonunun "*ortalama değer (DA/DC)*" ve "*etkin değer (RMS)*" eşitlikleri sürekli ve ayrık zaman için aşağıda verilmektedir. Buna göre klavyeden parametreleri girilen $f(t) = a.Sin(2\pi ft)$ gerilimi-nin/akımının ortalama ve etkin değerlerini aşağıdaki eşitliklerle hesaplayan program. (Not: Belirli integral için yamuklar yöntemini kullanabilirsiniz. Bknz. Bölüm 15)

	Sürekli	Ayrık
Ortalama (DC)	$f_{ort} = \frac{1}{T}\int_0^T f(t).dt$	$f_{ort} = \frac{\sum_{k=1}^{n} f[k]}{n}$
Etkin (RMS)	$f_{rms} = \sqrt{\frac{1}{T}\int_0^T f(t)^2.dt}$	$f_{rms} = \sqrt{\frac{\sum_{k=1}^{n} f[k]^2}{n}}$



Örnek-7.17: Klavyeden girilen N elemanlı A dizisinin en büyük ve en küçük elemanlarını ve bu elemanların kaçıncı sırada olduklarını bulan program (Maksimum-minimum problemi).

Bu tür problemlerde izlenebilecek en basit ve en temel yol; dizinin bir tane elemanını en büyük ve en küçük varsayıp diğer elemanlarla teker teker kıyaslamaktır. Kıyaslama esnasında daha büyük/küçük dizi elemanı varsa, varsayılan değer yerine bu yeni eleman değeri atanır. Öncelikle kullanacak değişkenleri

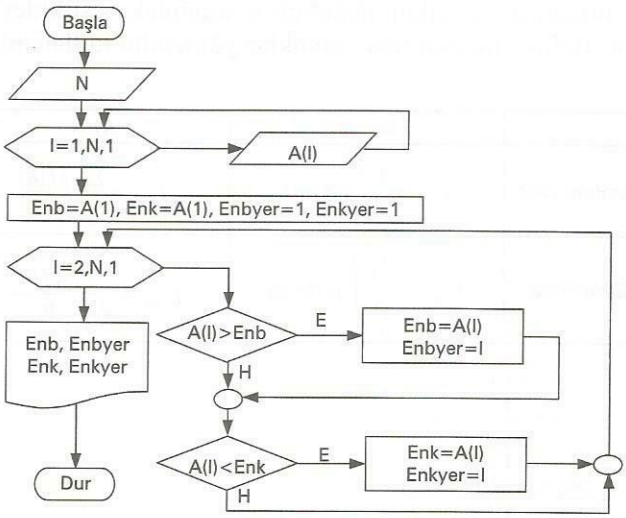
- Dizinin en büyük elemanı: *Enb*

- Dizinin en küçük elemanı: *Enk*
- Dizinin en büyük elemanının bulunduğu sıra: *Enbyer*
- Dizinin en küçük elemanının bulunduğu sıra: *Enkyer*

olarak belirlenir ve karşılaştırmaya başlamadan önce varsayılan olarak

- Dizinin en büyük elemanı *Enb=A(1)* ve sırası *Enbyer=1*
- Dizinin en küçük elemanı *Enk=A(1)* ve sırası *Enkyer=1*

alınabilir.



C

```
/* Dizinin en büyük-küçük elemanları */
#include <stdio.h>
#include <conio.h>
int i,N,enk,enkyer,enb,enbyer,A[100];
main()
{  clrscr();printf("Dizinin eleman sayısını giriniz: ");
   scanf("%u",&N);printf("\n");
   for(i=0;i<N;i++) { printf("A(%u)=",i+1);scanf("%d",&A[i]); }
   printf("\n");enk=A[0];enkyer=0;enb=A[0];enbyer=0;
   for(i=1;i<N;i++)
      {  if (A[i]<enk) { enk=A[i];enkyer=i; }
         if (A[i]>enb) { enb=A[i];enbyer=i; } }
   printf("Dizinin en küçük elemanı %d dir ve %i. sıradadır.\n",
    enk,enkyer+1);
printf("Dizinin en büyük elemanı %d dir ve %i. sıradadır.\n",
   enb,enbyer+1); getch(); return 0; }
```

```
Dizinin eleman sayısını giriniz: 5
A(1)= 2
A(2)= 3
A(3)= 1
A(4)= 6
A(5)= 4

Dizinin en büyük elemanı: 6
Dizinin en küçük elemanı: 1
```

C++

```
// Dizinin en büyük-küçük elemanları
#include <iostream>
using namespace std;
int i,N,enb,enk,enbyer,enkyer;
int main()
{   system("cls");cout<<"Dizinin eleman sayısını giriniz: "; cin>>N;
    int A[N]; cout << "\n";
    for(i=0;i<N;i++) { cout << "A(" << i+1 << ")="; cin >> A[i]; }
    cout << "\n"; enk=enb=A[0]; enkyer=enbyer=0;
    for(i=1;i<N;i++)
      { if(A[i]<enk) { enk=A[i]; enkyer=i; }
        if(A[i]>enb) { enb=A[i]; enbyer=i; } }
    cout << "Dizinin en küçük elemanı "<<enk<<" ve "<<
            enkyer+1 << ". sıradadır.\n";
    cout << "Dizinin en büyük elemanı "<<enb<<" ve "<<
            enbyer+1 << ". sıradadır.\n";
    system("pause");return 0; }
```

C#

```
// Dizinin en büyük-en küçük elemanları
using System;
namespace ornek717
{ class Program
   { static void Main(string[] args)
     { int i, N, enk, enkyer, enb, enbyer;
       Console.Write("Dizinin eleman sayısını giriniz: ");
       N = Convert.ToInt32(Console.ReadLine()); Console.WriteLine();
       int[] A=new int[N];
       for (i = 0; i < N; i++)
        {Console.Write("A({0})= ",i+1);
         A[i]=Convert.ToInt32(Console.ReadLine());}
       enk = enb = A[0]; enkyer = enbyer = 0;
       for (i = 1; i < N; i++)
         { if (A[i] < enk) { enk = A[i]; enkyer = i; }
           if (A[i] > enb) { enb = A[i]; enbyer = i; }}
       Console.WriteLine("\nDizinin en küçük elemanı {0} dir ve {1}.
sıradadır.", enk,enkyer+1);
       Console.WriteLine("Dizinin en büyük elemanı {0} dir ve {1}. sı-
radadır.", enb, enbyer + 1);
       Console.ReadLine();       }    }    }
```

JAVA

```
// Dizinin en büyük-küçük elemanları
import java.util.Scanner;
public class ornek717 {
public static void main(String[] args)
{  int i,N,enk,enkyer,enb,enbyer;
   Scanner klavye=new Scanner(System.in);
   System.out.print("Dizinin eleman sayısını giriniz: ");
   N=klavye.nextInt();System.out.println("");
   int A[]=new int[N];
   for(i=0;i<N;i++)
       { System.out.print("A("+(i+1)+")= ");A[i]=klavye.nextInt(); }
   System.out.println(""); enk=enb=A[0];enkyer=enbyer=0;
   for(i=1;i<N;i++)
       { if(A[i]<enk){enk=A[i]; enkyer=i;}
         if(A[i]>enb){enb=A[i]; enbyer=i;}  }
   System.out.println("Dizinin en küçük elemanı "+enk+" dir
                       ve "+(enkyer+1)+". sıradadır.");
   System.out.println("Dizinin en büyük elemanı "+enb+" dir
                       ve "+(enbyer+1)+". sıradadır.");        }}
```

MATLAB

```
% Dizinin en büyük-en küçük elemanları
clear all; clc;
N=input('Dizinin eleman sayısını giriniz: ');disp(' ');
for i=1:N
    fprintf('A(%d)= ',i);A(i)=input('');
end; [enk,enkyer]=min(A); [enb,enbyer]=max(A);
fprintf('\nDizinin en küçük elemanı %d dir ve %d. sırada-
dır.',enk,enkyer);
fprintf('\nDizinin en büyük elemanı %d dir ve %d. sırada-
dır.\n',enb,enbyer);
```

PYTHON

```
# Dizinin en büyük-en küçük elemanları
A=[]; N=int(input("Dizinin eleman sayısını giriniz: "))
for i in range(N):
   print("A(%d)= "%(i+1),end='')
   A.append(eval(input()))
print("\nDizinin en büyük elemanı: %d"%max(A))
print("Dizinin en küçük elemanı: %d"%min(A))
```

Örnek-7.18: Klavyeden girilen N elemanlı A ve B dizileri ile aşağıdaki işlemi yapıp sonucu yazan program.

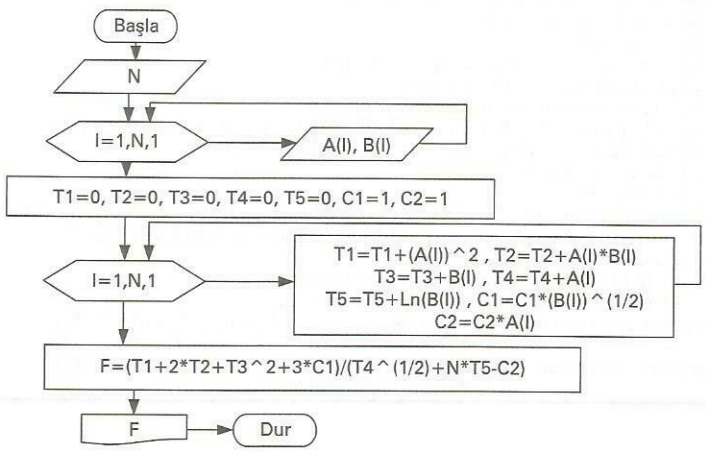
$$F=\frac{\sum_{i=1}^{N}A_i^2+2\sum_{i=1}^{N}A_i.B_i+(\sum_{i=1}^{N}B_i)^2+3\prod_{i=1}^{N}\sqrt{B_i}}{\sqrt{\sum_{i=1}^{N}A_i+N\sum_{i=1}^{N}Ln(B_i)-\prod_{i=1}^{N}A_i}}$$

Yukarıdaki karmaşık ifadeyi kolaylaştırmak için alt işlemlere ayrıştırılarak

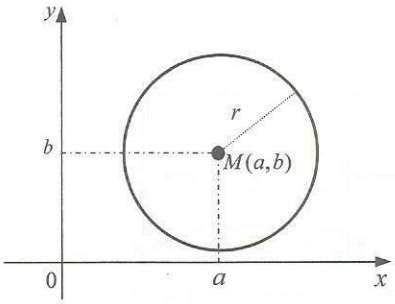
$$T1=\sum_{i=1}^{N}A_i^2 \quad , \quad T2=\sum_{i=1}^{N}A_i.B_i \quad , \quad T3=\sum_{i=1}^{N}B_i \quad , \quad T4=\sum_{i=1}^{N}A_i \quad ,$$

$$T5=\sum_{i=1}^{N}Ln(B_i) \ , \ C1=\prod_{i=1}^{N}\sqrt{B_i} \ , \ C2=\prod_{i=1}^{N}A_i \ , \ F=\frac{T1+2.T2+(T3)^2+3.C1}{\sqrt{T4}+N.T5-C2}$$

ile sonuca ulaşılabilir.



Örnek-7.19: Klavyeden girilen N adet nokta çiftinin, $(x-a)^2+(y-b)^2=r^2$ (Şekil 7.1) dairesindeki dağılımını (içinde, üzerinde, dışında) bulan program.

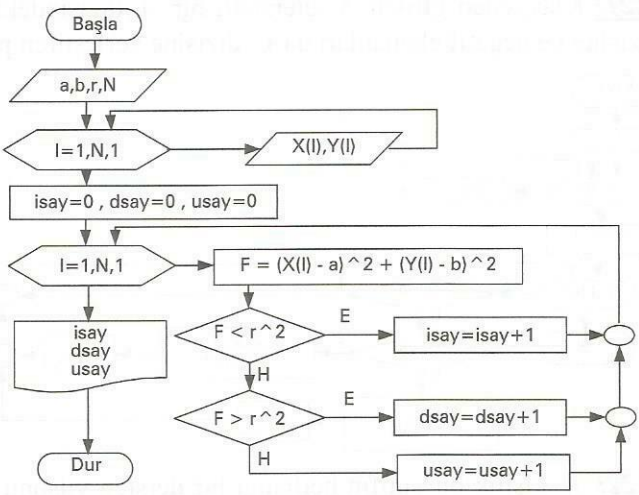


Şekil 7.1 Daire denklemi verileri

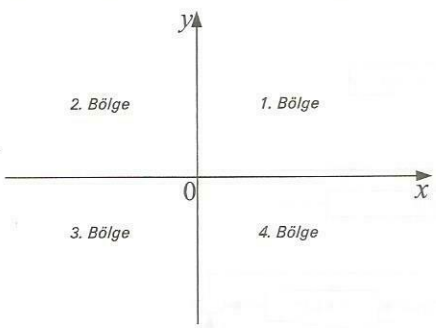
Klavyeden girilen nokta parametrelerinden x-koordinatları (apsis) X dizisine, y-koordinatları (ordinat) da Y dizisine yerleştirilir. 1'den nokta sayısına kadar döngü açılarak X_i ve Y_i 'ler, daire denkleminde yerine koyulup hesaplandığında bulunan değer:

- r^2 'den küçükse, dairenin içinde,
- r^2 'den büyükse, dairenin dışında,
- r^2 'ye eşitse, dairenin üzerindedir.

İlgili sayaç arttırılarak sonuca ulaşılır.



Örnek-7.20: Klavyeden girilen N adet nokta çiftinin (x ve y 'ler sıfırdan farklı) kartezyen koordinat sisteminindeki (Şekil 7.2) dağılımını bulan program.

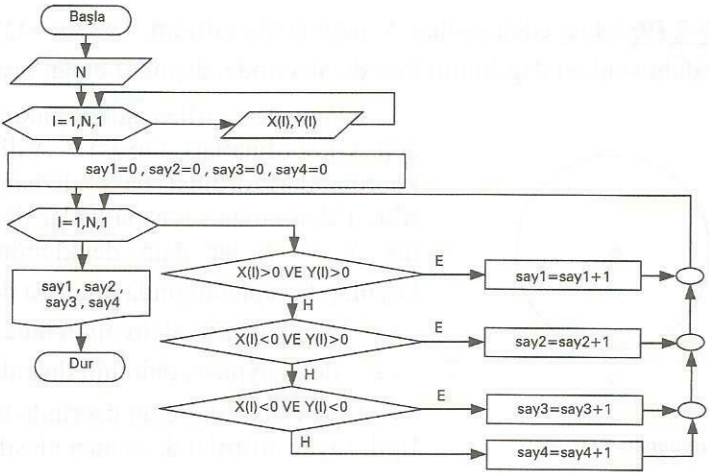


Şekil 7.2: Kartezyen koordinat sistemi

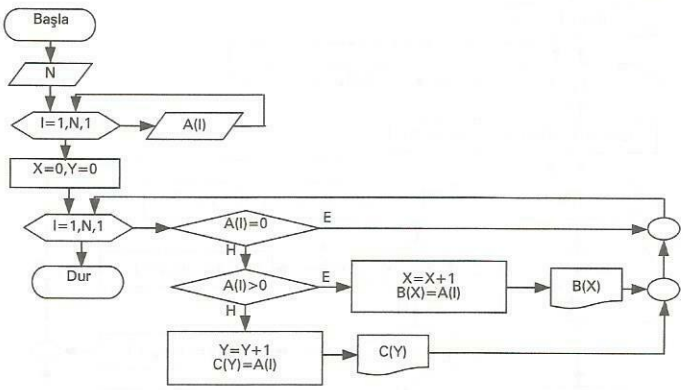
Kartezyen koordinat sisteminde bir noktanın koordinatları x ve y ise bölgelere göre özellikleri (işaretleri):

- 1. bölgede => $x>0$ ve $y>0$
- 2. bölgede => $x<0$ ve $y>0$
- 3. bölgede => $x<0$ ve $y<0$
- 4. bölgede => $x>0$ ve $y<0$

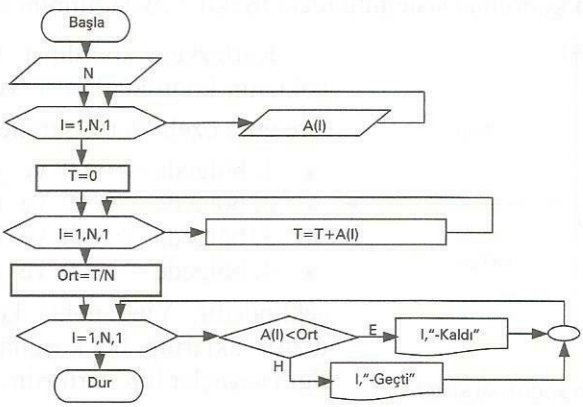
şeklindedir. Yine nokta koordinatları iki diziye aktarılıp bu koşullar test edilerek ilgili sayaçlar birer arttırılır.



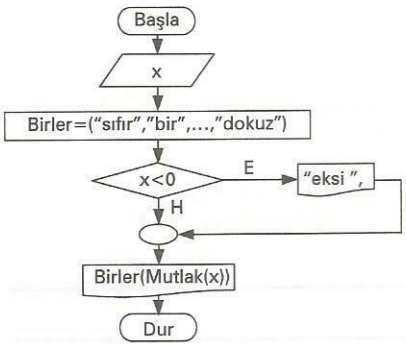
Örnek-7.21: Klavyeden girilen N elemanlı bir A dizisindeki pozitif elemanları B dizisine ve negatif elemanları da C dizisine yerleştiren program.



Örnek-7.22: N kişilik bir sınıfın herhangi bir dersten yılsonu notlarını girilmektedir. Buna göre sınıf ortalamasının altında kalanlara "Kaldı", diğerlerine de "Geçti" yazan program.



Örnek-7.23: Klavyeden girilen -9 ile 9 arasındaki bir basamaklı tamsayıyı, kelime olarak yazdıran program.



Bu işlemi gerçekleştirmek için *'Birler'* dizisi tanımlanır. Bu dizide *("sıfır","bir","iki","üç","dört","beş","altı","yedi","sekiz","dokuz")* rakam isimleri tutulur. Bu şekilde klavyeden girilen tek basamaklı tamsayı, dizinin eleman indisi olarak kullanılarak kelime olarak yazdırılmış olur. Eğer girilen tamsayı negatifse; mutlak değeri alınarak indis olarak kullanılır ve dizi elemanın önüne "eksi" yazdırılır.

C

```
/* Tek basamak */
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{ char *R[10]={"sıfır","bir","iki","üç","dört",
               "beş","altı","yedi","sekiz","dokuz"};
  int x,y;
  clrscr();printf("Tek basamaklı tamsayı giriniz: "); scanf("%u",&x);
  printf("\nGirdiğiniz sayı => "); if (x<0) printf("eksi ");
  y=abs(x); printf("%s\n",R[y]); getch(); }
```

```
Tek basamaklı tamsayı giriniz: -7

Girdiğiniz sayı => eksi yedi
```

C++

```
// Tek basamak
#include <iostream>
#include <cmath>
using namespace std;
const char *R[10]={"sifir","bir","iki","uc","dort",
                   "bes","alti","yedi","sekiz","dokuz"};
int x;
main()
{   system("cls"); cout << "Tek basamaklı tamsayı giriniz: "; cin >> x;
    cout <<"\nGirdiginiz sayi => ";
    if (x<0) cout << "eksi "; cout << R[abs(x)] << endl;
    system("pause"); return 0; }
```

C#

```
// Tek basamak
using System;
namespace ornek723
{class Program
  {static void Main(string[] args)
    {int x; String[] R={"sıfır","bir","iki","üç","dört",
                        "beş","altı","yedi","sekiz","dokuz"};
     Console.Write("Tek basamaklı bir tamsayı giriniz: ");
     x = Convert.ToInt32(Console.ReadLine());
     Console.Write("\nGirdiğiniz sayı => ");
     if(x<0) Console.Write("eksi ");Console.WriteLine(R[Math.Abs(x)]);
     Console.ReadLine();          }      }      }
```

JAVA

```
// Tek basamak
import java.util.Scanner;
public class ornek723 {
public static void main(String[] args)
{  int x;
   String R[]={"sıfır","bir","iki","üç","dört","beş",
               "altı","yedi","sekiz","dokuz"};
   Scanner klavye=new Scanner(System.in);
   System.out.print("Tek basamaklı bir tamsayı giriniz: ");
   x=klavye.nextInt(); System.out.print("\nGirdiğiniz sayı => ");
   if(x<0) System.out.print("eksi ");
   System.out.println(R[Math.abs(x)]); }}
```

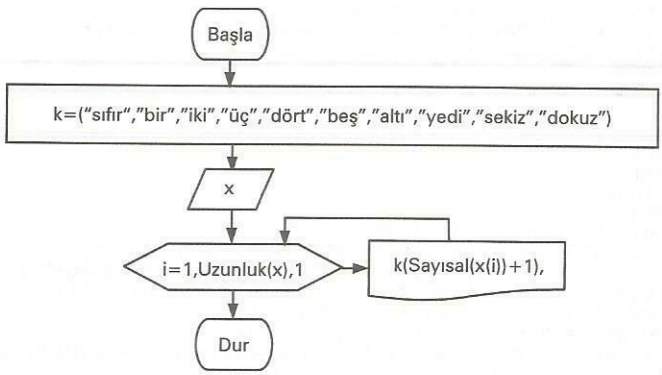
MATLAB

```
% Tek basamak
clear all; clc;
x=input('Tek basamaklı tamsayı giriniz: ');
r={'sıfır','bir','iki','üç','dört','beş','altı',
   'yedi','sekiz','dokuz'}; fprintf('\nGirdiğiniz sayı => ');
if (x<0)
   fprintf('eksi ');
end; fprintf('%s\n',char(r(abs(x)+1)));
```

PYTHON

```
# Tek basamak
r=["sıfır","bir","iki","üç","dört","beş","altı","yedi","sekiz","dokuz"]
x=int(input("Tek basamaklı tamsayı giriniz: "))
print("\nGirdiğiniz sayı => ",end='')
if (x<0):
   print("eksi ",end='')
print(r[abs(x)])
```

Örnek-7.24: Klavyeden girilen pozitif tamsayının hanelerini kelime olarak kodlayan program.



Örnek-7.25: Klavyeden girilen iki basamaklı tamsayıyı kelime olarak yazdıran program.

İki basamaklı tamsayılarda onlar ve birler basamağı olduğu için

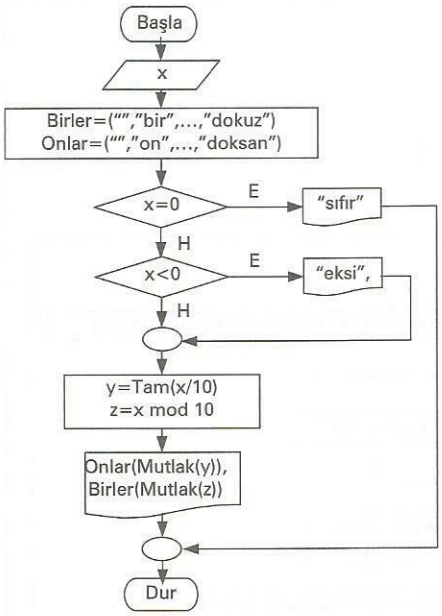
Onlar=("on","yirmi","otuz","kırk","elli","altmış","yetmiş","seksen","doksan")

Birler=("bir","iki","üç","dört","beş","altı","yedi","sekiz", dokuz")

şeklinde iki tane dizinin tanımlanması gerekmektedir. Bu dizi elemanları çağırılırken de eleman indisi olarak

- *Onlar* için: Sayının 10'a bölünüp tam kısmının (*Tamsayı bölme*),
- *Birler* için: Sayının 10'a bölünüp kalan kısmının (*Mod alma*)

mutlak değeri kullanılabilir.



Örneğin klavyeden 17 girilirse

- 17/10 işleminin tam kısmı: 1
- 17 mod 10 işleminin sonucu: 7

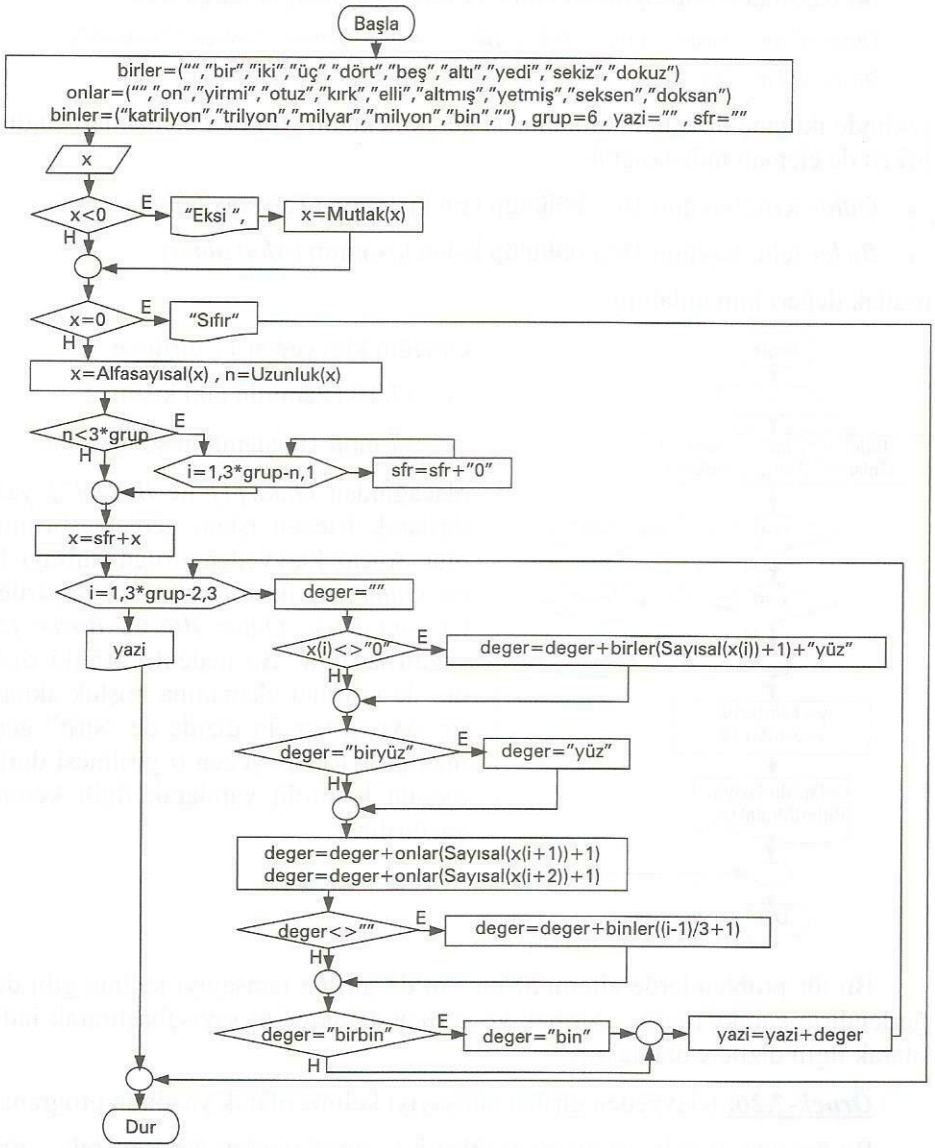
olacağından *Onlar[1]* ile *Birler[7]* yazdırılarak istenen işlem gerçekleştirilmiş olur. Ancak klavyeden girilen tamsayı 30 ise *Onlar [3]* ile *Birler [0]* veya girilen tamsayı 5 ise *Onlar [0]* ile *Birler [5]* yazdırılacaktır. Bu nedenle her iki dizinin de sıfırıncı elemanına boşluk aktarılır. Ayrıca her iki dizide de "sıfır" geçmediği için, klavyeden 0 girilmesi durumunun kontrolü yapılarak ilgili kelime yazdırılır.

Bu tür problemlerde alternatif bir yol da girilen tamsayıyı kelime gibi değerlendirip karakterlerine ayırmak ve ayrılan karakterleri sayısallaştırarak indis olarak ilgili dizilere atamaktır.

Örnek-7.26: Klavyeden girilen tamsayıyı kelime olarak yazdıran program.

Bu örnekte yazıyla yazdırma problemini - öncekilerden farklı olarak – girilen tamsayıyı kelime olarak işleyerek çözelim. Bir ve iki basamaklı tamsayılar için iki tane dizi tanımlamak gerekmekteydi. Ancak üç basamaklı sayılar için sadece "yüz" kelimesinin, dört basamaklı sayılar için "bin" kelimesinin, altı basamaklı sayılar için "milyon" kelimesinin vb. tanımlanması yeterlidir (Neden?). Bu durumda girilen sayıyı üçlü gruplar olarak işleyip özel tanımlayıcı kelimeleri arkalarına ekleyerek sonuca ulaşmış oluruz. Aşağıda katrilyona kadar

tamsayıların yazdırılması sunulmuştur. Aralığın daha da genişletilmesi gerekirse sadece "grup" değişkeninin arttırılması ve "binler" dizisine yeni kelimelerin eklenmesi yeterli olacaktır.

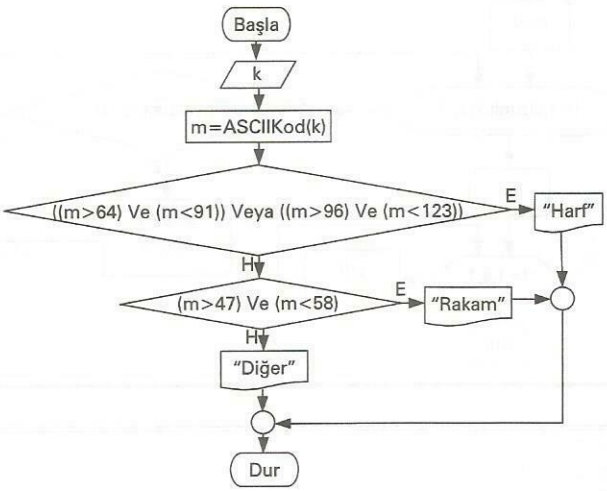


Örnek-7.27: Klavyeden girilen karakterin (basılan tuşun) türünü (harf, rakam, simge (diğer)) tespit eden program.

Programlama dillerinin bazılarında, basılan tuşun türünü tespit eden hazır komutlar vardır. Bu problemin çözümü ASCII kodlar kullanılarak kolaylıkla gerçekleştirilebilir. ASCII kod tablosu incelenirse (EK-A)

- Harflerin aralığı: A-Z → 65-90 , a-z: → 97-122
- Rakamların aralığı: 0-9 → 48-57

olduğu görülür.



Örnek-7.28: Klavyeden girilen bir cümlede, belirtilen harften kaç tane olduğunu bulup yerlerini yazdıran program.

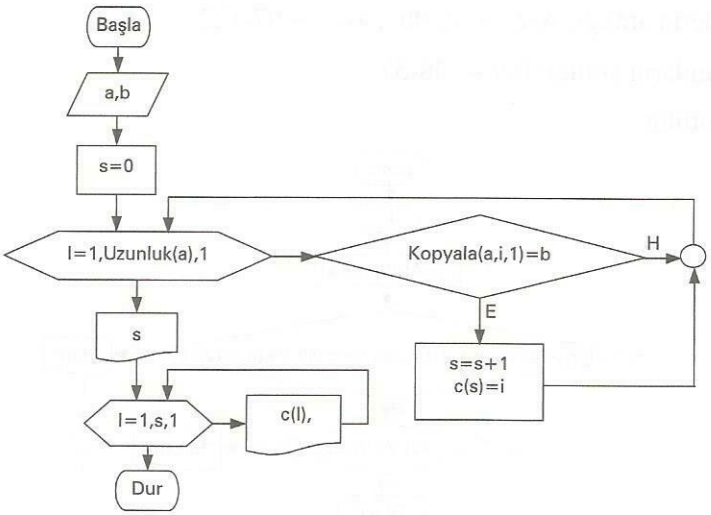
Bu programın akış diyagramı çizilirken programlama dillerinde olan ve *'Kopyala'* olarak adlandırılan bir komut kullanılacaktır. Eğer, A bilgisi *"Sakarya"* ise

- Kopyala(A,1,2)'nin sonucu : *Sa*
- Kopyala(A,3,3)'un sonunu : *kar*
- Kopyala(A,2,4)'un sonucu : *akar*

şeklinde olur. Yani *'Kopyala'* komutundaki parametrelerden ikincisi, kopyalanacak (alınacak) karakterin/karakterlerin başlangıç konumunu; üçüncüsü ise kopyalanacak (alınacak) karakterlerin sayısını belirtir. Ayrıca *'Uzunluk'* adlı diğer bir komut da alfasayısal bilginin (kelime, cümle vs.) kaç karakterden oluştuğunu vermektedir.

Bu iki komut kullanılarak bir kelimenin veya cümlenin karakterlerini (harflerini) tek tek almak için izlenebilecek en basit yollardan birisi 1'den kelimenin/cümlenin karakter sayısına (uzunluğuna) kadar bir döngü açıp "Kopyala" komutuyla döngü değişkeninin konumundan başlayarak birer karakter almaktır. Ayrıca alfasayısal bilgiler, bellekte dizi olarak saklandıklarından; birçok programlama dilinde karakterleri almak, dizi elemanını çağırmaktan ibarettir. Yani "`Kopyala(A,i,1)`" ile "`A(i)`" aynı görevde kullanılabilmektedir.

Programa dönülürse, cümlenin karakterleriyle belirtilen harf tek tek karşılaştırılır ve eşit olanlar sayılarak yerleri (konumları) C dizisine aktarılır.



C

```
/* Cümle-harf*/
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{   char a[200],b; int c[100],i,s=0;
    clrscr();printf("Cümleyi giriniz => ");gets(a);
    printf("Harfi giriniz => ");scanf("%s",&b);printf("\n");
    for(i=0;i<strlen(a);i++) { if (a[i]==b) {s++;c[s-1]=i+1;} }
    printf("Belirtilen harften %u tane vardır.\nYerleri:\n",s);
    for(i=0;i<s;i++) printf("%u\t",c[i]); getch(); }
```

C++

```
// Cümle-harf
#include <iostream>
using namespace std;
int main()
{   string a; char b; int c[100],i,s=0;
    system("cls"); cout << "Cümleyi giriniz => "; getline(cin, a);
    cout <<"Harfi giriniz => "; cin >> b;
    for(i=0;i<a.size();i++) { if(a[i]==b) { s++;c[s-1]=i+1; }}
    cout<<"\nBelirtilen harften "<<s<<" tane vardır.\nYerleri:\n";
    for(i=0;i<s;i++) cout << c[i] << "\t";
    system("pause");return 0; }
```

```
Cümleyi giriniz => Sakarya Üniversitesi
Harfi giriniz => r

Belirtilen harften  2  tane vardır.
Yerleri:
 5        14
```

C#

```
// Cümle - harf
using System;
namespace ornek728
{ class Program
   { static void Main(string[] args)
      {int i,s=0;String a;char b;int[] c=new int[100];
       Console.Write("Cümleyi giriniz => "); a = Console.ReadLine();
       Console.Write("Harfi giriniz => ");
       b = Convert.ToChar(Console.ReadLine());
       for (i=0;i<a.Length;i++) { if(a[i]==b){ s++;;c[s-1]=i+1;} }
       Console.WriteLine("\nBelirtilen harften {0} tane var",s);
        Console.WriteLine("Yerleri:");
        for(i=0;i<s;i++)  Console.Write("{0}\t",c[i]);
       Console.ReadLine();       }       }     }
```

JAVA

```
// Cümle-harf
import java.util.Scanner;
public class ornek728 {
public static void main(String[] args)
{  int i,s=0; String a; char b; int c[]=new int[100];
   Scanner klavye=new Scanner(System.in);
   System.out.print("Cümleyi giriniz => ");
   a=klavye.nextLine();
   System.out.print("Harfi giriniz => ");
   b=klavye.nextLine().charAt(0);
   for (i=0;i<a.length();i++) { if(a.charAt(i)==b){ s++;;c[s-1]=i+1;} }
   System.out.println("\nBelirtilen harften "+s+" tane var");
   System.out.println("Yerleri:");
   for(i=0;i<s;i++)  System.out.print(c[i]+"\t"); }}
```

MATLAB

```
% Cümle-harf
clear all; clc; a=input('Cümleyi giriniz => ','s');
b=input('Harfi giriniz => ','s'); c=findstr(a,b);
fprintf('\nBelirtilen harften %d tane vardır.\nYerleri:\n',length(c));
disp(c)
```

PYTHON

```
# Cümle-harf
import re
a=input("Cümleyi giriniz => ")
b=input("Harfi giriniz => ")
c=[i.start() for i in re.finditer(b,a)]
print("\nBelirtilen harften %d tane vardır.\nYerleri:"%len(c))
print([x+1 for x in c])
```

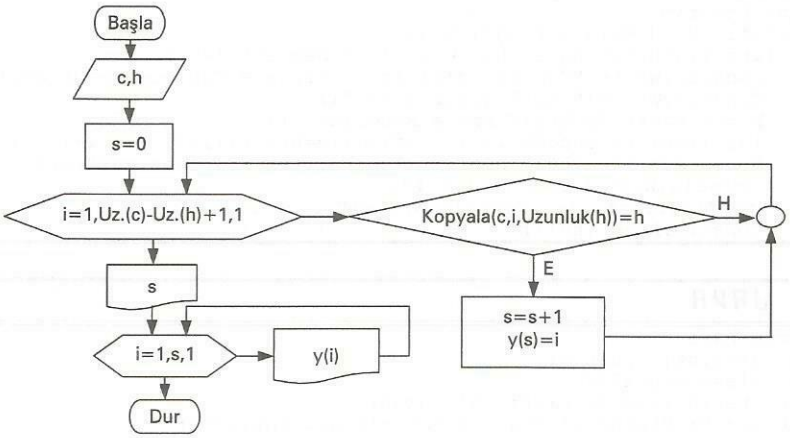
Not

Kelime/cümlelerle ilgili problemlerde büyük-küçük harf sorunu yaşamamak için girilen veri, program başında büyük veya küçük harfe dönüştürülüp işlenebilir.

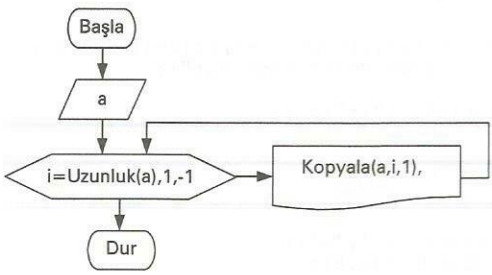
Örnek-7.29: Girilen bir cümlede, yine girilen bir heceden kaç tane olduğunu ve bunların yerlerini bulup yazdıran program.

Harf aratma probleminden farklı olarak burada karşılaştırma için cümleden birer karakter (harf) değil, girilen hecenin uzunluğu kadar karakter alınmaktadır.

Ayrıca heceleri almak için kullanılan döngü, bir ile cümledeki karakter sayısına kadar açılabildiği gibi, birden cümledeki karakter sayısının hecedeki karakter sayısınca eksiğinin bir fazlasına kadar açılabilir (Neden ?).



Örnek-7.30: Klavyeden girilen bir cümleyi, tersten yazdıran program.



Bir önceki örnekte olduğu gibi yine '*Kopyala*' ve '*Uzunluk*' komutları kullanılarak cümlenin sonundan başına doğru döngü açılır ve tek tek alınan karakterler yan yana yazdırılır.

C

```
/* Cümlenin tersten yazdırılması */
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{ char a[255]; int i; clrscr();printf("Cümleyi giriniz: ");gets(a);
  printf("\nCümlenin tersten yazımı:\n");
  for(i=strlen(a)-1;i>-1;i--) printf("%c",a[i]);
  /* veya   printf("%s\n",strrev(a)); */        getch();}
```

C++

```
// Cümlenin tersten yazdırılması
#include <iostream>
#include <cstring>
using namespace std;
int main()
{   char a[255]; system("cls"); cout << "Cümleyi giriniz: "; gets(a);
    cout << "\nCümlenin tersten yazımı:\n";
    for(int i=strlen(a)-1;i>=0;i--) cout << a[i];
    // veya  cout << strrev(a);       system("pause");return 0; }
```

C#

```
// Cümlenin tersten yazdırılması
using System;
namespace ornek730
{ class Program
   { static void Main(string[] args)
      { String a;
        Console.Write("Cümleyi giriniz: ");a=Console.ReadLine();
        Console.WriteLine("\nCümlenin tersten yazımı:");
        for(int i=a.Length-1;i>=0;i--) Console.Write(a[i]);
        // veya  char[] dizi = a.ToCharArray();
        //  Array.Reverse(dizi); Console.WriteLine(dizi);
        Console.ReadLine();    }    }    }
```

```
Cumleyi giriniz: Geyve Meslek Yuksek Okulu

Cumlenin tersten yazimi:
ulukO keskuY kelseM evyeG
```

JAVA

```
// Gümlenin tersten yazdırılması
import java.util.Scanner;
public class ornek730 {
public static void main(String[] args)
{  String a; Scanner klavye=new Scanner(System.in);
   System.out.print("Cümleyi giriniz: "); a=klavye.nextLine();
  // veya String ters=new StringBuffer(a).reverse().toString();
   System.out.println("\nCümlenin tersten yazımı: ");
   for(int i=a.length()-1;i>=0;i--)
     System.out.print(a.charAt(i));
  // veya System.out.println(ters);      }}
```

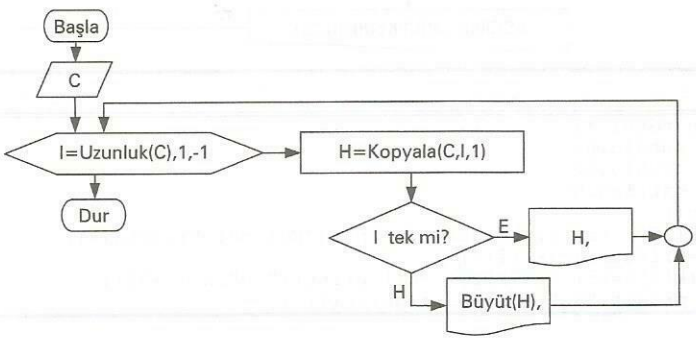
MATLAB

```
% Cümlenin tersten yazdırılması
clear all; clc; a=input('Cümleyi giriniz: ','s');
fprintf('\nCümlenin tersten yazımı:\n%s\n',a(end:-1:1));
```

PYTHON

```
# Cümlenin tersten yazdırılması
a=input("Cümleyi giriniz: ")
print("\nCümlenin tersten yazımı:\n%s"%a[::-1])
```

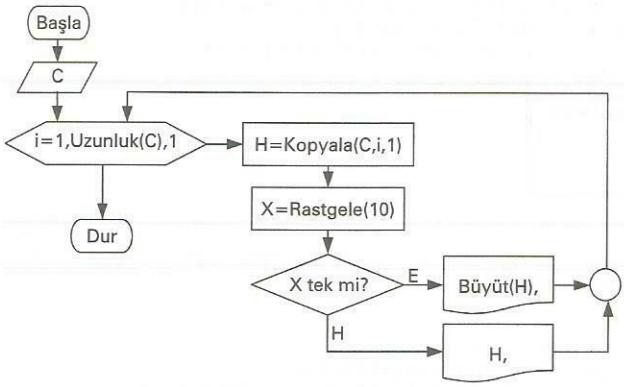
Örnek-7.31: Girilen bir cümleyi; *tersten* ve harflerini *küçük/büyük* yazdıran program



Not

Programlama dillerinde harfleri veya harf dizilerini (kelime, cümle) büyük veya küçük harfe dönüştüren komutlar mevcuttur. Harf dönüşümü için diğer bir yol da harfi küçültmek için ASCII koduna 32 eklemek ve harfi büyütmek için de ASCII kodundan 32 çıkarmaktır. Çünkü "A" harfinin ASCII kodu 65 ve "a" harfinin de ASCII kodu 97'dir.

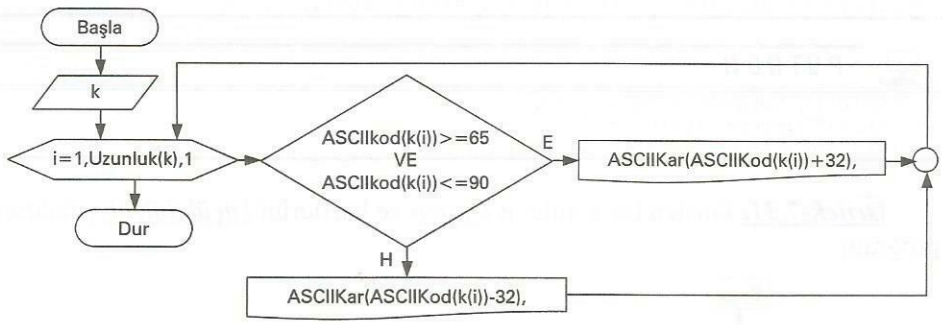
Örnek-7.32: Klavyeden küçük harflerle girilen bir cümlenin harflerini rastgele büyüklükte yeniden yazdıran program.



Büyüklük-küçüklük için rastgele bir tamsayı üretilir ve teklik-çiftlik durumuna göre cümlenin ilgili harfi büyütülür veya aynen yazdırılır. Diğer bir yol da cümlenin tamamını büyük harfe dönüştürüp, üretilen rastgele tamsayının tek/çift olmasına göre küçük veya büyük harfli halindekini yazdırmaktır.

Örnek-7.33: Klavyeden rastgele büyüklükte harfle girilen kelimenin harflerinin büyüklüklerinin tersini alarak (büyükse küçük, küçükse büyük yaparak) yeniden yazdıran program.

Bu programda kelimenin en fazla 25 harfli olduğu ve Türkçe karakterler içermediği kabul edilecek, ayrıca harf büyütüp küçültme komutları kullanılmayacaktır.



C

```
/* Harfin tersi */
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{  char k[25]; int i; clrscr();printf("Kelime: ");gets(k);
   for(i=0;i<strlen(k);i++)
    if ((k[i]>=65) && (k[i]<=90)) printf("%c",k[i]+32);
    else printf("%c",k[i]-32); getch(); }
```

C++

```
// Harfin tersi
#include <iostream>
#include <cstring>
using namespace std;
main()
{ char k[25]; system("cls"); cout << "Kelime: "; gets(k);
  for(int i=0;i<strlen(k);i++)
    if((k[i]>=65) && (k[i]<=90)) cout<< (char)(k[i]+32);
    else cout<<(char)(k[i]-32); system("pause"); return 0; }
```

C#

```
// Harfin tersi
using System;
namespace ornek733
{ class Program
   { static void Main(string[] args)
     { String k; Console.Write("Kelime: ");k=Console.ReadLine();
       for (int i=0; i<k.Length; i++)
         if ((k[i]>=65) && (k[i]<=90)) Console.Write((char)(k[i]+32));
         else Console.Write((char)(k[i]-32)); Console.ReadLine(); }}}
```

JAVA

```
// Harfin tersi
import java.util.Scanner;
public class ornek733 {
public static void main(String[] args)
{  String k; Scanner klavye=new Scanner(System.in);
   System.out.print("Kelime: ");k=klavye.nextLine();
   for(int i=0;i<k.length();i++)
    if ((k.charAt(i)>=65) && (k.charAt(i)<=90))
       System.out.print((char)(k.charAt(i)+32));
    else System.out.print((char)(k.charAt(i)-32)); }}
```

```
Kelime: BuRsa
bUrSA
```

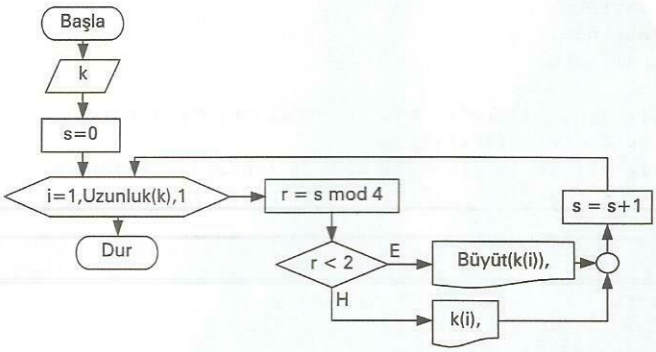
MATLAB

```
% Harfin tersi
clear all; clc; k=input('Kelime: ','s');
for i=1:length(k)
  if ((double(k(i))>=65) && (double(k(i))<=90)) fprintf('%c',k(i)+32);
  else fprintf('%c',k(i)-32);   end; end
```

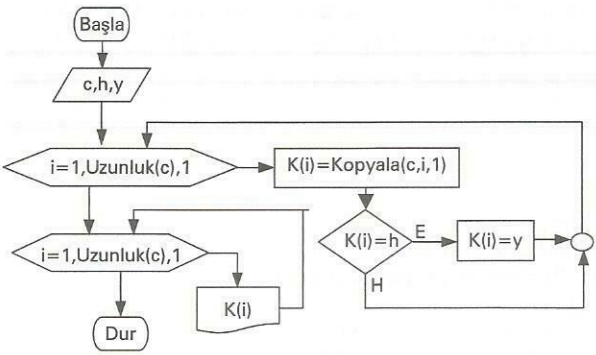
PYTHON

```
# Harfin tersi
k=input("Kelime: ")
for i in range(len(k)):
   if ((ord(k[i])>=65) & (ord(k[i])<=90)):
      print("%c"%(ord(k[i])+32),end='')
   else:
      print("%c"%(ord(k[i])-32),end='')
```

Örnek-7.34: Klavyeden küçük harflerle girilen bir kelimenin harflerini iki büyük – iki küçük harf şeklinde yaparak tekrar yazdıran program.



Örnek-7.35: Klavyeden girilen bir cümlede; belirtilen karakteri, yeni değeriyle değiştirip (Microsoft Word'te "Düzen" menüsündeki "*Değiştir*" seçeneği) yeniden yazdıran program.



İlk karakterden son karaktere kadar döngü açılıp, değiştirilecek karakter bulunduğunda dizinin ilgili elemanına yeni değeri aktarılır.

C

```
/* Harf değiştirme */
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{  char c[255],h,y,k[255]; int i;
   clrscr();printf("Cümleyi giriniz: ");gets(c);
   printf("\nDeğiştirilecek harfi giriniz: ");h=getche();
   printf("Harfin yeni değerini giriniz: ");
   y=getche();printf("\n");printf("\nYeni cümle -> ");
   for(i=0;i<strlen(c);i++)
     if (c[i]==h) k[i]=y;else k[i]=c[i];
   for(i=0;i<strlen(c);i++) printf("%c",k[i]);    getch();}
```

```
Cümleyi giriniz: Bılgısayar programlama
dılleri

Değiştirilecek harfi giriniz: ı
Harfin yeni değerini giriniz: i

Yeni cümle -> Bilgisayar programlama dilleri
```

C++

```
// Harf değiştirme
#include <iostream>
#include <cstring>
using namespace std;
char c[255],h,y,k[255]; int i;
main()
{   system("cls"); cout << "Cümleyi giriniz: "; gets(c);
    cout << "\nDeğiştirilecek harfi giriniz: "; cin >> h;
    cout << "\nHarfin yeni değerini giriniz: "; cin >> y;
    cout << "\nYeni cümle => ";
    for(i=0;i<strlen(c);i++)
      if(c[i]==h) k[i]=y; else k[i]=c[i];
    for(i=0;i<strlen(c);i++) cout << k[i]; system("pause"); return 0;}
```

C#

```
// Değiştirme
using System;
namespace ornek735
{ class Program
   { static void Main(string[] args)
     { String c; char h, y;
       Console.Write("Cumleyi giriniz: ");c=Console.ReadLine();
       Console.Write("\nDeğiştirilecek harfi giriniz: ");
       h = Convert.ToChar(Console.ReadLine());
       Console.Write("Harfin yeni değerini giriniz: ");
       y = Convert.ToChar(Console.ReadLine());
       Console.WriteLine("\nYeni cümle-> {0}", c.Replace(h, y));
       Console.ReadLine();          }    }    }
```

JAVA

```
// Değiştirme
import java.util.Scanner;
public class ornek735 {
public static void main(String[] args)
{  String c; char h,y; Scanner klavye=new Scanner(System.in);
   System.out.print("Cümleyi giriniz: "); c=klavye.nextLine();
   System.out.print("\nDeğiştirilecek harfi giriniz: ");
   h=klavye.nextLine().charAt(0);
   System.out.print("Harfin yeni değerini giriniz: ");
   y=klavye.nextLine().charAt(0);
   System.out.println("\nYeni cümle-> "+c.replace(h,y)); }}
```

MATLAB

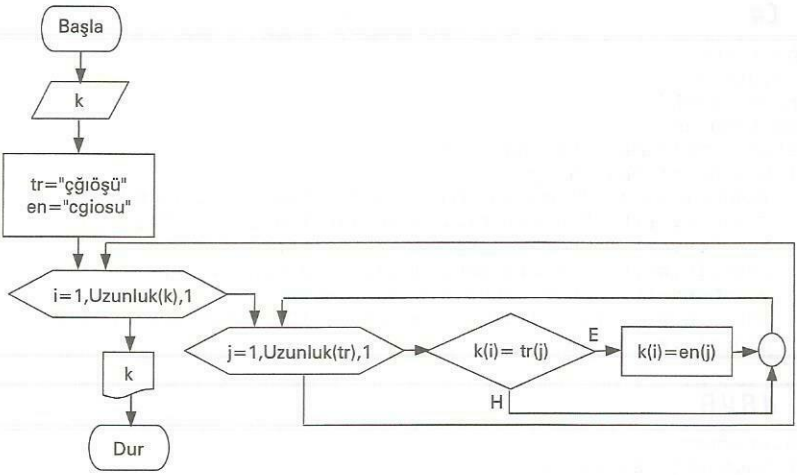
```
% Harf değiştirme
clear all; clc; c=input('Cümleyi giriniz: ','s');disp(' ');
h=input('Değiştirilecek harfi giriniz: ','s');
y=input('Harfin yeni değerini giriniz: ','s');
k=strrep(c,h,y); fprintf('\nYeni cümle -> %s\n',k);
```

PYTHON

```
# Harf değiştirme
c=input("Cümleyi giriniz: "); h=input("Değiştirilecek harfi giriniz: ")
y=input("Harfin yeni değerini giriniz: ")
print("\nYeni cümle -> %s"%c.replace(h,y))
```

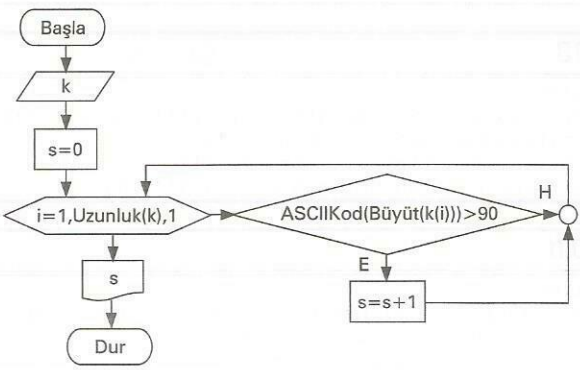
Örnek-7.36: Klavyeden küçük harflerle girilen kelimedeki Türkçe harfleri değiştiren/kaldıran - yani "ç→c", "ı→i" gibi dönüştüren - program.

Öncelikle Türkçe'ye özgü olan harfler "tr" dizisine ve buna karşılık değiştirilecek harfler de "en" dizisine aktarılmaktadır. Sonra kelimenin ilk harfinden son harfine (uzunluğuna) kadar döngü açılır ve sıradaki harfin, "tr" dizisinde olup olmadığı kontrol edilir. Eğer yoksa değiştirilmeden, varsa ise "en" dizisindeki karşılığı aktarılarak döngü tamamlanır. Aynı programda kelimenin ilk harfinden son harfine kadar döngü açılır, döngü içinde altı tane karşılaştırma yapılır ve ilgili harfi değişikliği gerçekleştirilerek de çözüme ulaşılabilir. Klavyeden girilen kelimenin harfleri rastgele büyüklükte girildiğinde akış diyagramında nasıl bir değişiklik olmalıdır?

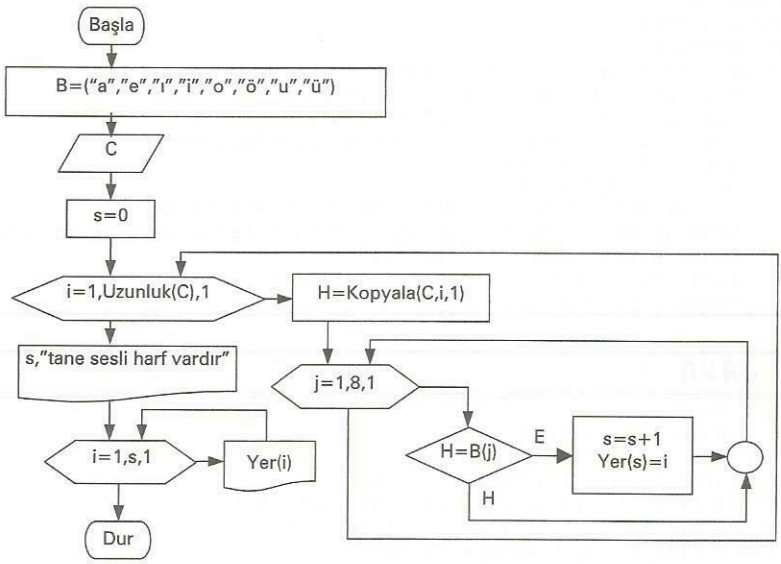


Örnek-7.37: Klavyeden girilen kelimede kaç tane İngiliz alfabesinde olmayan harf olduğunu tespit eden program.

ASCII tablosunda "A"→65 ve "Z"→90 ile kodlanmıştır. Dolayısıyla kelimenin tüm harflerin ASCII kodlarının bu aralığa düşüp düşmediği kontrol edilerek çözüme ulaşılabilir.



Örnek-7.38: Klavyeden küçük harflerle girilen bir cümledeki sesli harflerin sayısını ve cümledeki yerlerini bulan program.



C

```
/* Sesli harfler */
#include <stdio.h>
#include <conio.h>
#include <string.h>
char C[255],B[8]={"aeıioöuü"}; int i,j,Yer[255],s=0;
void main()
{  clrscr();printf("Cümle: ");gets(C);
   for(i=0;i<strlen(C);i++)
     { for(j=0;j<=7;j++) { if (C[i]==B[j]) {s++;Yer[s-1]=i+1;break;}}}
   printf("\nCümlede %u tane sesli harf vardır.\nYerleri: ",s);
   for(i=0;i<s;i++) printf("%u-",Yer[i]); getch(); }
```

C++

```
// Sesli harfler
#include <iostream>
using namespace std;
string C,B="aeıioöuü"; int i,j,yer[255],s=0;
int main()
{   system("cls"); cout << "Cumle: "; getline(cin,C);
    for(i=0;i<C.size();i++)
      { for(j=0;j<8;j++) if (C[i]==B[j]) { s++;yer[s-1]=i+1;break; }}
   cout << "\nCumlede "<< s << " tane sesli harf vardir.";
   cout << "\nYerleri:"; for (i=0;i<s;i++) cout << yer[i] <<" - ";
   system("pause");return 0; }
```

C#

```
// Sesli harfler
using System;
namespace ornek738
{ class Program
   { static void Main(string[] args)
      { int i,j,s=0; String c; int[] yer=new int[255];
        char[] b={'a','e','ı','i','o','ö','u','ü'};
        Console.Write("Cümle: ");c=Console.ReadLine();
        for(i=0;i<c.Length;i++)
        { for(j=0;j<8;j++)
            if (b[j]==c[i]) { s++; yer[s-1]=i+1; break; }}
        Console.WriteLine("\nCümlede {0} tane sesli harf vardır.",s);
        Console.WriteLine("\nYerleri:");
        for(i=0;i<s;i++) Console.Write("{0}-",yer[i]);
        Console.ReadLine();   }    }    }
```

JAVA

```
// Sesli harfler
import java.util.Scanner;
public class ornek738 {
public static void main(String[] args)
{  int i,j,s=0; String c; int yer[]=new int[255];
   char B[]={'a','e','ı','i','o','ö','u','ü'};
   Scanner klavye=new Scanner(System.in);
   System.out.print("Cümle: ");c=klavye.nextLine();
   for (i=0;i<c.length();i++)
       { for(j=0;j<8;j++) if(c.charAt(i)==B[j]){ s++;yer[s-1]=i+1;}}
   System.out.println("\nCümlede "+s+" tane sesli harf vardır.");
   System.out.println("\nYerleri:");
   for(i=0;i<s;i++) System.out.print(yer[i]+"-"); }}
```

```
Cümle: algoritma geliştirme ve programlamaya giriş

Cümlede 16 tane sesli harf vardır.

Yerleri:
1-4-6-9-12-14-17-20-23-27-30-33-35-37-40-42-
```

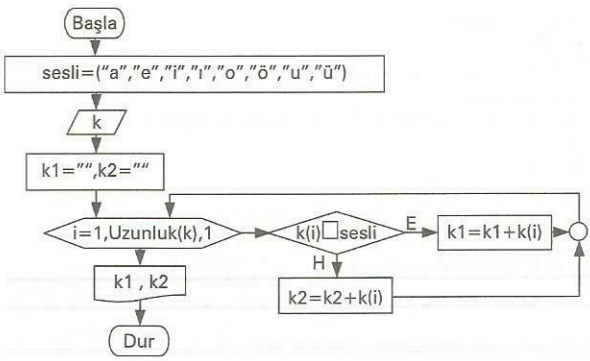
MATLAB

```
% Sesli harfler
clear all; clc; b='aeıioöuü' ; s=0; c=input('Cümle: ','s');
for i=1:length(c)
    if ismember(c(i),b)
        s=s+1; yer(s)=i;
    end;end;
fprintf('\nCümlede %d tane sesli harf vardır.\n\nYerleri:\n',s);
fprintf('%d\t',yer);
```

PYTHON

```
# Sesli harfler
b="aeıioöuü"; s=0; yer=[]; c=input("Cümle: ")
for i in range(len(c)):
    if (c[i] in b):
        s+=1; yer.append(i+1)
print("\nCümlede %d tane sesli harf vardır.\nYerleri:"%s)
print(yer)
```

Örnek-7.39: Klavyeden küçük harflerle girilen bir kelimenin sırasıyla sesli harflerinden k1 ve sessiz harflerinden k2 kelimesini oluşturup ekrana yazdıran program.



Bu örnekte ise öncekinden farklı olarak; girilen kelimenin ilk harfinden son harfine kadar bir döngü açılır ve eğer sıradaki harf sesli ise k1, sessiz ise k2 değişkenine ardışık olarak eklenerek (toplanarak) ilgili kelimeler oluşturulur.

C

```
/* Sesli sessiz harflerden kelime */
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{  char sesli[]="aeıioöuü",k[50],k1[50]="",k2[50]="";
   int i,j,kon,s1=-1,s2=-1;
   clrscr();printf("Küçük harflerle kelime: "); gets(k);
   for(i=0;i<strlen(k);i++)
     { kon=0;for(j=0;j<8;j++) if (k[i]==sesli[j]) { kon=1; break; }
       if (kon==1) {s1++;k1[s1]=k[i];} else {s2++;k2[s2]=k[i];} }
   printf("\nSesli harflerden kelime: %s\n",k1);
   printf("Sessiz harflerden kelime: %s\n",k2);     getch();    }
```

C++

```
// Sesli sessiz harflerden kelime
#include <iostream>
#include <string.h>
using namespace std;
int main()
{char sesli[]="aeıioöuü",k[50],k1[50]="",k2[50]="";int kon,s1=-1,s2=-1;
 system("cls");cout<<"Kucuk harflerle kelime: ";cin>>k;
 for(int i=0;i<strlen(k);i++)
  {kon=0;for(int j=0;j<8;j++) if (k[i]==sesli[j]) { kon=1;break; }
        if (kon==1) {s1++;k1[s1]=k[i];} else {s2++;k2[s2]=k[i];} }
 cout<<"\nSesli harflerden kelime: "<<k1;
 cout<<"\nSessiz harflerden kelime: "<<k2; system("pause");return 0;}
```

C#

```
// Sesli sessiz harflerden kelime
using System;
namespace ornek739
{ class Program
   { static void Main(string[] args)
      { int kon; String k,k1="",k2="";
        char[] sesli={'a','e','ı','i','o','ö','u','ü'};
        Console.Write("Küçük harflerle kelime: ");k=Console.ReadLine();
        for(int i=0;i<k.Length;i++)
         { kon=0; for (int j=0; j<8; j++)
           if (k[i]==sesli[j]) {kon=1;break;}
           if (kon==1) k1+=k[i]; else k2+=k[i];    }
        Console.WriteLine("\nSesli harflerden kelime: {0}",k1);
        Console.WriteLine("Sessiz harflerden kelime: {0}",k2);
        Console.ReadLine();          }    }    }
```

JAVA

```
// Sesli sessiz harflerden kelime
import java.util.Scanner;
public class ornek739 {
public static void main(String[] args)
{  int kon; String k,k1="",k2="";
   char sesli[]={'a','e','ı','i','o','ö','u','ü'};
   Scanner klavye=new Scanner(System.in);
   System.out.print("Küçük harflerle kelime: ");k=klavye.nextLine();
   for (int i=0;i<k.length();i++)
     { kon=0;for(int j=0;j<8;j++)
       if(k.charAt(i)==sesli[j]){kon=1;break;}
       if (kon==1) k1+=k.charAt(i); else k2+=k.charAt(i);   }
   System.out.printf("\nSesli harflerden kelime: %s",k1);
   System.out.printf("\nSessiz harflerden kelime: %s",k2); }}
```

MATLAB

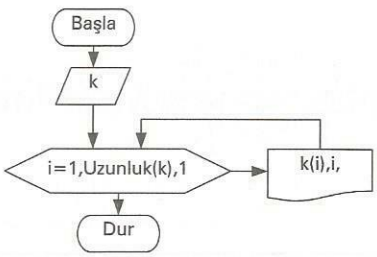
```
% Sesli sessiz harflerden kelime
clear all; clc; sesli='aeıioöuü' ; k1='';k2='';
k=input('Küçük harflerle kelime: ','s');
for i=1:length(k)
    if (ismember(k(i),sesli)) k1=strcat(k1,k(i));
    else k2=strcat(k2,k(i));
end; end; fprintf('\nSesli harflerden kelime: %s',k1);
fprintf('\nSesiz harflerden kelime: %s\n',k2);
```

```
Küçük harflerle kelime: istanbul

Sessli harflerden kelime: iau
Sessiz harflerden kelime: stnbl
```

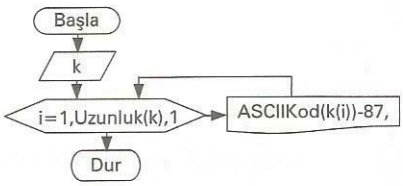
PYTHON

```
# Sesli sessiz harflerden kelime
sesli="aeıioöuü"; k1=""; k2=""; k=input("Küçük harflerle kelime: ")
for i in range(len(k)):
    if (k[i] in sesli):
        k1+=k[i]
    else:
        k2+=k[i]
print("\nSesli harflerden kelime: %s"%k1)
print("Sessiz harflerden kelime: %s"%k2)
```

Örnek-7.40: Klavyeden girilen bir kelimeyi, her harfinden sonra sırasını gösteren sayıyı da yanına koyarak (girilen kelime "Uludağ" ise ekran çıktısı "U1l2u3d4a5ğ6" şeklinde olmalı) tekrar yazdıran program.



Örnek-7.41: Klavyeden küçük harflerle girilen kelimenin tüm harflerini büyüterek ve alfabedeki "A..Z" harfleri yerine "10..35" tamsayılarıyla şifreleyip yazdıran program.



Örnek-7.42: Klavyeden girilen cümleyi; cümledeki tek ve çift konumda bulunan karakterlerin yerlerini karşılıklı değiştirerek şifreli şekilde yazdıran program.

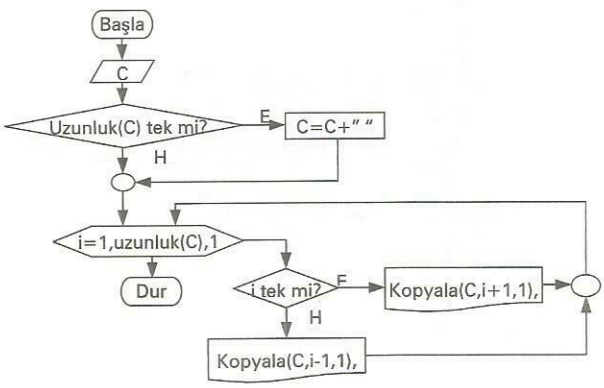
İlk adımda, klavyeden cümle girilir. Eğer cümledeki karakter sayısı çift ise, tek-çift değişiminde herhangi bir sorun oluşmaz. Örneğin "Pascal" kelimesinde karakter sayısı 6 olduğundan, sorunsuz şifreleme işlemi aşağıdaki gibidir.

	1 ↔ 2		3 ↔ 4		5 ↔ 6	
Normal	P	a	s	c	a	l
Şifrelenmiş	***a***	***P***	***c***	***s***	***l***	***a***

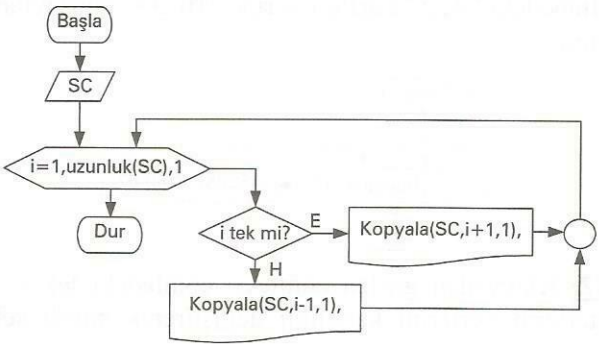
Ancak cümledeki karakter sayısı tek ise, son karakterin yer değiştireceği alternatif bir karakter yoktur. Bunun için karakter sayısı tek olan cümleye bir boşluk eklenir. Örneğin "Basic" kelimesinde 5 karakter olduğundan, 1 tane boşluk karakteri eklenir ve şifreleme işlemi aşağıdaki gibi olur.

	1 ↔ 2		3 ↔ 4		5 ↔ 6	
Normal	B	a	s	i	c	
Şifrelenmiş	***a***	***B***	***i***	***s***		***c***

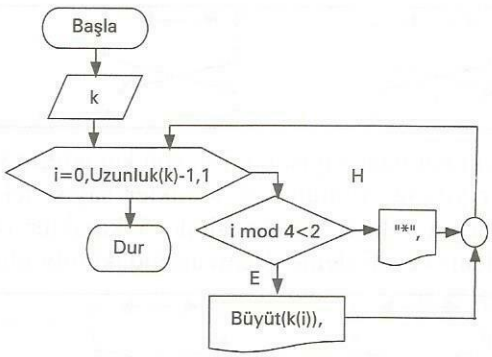
Tek ve çift konumdaki harfleri tespit etmek için 1'den cümle uzunluğuna kadar bir döngü (i) açılır. Eğer döngü değişkeni i tek ise cümledeki $(i+1)$. karakter, çift ise cümledeki $(i-1)$. karakter yazdırılır.



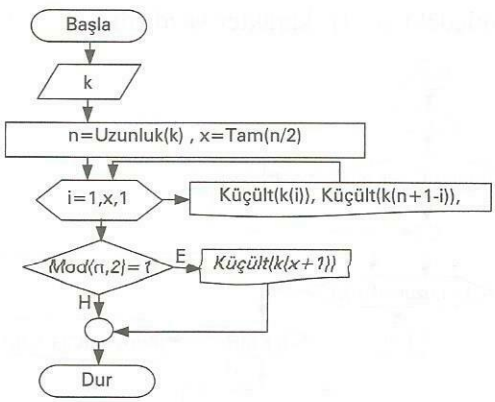
Örnek-7.43: Örnek-7.42'de şifrelenen cümleyi çözen program.



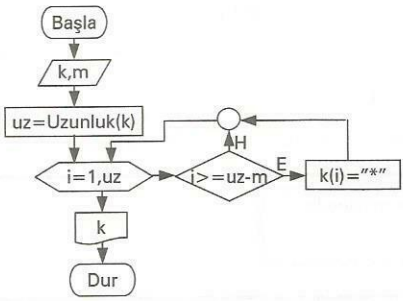
Örnek-7.44: Klavyeden küçük harflerle girilen kelimenin tüm harflerini büyüterek ve "iki harf açık, iki harf yıldızla kapalı" olarak yeniden yazdıran program.



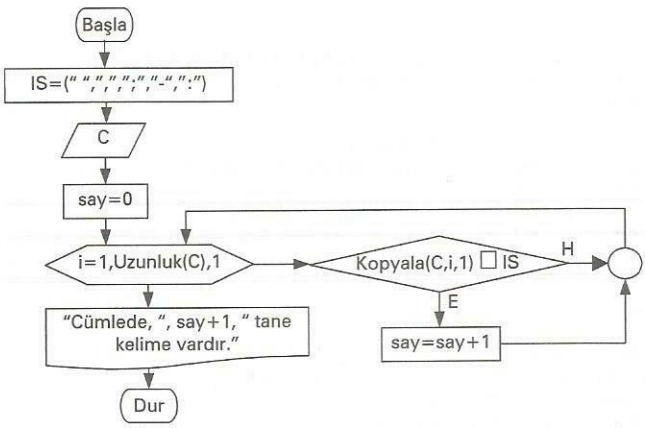
Örnek-7.45: Klavyeden büyük harflerle girilen kelimenin harflerini küçülterek ve sırasıyla bir başından (baştan sona doğru, soldan sağa) bir sonundan (sondan başa doğru, sağdan sola) alarak yeniden yazdıran program.



Örnek-7.46: Klavyeden girilen kelimenin belirtilen sayıdaki son harflerini "*" yapan program. Hata denetimi (örneğin 5 harfli kelime girildiğinde son 10 harfini "*" yapma gibi) yapacak şekilde örneği değiştiriniz.



Örnek-7.47: Klavyeden girilen bir cümlede kaç tane kelime olduğunu bulan program.



Cümledeki kelimeler birbirinden boşluk, virgül, noktalı virgül gibi karakterlerle ayrılırlar. Bu sembolleri sayarak, cümledeki kelime sayısına ulaşılabilir. Cümlede adı geçen sembollerden kaç tane varsa, bu sayının bir fazlası kadar kelime (ilk kelime için) vardır. Örneğin "*Gülme komşuna,gelir başına.*" cümlesinde 2 tane boşluk ve 1 tane virgül işareti vardır. Dolayısıyla kelime sayısı 2+1+1=4'tür. Bir diğer yol da; adı geçen sembollere, cümle sonlarında kullanılan nokta, ünlem işareti, soru işareti gibi karakterleri ekleyip saymaktır. Eğer problemde; girilen cümlede noktalama işaretleri kullanılmadığı belirtilirse, sadece boşlukları saymak yeterlidir.

C

```
/* Kelime sayısı */
#include <stdio.h>
#include <conio.h>
#include <string.h>
char c[255],is[5]={" ,;-:"}; int i,j,say=0;
void main()
{ clrscr();printf("Cümle: ");gets(c);
  for(i=0;i<strlen(c);i++)
    {for(j=0;j<5;j++)
       if (c[i]==is[j]) {say++;break; }}
  printf("\nCümlede %u tane kelime vardır.",say+1); getch(); }
```

C++

```
// Kelime sayısı
#include <iostream>
using namespace std;
string c,is=" ,;-:"; int i,j,say=0;
int main()
{   system("cls"); cout << "Cumle: "; getline(cin,c);
    for(i=0;i<c.size();i++)
      { for(j=0;j<5;j++)
          if(c[i]==is[j]) { say++; break; }}
    cout << "\nCumlede " << (say+1) <<" tane kelime vardir.";
    system("pause"); return 0; }
```

```
Cumle: Bilgisayar,elektronik bir makinedir.

Cumlede 4 tane kelime vardir.
```

C#

```
// Kelime sayısı
using System;
namespace ornek747
{ class Program
   { static void Main(string[] args)
    { int i,j,say=0; String c; char[] sim={' ',',',';','-',':'};
      Console.Write("Cümle: ");c=Console.ReadLine();
      for(i=0;i<c.Length;i++)
       { for(j=0;j<5;j++)
           if(sim[j]==c[i]) { say++;break;}}
      Console.WriteLine("\nCümlede {0} tane kelime vardır.", say+1);
      Console.ReadLine();  }    }   }
```

JAVA

```
// Kelime sayısı
import java.util.Scanner;
public class ornek747 {
public static void main(String[] args)
{  int i,j,say=0; String c; char is[]={' ',',',';','-',':'};
   Scanner klavye=new Scanner(System.in);
   System.out.print("Cümle: "); c=klavye.nextLine();
   for(i=0;i<c.length();i++)
       { for(j=0;j<5;j++)
           if(is[j]==c.charAt(i)) { say++;break;}}
   System.out.println("\nCümlede "+(say+1)+" tane kelime vardır."); }}
```

MATLAB

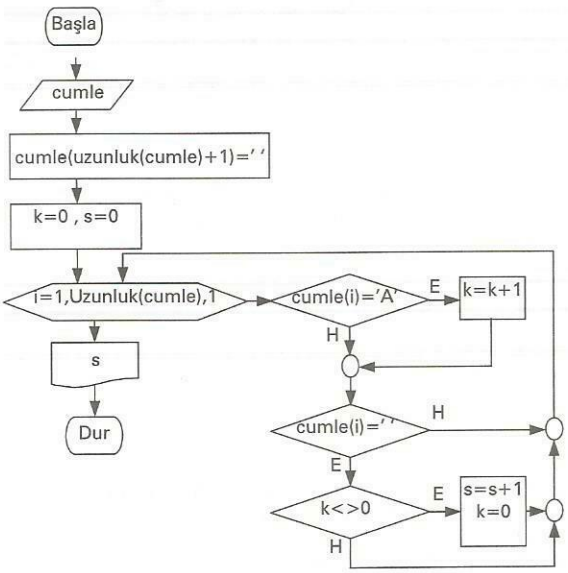
```
% Kelime sayısı
clear all; clc; is=' ,;-:' ; say=0; c=input('Cümle: ','s');
for i=1:length(c)
    if (ismember(c(i),is)) say=say+1;
end; end; fprintf('\nCümlede %d tane kelime vardır.\n',say+1);
```

PYTHON

```
# Kelime sayısı
ni=" ,;-:"; s=0; c=input("Cümle: ")
for i in range(len(c)):
    if (c[i] in ni):
        s+=1
print("\nCümlede %d tane kelime vardır."%(s+1))
```

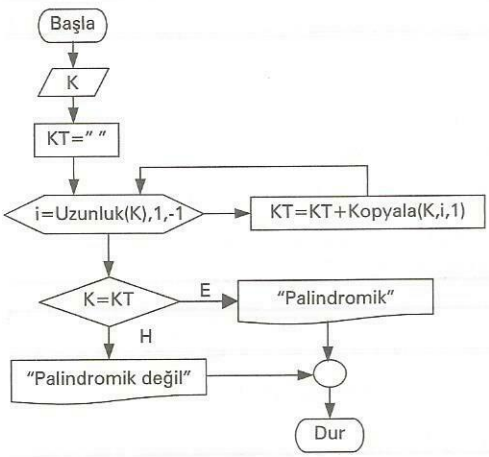
Akış diyagramında kolaylık olması için "$\in$" sembolü kullanılmıştır. Programlama dillerinde; bir elemanın, belirtilen kümenin elemanı olup olmadığını veren komut/ifadeler vardır. Bu komut/ifadeler yerine yukarıdaki kodlarda olduğu gibi döngü açılarak; belirtilen elemanın, dizide bulunup bulunmadığı arattırılabilir.

Örnek-7.48: Klavyeden büyük harflerle ve noktalama işaretleri kullanılmadan girilen bir cümlede, içinde "A" harfi olan kaç tane kelime olduğunu sayıp sonucu ekranda gösteren program.



Çözüm için cümledeki kelimeler ayrıştırılmalı ve kelime içinde "A" harfi olup olmadığına bakılmalıdır. Herhangi bir noktalama işareti kullanılmadığı için boşluklar kontrol edilerek "A" harfi geçip geçmediği denetlenir. Son kelimeden sonra boşluk olmadığından programın içinde, girilen cümlenin sonuna bir tane boşluk karakteri eklenebilir.

Örnek-7.49: Baştan ve sondan yazımları/okunuşları aynı olan kelimelere, "*palindromik (palindrom) kelime*"ler denir. Buna göre klavyeden küçük harflerle girilen kelimenin '*palindromik*' olup olmadığını tespit eden program.



Girilen kelimenin palindormik olup olmadığı tespit etmek için; tersten yazımı/okunuşu başka bir değişkene aktarılır ve girilen kelimeyle karşılaştırılır. Büyük /küçük harf ayrımından kaynaklanan hatanın oluşmaması için başlangıçta girilen kelimenin tüm harfleri büyütülebilir veya küçültülebilir.

C

```
/* Palindromik kelimeler */
#include <stdio.h>
#include <conio.h>
#include <string.h>
char k[255],kt[255];
void main()
{ clrscr();printf("Kelime: ");gets(k); strcpy(kt,k); strrev(kt);
  if (strcmp(k,kt)==0) printf("\nPalindromik");
  else printf("\nPalindromik değil"); getch(); }
```

C++

```
// Palindromik kelimeler
#include <iostream>
using namespace std;
main()
{  string k; system("cls"); cout << "Kelime: "; cin>>k; cout <<"\n";
   if(k==string(k.rbegin(),k.rend())) cout << "Palindromik" << endl;
   else cout << "Palindromik degil" << endl;
   system("pause"); return 0; }
```

C#

```
// Palindromik kelime
using System;
namespace ornek749
{ class Program
   { static void Main(string[] args)
     {  Console.Write("Kelime: "); String k = Console.ReadLine();
        char[] x = k.ToCharArray(); Array.Reverse(x);
        String kt=new String(x);
        if (k == kt) Console.WriteLine("Palindromik");
        else Console.WriteLine("Palindromik değil");
        Console.ReadLine();        }    }    }
```

```
Kelime: bursa
Palindromik değil
```

JAVA

```
// Palindromik kelime
import java.util.Scanner;
public class ornek749 {
public static void main(String[] args)
{  String k; Scanner klavye=new Scanner(System.in);
   System.out.print("Kelime: ");k=klavye.nextLine();
   // String kt = new StringBuffer(k).reverse().toString();
   String kt = new StringBuilder(k).reverse().toString();
   if(k.equals(kt)) System.out.println("\nPolindromik");
   else System.out.println("\nPolindromik değil"); }}
```

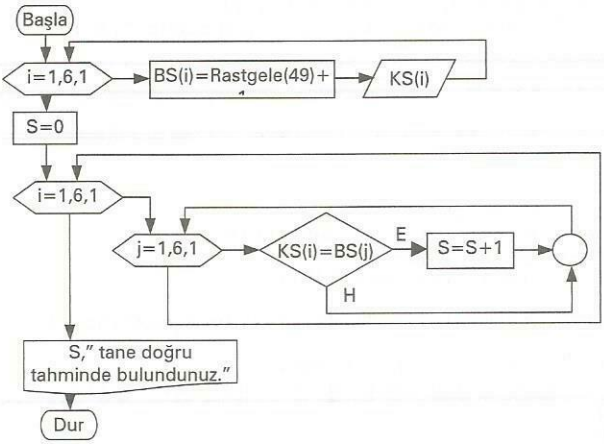
MATLAB

```
% Palindromik kelime
clear all; clc; k=input('Kelime: ','s'); kt=k(end:-1:1);
if (k==kt) fprintf('Palindromik\n');
else fprintf('Palindromik değil\n'); end
```

PYTHON

```
# Palindromik kelime
k=input("Kelime: "); kt=k[::-1]
if (k==kt):
    print("Palindromik")
else:
    print("Palindromik değil")
```

Örnek-7.50: Bilgisayar 1-49 arası rastgele 6 tane tamsayı üretmektedir. Kullanıcının 1-49 arası girdiği 6 tahminden kaçının doğru olduğunu veren program.



Bilgisayarın ürettiği rastgele tamsayılar, 6 elemanlı *BS* dizisine; kullanıcın girdiği tamsayılar da *KS* dizisine aktarılır. Bu aşamadan sonra problem, *KS* dizisinin elemanlarından kaç tanesinin *BS* dizisinin de elemanı olduğunu bulmaya dönüşmektedir.

C

```
/* Sayısal loto */
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int i,j,BS[6],KS[6],s=0;
void main()
{  clrscr();randomize();for(i=0;i<6;i++) BS[i]=random(49)+1;
   for(i=0;i<6;i++)
     {printf("%u. tahmininiz: ",i+1);scanf("%u",&KS[i]);}
   for(i=0;i<6;i++)
     for(j=0;j<6;j++) if (KS[i]==BS[j]) {s++;break}
   printf("\n%u tane doğru tahminde bulundunuz\n",s); getch(); }
```

C++

```
// Sayısal loto
#include <iostream>
#include <cstdlib>
using namespace std;
int i,j,BS[6],KS[6],s=0;
int main()
{  system("CLS");srand(time(NULL));for(i=0;i<6;i++) BS[i]=rand()%49+1;
   for(i=0;i<6;i++)
     { cout << i+1 <<". tahmininiz: "; cin >> KS[i]; }
   for(i=0; i<6;i++)
     for(j=0;j<6;j++) if(KS[i]==BS[j]) {s++;break;}
   cout <<"\n"<<s<<" tane dogru tahminde bulundunuz.";
   system("pause");return 0; }
```

C#

```
// Sayısal loto
using System;
namespace ornek750
{ class Program
   { static void Main(string[] args)
      {int i,j,s=0; int[] BS=new int[6]; int[] KS=new int[6];
       Random rastgele = new Random();
       for (i=0;i<6;i++) BS[i]=rastgele.Next(1,50);
       for (i=0;i<6;i++) {Console.Write("{0}. tahmininiz: ",i+1);
          KS[i]=Convert.ToInt32(Console.ReadLine());}
       for (i=0;i<6;i++)
        for (j=0;j<6;j++) if(KS[i]==BS[j]) {s++;break;}
      Console.WriteLine("\n{0} tane doğru tahminde bulundunuz.",s);
      Console.ReadLine(); } } }
```

JAVA

```
// Sayısal loto
import java.util.Scanner;
public class ornek750 {
public static void main(String[] args)
{  int i,j,s=0; int BS[]=new int[6]; int KS[]=new int[6];
   Scanner klavye=new Scanner(System.in);
   for (j=0;j<6;j++) BS[j]=(int)(Math.random()*49)+1;
   for (i=0;i<6;i++)
    {System.out.print((i+1)+". tahmininiz: ");KS[i]=klavye.nextInt();}
   for (i=0;i<6;i++)
     for (j=0;j<6;j++) if(KS[i]==BS[j]) {s++;break;}
   System.out.print("\n"+s+" tane doğru tahminde bulundunuz."); }}
```

```
1. tahmininiz: 3
2. tahmininiz: 11
3. tahmininiz: 25
4. tahmininiz: 33
5. tahmininiz: 39
6. tahmininiz: 44

5 tane doğru tahminde bulundunuz.
```

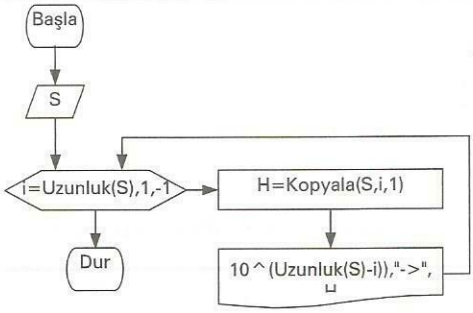
MATLAB

```
% Sayısal loto
clear all; clc; s=0; bs=randi(49,1,6);
for i=1:6
  fprintf('%d. tahmininiz: ',i); ks(i)=input(' '); end
for i=1:6
    if ismember(ks(i),bs)
        s=s+1;
    end; end; fprintf('\n%d tane doğru tahminde bulundunuz.\n',s);
```

PYTHON

```
# Sayısal loto
import random
s=0; ks=[]; bs=[]
for i in range(6):
    bs.append(random.randint(1,49))
for i in range(6):
    print("%d. tahmininiz: "%(i+1),end=''); ks.append(int(input()))
for i in range(6):
    if (ks[i] in bs):
        s+=1
print("\n%d tane doğru tahminde bulundunuz."%s)
```

Örnek-7.51: Girilen tamsayıyı basamaklarına ayrıştıran program.



Tamsayı kelime gibi alınarak uzunluğundan ilk karakterine kadar döngü açılır ve sıradaki alfasayısal rakam yazdırılır. Bu rakamları sayısallaştırmak için hazır komutlar kullanılabilir veya ASCII kodlarından 48 çıkarılabilir (Neden?).

C

```
/* Basamaklara ayrıştırma */
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <math.h>
void main()
{ char s[100]; int i; clrscr();printf("Tamsayı: ");scanf("%s",&s);
  printf("\nHaneler:\n");printf("--------\n");
  for(i=strlen(s)-1;i>=0;i--)
     printf("%0.0f -> %c\n",pow(10,strlen(s)-1-i),s[i]); getch();}
```

C++

```
// Basamaklara ayrıştırma
#include <iostream>
#include <cstring>
#include <math.h>
using namespace std;
int main()
{  char s[100]; system("cls"); cout << "Tamsayi: "; cin >> s;
   cout << "\nBasamaklar:\n";cout << "-----------\n";
   for(int i=strlen(s)-1;i>=0;i--)
   cout<<pow(10,strlen(s)-1-i)<<" -> "<<s[i]<<"\n";
   system("pause");return 0; }
```

C#

```
// Basamaklara ayrıştırma
using System;
namespace ornek751
{ class Program
   { static void Main(string[] args)
      { String s; Console.Write("Tamsayı: ");s=Console.ReadLine();
        Console.WriteLine("\nBasamaklar:");
        Console.WriteLine("-----------");
        for (int i=s.Length-1;i>=0;i--)
Console.WriteLine("{0:F0} -> {1}",(int)Math.Pow(10,s.Length-1-i),s[i]);
        Console.ReadLine();       }    }    }
```

JAVA

```
// Basamaklara ayrıştırma
import java.util.Scanner;
public class ornek751 {
public static void main(String[] args)
{  String s; Scanner klavye=new Scanner(System.in);
   System.out.print("Tamsayı: ");s=klavye.nextLine();
   System.out.println("\nBasamaklar: ");
   System.out.println("---------");
   for (int i=s.length()-1;i>=0;i--)
 System.out.println((int)Math.pow(10,s.length()-1-i)+"->"+s.charAt(i));
}}
```

```
Tamsayı: 2017

Basamaklar:
----------
1 -> 7
10 -> 1
100 -> 0
1000 -> 2
```

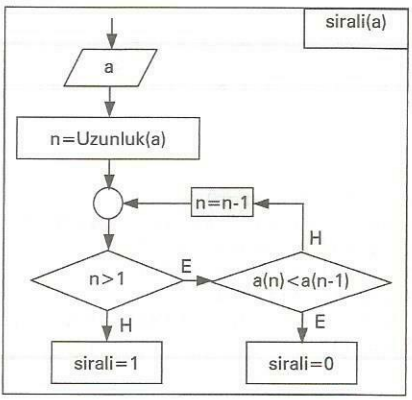
MATLAB

```
% Basamaklara ayrıştırma
clear all; clc; s=input('Tamsayı: ','s');
fprintf('\nBasamaklar\n----------\n'); n=length(s);
for i=n:-1:1
    fprintf('%d -> %c\n',10^(n-i),s(i));
end;
```

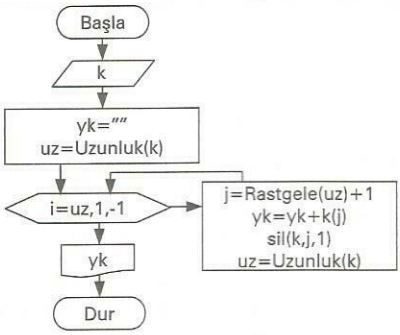
PYTHON

```
# Basamaklara ayrıştırma
s=input("Tamsayı: "); print("\nBasamaklar\n----------"); n=len(s)
for i in range(n-1,-1,-1):
    print("%d -> %c"%(10**(n-i-1),s[i]))
```

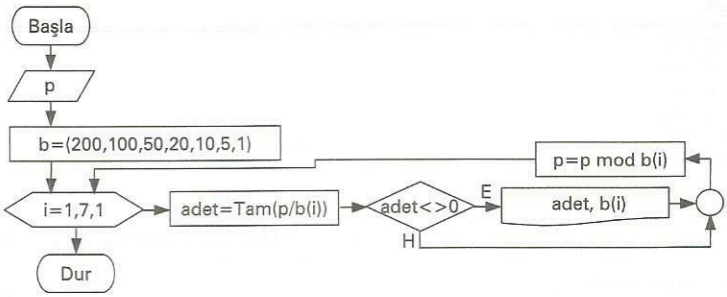
Örnek-7.52: Verilen bir dizinin küçükten büyüğe doğru sıralı olup olmadığını test eden fonksiyon/alt program.



Örnek-7.53: Klavyeden girilen kelimenin harflerini karıştıran program.

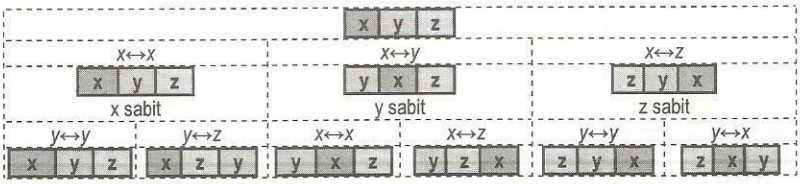


Örnek-7.54: Klavyeden girilen para miktarını banknotlara ayıran program.



Örnek-7.55: Klavyeden girilen kelimenin tüm permütasyonlarını listeleyen program.

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



C

```
/* Permütasyonlar */
#include <stdio.h>
#include <string.h>
#include <conio.h>
void permutasyon(char *a,int sol,int sag)
{ int i; char g;
  if(sol==sag) printf("%s\n",a);
  else { for(i=sol;i<=sag;i++)
         {  g=a[sol]; a[sol]=a[i]; a[i]=g; permutasyon(a,sol+1,sag);
            g=a[sol]; a[sol]=a[i]; a[i]=g; }}}
void main()
{ char k[100];
  clrscr();printf("Kelime: "); gets(k); printf("\nPermütasyonlar:\n");
  permutasyon(k,0,strlen(k)-1); getch(); }
```

C++

```
// Permütasyonlar
#include <iostream>
#include <string>
using namespace std;
void permutasyon(char *a,int sol,int sag)
{ int i; char g;
  if(sol==sag) printf("%s\n",a);
  else { for(i=sol;i<=sag;i++)
         { g=a[sol]; a[sol]=a[i]; a[i]=g; permutasyon(a,sol+1,sag);
           g=a[sol]; a[sol]=a[i]; a[i]=g; }}}
int main()
{ char k[100]; system("cls"); cout<<"Kelime: "; cin>>k;
  cout<<"\nPermütasyonlar:\n"; permutasyon(k,0,strlen(k)-1);
  system("pause"); return 0; }
```

C#

```
// Permütasyonlar
using System;
namespace ornek755
{class Program {
  static void permutasyon(char[] a, int sol, int sag)
   { int i; char g;
     if (sol==sag) Console.WriteLine(a);
     else { for (i=sol;i<=sag;i++)
            { g=a[sol];a[sol]=a[i];a[i]=g;permutasyon(a,sol+1,sag);
              g=a[sol];a[sol]=a[i];a[i]=g; }}}
  static void Main(string[] args)
   { string k;Console.Write("Kelime: "); k = Console.ReadLine();
     Console.WriteLine("\nPermütasyonlar:");
     permutasyon(k.ToCharArray(), 0, k.Length-1);
     Console.ReadLine(); }}}
```

```
Kelime: xyz

Permütasyonlar:
xyz
xzy
yxz
yzx
zyx
zxy
```

JAVA

```
// Permütasyonlar
import java.util.Scanner;
public class ornek755 {
 public static void permutasyon(char[] a, int sol, int sag)
  { int i; char g;
    if (sol==sag) System.out.println(a);
    else { for (i=sol;i<=sag;i++)
           { g=a[sol];a[sol]=a[i];a[i]=g; permutasyon(a,sol+1,sag);
             g=a[sol];a[sol]=a[i];a[i]=g; }}}
 public static void main(String[] args) {
  String k; Scanner klavye=new Scanner(System.in);
  System.out.print("Kelime: ");k=klavye.nextLine();
  System.out.printf("\nPermütasyonlar:\n");
  permutasyon(k.toCharArray(), 0, k.length()-1); }}
```

MATLAB

```
% Permütasyon
clear all; clc; k=input('Kelime: ','s');
fprintf('\nPermütasyonlar:\n'); permutasyon(k,1,length(k));
---
function permutasyon(a,sol,sag)
if (sol == sag)
    fprintf('%s\n',a);
else
   for i=sol:sag
       g=a(sol); a(sol)=a(i); a(i)=g; permutasyon(a,sol+1,sag);
       g=a(sol); a(sol)=a(i); a(i)=g;
   end; end;
```

PYTHON

```
# Permütasyon
import itertools
k=input("Kelime: "); print("\nPermütasyonlar:")
p=itertools.permutations(k)
for i in list(p):
    print(i)
```

Not

Bölüm 12 ve 13'teki sıralama ve arama algoritmalarını inceleyiniz.

SORULAR

1. Klavyeden girilen N elemanlı bir dizide her bir elemandan kaç tane olduğunu bulan programın akış diyagramını çizip yazılımını yapınız.

2. Klavyeden girilen N elemanlı bir A dizisinin aritmetik ortalamasını bulup, bu aritmetik ortalamadan küçük elemanları B dizisine ve büyük olan elemanları da C dizisine yerleştiren programın akış diyagramını çizip yazılımını yapınız.

3. Klavyeden girilen N elemanlı A, B, C dizilerini kullanarak D=A+B-C işlemiyle D dizisini oluşturup bu yeni diziyi büyükten küçüğe doğru sıralayan programın akış diyagramını çiziniz ve programını yazınız.

4. Klavyeden girilen N elemanlı bir A dizisinin elemanlarının karelerini alarak B dizisini; kareköklerini alarak C dizisini oluşturup bu yeni iki dizinin skaler çarpımını bulan programın akış diyagramını çizip yazılımını yapınız.

5. Klavyeden girilen bir A dizisinin elemanlarını ters sıra ve işarette B dizisine aktaran programın akış diyagramını çizip yazılımını yapınız.

6. Klavyeden girilen N değerine göre tek indisli elemanları '0', çift indisli elemanları '1' olan N elemanlı bir A dizisini oluşturan programın akış diyagramını çizip yazılımını herhangi bir programlama diliyle yapınız.

7. Klavyeden küçük harflerle girilen bir cümlede kelimelerinin sadece ilk harflerini büyütüp yeniden yazdıran programın akış diyagramını çizip yazılımını herhangi bir programlama diliyle yapınız.

8. Klavyeden büyük harflerle girilen bir cümlenin kelimelerini büyük-harfli/küçük-harfli şeklinde yeniden yazdıran programın akış diyagramını çiziniz ve bir dilde programını yazınız.

9. Klavyeden girilen bir cümle içinde yine klavyeden girilen bir kelimeyi arattıran programın akış diyagramını çizip yazılımını yapınız.

10. Klavyeden girilen bir cümleyi, her karakterinden ikişer tane alarak tekrar yazdıran programın akış diyagramını çizip yazılımını yapınız. Örneğin girilen cümle/kelime "BURSA" ise çıktı "BBUURRSSAA" şeklinde olacaktır.

11. Klavyeden girilen bir kelimenin harfleri arasına birer tane boşluk koyarak yeniden yazdıran programın akış diyagramını çizip yazılımını yapınız.

12. Klavyeden girilen bir cümleyi; önce mavi zemin üzerine sarı ile yazıp yazı rengini, cümlenin başından sonuna doğru 1 s aralıklarla kırmızıya dönüştüren programın akış diyagramını çizip yazılımını yapınız.

13. Girilen bir cümlede, kaç tane "a" ile biten kelime olduğunu bulan programın akış diyagramını çizip herhangi bir dilde kodlayınız.

14. Girilen ay adına göre, bunun yılın kaçıncı ayı olduğunu bulan programın akış diyagramını çizip yazılımını yapınız.

15. Sayısal olarak girilen doğum tarihini kelimelerle yazdıran programın akış diyagramını çizip kodlayınız.

16. Girilen doğum tarihine göre kişinin burcunu yazdıran programın akış diyagramını çizip kodlayınız.

17. Klavyeden " ss:dd:ss " formatında girilen saat bilgisini, toplam dakika olarak yazdıran programın akış diyagramını çizip kodlayınız.

18. n elemanlı bir x dizisinin ω ağırlıklar dizisi olmak üzere ağırlıklı ortalama türleri aşağıda verilmektedir. Buna göre klavyeden ağırlıklarıyla girilen dizinin ağırlıklı ortalamalarını hesaplayan programın akış diyagramını çiziniz ve kodlayınız.

Ağırlıklı aritmetik ortalama	Ağırlıklı geometrik ortalama	Ağırlıklı harmonik ortalama
$AAO = \frac{\sum_{i=1}^{n} \omega_i . x_i}{\sum_{i=1}^{n} \omega_i}$	$AGO = \left(\prod_{i=1}^{n} x_i^{\omega_i} \right)^{\frac{1}{\sum_{i=1}^{n} \omega_i}}$ $AGO = e^{\frac{\sum_{i=1}^{n} \omega_i . \ln(x_i)}{\sum_{i=1}^{n} \omega_i}}$	$AHO = \frac{\sum_{i=1}^{n} \omega_i}{\sum_{i=1}^{n} \frac{\omega_i}{x_i}}$

19. Pozitif iki a, b sayısı için aşağıda bazı ortalamalar verilmektedir. Buna göre klavyeden girilen iki pozitif tamsayının seçilen türdeki ortalamasını hesaplayan programın akış diyagramını çiziniz ve kodlayınız.

Heinz ortalama	Heronian ortalama
$H_p(a,b) = \frac{a^p b^{1-p} + a^{1-p} b^p}{2}$ $0 \le p \le 0.5$	$H(a,b) = \frac{a + \sqrt{ab} + b}{3}$ $H(a,b) = \frac{2}{3} \underbrace{\frac{a+b}{2}}_{\text{Aritmetik ortalama}} + \frac{1}{3} \underbrace{\sqrt{ab}}_{\text{Geometrik ortalama}}$
Stolarsky ortalama	**Identric ortalama**
$SO_p(a,b) = \begin{cases} a & , a = b \\ \left(\frac{a^p - b^p}{p(a-b)} \right)^{\frac{1}{p-1}} & , diger \end{cases}$	$IO(a,b) = \begin{cases} a & , a = b \\ \frac{1}{e} \sqrt[a-b]{\frac{a^a}{b^b}} & , diger \end{cases}$ $IO(a,b) = \begin{cases} a & , a = b \\ \frac{1}{e} \left(\frac{b^b}{a^a} \right)^{\frac{1}{b-a}} & , diger \end{cases}$
Logaritmik ortalama	
$LO(a,b) = \begin{cases} 0 & , a = 0 \vee b = 0 \\ a & , a = b \\ \frac{b-a}{\ln(b) - \ln(a)} & , diger \end{cases}$	

20. Klavyeden küçük harflerle girilen cümledeki her kelimenin ilk harfini büyüten programın akış diyagramını çizip kodlayınız.

21. Ondalık işareti olarak virgül kullanılarak girilen ondalıklı sayıyı düzelten, yani virgülü nokta yapan programın akış diyagramını çizip kodlayınız.

22. Klavyeden girilen cümledeki kelimeleri, uzunluklarına göre sıralayan programın akış diyagramını çizip kodlayınız.

23. Klavyeden girilen pozitif tamsayıyı, basamak sayısına bölen programın akış diyagramını çizip kodlayınız.

24. Klavyeden girilen pozitif tamsayının herbir hanesini, basamak sayısına bölen programın akış diyagramını çizip kodlayınız.

25. Klavyeden girilen ondalıklı sayıyı ayrıştıran programın akış diyagramını çizip kodlayınız.

26. Klavyeden girilen n pozitif tamsayısına göre n basamaklı en büyük tamsayı ile n basamaklı en küçük tamsayı arasındaki farkı hesaplayan programın akış diyagramını çizip kodlayınız.

27. Klavyeden girilen cümledeki kelimeleri tersten yazdırarak yeni cümle oluşturan programın akış diyagramını çizip kodlayınız.

28. Klavyeden girilen cümledeki en uzun kelimeyi bulan programın akış diyagramını çizip kodlayınız.

29. Klavyeden girilen cümledeki en uzun ile en kısa kelime arasındaki karakter sayısı farkını bulan programın akış diyagramını çizip kodlayınız.

30. Klavyeden girilen cümlede kaç tane noktalama işareti olduğunu bulan programın akış diyagramını çizip kodlayınız.

31. Klavyeden girilen cümlede, noktalama işaretlerinden sonra bir tane boşluk ekleyen programın akış diyagramını çizip kodlayınız.

32. Noktalama işaretleri kullanılmadan klavyeden girilen cümledeki kelimeler arasına "*" karakteri koyan programın akış diyagramını çizip kodlayınız.

BÖLÜM

8

İKİ BOYUTLU DİZİ UYGULAMALARI

Bu bölümde

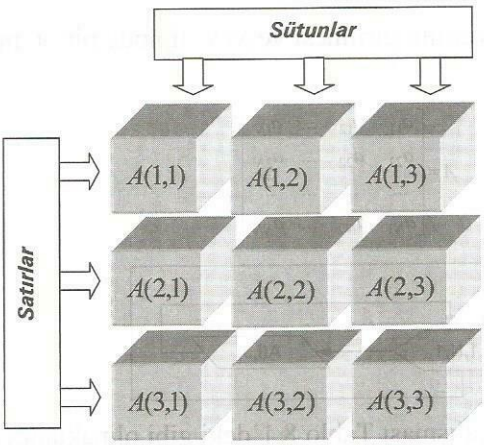
- İki boyutlu diziler hakkındaki temel bilgileri,
- İki boyutlu dizileri kullanmayı,
- İki boyutlu diziler üzerinde birçok işlem gerçekleştirmeyi

öğreneceksiniz.

Bölüm sonunda

- Çoklu verileri matris/tablo olarak işlemeyi öğrenecek,
- Programlarda matrisleri kullanabilecek,
- Sayısal ve alfasayısal matrisler üzerinde birçok işlem gerçekleştirebileceksiniz.

Mühendislik uygulamalarında, özellikle de çoklu verilerle ilgili olan işlemlerde yoğun olarak kullanılan yapılardan birisi de matrislerdir. Tablo gibi olan iki boyutlu diziler (dizilerin yatay [satır] ve düşey doğrultulu [sütun] olarak oluşturulması), '*matris*' olarak isimlendirilir. Bazen de "*diziler dizisi*" olarak da adlandırılan matrislerin elemanları, bellekte aynı tek boyutlu dizi elemanları gibi doğrusal olarak yerleşirler. Satır ve sütun sayısı eşit olan matrislere "*kare matris*" denir (Şekil 8.1).



$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Şekil 8.1: $(A)_{3x3}$ kare matrisinin yapısı ve matematiksel gösterimi

Bölüm 7'de sınıftaki 100 öğrencinin bir dersten aldıkları notlarının, diziye aktarılarak daha kolay işlem yapılabileceği gösterilmişti. Diziye aktarılarak, notlar düzenli hale gelmekte ve 100 tane ayrı değişken tanımlama gereksinimi ortadan kalkmaktaydı. Ancak 100 kişilik bir sınıfın 2 dersten aldıkları notlar işleneceği zaman, iki tane dizi tanımlamak gerekecektir. Tabi ki ders sayısı arttıkça, tanımlanacak olan dizi sayısı da paralel olarak artacaktır. Bu zorluk aşmak için rahatlıkla matris/tablo yapıları kullanılabilmektedir.

Örneğin 3 kişilik bir sınıftaki öğrencilerin 2 ayrı dersten (Matematik ve Fizik) aldıkları notlar;

Matematik dersi için: $A = [a_1, a_2, a_3] = [65,80,40]$

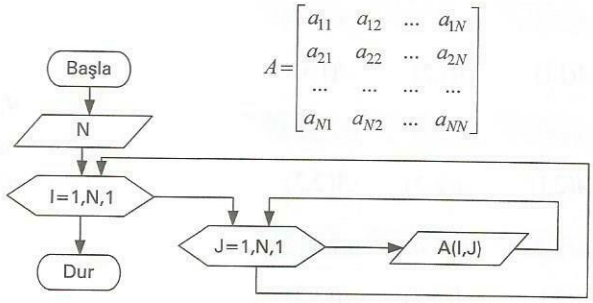
Fizik dersi için: $B = [b_1, b_2, b_3] = [90,35,78]$

şeklinde A ve B dizilerine aktarılabilir. Verilerin, ayrı yerlerde (farklı değişkenlerde) olması işlemleri zorlaştıracaktır. Dolayısıyla bu veriler, satırlarında diziler bulunan yeni bir yapıya yerleştirebilir. İki boyutlu veriler için çok uygun olan bu yapının adı, matristir. Örnekteki notlar, C matrisine;

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix} = \begin{bmatrix} 65 & 80 & 40 \\ 90 & 35 & 78 \end{bmatrix}$$

şeklinde yerleştirilebilirler. Yani matrisin 1. satırında Matematik dersinden alınan notlar, ikinci satırında da Fizik dersinden alınan notlar bulunmaktadır. Örneğin Fizik dersinden 1. öğrencinin aldığı not $C(2,1)$'den, Matematik dersinden 3. öğrencinin aldığı not $C(1,3)$'den okunarak öğrenilebilir. Görüldüğü gibi matrisin sütunlarında öğrenciler, satırlarında da dersler yer almaktadır. Bir boyutlu dizilerden farklı olarak matris adından sonra iki indis yer almaktadır. Birinci indis satırı gösterirken, ikinci indis de sütunu belirtmektedir.

Örnek-8.1: Klavyeden çoklu verinin girilmesi ve NxN tipinde bir A matrisine yerleştirilmesi.



$$A=\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{bmatrix}$$

$N=2$ durumunda programın çalışması Tablo 8.1'deki gibi olmaktadır. Görüldüğü gibi ilk önce içteki döngü (sütun değerini gösteren J), daha sonra dış döngü (satır değerini gösteren I) tamamlanmaktadır.

Tablo 8.1: Örnek 8.1 çalışma prensibi

Matrisin satır indisi	Matrisin sütun indisi	Girilecek eleman
I	*J*	*A(I,J)*
1	1	A(1,1)
1	2	A(1,2)
2	1	A(2,1)
2	2	A(2,2)

C

```
/* Matris girme */
#include <stdio.h>
#include <conio.h>
int A[10][10],i,j,N;
void main()
{ clrscr();printf("Kare matrisin tipini giriniz: ");scanf("%u",&N);
  for(i=0;i<N;i++)
   { for(j=0;j<N;j++)
    { printf("A(%u,%u)= ",i+1,j+1);scanf("%d",&A[i][j]);}} getch();}
```

```
Kare matrisin tipini giriniz: 2

A(1,1)= 2
A(1,2)= 5
A(2,1)= 4
A(2,2)= 9
```

C

```
/* Matris girme */
#include <stdio.h>
#include <conio.h>
int A[10][10],i,j,N;
void main()
{   clrscr();printf("Kare matrisin tipini giriniz: ");scanf("%u",&N);
    printf("\n  A matrisi\n");printf("   ------------\n");
    for(i=0;i<N;i++)
      { for(j=0;j<N;j++)
         { gotoxy(5*j+5,5+i);scanf("%u",&A[i][j]); }} getch(); }
```

```
Kare matrisin tipini giriniz: 3

  A matrisi
  ------------
  2  5  8
  6  4  9
  3  8  4
```

C++

```
// Matris girme
#include <iostream>
using namespace std;
int main()
{ int N; system("cls"); cout<<"Kare matrisin tipini giriniz: "; cin>>N;
  int A[N][N]; for(int i=0;i<N;i++)
  { for(int j=0;j<N;j++)
     { cout <<"A("<<i+1<<","<<j+1<<")= ";cin >> A[i][j]; } }
  system("pause");return 0; }
```

C++

```
// Matris girme
#include <iostream>
#include <windows.h>
using namespace std;
int N;
int main()
{   system("cls");cout << "Kare matrisin tipini giriniz: "; cin >> N;
    int A[N][N]; cout<< "\n   A matrisi\n";cout<< "   ------------\n";
    for(int i=0;i<N;i++)
    { for(int j=0;j<N;j++)
      { COORD konum = {5*j+5,4+i};
        SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),konum);
           cin >> A[i][j]; } }    system("pause");return 0; }
```

C#

```
// Matris girme
using System;
namespace ornek81
{ class Program
   { static void Main(string[] args)
     {Console.Write("Kare matrisin tipini giriniz: ");
      int N=Convert.ToInt32(Console.ReadLine());int[,] A=new int[N,N];
      for(int i=0;i<N;i++)
        for (int j=0;j<N;j++)
          { Console.Write("A({0},{1})= ", i, j);
            // veya Console.Write("A("+i+1+","+j+1+")= ");
            A[i,j]=Convert.ToInt32(Console.ReadLine()); }
       Console.ReadLine();       }    }    }
```

C#

```
// Matris girme
using System;
namespace ornek81
{ class Program
   { static void Main(string[] args)
    { Console.Write("Kare matrisin tipini giriniz: ");
      int N=Convert.ToInt32(Console.ReadLine()); int[,] A=new int[N,N];
      Console.WriteLine("\n      A matrisi:");
      Console.WriteLine("      ----------\n");
      for(int i=0;i<N;i++)
       for (int j=0;j<N;j++)
        { Console.SetCursorPosition(5*j+5,4+i);
          A[i,j]=Convert.ToInt32(Console.ReadLine()); }
       Console.ReadLine();}}}
```

JAVA

```
// Matris girme
import java.util.Scanner;
public class ornek81 {
public static void main(String[] args)
{int N; Scanner klavye=new Scanner (System.in);
 System.out.print("Kare matrisin tipini giriniz: ");N=klavye.nextInt();
 int A[][]=new int[N][N]; System.out.println("\nA matrisi: ");
 for(int i=0;i<N;i++){
  for(int j=0;j<N;j++)
   { System.out.print("A("+(i+1)+","+(j+1)+")= ");
    A[i][j]=klavye.nextInt();}}}}
```

MATLAB

```
% Matris girme
clear all; clc; N=input('Kare matrisin tipini giriniz: ');
for i=1:N
    for j=1:N
        fprintf('A(%d,%d)= ',i,j);A(i,j)=input(''); end; end;
```

MATLAB

```
% Matris girme
clear all; clc; A=input('Matrisi [a b;c d] şeklinde giriniz!\n');
```

```
Kare matrisin tipini giriniz: 2
Matrisi [a b;c d] şeklinde giriniz!
[1 2;4 3]
```

MATLAB

```
% Matris girme
clear all; clc; N=input('Kare matrisin tipini giriniz: ');
x=input('Matris elemanlarını boşluk bırakarak [] dizi şeklinde giri-
niz!\n'); A=reshape(x,N,N)';
```

```
Kare matrisin tipini giriniz: 2
Matris elemanlarını boşluk bırakarak [] dizi şeklinde giriniz!
[1 2 4 3]
```

PYTHON

```
# Matris girme
N=int(input("Kare matrisin tipini giriniz: "))
A=[[] for i in range(N)] for j in range(N)]
for i in range(N):
      for j in range(N):
          A[i][j]=int(input("A(%d,%d)= "%(i+1,j+1)))
```

PYTHON

```
# Matris girme
import numpy as np
x=input("Matrisi [a b;c d] şeklinde giriniz!\n")
A=np.matrix(x)
```

```
Matrisi [a b;c d] şeklinde giriniz!
1 2;4 3
```

Not

"NxN" tipinde bir matrisin;

- ✓ MATLAB dilindeki ilk elemanının indisleri "(1,1)" ve son elemanının indisleri de "(N,N)"dir.
- ✓ C, C++, C#, Java ve Python dillerinde ilk elemanının indisleri "(0,0)" ve son elemanının indisleri de "(N-1,N-1)"dir.

Not

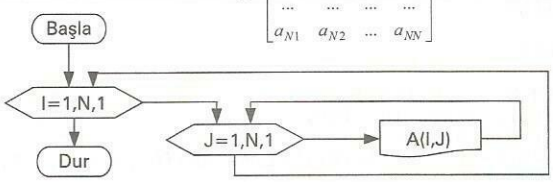
Kitaptaki uygulamaların akış diyagramlarında, matematik notasyona uygun olması için 1-N arası indis kullanılmıştır. Döngü değişkenleri 0'dan başlatılıp N-1'e kadar da açılabilir. Böylece ilk elemanın indisi 0 ile başlayıp son elemanın indisi N-1'de biter.

Not

Matrislere, dizilerde olduğu gibi statik veya dinamik bellek tahsisi yapılabilir.

Örnek-8.2: *NxN* tipindeki *A* kare matrisinin elemanlarının ekrana yazdırılması.

$$A=\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{bmatrix}$$



$N=2$ durumunda programın çalışma adımları Tablo 8.2'deki gibidir.

Tablo 8.2: Örnek-8.2'in çalışma adımları

Matrisin satır indisi	Matrisin sütun indisi	Ekrana yazılacak eleman
I	J	A(I,J)
1	1	A(1,1)
1	2	A(1,2)
2	1	A(2,1)
2	2	A(2,2)

C

```
/* Matris yazdırma */
#include <stdio.h>
#include <conio.h>
void main()
{ int i,j,N,A[100][100]; clrscr();
  for(i=0;i<N;i++)
    for(j=0;j<N;j++) printf("A(%u,%u)= %d\n",i,j,A[i][j]); getch(); }
```

C

```
/* Matris yazdırma */
#include <stdio.h>
#include <conio.h>
void main()
{ int i,j,N,A[100][100]; clrscr(); printf("A matrisi:\n");
  for(i=0;i<N;i++)
    {for(j=0;j<N;j++) printf("%d\t",A[i][j]); printf("\n");} getch();}
```

C++

```
// Matris yazdırma
#include <iostream>
using namespace std;
int main()
{ int A[100][100],N; system("cls");
  for(int i=0;i<N;i++)
    for (int j=0;j<N;j++) cout <<"A("<<i<<","<<j<<")= "<<A[i][j]<<endl;
  system("pause"); return 0; }
```

C++

```
// Matris yazdırma
#include <iostream>
using namespace std;
int main()
{   int A[100][100],N; system("cls"); cout << "A matrisi:\n";
    for(int i=0;i<N;i++)
      { for (int j=0;j<N;j++) cout << A[i][j] << "\t"; cout << "\n"; }
    system("pause"); return 0; }
```

C#

```
// Matris yazdırma
using System;
namespace ornek82
{ class Program
   { static void Main(string[] args)
     { int N; int[,] A=new int[100,100];
       for (int i=0;i<N;i++)
 for (int j=0;j<N;j++) Console.WriteLine("A({0},{1})={2}",i,j,A[i, j]);
       Console.ReadLine();       }    }   }
```

C#

```
// Matris yazdırma
using System;
namespace ornek82
{class Program
  {static void Main(string[] args)
    {int N; int[,] A=new int[100,100];Console.WriteLine("A matrisi");
        for (int i=0;i<N;i++)
         { for (int j=0;j<N;j++) Console.Write("{0}\t", A[i, j]);
           Console.WriteLine(); }  Console.ReadLine();   } } }
```

JAVA

```
// Matris yazdırma
public class ornek82 {
public static void main(String[] args)
{int N; int A[][]=new int[100][100];
 for(int i=0;i<N;i++)
 for(int j=0;j<N;j++) System.out.println("A("+i+","+j+")= "+A[i][j]);}}
```

JAVA

```
// Matris yazdırma
public class ornek82 {
public static void main(String[] args)
{  int N; int A[][]=new int[100][100];
   for(int i=0;i<N;i++)
     { for(int j=0;j<N;j++) System.out.print(A[i][j]+"\t");
       System.out.println(); } }}
```

MATLAB

```
% Matris yazdırma
clear all; clc;
for i=1:N
    for j=1:N
        fprintf('A(%d,%d)= %d\n',i,j,A(i,j)); end; end;
```

MATLAB

```
% Matris yazdırma
clear all; clc;
A            % veya disp(A)
```

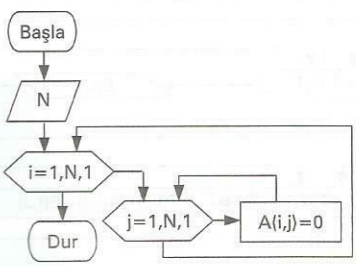
PYTHON

```
# Matris yazdırma
for i in range(N):
    for j in range(N):
        print("A(%d,%d)= %d"%(i,j,A[i][j]))
```

PYTHON

```
# Matris yazdırma
import numpy as np
print(A)
```

Örnek-8.3: Tüm elemanları sıfır olan *NxN* tipi *A* kare matrisini oluşturan program.



Örnek-8.4: Klavyeden girilen NxN tipindeki A ve B kare matrislerini toplayan ($C = A + B$) program.

Dizi işlemlerinde olduğu gibi matris işlemlerinde de indisler arasındaki ilişkiler ortaya çıkarılmalıdır. En basitinden $2x2$ veya $3x3$ tipinde örnek kare matrisler alınarak işlem sonucu oluşan matris açıkça yazılmalıdır. Böylece satır/sütun indislerinin değişimi gözlenerek gerekli işlem/işlemlerin yapılmasını sağlayan ifadeler elde edilir.

Örnek olarak $N = 2$ alınırsa, toplama işlemi aşağıdaki gibidir (Tablo 8.3):

$$\left.\begin{aligned} A &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \\ B &= \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \end{aligned}\right\} \Rightarrow C = A + B = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}$$

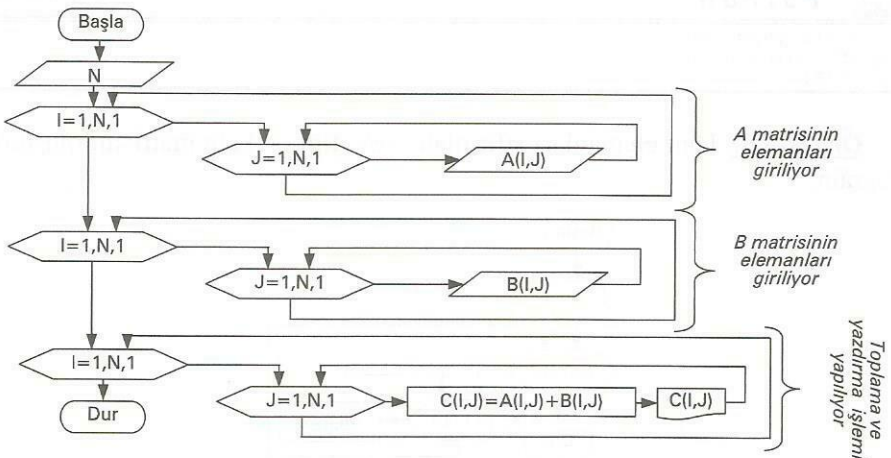
Tablo 8.3: Örnek-8.4 çalışma prensibi

Satır indisi	Sütun indisi	A matrisinin eleman değeri	B matrisinin eleman değeri	C matrisinin eleman değeri
I	J	A(I,J)	B(I,J)	C(I,J)=A(I,J)+B(I,J)
1	1	A(1,1)	B(1,1)	C(1,1)=A(1,1)+B(1,1)
1	2	A(1,2)	B(1,2)	C(1,2)=A(1,2)+B(1,2)
2	1	A(2,1)	B(2,1)	C(2,1)=A(2,1)+B(2,1)
2	2	A(2,2)	B(2,2)	C(2,2)=A(2,2)+B(2,2)

İki matris toplanırken, matrisin karşılıklı elemanları toplanır. Dikkat edilirse, karşılıklı elemanlar toplandığı için indisler aynı şekilde değişmektedir. Dolayısıyla toplama işlemi için iki döngü açılır ve

$$C(I,J) = A(I,J) + B(I,J)$$

ifadesi yazılır. Burada I (satır) ve J (sütun), $1-N$ arasında açılan döngü değişkenleridir.



C

```
/* Matris toplamı */
#include <stdio.h>
#include <conio.h>
int i,j,N,A[10][10],B[10][10],C[10][10];
void main()
{  clrscr();printf("Kare matrisin tipini giriniz: ");scanf("%u",&N);
   printf("\n   A matrisi\n");printf(" ------------\n");
   for(i=0;i<N;i++)
    { for(j=0;j<N;j++) { gotoxy(5*j+5,5+i);scanf("%d",&A[i][j]); }}
   printf("\n   B matrisi\n");printf(" ------------\n");
   for(i=0;i<N;i++)
    { for(j=0;j<N;j++) { gotoxy(5*j+5,8+i+N);scanf("%d",&B[i][j]); }}
   printf("\n  C=A+B matrisi\n");printf(" ------------\n");
   for(i=0;i<N;i++)
    { for(j=0;j<N;j++)
      { C[i][j]=A[i][j]+B[i][j];
        gotoxy(5*j+5,11+i+2*N);printf("%d",C[i][j]);}} getch(); }
```

C++

```
// Matris toplamı
#include <iostream>
#include <windows.h>
using namespace std;
int i,j,N;
int main()
{ system("cls");
  cout << "Kare matrisin tipini giriniz: "; cin >> N;
  int A[N][N],B[N][N],C[N][N];
  cout << "\n   A matrisi:\n";cout << "   ------------\n";
  for(i=0;i<N;i++)
   {for(j=0;j<N;j++)
     {COORD konum = {5*j+5,4+i};
      SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),konum);
      cin >> A[i][j]; } }
  cout << "\n   B matrisi:\n";cout << "   ------------\n";
  for(i=0;i<N;i++)
   {for(j=0;j<N;j++)
     {COORD konum = {5*j+5,7+i+N};
      SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),konum);
      cin >> B[i][j]; } }
  cout<<"\n  C=A+B matrisi:\n";cout<<"   ------------\n";
  for(i=0;i<N;i++)
   {for(j=0;j<N;j++)
     {C[i][j]=A[i][j]+B[i][j]; COORD konum = {5*j+5,10+i+2*N};
      SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),konum);
      cout << C[i][j]; } }     system("pause");return 0; }
```

```
Kare matrisin tipini giriniz: 2

  A matrisi
------------
  3  5
  1  2

  B matrisi
------------
  8  4
  3  6

  C=A+B matrisi
----------------
  11  9
   4  8
```

C#

```
// Matris toplamı
using System;
namespace ornek84
{ class Program
  { static void Main(string[] args)
   {Console.Write("Kare matirisn tipini giriniz: ");
    int N = Convert.ToInt32(Console.ReadLine());
    int[,] A=new int[N,N];int[,] B=new int[N,N];int[,] C=new int[N, N];
    Console.WriteLine("\n      A matrisi");
    Console.WriteLine("      ---------");
    for (int i=0;i<N;i++)
     { for (int j=0;j<N;j++)
        { Console.SetCursorPosition(5*j+5,4+i);
          A[i,j] = Convert.ToInt32(Console.ReadLine()); }}
    Console.WriteLine("\n      B matrisi");
    Console.WriteLine("      ---------");
    for (int i=0;i<N;i++)
     { for (int j=0;j<N;j++)
        { Console.SetCursorPosition(5*j+5,7+i+N);
          B[i,j] = Convert.ToInt32(Console.ReadLine()); }}
    Console.WriteLine("\n      C matrisi");
    Console.WriteLine("      ---------");
    for (int i=0;i<N;i++)
     { for (int j=0;j<N;j++) { C[i,j]=A[i,j]+B[i,j];
    Console.SetCursorPosition(5*j+5,10+i+2*N);Console.Write(C[i,j]);}}
    Console.ReadLine();       }    }   }
```

JAVA

```
// Matris toplamı
import java.util.Scanner;
public class ornek84 {
public static void main(String[] args)
{  int i,j,N; Scanner klavye=new Scanner (System.in);
   System.out.print("Kare matrisin tipini giriniz: ");
   N=klavye.nextInt();
   int A[][]=new int[N][N];int B[][]=new int[N][N];
   int C[][]=new int[N][N];
   System.out.println("\nA matrisi");System.out.println("---------");
   for(i=0;i<N;i++){
    for(j=0;j<N;j++)
     {System.out.print("A("+i+","+j+")= ");A[i][j]=klavye.nextInt();}}
   System.out.println("\nB matrisi");System.out.println("---------");
   for(i=0;i<N;i++){
    for(j=0;j<N;j++)
     {System.out.print("B("+i+","+j+")= ");B[i][j]=klavye.nextInt();}}
   System.out.println("\nC=A+B matrisi");
   System.out.println("-------------");
   for(i=0;i<N;i++){ for(j=0;j<N;j++)
      { C[i][j]=A[i][j]+B[i][j]; System.out.print(C[i][j]+"\t");}
       System.out.println();} } }
```

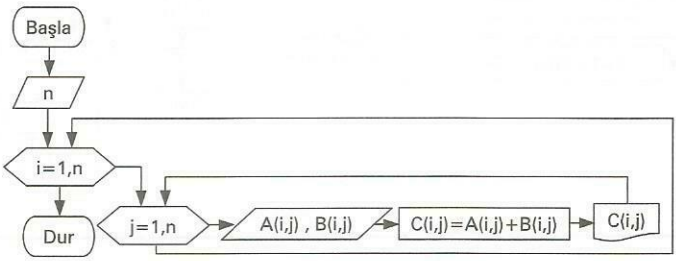
MATLAB

```
% Matris toplamı
clear all; clc;fprintf('Matrisleri [a b;c d] şeklinde giriniz!\n');
fprintf('\nA matrisi\n---------\n'); A=input('');
fprintf('\nB matrisi\n---------\n'); B=input('');
C=A+B; fprintf('\nC=A+B matrisi\n-------------\n');disp(C)
```

PYTHON

```
# Matris toplamı
import numpy as np
print("Matrisleri [a b;c d] şeklinde giriniz!\n")
print("\nA matrisi\n---------"); x=input(); A=np.matrix(x)
print("\nB matrisi\n---------"); x=input(); B=np.matrix(x)
C=A+B; print("\nC=A+B matrisi\n-------------"); print(C)
```

Program kodlarını kısaltmak için birinci matrisin elemanlarının girildiği döngülerde ikinci matris elemanları da girilip toplama ve yazdırma işlemi aynı anda gerçekleştirilebilir. Böyle altı döngü yerine sadece iki döngü ile program tamamlanmış olur.



Matrislerdeki çıkarma işlemi de aynı şekildedir. Akış diyagramı ve kodlardaki işlem kısmında "+" yerine "-" yazılması yeterlidir. Ayrıca karşılıklı elemanların çarpımıyla yeni matris oluşturmada "+" yerine "*" ve karşılıklı elemanların bölümüyle de yeni matris oluşturmada "+" yerine "/" yazılır.

Not

Kitaptaki matris uygulamalarında, daha açıklayıcı ve anlaşılır olması açısından fazla döngü kullanımından kaçınılmamıştır. Ancak unutmayınız ki daha az işlem içeren programlar, doğal olarak daha hızlı çalışacaklardır.

Örnek-8.5: Klavyeden girilen NxN tipindeki A matrisinin izini (ana köşegen üzerindeki elemanların toplamını) hesaplayıp yazdıran program.

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2N} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3N} \\ \dots & \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & a_{N3} & \dots & a_{NN} \end{bmatrix}$$

Ana köşegen

"*Matrisin izi*", ana köşegen üzerindeki elemanların toplamı olup

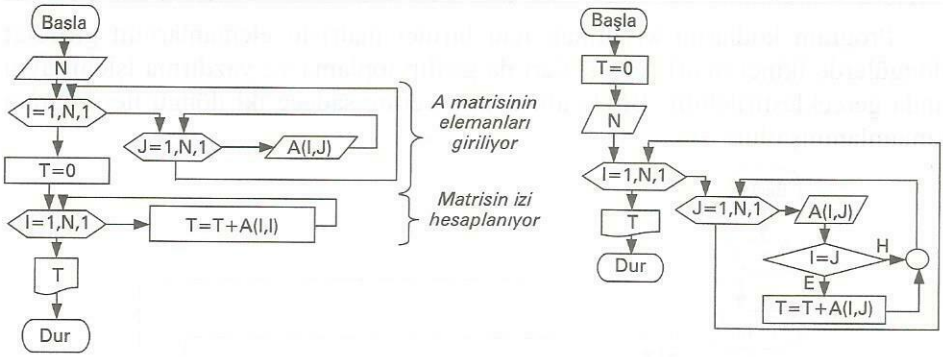
$$iz(A) = a_{11} + a_{22} + a_{33} + \dots\dots + a_{NN}$$

matematiksel eşitliğiyle hesaplanır. Dikkat edilirse ana köşegen üzerindeki elemanların satır ve sütun sayılarını gösteren indisler daima birbirine eşittir (I=J). Dolayısıyla bu toplama işlemini gerçekleştirmek için tek döngü yeterli olmaktadır.

Programdaki döngü içinde $T = T + A(I,I)$ yazılarak matrisin izi hesaplanır. Örnek olarak $N = 3$ için matrisin izi, $T = a_{11} + a_{22} + a_{33}$ olur (Tablo 8.4). Yine açık olması için akış diyagramı ve kodlarda matrisin girilmesi ve izinin hesaplanması ayrı döngülerde yapıldı. Ancak programda matrisin izi, elemanlarının girildiği döngülerde de hesaplanabilir.

Tablo 8.4: Örnek-8.5'in çalışma prensibi

Döngü Sayacı	Matrisin eleman değeri	Toplamın eski değeri	Toplamın yeni değeri
I	A(I,I)	T	T=T+A(I,I)
1	A(1,1)	0	T=0+A(1,1)
2	A(2,2)	A(1,1)	T=A(1,1)+A(2,2)
3	A(3,3)	A(1,1)+A(2,2)	T=A(1,1)+A(2,2)+A(3,3)



C

```
/* Matrisin izi */
#include <stdio.h>
#include <conio.h>
int i,j,N,A[10][10],T=0;
void main()
{ clrscr();printf("Kare matrisin tipini giriniz: ");scanf("%u",&N);
  printf("\nA matrisi\n");printf(" ------------\n");
  for(i=0;i<N;i++)
    { for(j=0;j<N;j++) { gotoxy(5*j+5,5+i);scanf("%d",&A[i][j]); }}
  for(i=0;i<N;i++) T+=A[i][i]; printf("\nMatrisin izi= %d",T);getch(); }
```

```
Kare matrisin tipini giriniz: 3

 A matrisi
 ------------
 1  2  3
 6  5  4
 7  8  9

Matrisin izi= 15
```

C++

```
// Matrisin izi
#include <iostream>
#include <windows.h>
using namespace std;
int i,j,N,T=0;
int main()
{ system("cls");cout<<"Kare matrisin tipini giriniz: ";cin>>N;
  int A[N][N]; cout<<"\n   A matrisi:\n"; cout<<"   ------------\n";
  for(i=0;i<N;i++)
   {for(j=0;j<N;j++)
     {COORD konum = {5*j+5,4+i};
      SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),konum);
      cin >> A[i][j]; } }
    for(i=0;i<N;i++) T+=A[i][i]; cout<<"\nMatrisin izi= "<<T<<endl;
    system("pause");return 0; }
```

C#

```
// Matrisin izi
using System;
namespace ornek85
{ class Program
   { static void Main(string[] args)
    { Console.Write("Kare matirisn tipini giriniz: ");
      int N = Convert.ToInt32(Console.ReadLine());
      int[,] A = new int[N, N];
      Console.WriteLine("\nA matrisi");Console.WriteLine("---------");
      for (int i=0;i<N;i++)
        { for (int j=0;j<N;j++)
            { Console.SetCursorPosition(5*j+5,4+i);
              A[i,j] = Convert.ToInt32(Console.ReadLine()); }}
      int T = 0; for (int i = 0; i < N; i++) T += A[i,i];
      Console.WriteLine("\nMatrisin izi=" + T); Console.ReadLine(); }}}
```

JAVA

```
// Matrisin izi
import java.util.Scanner;
public class ornek85 {
public static void main(String[] args)
{ int i,j,N,T=0;   Scanner klavye=new Scanner (System.in);
  System.out.print("Kare matrisin tipini giriniz: ");
  N=klavye.nextInt(); int A[][]=new int[N][N];
  System.out.println("\nA matrisi");System.out.println("---------");
  for(i=0;i<N;i++){
    for(j=0;j<N;j++){ System.out.print("A("+i+","+j+")= ");
                      A[i][j]=klavye.nextInt(); }}
  for(i=0;i<N;i++) T+=A[i][i];
  System.out.println("\nMatrisin izi= "+T); }}
```

MATLAB

```
% Matrisin izi
clear all; clc;
fprintf('Matrisi [a b;c d] şeklinde giriniz!\n\n'); A=input('');
fprintf('\nMatrisin izi: %d\n',trace(A));
% fprintf('\nMatrisin izi: %d\n',sum(diag(A)));
```

PYTHON

```
# Matrisin izi
import numpy as np
print("Matrisi [a b;c d] şeklinde giriniz!\n"); x=input()
A=np.matrix(x); print("\nMatrisin izi: %d"%np.trace(A))
```

Örnek-8.6: Klavyeden girilen NxN tipindeki A matrisinin yedek köşegen üzerindeki elemanlarının toplamını bulan program.

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2N} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3N} \\ \dots & \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & a_{N3} & \dots & a_{NN} \end{bmatrix}$$

Yedek köşegen

Yedek köşegenin üzerindeki elemanların toplamı $T = a_{1N} + a_{2N-1} + \dots + a_{N1}$ olup aradaki ilişkiyi daha açık görebilmek için $N = 3$ alınır ve indislerin değişimi gözlenir. Dikkat edilirse toplanan matris elemanlarının indislerinin toplamı 4'tür, yani $(N+1)$ dir. Satır indisleri artarken, aynı oranda sütun indisleri azalmaktadır (Tablo 8.5).

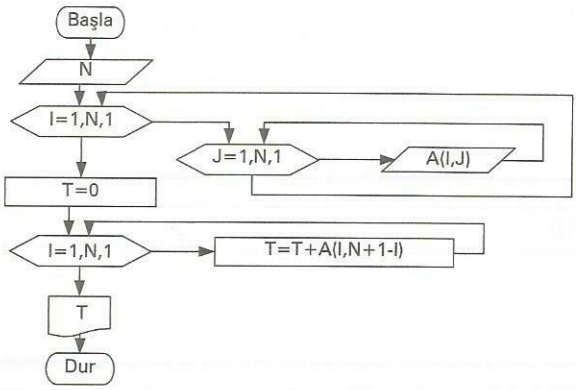
$$T = a_{13} + a_{22} + a_{31}$$

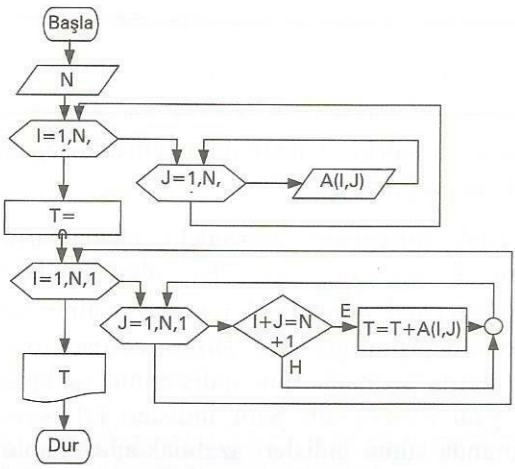
Satır indisi	Sütun indisi	Sütun indisi	Toplam
i	*j*	*j=4-i*	*i+j*
1	3	4 – *1* = 3	1+3=4
2	2	4 – *2* = 2	2+2=4
3	1	4 – *3* = 1	3+1=4

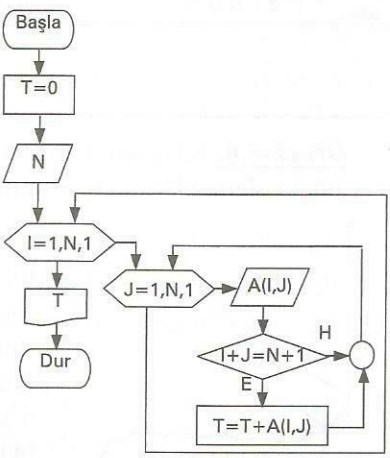
Tablo 8.5: Örnek-8.6'ın çalışma prensibi

Döngü sayacı	Matrisin eleman değeri	Toplamın eski değeri	Toplamın yeni değeri
I	A(I,N+1-I)	T	T=T+A(I,N+1-I)
1	A(1,3)	0	T=0+A(1,3)
2	A(2,2)	A(1,3)	T=A(1,3)+A(2,2)
3	A(3,1)	A(1,3)+A(2,2)	T=A(1,3)+A(2,2)+A(3,1)

Programdaki tek döngü ile bu işlem $T = T + A(I, N+1-I)$ ile yapılabilir. Bu işlemi gerçekleştirmek için farklı bir mantık da geliştirilebilir. Dikkat edilirse yedek köşegen üzerindeki indislerin toplamı daima $(N+1)$'e eşittir (Tablo 8.6). Buna göre çözüm yapıldığında, iki döngü kullanılması gerekecektir.







Tablo 8.6: Örnek-8.6 için değişik çalışma prensibi

I	J	A(I,J)	I+J	N+1	Eski T	Yeni T
1	*1*	A(1,1)	2	4	0	---
1	*2*	A(1,2)	3	4	0	---
1	*3*	A(1,3)	4	4	0	T=0+*A(1,3)*
2	*1*	A(2,1)	3	4	A(1,3)	---
2	*2*	A(2,2)	4	4	A(1,3)	T=A(1,3)+*A(2,2)*
2	*3*	A(2,3)	5	4	A(1,3)+A(2,2)	---
3	*1*	A(3,1)	4	4	A(1,3)+A(2,2)	T=A(1,3)+A(2,2)+*A(3,1)*
3	*2*	A(3,2)	5	4	A(1,3)+A(2,2)+A(3,1)	---
3	*3*	A(3,3)	6	4	A(1,3)+A(2,2)+A(3,1)	---

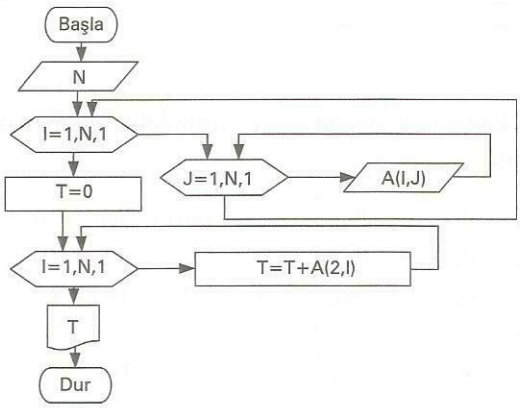
Örnek-8.7: Klavyeden girilen *NxN* tipindeki *A* matrisinin 2. satır elemanlarının toplamını bulan program.

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2N} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3N} \\ \dots & \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & a_{N3} & \dots & a_{NN} \end{bmatrix}$$

2. satır

2. satır elemanlarının toplamı $T = a_{21} + a_{22} + a_{23} + \dots\dots + a_{2N}$ dır. Örnek olarak $N = 3$ için matrisin 2. satır elemanları toplamı $T = a_{21} + a_{22} + a_{23}$ olur. Dikkat edilirse 2. satır üzerindeki elemanların satır sayısını gösteren indis daima *2'ye eşittir.*

Matrisin 2. satır elemanlarının toplamı için açılan tek döngüde $T = T + A(2,I)$ yazılması yeterlidir (Tablo 8.7).



Tablo 8.7: Örnek-8.7'nin çalışma prensibi

Döngü sayacı	Matrisin eleman değeri	Toplamın eski değeri	Toplamın yeni değeri
I	**A(2,I)**	**T**	**T=T+A(2,I)**
1	A(2,1)	0	T=0+A(2,1)
2	A(2,2)	A(2,1)	T=A(2,1)+A(2,2)
3	A(2,3)	A(2,1)+A(2,2)	T=A(2,1)+A(2,2)+A(2,3)

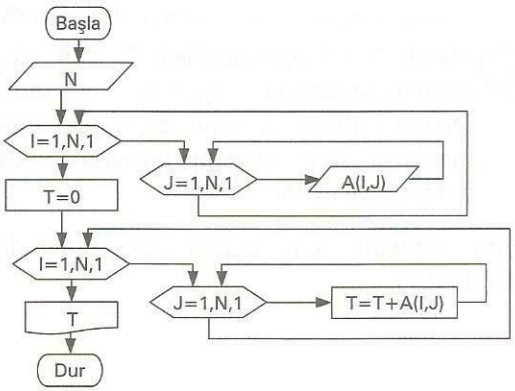
Aynı şekilde klavyeden girilen *NxN* tipindeki *A* matrisinin 2. sütun elemanlarının toplamını bulan program da küçük bir değişiklikle yazılabilir. Akış diyagramında, işlem kutusu içindeki $T = T + A(2,I)$ yerine $T = T + A(I,2)$ yazılması yeterlidir. Bu akış diyagramını, önceki örneklerdeki gibi kısaltınız.

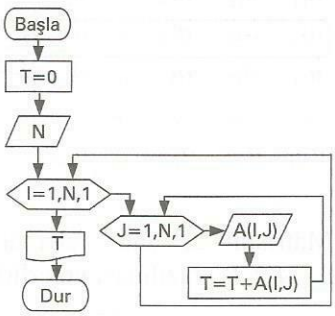
Örnek-8.8: Klavyeden girilen NxN tipindeki A matrisinin tüm elemanlarının toplamını hesaplayıp ekrana yazdıran program.

Matrisin tüm elemanlarının toplamı istendiği için iki döngü kullanılır ve döngülerin içine $T = T + A(I,J)$ ifadesi yazılır. $N = 2$ için programın ayrıntılı çalışması Tablo 8.8'de verilmektedir.

Tablo 8.8: Örnek-8.8'in çalışma prensibi

I	J	A(I,J)	Eski T	Yeni T
1	1	A(1,1)	0	T=0+A(1,1)
1	2	A(1,2)	A(1,1)	T=A(1,1)+A(1,2)
2	1	A(2,1)	A(1,1)+A(1,2)	T=A(1,1)+A(1,2)+A(2,1)
2	2	A(2,2)	A(1,1)+A(1,2)+A(2,1)	T=A(1,1)+A(1,2)+A(2,1)+A(2,2)





Örnek-8.9: Klavyeden girilen NxN tipindeki A matrisinin transpozesini hesaplayan program.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2N} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3N} \\ \dots & \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & a_{N3} & \dots & a_{NN} \end{bmatrix} \Rightarrow A^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} & \dots & a_{N1} \\ a_{12} & a_{22} & a_{32} & \dots & a_{N2} \\ a_{13} & a_{23} & a_{33} & \dots & a_{N3} \\ \dots & \dots & \dots & \dots & \dots \\ a_{1N} & a_{2N} & a_{3N} & \dots & a_{NN} \end{bmatrix}$$

Bir matrisin "*transpozesi*", o matrisin satır ve sütunlarının yer değiştirmesiyle elde edilir. Orijinal matristeki satır elemanları, transpoz matriste sütun elemanları olmakta; sütun elemanları da satır elemanı olmaktadır. Örneğin A matrisinin a_{21} elemanı, A matrisinin transpozesi olan A^T'da a_{12} olmaktadır. Yani,

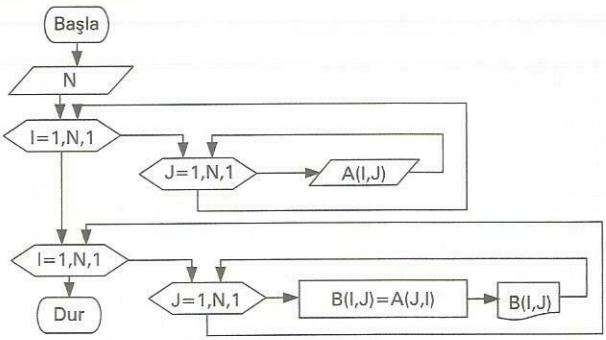
Satır indisi $\Rightarrow$ *Sütun indisi* I $\Rightarrow$ J
Sütun indisi $\Rightarrow$ *Satır indisi* J $\Rightarrow$ I

şekline dönüşmektedir. Bu işlem; I satır indisini gösteren döngü ve J de sütun indisini gösteren döngü olmak üzere $B(I,J) = A(J,I)$ atamasıyla gerçekleştirilir.

Örnek olarak $N=2$ için A matrisinin transpozesi olan B matrisinin elemanları Tablo 8.9'daki gibi oluşacaktır.

Tablo 8.9: Örnek-8.9'un çalışma prensibi

Satır sayısı	Sütun sayısı	A matrisinin eleman değeri	$B=A^T$ matrisinin eleman değeri
I	J	A(I,J)	B(I,J)
1	1	A(1,1)	B(1,1)=A(1,1)
1	2	A(1,2)	B(1,2)=A(2,1)
2	1	A(2,1)	B(2,1)=A(1,2)
2	2	A(2,2)	B(2,2)=A(2,2)



C

```
/* Matrisin transpozesi*/
#include <stdio.h>
#include <conio.h>
int i,j,N,A[10][10],B[10][10];
main()
{ clrscr();printf("Kare matrisin satır/sütun sayısı: ");scanf("%u",&N);
  printf("\n   A matrisi\n");printf("  ------------\n");
  for(i=1;i<=N;i++)
    { for(j=1;j<=N;j++) { gotoxy(5*j,4+i);scanf("%d",&A[i-1][j-1]); }}
  printf("\nMatrisin transpozesi\n");printf("--------------------\n");
  for(i=0;i<N;i++)
   { for(j=0;j<N;j++)
      { B[i][j]=A[j][i];
        gotoxy(5*j+5,8+i+N);printf("%d",B[i][j]);}} getch();return 0;}
```

C++

```
// Matrisin transpozesi
#include <iostream>
#include <windows.h>
using namespace std;
int main()
{ int i,j,N; system("cls");cout<<"Kare matrisin satır/sütun sayısı: ";
  cin >> N; int A[N][N],B[N][N];
  cout << "\n   A matrisi \n";cout << "  ---------- \n";
  for (i=1;i<=N;i++) { for (j=1;j<=N;j++)
      { COORD xy = {5*j,3+i};
        SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),xy);
        cin >> A[i-1][j-1]; }}
  cout << "\nMatrisin transpozesi\n";cout << "--------------------\n";
  for (i=0;i<N;i++)
   { for (j=0;j<N;j++) { B[i][j]=A[j][i]; COORD xy = {5*j+5,7+i+N};
      SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),xy);
      cout << B[i][j]; }}   system("pause");return 0; }
```

```
Kare matrisin satır/sütun sayısı: 3

 A matrisi
 ---------
 1  2  3
 4  5  6
 7  8  9

Matrisin transpozesi
--------------------
 1  4  7
 2  5  8
 3  6  9
```

C#

```
// Matrisin transpozesi
using System;
namespace ornek89
{ class Program
   { static void Main(string[] args)
    { Console.Write("Kare matrisin satır/sütun sayısı: ");
      int N = Convert.ToInt32(Console.ReadLine());
      int[,] A = new int[N, N]; int[,] B = new int[N, N];
      Console.WriteLine("\n     A matrisi");
      Console.WriteLine("     ---------");
      for (int i=0;i<N;i++) { for (int j=0;j<N;j++)
        { Console.SetCursorPosition(5*j+5,4+i);
          A[i,j] = Convert.ToInt32(Console.ReadLine()); }}
      Console.WriteLine("\nMatrisin transpozesi");
      Console.WriteLine("--------------------");
      for (int i=0;i<N;i++) { for (int j=0;j<N;j++)
        { B[i,j]=A[j,i];Console.SetCursorPosition(5*j+5,7+i+N);
          Console.Write(B[i,j]); }}        Console.ReadLine();  }}}
```

JAVA

```
// Matrisin transpozesi
import java.util.Scanner;
public class ornek89 {
public static void main(String[] args)
{  int i,j,N; Scanner klavye=new Scanner (System.in);
   System.out.print("Kare matrisin satır/sütun sayısı: ");
   N=klavye.nextInt(); int A[][]=new int[N][N];int B[][]=new int[N][N];
   System.out.println("\nA matrisi");System.out.println("---------");
   for(i=0;i<N;i++){
    for(j=0;j<N;j++)
     {System.out.print("A("+i+","+j+")= ");A[i][j]=klavye.nextInt();}}
   System.out.println("\nMatrisin transpozesi");
   System.out.println("--------------------");
   for(i=0;i<N;i++)
    { for(j=0;j<N;j++)
       { B[i][j]=A[j][i];System.out.print(B[i][j]+"\t");}
       System.out.println(); }  }}
```

MATLAB

```
% Matrisin transpozesi
clear all; clc;
fprintf('Matrisi [a b;c d] şeklinde giriniz!\n\n'); A=input('');
fprintf('\nMatrisin transpozesi\n--------------------\n');
B=A' ; disp(B)
```

PYTHON

```
# Matrisin transpozesi
import numpy as np
print("Matrisi [a b;c d] şeklinde giriniz!\n")
x=input(); A=np.matrix(x); B=np.transpose(A)
print('\nMatrisin transpozesi\n--------------------'); print(B)
```

Örnek-8.10: Klavyeden girilen N satır/sütun sayısına göre NxN tipinde "*birim matris*" oluşturan program.

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2N} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3N} \\ \dots & \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & a_{N3} & \dots & a_{NN} \end{bmatrix}$$

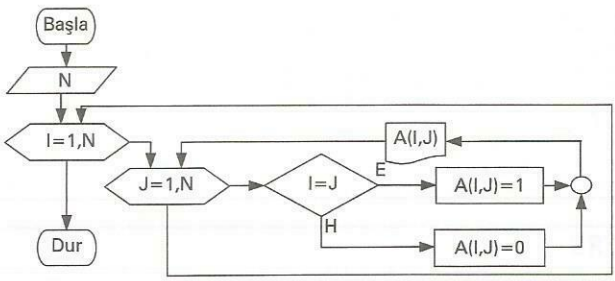
Ana köşegen üzerindeki elemanları "1", diğer bütün elemanları "0" olan matrise "*birim matris*" denir. A matrisinin ana köşegen üzerindeki elemanları a_{ii}'dır ve bunların tamamı "1", diğer tüm a_{ij} elemanları ise "0" olmalıdır.

$$\Rightarrow A(I,J)=\begin{cases} 1 \Rightarrow I=J \\ 0 \Rightarrow I\neq J \end{cases}$$

$N=2$ için programın çalışması, Tablo-8.10'da özetlenmektedir.

Tablo 8.10: Örnek-8.10'un çalışma prensibi

I	J	A(I,J)	I=J	İşlem
1	1	A(1,1)	Evet	A(1,1)=1
1	2	A(1,2)	Hayır	A(1,2)=0
2	1	A(2,1)	Hayır	A(2,1)=0
2	2	A(2,2)	Evet	A(2,2)=1



C

```
/* Birim matris */
#include <stdio.h>
#include <conio.h>
short i,j,N,A[10][10];
void main()
{  clrscr();printf("Birim matrisin tipini giriniz: ");scanf("%u",&N);
   printf("\n  Birim matris\n");printf("  ------------\n");
   for(i=0;i<N;i++)
    { for(j=0;j<N;j++)
       { if(i==j) A[i][j]=1;else A[i][j]=0;
         gotoxy(5*j+5,5+i);printf("%u",A[i][j]); }} getch(); }
```

C++

```
// Birim matris
#include <iostream>
#include <windows.h>
using namespace std;
short i,j,N;
int main()
{ system("cls");cout << "Birim matrisin tipini giriniz: ";cin >> N;
  int A[N][N]; cout<<"\n     Birim matris\n";cout<<"     ------------\n";
  for (i=0;i<N;i++)
    { for (j=0;j<N;j++)
       { if(i==j) A[i][j]=1;else A[i][j]=0;
          COORD xy = {5*j+5,4+i};
          SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),xy);
          cout << A[i][j]; }}      system("pause");return 0; }
```

C#

```
// Birim matris
using System;
namespace ornek810
{ class Program
   { static void Main(string[] args)
      { Console.Write("Birim matrisin tipini giriniz: ");
        int N=Convert.ToInt32(Console.ReadLine());
        int[,] A=new int[N, N];
        Console.WriteLine("\n  Birim matris");
        Console.WriteLine("  ------------");
        for (int i=0;i<N;i++)
         { for (int j=0;j<N;j++)
            { if (i==j) A[i,j]=1; else A[i,j]=0;
              Console.SetCursorPosition(5*j+5,4+i);
              Console.Write(A[i,j]); }}      Console.ReadLine(); }}}
```

JAVA

```
// Birim matris
import java.util.Scanner;
public class ornek810 {
public static void main(String[] args)
{  int i,j,N,T=0; Scanner klavye=new Scanner (System.in);
   System.out.print("Birim matrisin tipini giriniz: ");
   N=klavye.nextInt();    int A[][]=new int[N][N];
   System.out.println("\nBirim matris");
   System.out.println("------------");
   for(i=0;i<N;i++)
      { for(j=0;j<N;j++)
        { if(i==j) A[i][j]=1;else A[i][j]=0;
          System.out.print(A[i][j]+"\t");} System.out.println();} }}
```

MATLAB

```
% Birim matris
clear all; clc; N=input('Birim matrisin tipini giriniz: ');
fprintf('\n\tBirim matris\n\t------------\n'); A=eye(N); disp(A)
```

```
Birim matrisin tipini giriniz: 3

Birim matris
------------
   1   0   0
   0   1   0
   0   0   1
```

PYTHON

```
# Birim matris
import numpy as np
N=int(input("Birim matrisin tipini giriniz: ")); A=np.eye(N)
print('\nBirim matris\n------------'); print(A)
```

Örnek-8.11: Klavyeden girilen NxN tipindeki A ve B matrislerinin çarpımını ($C = A.B$) hesaplayan program.

$$\left.\begin{aligned} A &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \\ B &= \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \end{aligned}\right\} \Rightarrow C = A.B = \begin{bmatrix} a_{11}.b_{11} + a_{12}.b_{21} & a_{11}.b_{12} + a_{12}.b_{22} \\ a_{21}.b_{11} + a_{22}.b_{21} & a_{21}.b_{12} + a_{22}.b_{22} \end{bmatrix}$$

İki matrisin çarpımında; birinci matrisin satır elemanları ile ikinci matrisin sütun elemanları karşılıklı çarpılıp, sonuçlar toplanır. Olayı daha açık bir şekilde görmek ve indisler arasındaki ilişkiyi çıkarabilmek için $N = 2$ alınıp işlemler yazılırsa:

$$c_{11} = a_{11}.b_{11} + a_{12}.b_{21} \quad , \quad c_{12} = a_{11}.b_{12} + a_{12}.b_{22}$$
$$c_{21} = a_{21}.b_{11} + a_{22}.b_{21} \quad , \quad c_{22} = a_{21}.b_{12} + a_{22}.b_{22}$$

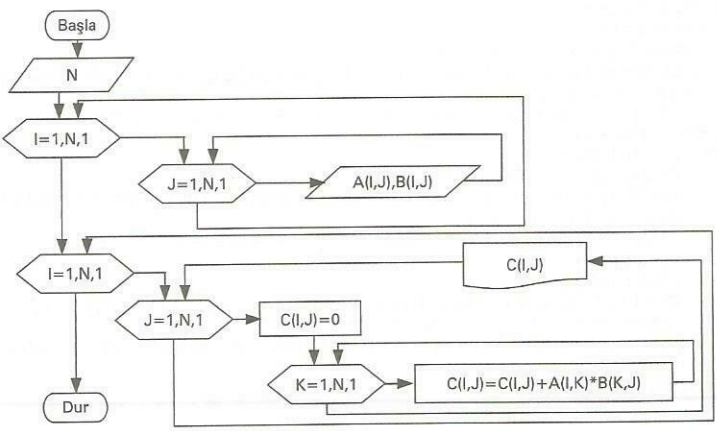
olur. Çarpma işleminde benzer değişimler görülmektedir. Satır sayısını gösteren indisler (I) koyu, sütun sayısını gösteren indisler (J) italik yazılmıştır. Bunların dışında bir indise (döngüye) daha gerek vardır. 'K' ile gösterilen bu döngü de, her çarpım teriminin elde edilmesinde 1'den N'e kadar değişmektedir. Bu durumda işlem için gerekli ifade, aşağıdaki gibi olmaktadır:

$$C(I,J) = C(I,J) + A(I,K) * B(K,J)$$

Burada satır sayısını gösteren I döngüsü, en dış döngü iken; K döngüsü ise en iç döngüdür. Yukarıda göz önüne alınan matris, kare matristir (NxN). Farklı tipteki matrislerin çarpılabilmesi için birinci matrisin sütun sayısı ile ikinci matrisin satır sayısı birbirine eşit olmalıdır. Aksi halde bu matrisler çarpılamaz.

$$(A)_{PxQ}.(B)_{QxR} = (C)_{PxR}$$

matrislerinin çarpımını gerçekleştirmek için I döngüsü $1-P$, J döngüsü $1-R$ ve K döngüsü de $1-Q$ değerleri arasında açılır.



$N = 2$ için çarpma işlemindeki adımlar, Tablo 8.11'deki gibi olmaktadır.

Tablo 8.11: Örnek-8.11'in çalışma prensibi

I	J	C(I,J)	K	Eski C(I,J)	Yeni C(I,J)
1	1	C(1,1)=0	1	0	C(1,1)=0+A(1,1)*B(1,1)
1	1	---	2	A(1,1)*B(1,1)	C(1,1)=A(1,1)*B(1,1)+A(1,2)*B(2,1)
1	2	C(1,2)=0	1	0	C(1,2)=0+A(1,1)*B(1,2)
1	2	---	2	A(1,1)*B(1,2)	C(1,2)=A(1,1)*B(1,2)+A(1,2)*B(2,2)
2	1	C(2,1)=0	1	0	C(2,1)=0+A(2,1)*B(1,1)
2	1	---	2	A(2,1)*B(1,1)	C(2,1)=A(2,1)*B(1,1)+A(2,2)*B(2,1)
2	2	C(2,2)=0	1	0	C(2,2)=0+A(2,1)*B(1,2)
2	2	---	2	A(2,1)*B(1,2)	C(2,2)=A(2,1)*B(1,2)+A(2,2)*B(2,2)

C

```
/* Matris çarpımı */
#include <stdio.h>
#include <conio.h>
int i,j,k,N,A[10][10],B[10][10],C[10][10];
void main()
{ clrscr();printf("Matrislerin tipini giriniz: ");scanf("%u",&N);
  printf("\n   A matrisi\n");printf("  ----------\n");
  for(i=0;i<N;i++)
    { for(j=0;j<N;j++)
        { gotoxy(5*j+5,5+i);scanf("%d",&A[i][j]); }}
  printf("\n   B matrisi\n");printf(" ----------\n");
  for(i=0;i<N;i++)
    { for(j=0;j<N;j++)
        { gotoxy(5*j+5,8+i+N);scanf("%d",&B[i][j]);}}
  printf("\n   C matrisi\n");printf("  ----------\n");
  for(i=0;i<N;i++)
    { for(j=0;j<N;j++)
        { C[i][j]=0; for(k=0;k<N;k++) C[i][j]+=A[i][k]*B[k][j];
        gotoxy(5*j+5,11+i+2*N);printf("%d",C[i][j]); }}   getch(); }
```

C++

```
// Matris çarpımı
#include <iostream>
#include <windows.h>
using namespace std;
int i,j,k,N;
int main()
{  system("cls");cout << "Matrislerin tipini giriniz: ";cin >> N;
   cout << "\n   A matrisi\n";cout << "  ---------- \n";
   int A[N][N],B[N][N],C[N][N];
   for(i=0;i<N;i++)
    { for(j=0;j<N;j++)
       { COORD xy = {5*j+5,4+i};
         SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),xy);
         cin >> A[i][j]; }}
   cout << "\n   B matrisi\n";cout << "  ---------- \n";
   for(i=0;i<N;i++)
     { for(j=0;j<N;j++)
       { COORD xy = {5*j+5,7+i+N};
         SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),xy);
         cin >> B[i][j]; }}
   cout << "\n   C matrisi\n";cout << "  ---------- \n";
   for(i=0;i<N;i++)
     { for(j=0;j<N;j++)
       { C[i][j]=0;for(k=0;k<N;k++) C[i][j]+=A[i][k]*B[k][j];
         COORD xy = {5*j+5,10+i+2*N};
         SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),xy);
         cout << C[i][j]; }}
   system("pause");return 0; }
```

C#

```
// Matris çarpımı
using System;
namespace ornek811
{ class Program
   { static void Main(string[] args)
    { Console.Write("Matrislerin tipini giriniz: ");
      int N = Convert.ToInt32(Console.ReadLine());
      int[,] A = new int[N, N]; int[,] B = new int[N, N];
      int[,] C = new int[N, N];
      Console.WriteLine("\n      A matrisi");
      Console.WriteLine("      ---------");
      for (int i=0;i<N;i++)
        { for (int j=0;j<N;j++)
            { Console.SetCursorPosition(5*j+5,4+i);
              A[i,j] = Convert.ToInt32(Console.ReadLine()); }}
      Console.WriteLine("\n      B matrisi");
      Console.WriteLine("      ---------");
      for (int i = 0; i < N; i++)
      {   for (int j = 0; j < N; j++)
            { Console.SetCursorPosition(5 * j + 5, 7 + i+N);
              B[i, j] = Convert.ToInt32(Console.ReadLine());}}
      Console.WriteLine("\n      C=A.B matrisi");
      Console.WriteLine("      -------------");
      for (int i = 0; i < N; i++)
      {   for (int j = 0; j < N; j++)
            { C[i,j]=0; for (int k=0;k<N;k++) C[i,j]+=A[i,k]*B[k, j];
              Console.SetCursorPosition(5 * j + 5, 10 + i+2*N);
              Console.Write(C[i, j]); } } Console.ReadLine();  } } }
```

```
Matrislerin tipini giriniz: 3

   A matrisi
   ---------
   1  2  3
   4  5  6
   7  8  9

   B matrisi
   ---------
   9  8  7
   6  5  4
   1  3  2

   C=A.B matrisi
   -------------
   24  27  21
   72  75  60
   120  123  99
```

JAVA

```
// Matris çarpımı
import java.util.Scanner;
public class ornek811 {
public static void main(String[] args)
{  int i,j,k,N;
   Scanner klavye=new Scanner (System.in);
   System.out.print("Matrislerin tipini giriniz: ");N=klavye.nextInt();
   int A[][]=new int[N][N];
   int B[][]=new int[N][N];
   int C[][]=new int[N][N];
   System.out.println("\nA matrisi");System.out.println("---------");
   for(i=0;i<N;i++){
    for(j=0;j<N;j++)
```

```
    {System.out.print("A("+i+","+j+")= ");A[i][j]=klavye.nextInt();}}
  System.out.println("\nB matrisi");System.out.println("---------");
  for(i=0;i<N;i++){
   for(j=0;j<N;j++)
    {System.out.print("B("+i+","+j+")= ");B[i][j]=klavye.nextInt();}}
  System.out.println("\nC=A.B matrisi");
  System.out.println("-------------");
  for(i=0;i<N;i++){
   for(j=0;j<N;j++)
    { C[i][j]=0;for(k=0;k<N;k++) C[i][j]+=A[i][k]*B[k][j];
      System.out.print(C[i][j]+"\t");} System.out.println();} }}
```

MATLAB

```
% Matris çarpımı
clear all; clc;fprintf('Matrisleri [a b;c d] şeklinde giriniz!\n');
fprintf('\nA matrisi\n---------\n'); A=input('');
fprintf('\nB matrisi\n---------\n'); B=input('');
C=A*B; fprintf('\nC=A.B matrisi\n-------------\n');disp(C)
```

PYTHON

```
# Matris çarpımı
import numpy as np
print("Matrisleri [a b;c d] şeklinde giriniz!")
print("\nA matrisi\n---------"); x=input(); A=np.matrix(x)
print("\nB matrisi\n---------"); x=input(); B=np.matrix(x); C=A*B
print("\nC=A.B matrisi\n-------------"); print(C)
```

Örnek-8.12: Klavyeden girilen *NxN* tipindeki *A* matrisinin ana köşegeninin üst kısmında kalan elemanların toplamını hesaplayan program.

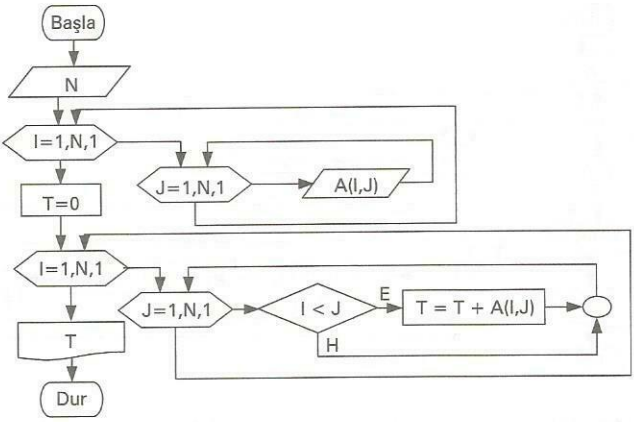
$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2N} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3N} \\ \dots & \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & a_{N3} & \dots & a_{NN} \end{bmatrix}$$

T

$N=3$ alınırsa toplam $T=a_{12}+a_{13}+a_{23}$ olur. Dikkat edilirse ana köşegen üstündeki bölgede kalan elemanların satır sayısını gösteren indis, sütun sayısını gösteren indisten daima küçüktür. Böylece, problemin çözümü için " I<J " koşulu yeterli olmaktadır. $N=3$ için işlem adımları Tablo 8.12'de verilmektedir.

Tablo 8.12: Örnek-8.12'nin çalışma prensibi

I	J	A(I,J)	I<J	Eski T	Yeni T
1	*1*	A(1,1)	Hayır	0	---
1	*2*	A(1,2)	***Evet***	0	T=0+A(1,2)
1	*3*	A(1,3)	***Evet***	A(1,2)	T=A(1,2)+A(1,3)
2	*1*	A(2,1)	Hayır	A(1,2)+A(1,3)	---
2	*2*	A(2,2)	Hayır	A(1,2)+A(1,3)	---
2	*3*	A(2,3)	***Evet***	A(1,2)+A(1,3)	T=A(1,2)+A(1,3)+A(2,3)
3	*1*	A(3,1)	Hayır	A(1,2)+A(1,3)+A(2,3)	---
3	*2*	A(3,2)	Hayır	A(1,2)+A(1,3)+A(2,3)	---
3	*3*	A(3,3)	Hayır	A(1,2)+A(1,3)+A(2,3)	---



Örnek-8.13: Klavyeden girilen *NxN* tipindeki *A* matrisinin ana köşegeninin alt kısmında kalan elemanların toplamını hesaplayan program.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

C

$$C = a_{21} \cdot a_{31} \cdot a_{32}$$

Dikkat edilirse ana köşegen altındaki bölgede satır sayısını gösteren indis, sütun sayısını gösteren indisten daima büyüktür. Bu nedenle problemi çözümü için Örnek-8.12'deki akış diyagramında " I>J " koşulu kontrol edilir.

Örnek-8.14: Klavyeden girilen *NxN* tipindeki *A* matrisinin yedek köşegeninin üstünde kalan elemanların toplamını hesaplayan program.

Yedek köşegenin üst bölgesindeki elemanların indisleri arasındaki ilişkiyi görebilmek için $N = 4$ alınırsa, toplam aşağıdaki gibi olur.

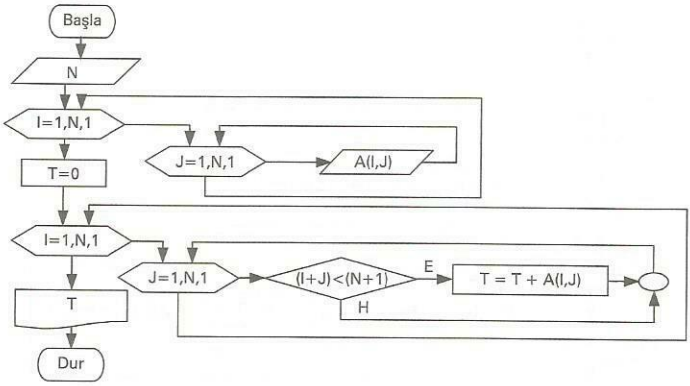
$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

T

I	J	I+J
1	1	2<5
1	2	3<5
1	3	4<5
2	1	3<5
2	2	4<5
3	1	4<5

$$T = a_{11} + a_{12} + a_{13} + a_{21} + a_{22} + a_{31}$$

$N = 4$ alındığında yedek köşegenin üstündeki elemanların indislerinin toplamı *daima 5'ten küçük* olmaktadır. Dolayısıyla *NxN* tipindeki bir matrisin yedek köşegeninin üstündeki elemanların indisleri toplamı daima $(N+1)$ den küçük olacaktır. Programda koşul olarak " (I+J)<(N+1) " yazılarak istenilen bölgedeki elemanlar toplanmış olur.



Örnek-8.15: Klavyeden girilen *NxN* tipindeki *A* matrisinin yedek köşegeninin altında kalan elemanların toplamını hesaplayan program.

Örnek-8.14'teki akış diyagramında "(I+J)>(N+1)" koşulu kontrol ettirilerek, istenilen sonuca ulaşılır.

Örnek-8.16: Klavyeden girilen *NxN* tipindeki *A* matrisinin sondan bir önceki satır ve sütunundaki elemanların toplamını hesaplayan program.

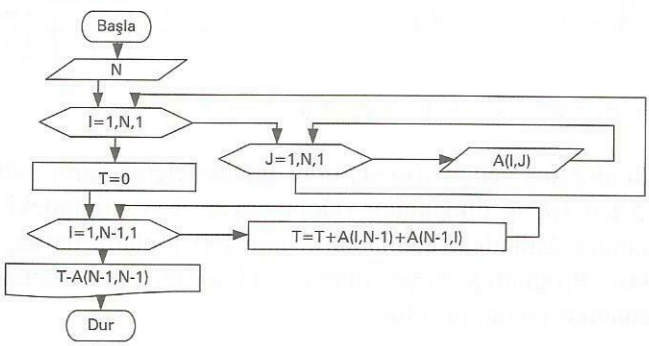
$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

$$T = \underbrace{a_{13} + a_{23} + a_{33}}_{A(I,3)} + \underbrace{a_{31} + a_{32} + a_{33}}_{A(3,I)}$$

Seçili bölgedeki elemanlar toplamı iki kısma ayrılır:

1. kısımdaki elemanların sütun indisleri daima 3
2. kısımdaki elemanların satır indisleri daima 3'tür.

Örnek matrisin tipi 4x4 olduğundan seçili bölgedeki elemanların indisleri değişimi 1-3 arasındadır. Dikkat edilmesi gerekli olan bir nokta da A(3,3) elemanının iki defa toplanmasıdır. Dolayısıyla sonuç yazdırılırken A(3,3) elemanı kadar azaltmak gerekecektir.



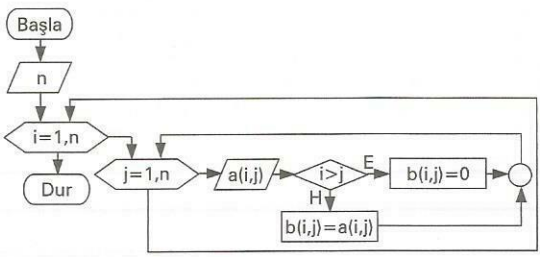
Not

- ✓ Matrisin ana köşegen üzerinde satır ve sütun indisleri daima birbirine eşittir.
- ✓ Yedek köşegen üzerinde satır ve sütun indislerinin toplamı daima sabit olup kare matrisin tipinin (satır/sütun sayısının) bir fazlasıdır.
- ✓ Ana köşegenin üst bölgesindeki elemanların satır indisi, daima sütun indisinden küçüktür.
- ✓ Ana köşegenin alt bölgesindeki elemanların satır indisi, daima sütun indisinden büyüktür.
- ✓ Yedek köşegenin üst bölgesindeki elemanların satır ve sütun indislerinin toplamı, daima kare matrisin tipinin bir fazlasından küçüktür.
- ✓ Yedek köşegenin alt bölgesindeki elemanların satır ve sütun indislerinin toplamı, daima kare matrisin tipinin bir fazlasından büyüktür.

Örnek-8.17: Klavyeden girilen nxn tipindeki a matrisinden üst üçgen b matrisini oluşturan program.

3x3 tipindeki matristen 3 elemanlı dizi oluşur. Ana köşegen altındaki elemanların sıfır olduğu matrislere, "*üst üçgen matris*" denir. $n = 3$ için

$$a = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \Rightarrow b = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a_{22} & a_{23} \\ 0 & 0 & a_{33} \end{bmatrix}$$ şeklindedir.



C

```
/* Üst üçgen matris */
#include <stdio.h>
#include <conio.h>
int i,j,n,a[10][10],b[10][10];
void main()
{ clrscr();printf("Matrisin tipini giriniz: ");scanf("%u",&n);
  printf("\n     Matris\n");printf("  ----------\n");
  for(i=0;i<n;i++) { for(j=0;j<n;j++)
      { gotoxy(5*j+5,5+i);scanf("%d",&a[i][j]);
        if (i>j) b[i][j]=0; else b[i][j]=a[i][j];}}
  printf("\nÜst üçgen matris\n");printf("----------------\n");
  for(i=0;i<n;i++) { for(j=0;j<n;j++)
      { gotoxy(5*j+5,8+i+n);printf("%d",b[i][j]);}} getch(); }
```

C++

```
// Üst üçgen matris
#include <iostream>
#include <windows.h>
using namespace std;
int i,j,n;
int main()
{  system("cls");cout << "Matrisin tipini giriniz: ";cin >> n;
   cout << "\n     Matris\n";cout << "  ---------- \n";
```

```
    int a[n][n],b[n][n];
    for(i=0;i<n;i++) { for(j=0;j<n;j++)
        { COORD xy = {5*j+5,4+i};
          SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),xy);
          cin >> a[i][j]; if (i>j) b[i][j]=0; else b[i][j]=a[i][j]; }}
    cout << "\nÜst üçgen matris\n";cout << "----------------\n";
    for(i=0;i<n;i++) { for(j=0;j<n;j++)
        { COORD xy = {5*j+5,8+i+n};
          SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),xy);
          cout << b[i][j]; }}     system("pause");return 0; }
```

C#

```
// Üst üçgen matris
using System;
namespace ornek817
{ class Program
   { static void Main(string[] args)
    { Console.Write("Matrisin tipini giriniz: ");
      int n = Convert.ToInt32(Console.ReadLine());
      int[,] a = new int[n, n]; int[,] b = new int[n, n];
      Console.WriteLine("\n        Matris");
      Console.WriteLine("      ---------");
      for (int i=0; i<n; i++) { for (int j=0; j<n; j++)
            {  Console.SetCursorPosition(5*j+5,4+i);
               a[i,j]=Convert.ToInt32(Console.ReadLine());
               if (i>j) b[i,j]=0; else b[i,j]=a[i,j]; }}
      Console.WriteLine("\nÜst üçgen matris");
      Console.WriteLine("----------------");
      for (int i=0;i<n;i++) { for (int j=0; j<n;j++)
            {  Console.SetCursorPosition(5*j+5,7+i+n);
               Console.Write(b[i,j]); }}   Console.ReadLine(); } } }
```

JAVA

```
// Üst üçgen matris
import java.util.Scanner;
public class ornek817 {
public static void main(String[] args)
{  int i,j,n; Scanner klavye=new Scanner (System.in);
   System.out.print("Matrisin tipini giriniz: ");n=klavye.nextInt();
   int a[][]=new int[n][n]; int b[][]=new int[n][n];
   System.out.println("\nMatris");   System.out.println("------");
   for(i=0;i<n;i++){ for(j=0;j<n;j++)
     { System.out.print("a("+(i+1)+","+(j+1)+")= ");
       a[i][j]=klavye.nextInt();
       if (i>j) b[i][j]=0; else b[i][j]=a[i][j]; }}
   System.out.println("\nÜst üçgen matris");
   System.out.println("----------------");
   for(i=0;i<n;i++){ for(j=0;j<n;j++)
    System.out.print(b[i][j]+"\t");System.out.println();}}}
```

MATLAB

```
% Üst üçgen matris
clear all; clc;fprintf('Matrisi [a b;c d] şeklinde giriniz!\n');
fprintf('\nMatris\n------\n'); a=input('');
b=triu(a);fprintf('\nÜst üçgen matris\n----------------\n');disp(b)
```

PYTHON

```
# Üst üçgen matris
import numpy as np
print("Matrisi [a b;c d] şeklinde giriniz!")
print("\nMatris\n------"); x=input(); a=np.matrix(x); b=np.triu(a)
print("\nÜst üçgen matris\n----------------"); print(b)
```

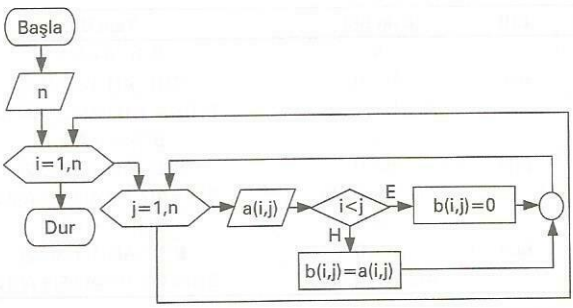
```
Matrisi [a b;c d] şeklinde giriniz!

Matris
------
1 2 3;4 5 6;7 8 9

Üst üçgen matris
----------------
[[1 2 3]
 [0 5 6]
 [0 0 9]]
```

Örnek-8.18: Klavyeden girilen *nxn* tipindeki *a* matrisinden alt üçgen *b* matrisini oluşturan program.

3x3 tipindeki matristen 3 elemanlı dizi oluşur. Ana köşegen üstündeki elemanların sıfır olduğu matrislere, "*alt üçgen matris*" denir. $n = 3$ için

$$a = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \Rightarrow b = \begin{bmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

şeklindedir.

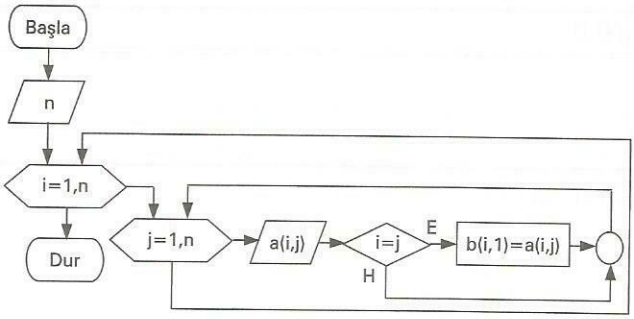


Örnek-8.19: Klavyeden girilen *nxn* tipindeki *a* matrisinden köşegen *b* matrisini oluşturan program.

3x3 tipindeki matristen 3 elemanlı dizi oluşur. Ana köşegen elemanlarının oluşturduğu matrislere, "*köşegen matris*" denir. $n = 3$ için

$$a = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \Rightarrow b = \begin{bmatrix} a_{11} \\ a_{22} \\ a_{33} \end{bmatrix}$$

şeklindedir.



Örnek-8.20: Klavyeden girilen NxN tipindeki A matrisinin her bir satır elemanları toplamından N elemanlı B dizisini oluşturan program.

$N = 3$ alınırsa 3x3 tipindeki matristen 3 elemanlı dizi oluşur.

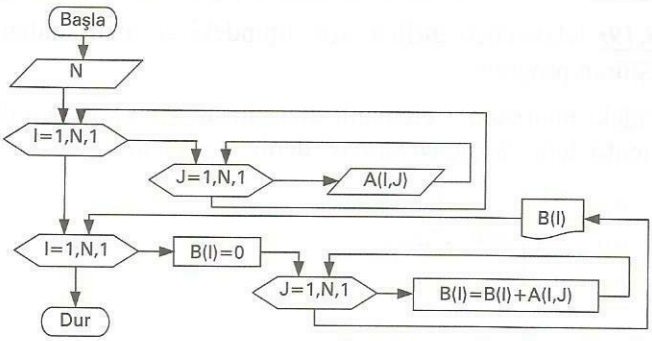
$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$B(1) = a_{11} + a_{12} + a_{13}$$
$$B(2) = a_{21} + a_{22} + a_{23}$$
$$B(3) = a_{31} + a_{32} + a_{33}$$

Satır elemanları toplamı olduğu için dizinin eleman indisi ile matris elemanların satır indisi *daima birbirine eşittir*. Programda iki döngü kullanılarak $B(I) = B(I) + A(I,J)$ ardışık toplama işlemiyle dizi oluşturulmuş olur. $N = 3$ için işlem adımları Tablo 8.13'te verilmektedir.

Tablo 8.13: Örnek-8.20'nin çalışma prensibi

I	J	B(I)	Eski B(I)	Yeni B(I)
1	1	B(1)	0	B(1)=0+A(1,1)
1	2		A(1,1)	B(1)=A(1,1)+A(1,2)
1	3		A(1,1)+A(1,2)	B(1)=A(1,1)+A(1,2)+A(1,3)
2	1	B(2)	0	B(2)=0+A(2,1)
2	2		A(2,1)	B(2)=A(2,1)+A(2,2)
2	3		A(2,1)+A(2,2)	B(2)=A(2,1)+A(2,2)+A(2,3)
3	1	B(3)	0	B(3)=0+A(3,1)
3	2		A(3,1)	B(3)=A(3,1)+A(3,2)
3	3		A(3,1)+A(3,2)	B(3)=A(3,1)+A(3,2)+A(3,3)



Örnek-8.21: Klavyeden girilen NxN tipindeki A matrisinin her bir sütun elemanları çarpımından N elemanlı B dizisini oluşturan program.

Örnek-8.20'dekine benzer şekilde bu program, küçük bir değişiklikle yazılabilir. $N=3$ alınırsa, 3x3 tipindeki matristen 3 elemanlı B dizisi oluşur. Bu dizinin elemanları;

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad \begin{aligned} B(1) &= a_{11} \,.\, a_{21} \,.\, a_{31} \\ B(2) &= a_{12} \,.\, a_{22} \,.\, a_{32} \\ B(3) &= a_{13} \,.\, a_{23} \,.\, a_{33} \end{aligned}$$

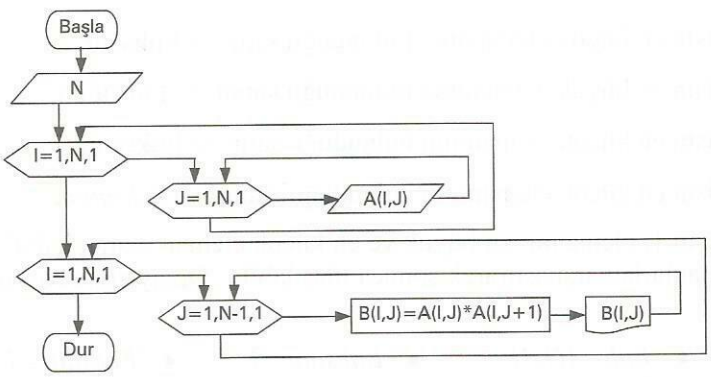
şeklindedir. Dikkat edilirse dizinin eleman indisiyle matris elemanların sütun sayısını gösteren indis *daima birbirine* eşittir. Örnek-8.20'de olduğu gibi iki döngü kullanılarak ve $B(I)=B(I)*A(J,I)$ ifadesi yazılarak dizi oluşturulur.

Örnek-8.22: Klavyeden girilen NxN tipindeki A matrisini; bir sonraki sütunu bir önceki sütunla çarpmak suretiyle NxN tipinden $NxN-1$ tipine indirgeyen program.

İstenilen indirgeme işlemini daha ayrıntılı irdelemek için $N=4$ alınıp, 4x4 tipindeki A matrisinden oluşan 4x3 tipindeki B matrisi

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \Rightarrow B=\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \\ b_{41} & b_{42} & b_{43} \end{bmatrix} = \begin{bmatrix} a_{11}.a_{12} & a_{12}.a_{13} & a_{13}.a_{14} \\ a_{21}.a_{22} & a_{22}.a_{23} & a_{23}.a_{24} \\ a_{31}.a_{32} & a_{32}.a_{33} & a_{33}.a_{34} \\ a_{41}.a_{42} & a_{42}.a_{43} & a_{43}.a_{44} \end{bmatrix}$$

şeklinde oluşur. $I=1,N,1$ ve $J=1,N-1,1$ dış ve iç döngü olmak üzere, indirgeme işlemi $B(I,J)=A(I,J)*A(I,J+1)$ işlemiyle gerçekleşmektedir.

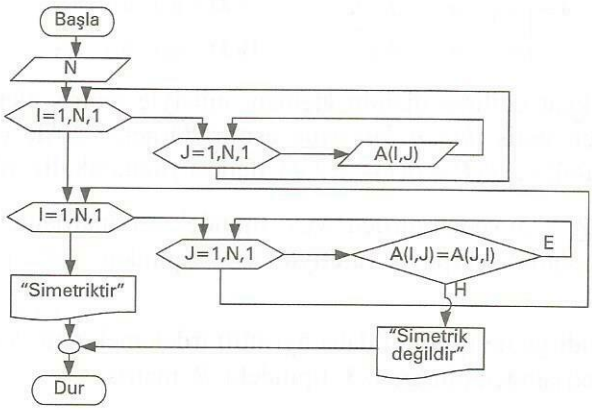


Örnek-8.23: Klavyeden girilen NxN tipindeki A matrisinin simetrik olup olmadığını kontrol eden program.

Bir matrisin "*simetrik*" olabilmesi için; ana köşegene göre karşılıklı bütün elemanlarının eşit olması gereklidir. Örneğin 3x3 tipindeki bir matriste;

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad \begin{matrix} a_{12}=a_{21} \\ a_{13}=a_{31} \\ a_{23}=a_{32} \end{matrix} \quad \text{ise } A \text{ matrisi simetriktir.}$$

Karşılıklı elemanlardan bahsedildiğinden bir elemandaki satır indisi, diğerinde sütun indisi olmaktadır. Sonucun bulunmasında $A(I,J)=A(J,I)$ koşulunun denetlenmesi yeterlidir.



Örnek-8.24: Klavyeden girilen *NxN* tipindeki *A* matrisinin en büyük ve en küçük elemanlarını ve bu elemanların yerlerini bulup sonuçları ekrana yazdıran program.

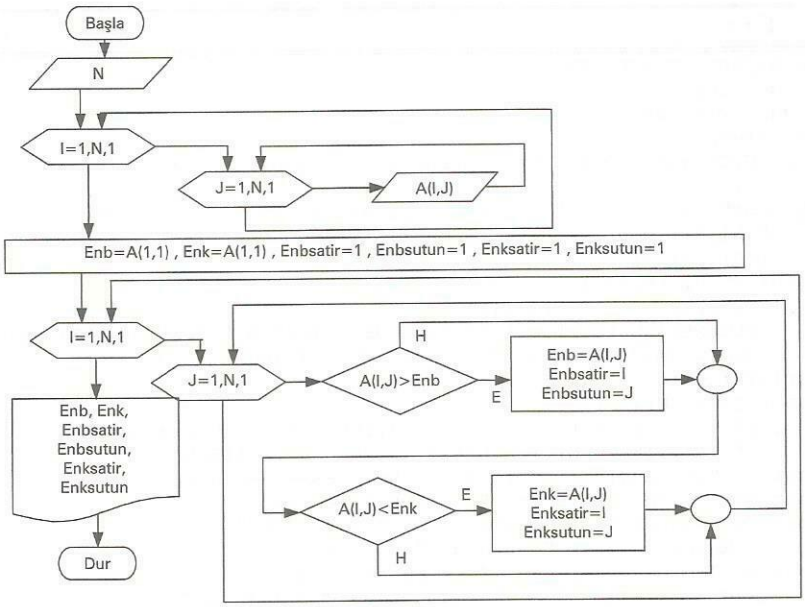
Önce değişken isimleri belirlenir:

- ✓ Matrisin en büyük elemanı => Enb,
- ✓ Matrisin en küçük elemanı => Enk,
- ✓ Matrisin en büyük elemanının bulunduğu satır => Enbsatir,
- ✓ Matrisin en büyük elemanının bulunduğu sütun => Enbsutun,
- ✓ Matrisin en küçük elemanının bulunduğu satır => Enksatir,
- ✓ Matrisin en küçük elemanının bulunduğu sütun => *Enksutun*,

Matrisin 1. elemanını en büyük ve en küçük eleman olarak kabul edip bunu diğer elemanlarla karşılaştırarak sonuca ulaşılabilir. Dolayısıyla başlangıç koşulu olarak;

- *Enb=A(1,1)*
- *Enk=A(1,1)*
- *Enbsatir=1*
- *Enbsutun=1*
- *Enksatir=1*
- *Enksutun=1*

atamaları yapılır. Bu varsayımdan sonra matrisin diğer elemanları, *Enb* ve *Enk* ile karşılaştırılır. Eğer eleman *Enb*'den büyük ise yeni *Enb* olarak; eğer eleman *Enk*'dan küçük ise yeni *Enk* olarak atanır. Satır ve sütun değişkenleri de bu değişikliklere uyarlanır.



C

```
/* En büyük-küçük eleman */
#include <stdio.h>
#include <conio.h>
int A[10][10],i,j,N,Enksatir=0,Enksutun=0,Enbsatir=0,Enbsutun=0,Enk,Enb;
main()
{ clrscr();printf("Kare matrisin satır sayısı: ");scanf("%u",&N);
  printf("\n   A matrisi\n");printf("  ------------\n");
  for(i=0;i<N;i++)
   { for(j=0;j<N;j++)
       { gotoxy(5*j+5,5+i);scanf("%d",&A[i][j]);}}
  Enk=A[0][0];Enb=A[0][0];
  for(i=0;i<N;i++)
   { for(j=0;j<N;j++)
      { if (A[i][j]<Enk) { Enk=A[i][j];Enksatir=i;Enksutun=j; }
        if (A[i][j]>Enb) { Enb=A[i][j];Enbsatir=i;Enbsutun=j; }}}
  printf("\nSonuçlar: \n");printf("------------\n");
  printf("En küçük eleman= %d\n",Enk);
  printf("Konum: %u.satır ve %u.sütunda.\n", Enksatir+1, Enksutun+1);
  printf("En büyük eleman= %d\n",Enb);
  printf("Konum: %u.satır ve %u.sütunda.\n",Enbsatir+1, Enbsutun+1);
  getch();return 0; }
```

```
Kare matrisin satır sayısı: 3

  A matrisi
  ------------
  2  1  6
  3  8  4
  7  9  2

Sonuçlar:
------------
En küçük eleman= 1
Konum: 1.satır ve 2.sütunda.
En büyük eleman= 9
Konum: 3.satır ve 2.sütunda.
```

C++

```
// En büyük-küçük eleman
#include <iostream>
#include <windows.h>
using namespace std;
int i,j,N,Enksatir=0,Enksutun=0,Enbsatir=0,Enbsutun=0;
int main()
{  system("cls");cout<<"Kare matrisin satır sayısı: ";cin>>N;
   int A[N][N]; cout << "\n   A matrisi\n";cout<<"  ---------- \n";
   for(i=0;i<N;i++)
    {for (j=0;j<N;j++)
      {COORD xy = {5*j+5,4+i};
       SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),xy);
       cin >> A[i][j]; } } int Enk=A[0][0],Enb=A[0][0];
   for(i=0;i<N;i++)
    {for(j=0;j<N;j++)
      { if (A[i][j]<Enk) { Enk=A[i][j];Enksatir=i;Enksutun=j; }
        if (A[i][j]>Enb) { Enb=A[i][j];Enbsatir=i;Enbsutun=j; }}}
   cout << "\nSonuçlar: \n";cout << "------------\n";
   cout << "En küçük eleman= " << Enk << endl;
   cout << "Konum: "<< Enksatir+1 << ".satır ve " <<
           Enksutun+1 << ".sütunda." << endl;
   cout << "En büyük eleman= " << Enb << endl;
        cout << "Konum: "<< Enbsatir+1 << ".satır ve " <<
           Enbsutun+1 << ".sütunda." << endl;
   system("pause"); return 0; }
```

C#

```
// Matrisin en büyük-küçük elemanları ve yerleri
using System;
namespace ornek824
{ class Program
   { static void Main(string[] args)
      { Console.Write("Kare matrisin satır sayısı: ");
        int N = Convert.ToInt32(Console.ReadLine());
        int[,] A = new int[N, N];
        int i,j,Enksatir=0,Enksutun=0,Enbsatir=0,Enbsutun=0;
        Console.WriteLine("\n     A matrisi");
        Console.WriteLine("     ------------");
        for (i = 0; i < N; i++)
         { for (j = 0; j < N; j++)
            { Console.SetCursorPosition(5 * j + 5, 4 + i);
              A[i, j] = Convert.ToInt32(Console.ReadLine()); }}
        int Enk = A[0, 0], Enb = A[0, 0];
        for (i = 0; i < N; i++)
         { for (j = 0; j < N; j++)
            { if (A[i,j]<Enk) { Enk=A[i,j]; Enksatir=i; Enksutun=j; }
              if (A[i,j]>Enb) { Enb=A[i,j]; Enbsatir=i; Enbsutun=j; }}}
        Console.WriteLine("\nSonuclar:");
        Console.WriteLine("------------");
        Console.WriteLine("En kucuk eleman= {0}",Enk);
        Console.WriteLine("Konum: {0}.satir ve {1}.sutunda.",
                          Enksatir+1,Enksutun+1);
        Console.WriteLine("En büyük eleman= {0}", Enb);
        Console.WriteLine("Konum: {0}.satir ve {1}.sutunda.",
                          Enbsatir + 1, Enbsutun + 1);
        Console.ReadLine();          }    }    }
```

JAVA

```
// Matris elemanları büyük-küçük
import java.util.Scanner;
public class ornek824 {
public static void main(String[] args)
{  int i,j,N,enksatir=0,enksutun=0,enbsatir=0,enbsutun=0;
   Scanner klavye=new Scanner (System.in);
   System.out.print("Kare matrisin satır sayısı: ");
   N=klavye.nextInt(); int A[][]=new int[N][N];
   System.out.println("\nA matrisi"); System.out.println("---------");
   for(i=0;i<N;i++){
    for(j=0;j<N;j++)
     {System.out.print("A("+i+","+j+")= ");A[i][j]=klavye.nextInt();}}
   int enk=A[0][0],enb=A[0][0];
   for(i=0;i<N;i++)
    { for(j=0;j<N;j++)
       { if(A[i][j]<enk) {enk=A[i][j];enksatir=i;enksutun=j;}
         if(A[i][j]>enb) {enb=A[i][j];enbsatir=i;enbsutun=j;}}}
   System.out.println("\nSonuçlar:"); System.out.println("---------");
   System.out.println("En küçük eleman= "+enk);
   System.out.println("Konum: "+(enksatir+1)+". satir ve
"+(enksutun+1)+". sutunda.");
   System.out.println("En büyük eleman= "+enb);
   System.out.println("Konum: "+(enbsatir+1)+". satir ve
"+(enbsutun+1)+". sutunda."); }}
```

MATLAB

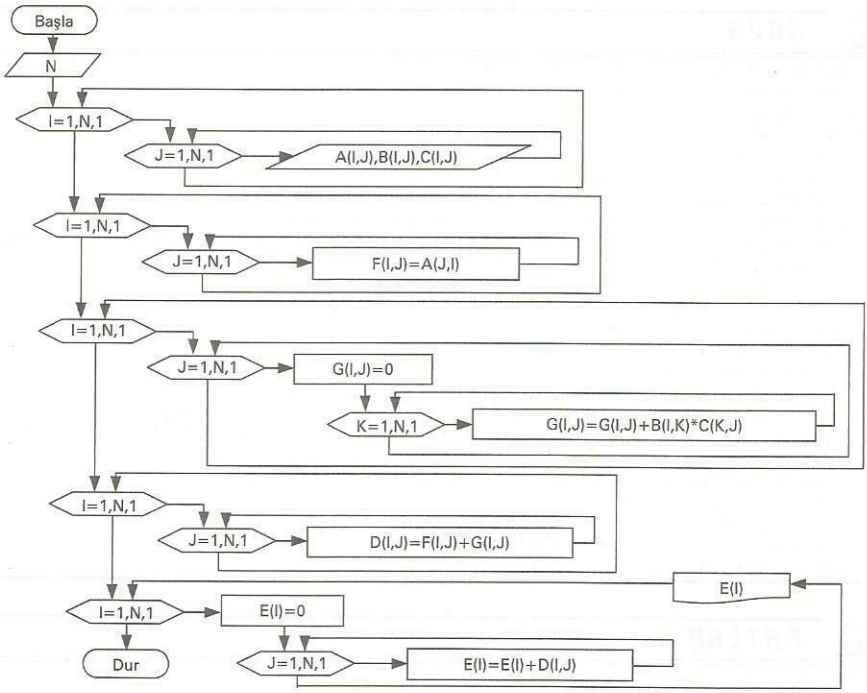
```
% Matris elemanları büyük-küçük
clear all; clc;fprintf('Matrisi [a b;c d] şeklinde giriniz!\n');
fprintf('\nMatris\n------\n'); a=input('');
enk=min(min(a));[enksatir,enksutun]=find(a==enk);
enb=max(max(a));[enbsatir,enbsutun]=find(a==enb);
fprintf('\nSonuçlar:\n---------\n');
fprintf('En küçük eleman= %d\n',enk);
fprintf('Konum: %d. satır ve %d. sütunda.\n',enksatir,enksutun);
fprintf('En büyük eleman= %d\n',enb);
fprintf('Konum: %d. satır ve %d. sütunda.\n',enbsatir,enbsutun);
```

PYTHON

```
# Matris elemanları büyük-küçük
import numpy as np
print("Matrisi [a b;c d] şeklinde giriniz!")
print("\nMatris\n------"); x=input(); a=np.matrix(x)
enk=a.min();enkyer=np.unravel_index(a.argmin(),a.shape)
enb=a.max();enbyer=np.unravel_index(a.argmax(),a.shape)
print("\nSonuçlar:\n---------"); print("En küçük eleman= %d"%enk)
print("Konum: %d. satır ve %d. sütunda."%(enkyer[0]+1,enkyer[1]+1))
print("En büyük eleman= %d"%enb)
print("Konum: %d. satır ve %d. sütunda."%(enbyer[0]+1,enbyer[1]+1))
```

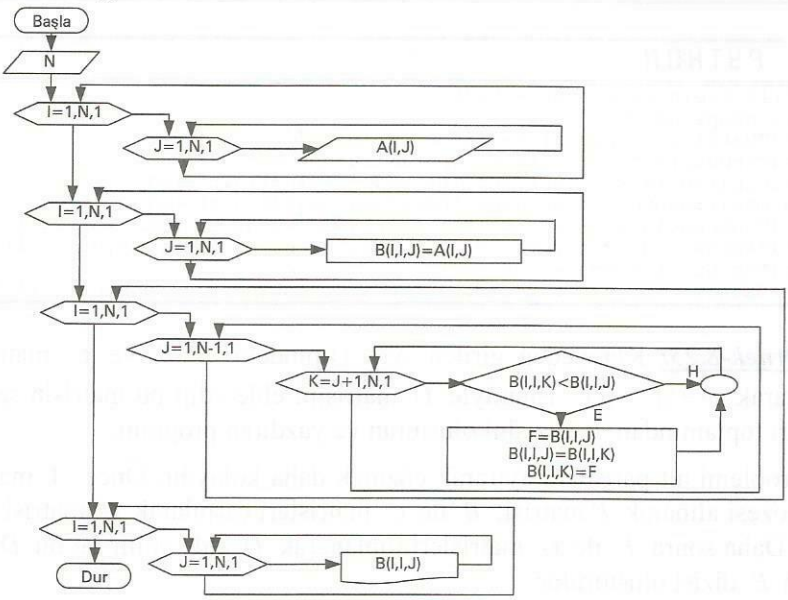
Örnek-8.25: Klavyeden girilen NxN tipindeki A, B ve C matrislerini kullanarak $D = A^T + B.C$ işlemiyle D matrisini elde edip bu matrisin satır elemanları toplamından E dizisini oluşturan ve yazdıran program.

Problemi alt parçalara ayırarak çözmek daha kolaydır. Önce A matrisinin transpozesi alınarak F matrisi; B ile C matrisleri çarpılarak G matrisi oluşturulur. Daha sonra F ile G matrisleri toplanarak D elde edilir ve bu D matrisinden E dizisi oluşturulur.



Örnek-8.26: Klavyeden girilen *NxN* tipindeki *A* matrisinin her bir satır elemanlarını kendi aralarında küçükten büyüğe doğru sıralayan program.

Bunun için matrisin her bir satırının elemanları diziye aktarılarak sıralanabilir. Başka bir çözüm yolu da matrisi üç boyutlu yapıya dönüştürüp, ilk indisini dizi göstericisi gibi kullanmaktır.



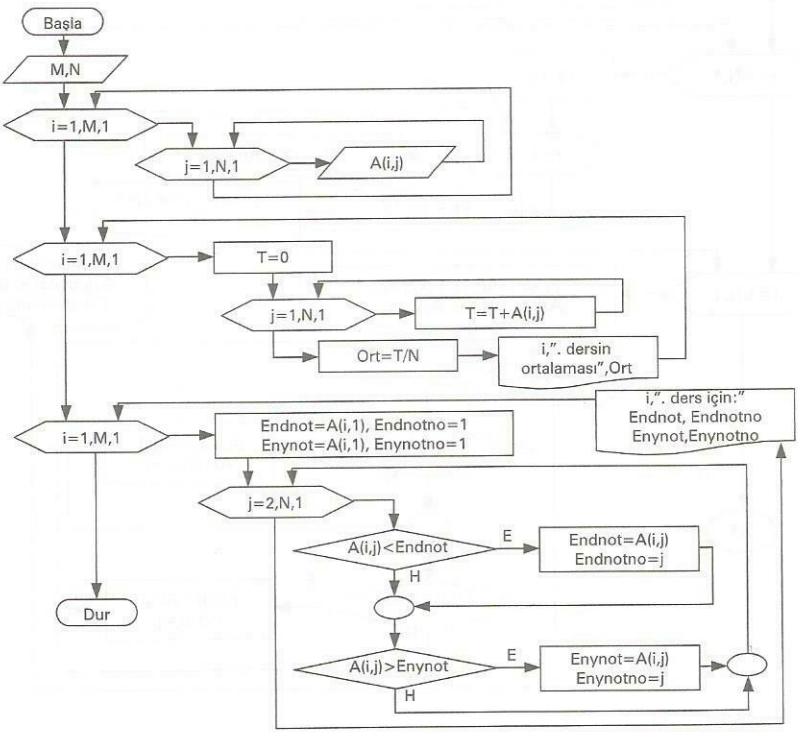
Örnek-8.27: N kişilik bir sınıfın M tane dersine ait notları, bilgisayara girilmektedir. Buna göre;

i. Herbir dersten sınıf ortalamasını,

ii. Herbir dersten en düşük ve en yüksek not ile bu notu alan öğrenci numaralarını veren program.

N kişilik bir sınıfın M tane dersine ait notları, klavyeden girilerek bellekte $(A)_{MxN}$ matrisine aktarılmaktadır. Bu matrisin satırlarında dersler, sütunlarında ise öğrenciler yer alır. Herbir satır elemanlarının toplamının öğrenci sayısına (N) bölümü ile derslerden sınıf ortalamaları bulunur. En düşük ve en yüksek notu bulmak için ise; matrisin herbir satırını dizi gibi ele alıp, bu dizilerin en küçük ve büyük elemanlarını ilgili not ve bu elemanların indislerini de ilgili öğrenci numarası olarak değerlendirilir.

$$A = Dersler \begin{cases} \overbrace{\begin{bmatrix} a_{11} & a_{12} & \cdots & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & \cdots & a_{2N} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{M1} & a_{M2} & \cdots & \cdots & a_{MN} \end{bmatrix}}^{Ogrenci\ numaralari} \end{cases} \quad 1.\ dersin\ ortalamasi:\ Ort_1 = \frac{\sum_{i=1}^{N} a_{1i}}{N}$$



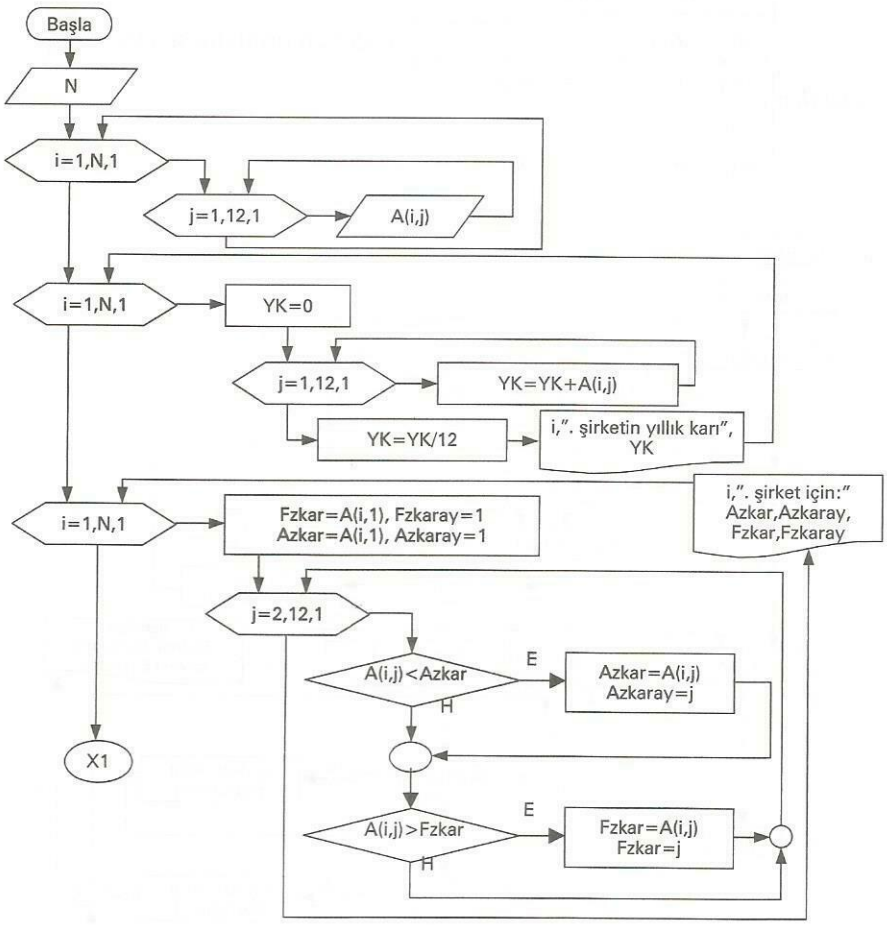
Örnek-8.28: N tane şirketin bir yıla ait aylık kâr oranları, bilgisayara girilmektedir. Buna göre;

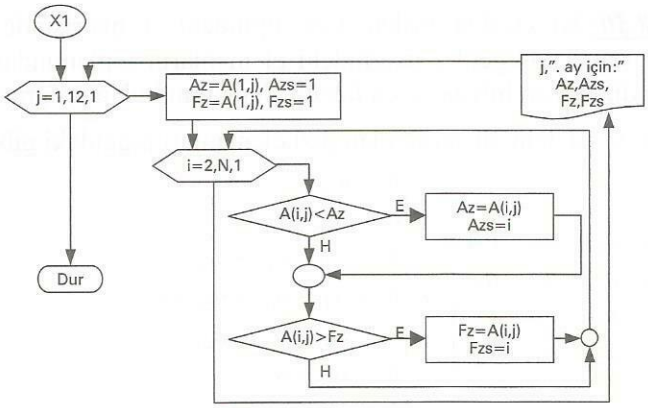
i. Herbir şirketin yıllık ortalama kâr miktarlarını,

ii. Herbir şirketin en az ve en fazla kâr elde ettiği ayları ve miktarlarını,

iii. Herbir aya ait en az ve en fazla kâr miktarları ve şirketleri veren program.

N tane şirketin bir yıla ait aylık kâr oranları, bilgisayara girilerek $(A)_{Nx12}$ matrisinde saklanır. Bu matrisin satırlarında şirketler, sütunlarında da aylar yer alır. Herbir şirketin yıllık ortalama kâr miktarları için satır elemanları toplanır ve 12'ye bölünür. Herbir şirketin en az ve en fazla kâr elde ettiği ayları ve miktarlarını tespit etmek için ise herbir satırdaki en küçük ve en büyük değerli eleman bulunur. Herbir sütundaki en küçük değerli eleman, o aya ait en az kâr miktarını ve satır indisi de şirketi; herbir sütundaki en büyük değerli eleman da en yüksek kâr miktarını gösterir.

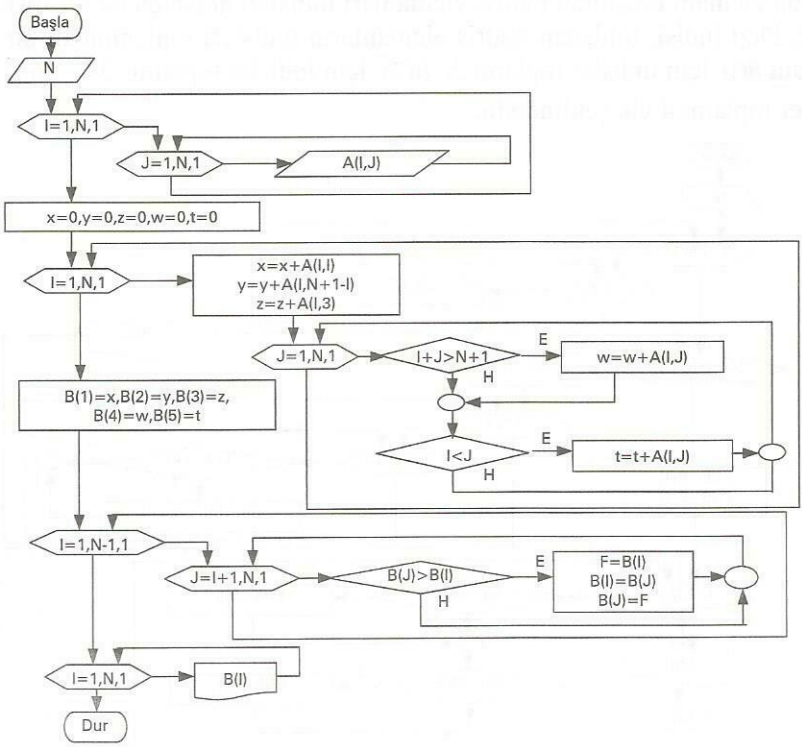




Örnek-8.29: Klavyeden girilen NxN tipindeki A matrisinde

$$N=4 \Rightarrow A=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \rightarrow \begin{cases} x = a_{11}+a_{22}+a_{33}+a_{44} \\ y = a_{14}+a_{23}+a_{32}+a_{41} \\ z = a_{13}+a_{23}+a_{33}+a_{43} \\ w = a_{24}+a_{33}+a_{34}+a_{42}+a_{43}+a_{44} \\ t = a_{12}+a_{13}+a_{14}+a_{23}+a_{24}+a_{34} \end{cases}$$

katsayılarını hesaplayıp bunları büyükten küçüğe doğru sıralayan program.



Örnek-8.30: Klavyeden girilen NxN tipindeki A matrisinde yedek köşegene paralel sanal köşegenler üzerindeki elemanların toplamından B dizisini oluşturup bu dizinin en büyük ve en küçük elemanını bulan program.

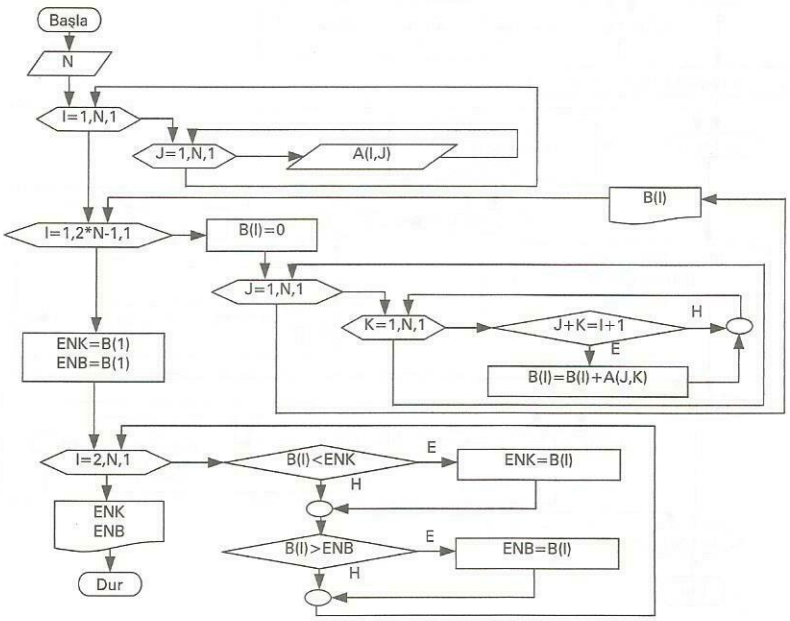
Örneğin $N = 4$ için oluşacak olan dizi elemanları aşağıdaki gibidir.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \rightarrow \begin{cases} b_1 = a_{11} \\ b_2 = a_{21} + a_{12} \\ b_3 = a_{31} + a_{22} + a_{13} \\ b_4 = a_{41} + a_{32} + a_{23} + a_{14} \\ b_5 = a_{42} + a_{33} + a_{24} \\ b_6 = a_{43} + a_{34} \\ b_7 = a_{44} \end{cases}$$

Öncelikle oluşacak olan B dizisinin eleman sayısının belirlenmesi gerekir. B dizisinin eleman sayısı;

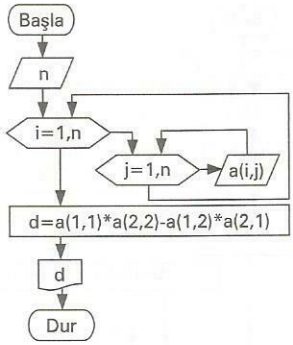
• $N = 2$ için 3 • $N = 3$ için 5 • $N = 4$ için 7 • $N = 5$ için 9 ...

şeklinde değişmektedir. Yani dizinin eleman sayısı, kare matrisin satır/sütun sayısının iki katının bir eksiğidir. Şimdi de dizi elemanlarını oluşturan matris elemanlarının indisleri arasındaki ilişki ortaya çıkarılmalıdır. Her dizi elemanını oluşturan matris elemanlarının indisleri toplamı eşittir. Ayrıca dizi elemanı indisi ile bu elemanı oluşturan matris elemanları indisleri arasında da bir ilişki mevcuttur. Dizi indisi, toplanan matris elemanların indisleri toplamından bir eksiktir. Yani $B(1)$ için indisler toplamı 2, $B(2)$ için indisler toplamı 3, $B(3)$ için indisler toplamı 4 vb. şeklindedir.



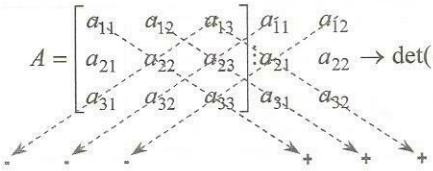
Örnek-8.31: Klavyeden girilen $2x2$ tipindeki A matrisinin determinantını hesaplayan program.

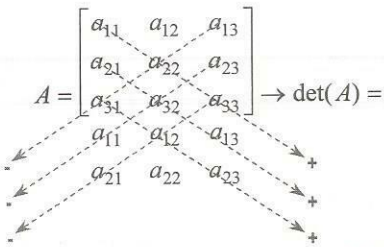
$$A=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \rightarrow \det(A) = a_{11}a_{22} - a_{12}a_{21}$$



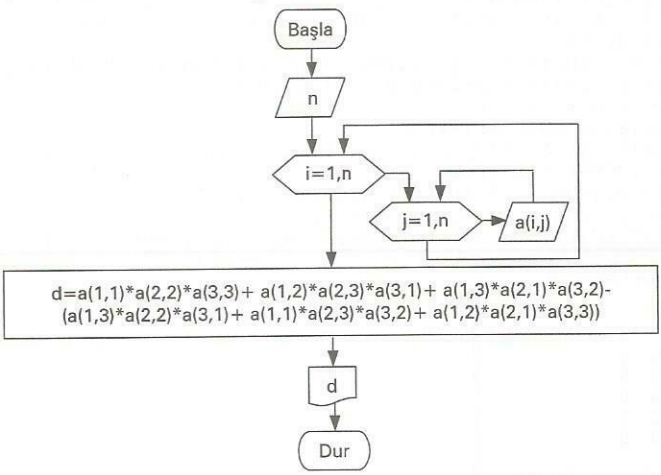
Örnek-8.32: Klavyeden girilen $3x3$ tipindeki A matrisinin determinantını "*Sarrus yöntemi*"yle hesaplayan program.

$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \rightarrow \det(A) = \begin{cases} (a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32}) - \\ (a_{13}a_{22}a_{31} + a_{11}a_{23}a_{32} + a_{12}a_{21}a_{33}) \end{cases}$$

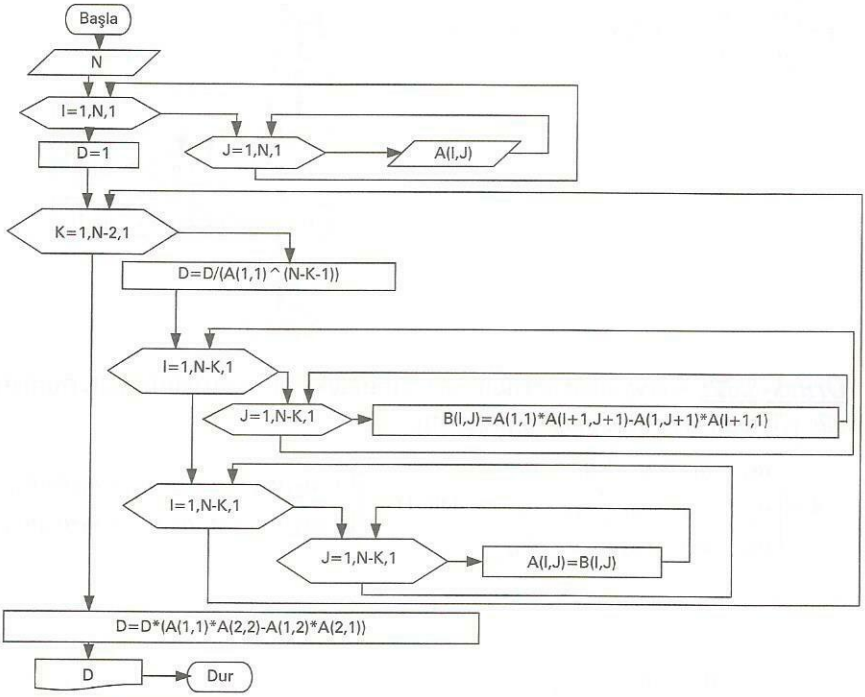




$$A=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \rightarrow \det(A) = \begin{cases} (a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32}) - \\ (a_{13}a_{22}a_{31} + a_{11}a_{23}a_{32} + a_{12}a_{21}a_{33}) \end{cases}$$



Örnek-8.33: Klavyeden girilen *NxN* tipindeki *A* matrisinin determinantını hesaplayan program.




```
/* Determinant */
#include <stdio.h>
#include <conio.h>
#include <math.h>
float D=1,A[10][10],B[10][10]; int i,j,k,N;
void main()
{ clrscr();printf("Kare matrisin satır sayısı: "); scanf("%u",&N);
  printf("\n  A matrisi\n"); printf(" ------------\n");
  for(i=0;i<N;i++)
    { for(j=0;j<N;j++) { gotoxy(5*j+5,5+i);scanf("%f",&A[i][j]); }}
  for(k=0;k<N-2;k++) { D=D/(pow(A[0][0],N-k-2));
    for(i=0;i<N-k-1;i++)
      { for(j=0;j<N-k-1;j++)
          B[i][j]=A[0][0]*A[i+1][j+1]-A[0][j+1]*A[i+1][0];}
    for(i=0;i<N-k-1;i++) {
      for(j=0;j<N-k-1;j++) A[i][j]=B[i][j]; }}
    D=D*(A[0][0]*A[1][1]-A[0][1]*A[1][0]);
    printf("\nMatrisin determinantı= %0.3f",D); getch(); }
```

```
Kare matrisin satır sayısı: 5

   A matrisi
   -------------
   2  3  4  8  6
   1  4  7  3  5
   8  1  3  9  5
   6  2  8  2  1
   0  3  4  9  6

Matrisin determinantı= 351.000
```

C++

```
// Determinant
#include <iostream>
#include <windows.h>
#include <cmath>
using namespace std;
float D=1; int i,j,k,N;
int main()
{ system("cls");cout<<"Kare matrisin satır sayısı: ";cin>>N;
  cout<<"\n  A matrisi\n"; cout<<" ------------\n";
  float A[N][N],B[N][N];
  for(i=0;i<N;i++)
   { for(j=0;j<N;j++)
      { COORD xy = {5*j+5,4+i};
        SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),xy);
        cin>>A[i][j]; }}
  for(k=0;k<N-2;k++)
   { D=D/(pow(A[0][0],N-k-2));
     for(i=0;i<N-k-1;i++)
      { for(j=0;j<N-k-1;j++)
        B[i][j]=A[0][0]*A[i+1][j+1]-A[0][j+1]*A[i+1][0];}
    for(i=0;i<N-k-1;i++)
      { for(j=0;j<N-k-1;j++)
          A[i][j]=B[i][j];}}
    D=D*(A[0][0]*A[1][1]-A[0][1]*A[1][0]);
    cout<<"\nMatrisin determinantı= "<<D<<endl;
    system("pause");return 0; }
```

C#

```
// Determinant
using System;
namespace ornek833
{ class Program
   { static void Main(string[] args)
    { int i, j, k, N; double D = 1;
      Console.Write("Kare matrisin satır sayısı: ");
      N = Convert.ToInt32(Console.ReadLine());
      double[,] A = new double[N, N]; double[,] B = new double[N, N];
      Console.WriteLine("\n      A matrisi");
      Console.WriteLine("      ------------");
      for (i=0;i<N;i++)
       { for (j=0;j<N;j++)
          { Console.SetCursorPosition(5*j+5,4+i);
            A[i,j]=Convert.ToDouble(Console.ReadLine()); }}
      for (k=0;k<N-2;k++)
       { D=D/(Math.Pow(A[0,0],N-k-2));
         for (i=0;i<N-k-1;i++)
          { for (j=0;j<N-k-1;j++)
              B[i,j]=A[0,0]*A[i+1,j+1]-A[0,j+1]*A[i+1,0]; }
         for (i=0;i<N-k-1;i++) {
           for (j=0;j<N-k-1;j++)
              A[i,j]=B[i,j]; }}
      D=D*(A[0,0]*A[1,1]-A[0,1]*A[1,0]);
      Console.WriteLine("\nMatrisin determinantı={0:F3}", D);
      Console.ReadLine();    }    }    }
```

JAVA

```
// Determinant
import java.util.Scanner;
public class ornek833 {
public static void main(String[] args)
{  int i,j,k,N; double D=1; Scanner klavye=new Scanner (System.in);
   System.out.print("Kare matrisin satır sayısı: ");
   N=klavye.nextInt();
   double A[][]=new double[N][N]; double B[][]=new double[N][N];
   System.out.println("\nA matrisi"); System.out.println("---------");
   for(i=0;i<N;i++)
    { for(j=0;j<N;j++)
       { System.out.print("A("+i+","+j+")= ");
         A[i][j]=klavye.nextFloat(); }}
   for(k=0;k<N-2;k++)
    { D=D/(Math.pow(A[0][0],N-k-2));
      for(i=0;i<N-k-1;i++)
       { for(j=0;j<N-k-1;j++)
           B[i][j]=A[0][0]*A[i+1][j+1]-A[0][j+1]*A[i+1][0];}
      for(i=0;i<N-k-1;i++) {
        for(j=0;j<N-k-1;j++)
         A[i][j]=B[i][j]; }}
   D=D*(A[0][0]*A[1][1]-A[0][1]*A[1][0]);
   System.out.printf("\nMatrisin deteminantı= %.3f\n",D); }}
```

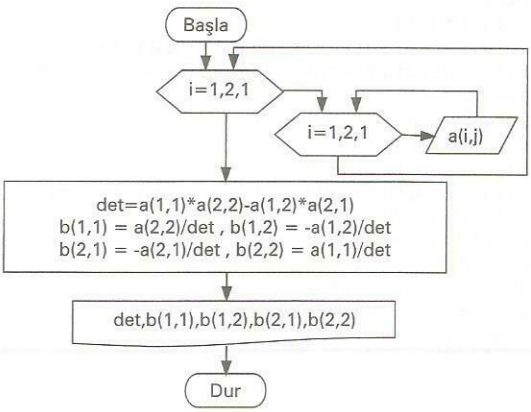
MATLAB

```
% Determinant
clear all; clc;fprintf('Matrisi [a b;c d] şeklinde giriniz!\n');
fprintf('\nMatris\n------\n'); A=input('');
fprintf('\nMatrisin determinantı= %0.3f\n',det(A));
```

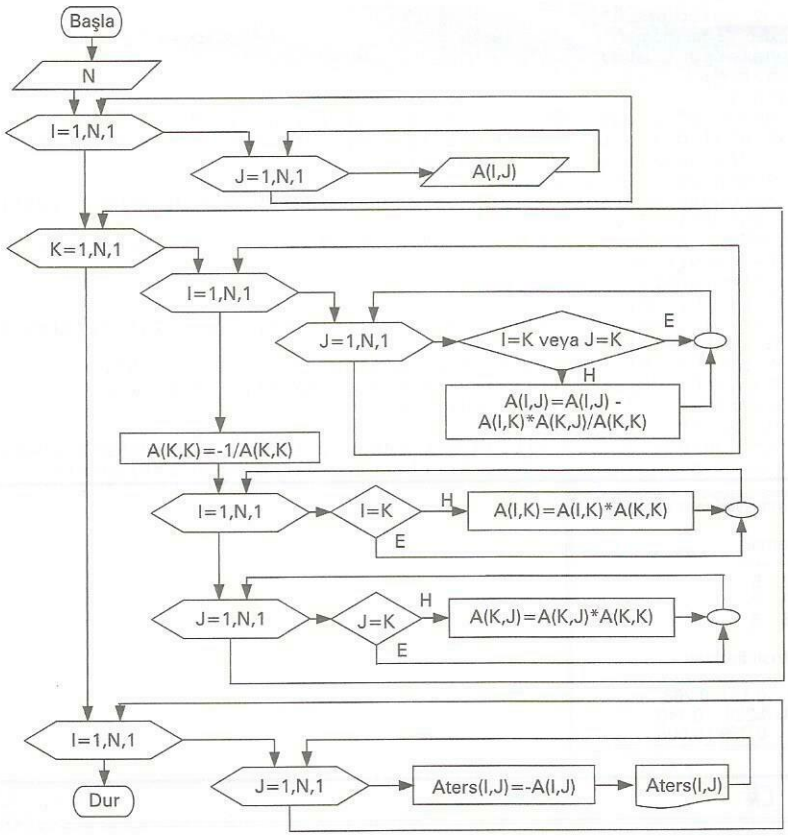
PYTHON

```
# Determinant
import numpy as np
print("Matrisi [a b;c d] şeklinde giriniz!")
print("\nMatris\n------")
x=input()
A=np.matrix(x)
print("\nMatrisin determinantı= %0.3f"%np.linalg.det(A))
```

Örnek-8.34: Klavyeden girilen ve determinantı sıfırdan farklı 2x2 tipindeki matrisin determinantını ve tersini hesaplayıp yazdıran program.



Örnek-8.35: Klavyeden girilen *NxN* tipindeki *A* kare matrisinin tersini hesaplayan program.



C

```
/* Matrisin tersi */
#include <stdio.h>
#include <conio.h>
int i,j,k,N; float A[10][10];
void main()
{ clrscr();printf("Kare matrisin satır sayısı: ");scanf("%u",&N);
  printf("\n  A matrisi\n");printf(" ------------\n");
  for(i=0;i<N;i++) { for(j=0;j<N;j++)
      { gotoxy(5*j+5,5+i);scanf("%f",&A[i][j]); }}
  printf("\n  A matrisinin tersi\n");printf(" -------------------\n");
  for(k=0;k<N;k++)
   {for(i=0;i<N;i++)
     {for(j=0;j<N;j++)
       if(!((i==k)||(j==k))) A[i][j]=A[i][j]-A[i][k]*A[k][j]/A[k][k];}
       A[k][k]=-1/A[k][k];
  for(i=0;i<N;i++) if(!(i==k)) A[i][k]=A[i][k]*A[k][k];
  for(j=0;j<N;j++) if(!(j==k)) A[k][j]=A[k][j]*A[k][k];}
  for(i=0;i<N;i++) { for(j=0;j<N;j++)
    {gotoxy(8*j+8,8+i+N);printf("%0.3f",-A[i][j]);}}   getch();}
```

C++

```
// Matrisin tersi
#include <iostream>
#include <windows.h>
#include <iomanip>
using namespace std;
int i,j,k,N;
int main()
{ system("cls");cout<<"Kare matrisin satır sayısı: "; cin>>N;
  float A[N][N];cout<<"\n  A matrisi\n";cout<<" ------------\n";
  for(i=0;i<N;i++) {for(j=0;j<N;j++)
     {COORD xy = {5*j+5,4+i};
      SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),xy);
      cin>>A[i][j]; }}
  cout<<"\n  A matrisinin tersi\n";cout<<" -------------------\n";
  for(k=0;k<N;k++)
   {for(i=0;i<N;i++)
     {for(j=0;j<N;j++)
       if(!((i==k)||(j==k))) A[i][j]=A[i][j]-A[i][k]*A[k][j]/A[k][k];}
      A[k][k]=-1/A[k][k];
      for(i=0;i<N;i++) if(!(i==k)) A[i][k]=A[i][k]*A[k][k];
      for(j=0;j<N;j++) if(!(j==k)) A[k][j]=A[k][j]*A[k][k];}
  for(i=0;i<N;i++) {for(j=0;j<N;j++)
     {COORD xy = {8*j+8,7+i+N};
      SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),xy);
      cout<<setprecision(3)<<-A[i][j];}} system("pause");return 0; }
```

```
Kare matrisin satır sayısı: 3

 A matrisi
 ------------
 1  2  5
 4  9  7
 6  5  3

 A matrisinin tersi:
 ---------------------------
 0.068  -0.161  0.263
 -0.254 0.229   -0.110
 0.288  -0.059  -0.008
```

C#

```
// Matrisin tersi
using System;
namespace ornek835
{ class Program
   { static void Main(string[] args)
      { int i,j,k,N; Console.Write("Kare matrisin satır sayısı: ");
        N=Convert.ToInt32(Console.ReadLine());
        double[,] A = new double[N, N];
        Console.WriteLine("\n      A matrisi");
        Console.WriteLine("      ------------");
        for (i=0;i<N;i++) { for (j=0;j<N;j++)
            { Console.SetCursorPosition(5*j+5,4+i);
              A[i,j]=Convert.ToDouble(Console.ReadLine()); }}
        for (k=0;k<N;k++)
         { for (i=0;i<N;i++)
            { for (j=0;j<N;j++)
               if (!((i==k) || (j==k)))
                A[i,j]=A[i,j]-A[i,k]*A[k,j]/A[k,k];  }
        A[k,k]=-1/A[k,k];
        for (i=0;i<N;i++) if (!(i==k)) A[i,k]=A[i,k]*A[k,k];
        for (j=0;j<N;j++) if (!(j==k)) A[k,j]=A[k,j]*A[k,k]; }
        Console.WriteLine("\n      A matrisinin tersi");
        Console.WriteLine("      -------------------");
        for (i=0;i<N;i++) { for (j=0;j<N;j++)
         { Console.SetCursorPosition(8*j+8,7+i+N);
           Console.Write("{0:F3}", -A[i, j]);}}Console.ReadLine();}}}
```

JAVA

```
// Matrisin tersi
import java.util.Scanner;
public class ornek835 {
public static void main(String[] args)
{  int i,j,k,N; Scanner klavye=new Scanner (System.in);
   System.out.print("Kare matrisin satır sayısı: ");
   N=klavye.nextInt();float A[][]=new float[N][N];
   System.out.println("\nA matrisi");System.out.println("---------");
   for(i=0;i<N;i++){ for(j=0;j<N;j++)
     { System.out.print("A("+i+","+j+")= ");
       A[i][j]=klavye.nextFloat(); }}
   for(k=0;k<N;k++)
    { for(i=0;i<N;i++)
       { for(j=0;j<N;j++)
          if(!((i==k) || (j==k)))
            A[i][j]=A[i][j]-A[i][k]*A[k][j]/A[k][k]; }
       A[k][k]=-1/A[k][k];
    for(i=0;i<N;i++) if(!(i==k)) A[i][k]=A[i][k]*A[k][k];
    for(j=0;j<N;j++) if(!(j==k)) A[k][j]=A[k][j]*A[k][k];}
   System.out.println("\nA matrisinin tersi");
   System.out.println("------------------");
   for(i=0;i<N;i++)
    {for(j=0;j<N;j++) System.out.printf("%.3f\t",-A[i][j]);
     System.out.println();} }}
```

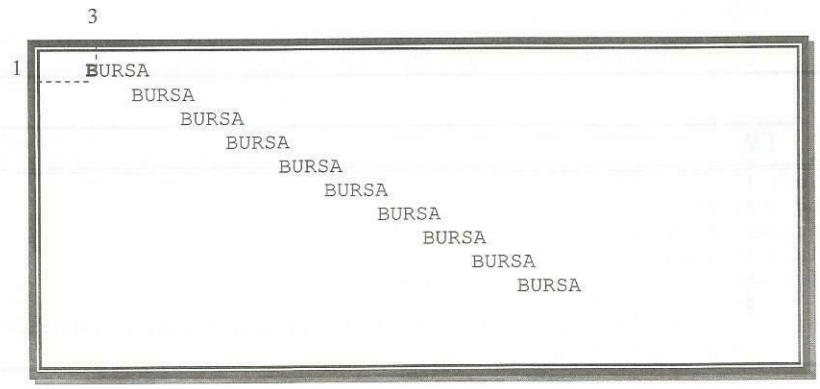
MATLAB

```
% Matrisin tersi
clear all; clc;fprintf('Matrisi [a b;c d] şeklinde giriniz!\n');
fprintf('\nA matrisi\n---------\n'); A=input('');
fprintf('\nA matrisinin tersi\n------------------\n');disp(inv(A));
```

PYTHON

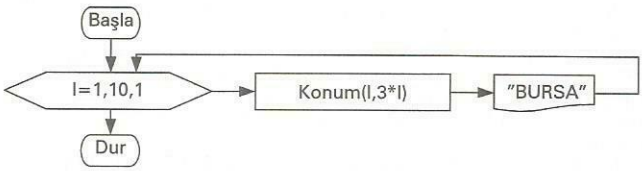
```
# Matrisin tersi
import numpy as np
print("Matrisi [a b;c d] şeklinde giriniz!")
print("\nA matrisi\n---------"); x=input(); A=np.matrix(x)
print("\nA matrisinin tersi\n------------------")
print(np.linalg.inv(A))
```

Örnek-8.36: "BURSA" ismini aşağıdaki gibi yazdıran program (ilk "BURSA" isminin konumu 1. satır-3. sütundur).



Bilgisayarın standart metin ekranı, 25 satır ve 80 sütundan oluşmaktadır. Yani 25x80 tipinde bir matris olarak karakter işlemlerinde rahatlıkla kullanılabilmektedir. Yukarıdaki yazının karakterlerinin yerlerine bakılırsa; satır sayısı birer birer artarken, sütun sayısı da üçer üçer artmaktadır. Tek döngü (*I*) kullanılarak imleç yönlendirme komutunun (*Konum*) satır parametresi *I* ve sütun parametresi de 3**I* yapılırsa, istenilen çıktı elde edilmektedir.

Yazı no	Satır	Sütun
1	1	3
2	2	6
3	3	9
...	...	...



C

```
/* Yazı 1*/
#include <stdio.h>
#include <conio.h>
unsigned char i;
main()
{  clrscr(); for(i=1;i<=10;i++){ gotoxy(3*i,i);printf("BURSA"); }
   getch();return 0; }
```

C++

```
// Yazi 1
#include <iostream>
#include <windows.h>
using namespace std;
main()
{  system("cls");
   for(int i=1;i<=10;i++)
     { COORD xy = {3*i,i};
       SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),xy);
       cout << "BURSA"; }  system("pause");return 0; }
```

C#

```
// Yazı 1
using System;
namespace ornek836
{ class Program
   { static void Main(string[] args)
      { for (int i=1;i<=10;i++)
          { Console.SetCursorPosition(3*i,i); Console.Write("BURSA"); }
        Console.ReadLine();     }     }     }
```

JAVA

```
// Yazı-1
import java.util.Scanner;
public class ornek836 {
public static void main(String[] args)
{ for(int i=1;i<=10;i++)
    { for(int j=1;j<=2*i;j++) System.out.print(" ");
      System.out.println("BURSA");}}}
```

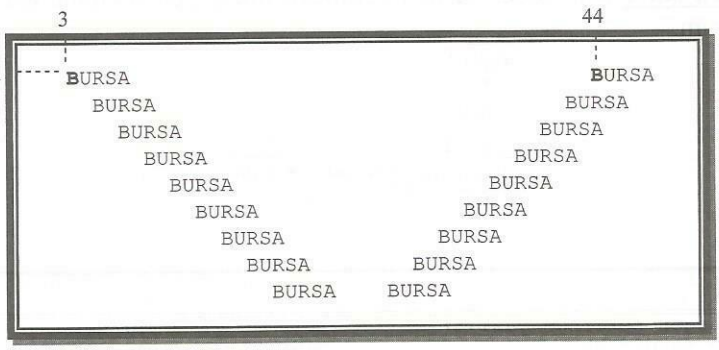
MATLAB

```
% Yazı-1
clear all;clc;for i=1:10
              for j=1:2*i fprintf(' '); end; fprintf('BURSA\n'); end;
```

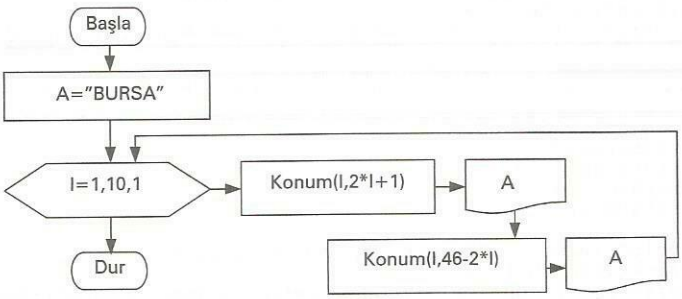
PYTHON

```
# Yazı-1
for i in range(1,11):
    for j in range(1,2*i+1):
        print(" ",end='')
    print("BURSA")
```

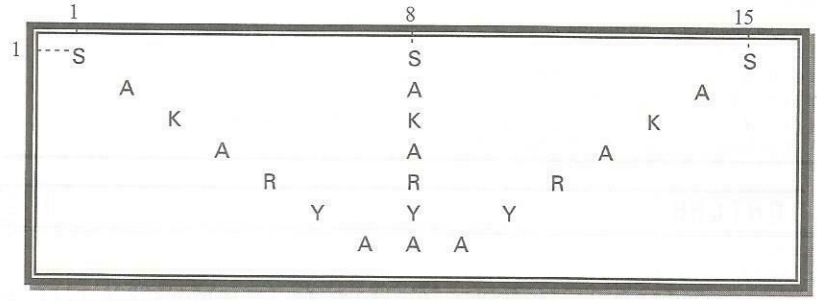
Örnek-8.37: Aşağıdaki çıktıyı veren program (1. satırdaki "BURSA" isminin sütun olarak konumları 3 ve 44'dir).

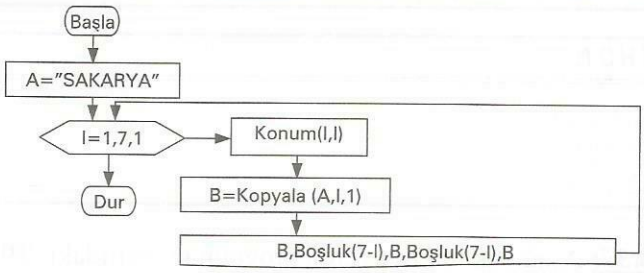


İsmin ekrandaki yerleşim konumlarına (satır ve sütun olarak) dikkat edilirse; imleç yönlendirme komutunun parametreleri ilk kayan yazı için $I,2*I+1$ ve ikincisi için de $I,46-2*I$ şeklindedir.

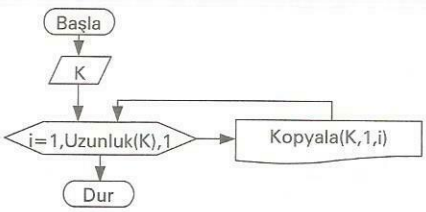


Örnek-8.38: Aşağıdaki çıktıyı veren program.





Örnek-8.39: Klavyeden girilen kelimeyi, dik üçgen şeklinde ekrana yazdıran program.



C

```
/* Dik üçgen şeklinde yazı */
#include <stdio.h>
#include <conio.h>
#include <string.h>
char k[20];
int i,j;
void main()
{ clrscr();printf("Kelime: ");scanf("%s",&k);printf("\n");
  for(i=1;i<=strlen(k);i++)
   { for(j=0;j<i;j++) printf("%c",k[j]);printf("\n"); }  getch(); }
```

C++

```
// Dik üçgen şeklinde yazı
#include <iostream>
#include <cstring>
using namespace std;
int i,j; string k;
main()
{ system("cls");cout << "Kelime: "; cin >> k;cout << endl;
  for(i=1;i<=k.length();i++)
  {for(j=0;j<i;j++) cout<<k[j];cout<<endl;} system("pause");return 0;}
```

C#

```
// Dik üçgen şeklinde yazı
using System;
namespace ornek839
{class Program
 {static void Main(string[] args)
  { String k; Console.Write("Kelime: "); k = Console.ReadLine();
    Console.WriteLine();
    for (int i=0;i<k.Length;i++) Console.WriteLine(k.Substring(0,i+1));
    Console.ReadLine();     }     }     }
```

JAVA

```
// Dik üçgen şeklinde yazı
import java.util.Scanner;
public class ornek839 {
public static void main(String[] args)
{ String k; Scanner klavye=new Scanner(System.in);
  System.out.print("Kelime: ");k=klavye.nextLine();
  System.out.println(); for(int i=0;i<k.length();i++)
                        System.out.println(k.substring(0,i+1)); }}
```

MATLAB

```
% Dik üçgen şeklinde yazı
clear all; clc; k=input('Kelime: ','s');disp(' ');
for i=1:length(k)
    for j=1:i fprintf('%c',k(j)); end;fprintf('\n'); end;
```

PYTHON

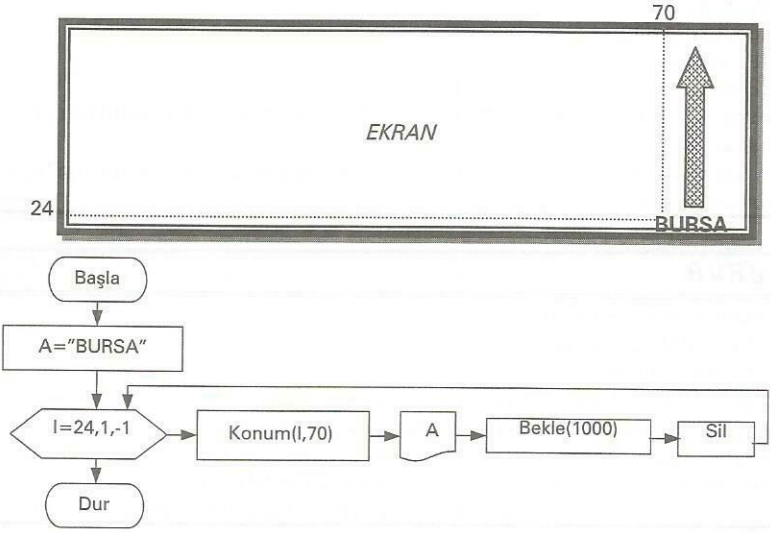
```
# Dik üçgen şeklinde yazı
k=input("Kelime: "); print("")
for i in range(len(k)):
    for j in range(i+1):
        print("%c"%k[j],end='')
    print("")
```

```
Kelime: ANKARA

A
AN
ANK
ANKA
ANKAR
ANKARA
```

Örnek-8.40: Belirtilen yazıyı ekranın alt kısmından üst kısmına doğru 1 saniye aralıklarla hareket ettiren program.

Yazının yukarıya doğru (dikey) hareketi, satır sayısının birer birer azaltılmasıyla sağlanır. İlk adımda yazı, örneğin 24. satır ve 70. sütuna yazılır. Daha

sonra yazı; bir üst satıra yazılır, belirli bir süre bekletilir ve ekran temizlenir. Bu şekilde dikey hareket sağlanmış olur.



C

```
/* Dikey hareket */
#include <stdio.h>
#include <conio.h>
#include <dos.h>
main()
{ static char A[]= "BURSA"; unsigned char i; clrscr();
  for(i=24;i>=1;i--){ gotoxy(70,i);printf(A);delay(1000);clrscr();}
  getch();return 0; }
```

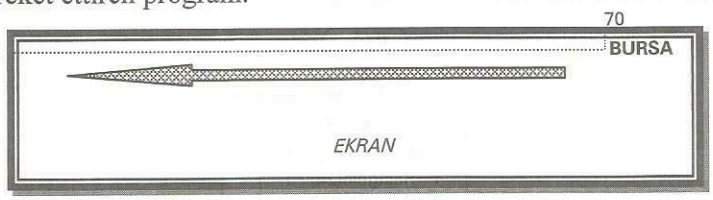
C++

```
// Dikey hareket
#include <iostream>
#include <windows.h>
using namespace std;
main()
{  static char A[]= "BURSA"; char i;system("cls");
   for(i=24;i>=0;i--) {COORD xy = {70,i};
      SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE),xy);
      cout << A; Sleep(1000);system("cls");} system("pause");return 0;}
```

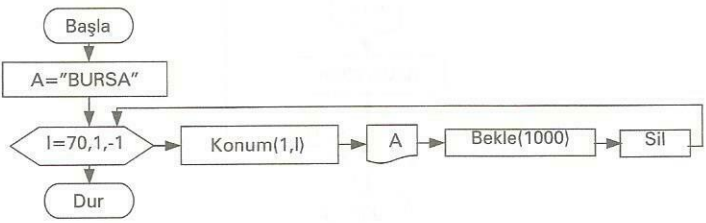
C#

```
// Dikey hareket
using System;
namespace ornek840
{ class Program
   { static void Main(string[] args) { String A = "BURSA";
      for (int i=24;i>=0;i--)
        { Console.SetCursorPosition(70,i); Console.Write(A);
          System.Threading.Thread.Sleep(1000); Console.Clear(); }
      Console.ReadLine();     }     }     }
```

Örnek-8.41: Belirtilen yazıyı ekranın sağından soluna doğru 1 saniye aralıklarla hareket ettiren program.

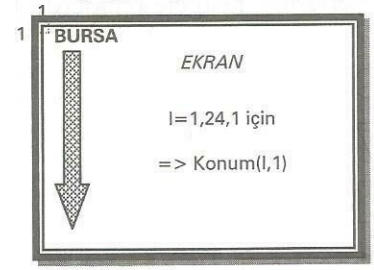


Yazının sağdan sola (yatay) hareketi, sütun sayısının birer birer azalmasından ibarettir. İlk adımda yazı, örneğin 1. satır ve 70. sütuna yazılır. Daha sonra sütun sayısı azaltılarak, imleç yönlendirme komutuna parametre olarak verilir ve belirtilen konuma yazı yazdırılır. Belirli bir süre bekledikten sonra ekran silinir ve 1. sütuna gelinceye kadar döngü bu şekilde devam ettirilir.

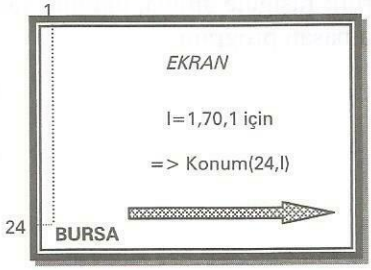


Benzer şekilde, ekrandaki diğer hareketler aşağıdaki gibi sağlanır:

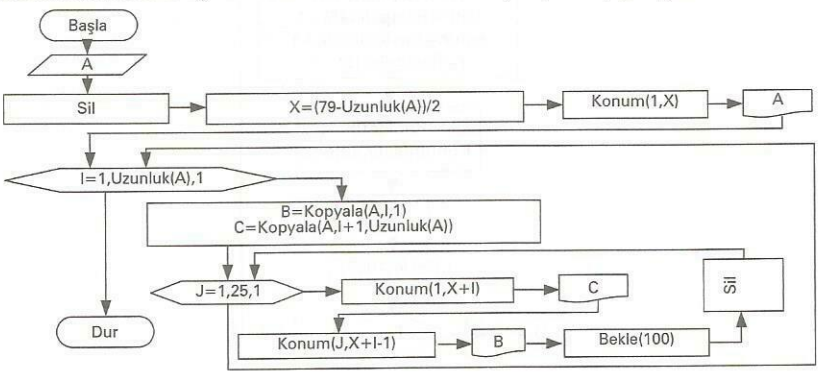
Yazının aşağıya doğru hareket etmesi



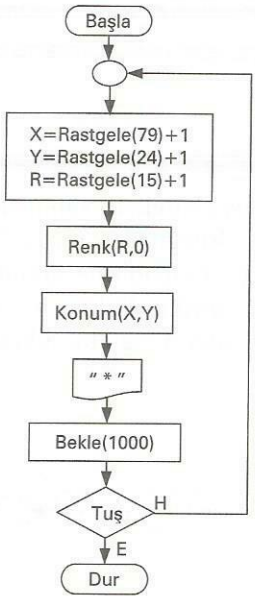
Yazının sağa doğru hareketi



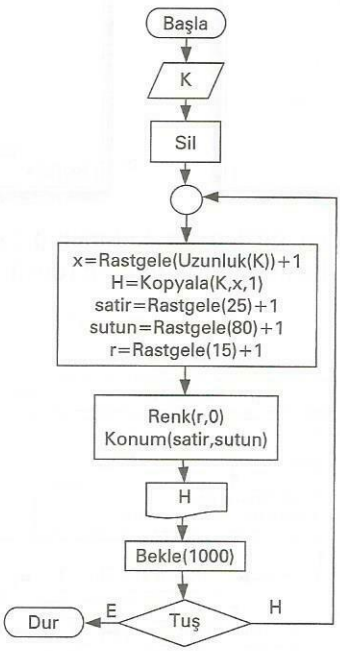
Örnek-8.42: Klavyeden girilen bir kelimenin harflerini, ekranın üst kısmından alt kısmına doğru 1 s aralıklarla indiren (düşüren) program.



Örnek-8.43: Klavyeden herhangi bir tuşa basılıncaya kadar ekranın rastgele yerlerine, 1 s aralıklarla rastgele renklerde "*" karakteri basan program.

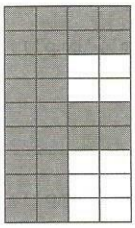


Örnek-8.44: Herhangi bir tuşa basıncaya kadar klavyeden girilen kelimenin harflerini; rastgele sırada, ekranın rastgele yerlerine, rastgele renklerde 1 s aralıklarla basan program.



Örnek-8.45: Siyah-beyaz ikili (binary) bir resmin/görüntünün negatifini (tümleyenini, tamamlayanını) alan program.

Tablo 8.14: Negatif alma işlemi

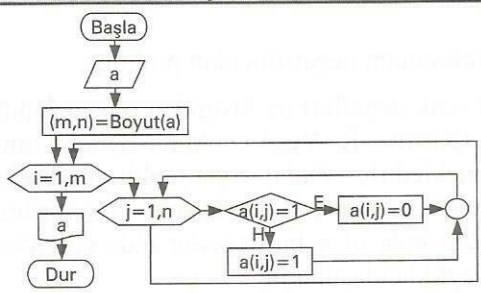
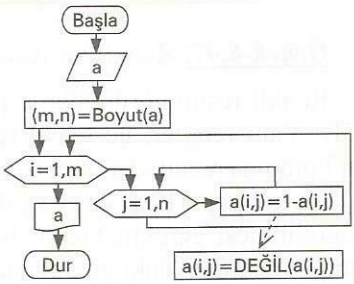
Orijinal matris	Görüntü benzetimi	Görüntü
$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$		

İşlem	Anlamı	Matris	Görüntü
Bitlerin tersi	Negatif alma	$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$	

Matris yapısında olan iki boyutlu görüntüler, *piksel* (pixel) olarak adlandırılan noktacıklardan oluşmaktadırlar. Yani görüntü matrisinin

- ✓ Elemanları pikseller,
- ✓ Satır sayısı, görüntünün yüksekliği,
- ✓ Sütun sayısı, görüntünün genişliğidir.

Siyah-beyaz ikili (binary) görüntülerde, piksel değerleri 0 ve 1'dir. 0, siyah renge; 1 ise beyaz renge karşılık gelmektedir. Siyah-beyaz görüntünün negatifini almak; beyaz pikselleri siyah, siyah pikselleri de beyaz yapmaktan ibarettir. Yani görüntü matrisinin elemanları arasında 0↔1 değişimin yapılması gerekmektedir (Tablo 8.14).

Çözüm 1	Çözüm 2
	

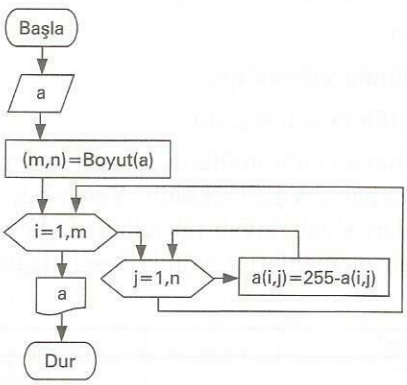
MATLAB

```
% Siyah-beyaz resmin negatifi
clear all; clc; a=imread('resim','bmp'); [y,g]=size(a);
for i=1:y
    for j=1:g
        a(i,j)=1-a(i,j);
    end; end; imshow(a)
-------------------------- veya ----------------------
% Siyah-beyaz resmin negatifi
clear all; clc; a=imread('resim','bmp'); [y,g]=size(a);
for i=1:y
    for j=1:g
        a(i,j)=~a(i,j);
    end; end; imshow(a)
-------------------------- veya ----------------------
% Siyah-beyaz resmin negatifi
clear all; clc; a=imread('resim','bmp');
a(:,:)=1-a(:,:); imshow(a)
-------------------------- veya ----------------------
% Siyah-beyaz resmin negatifi
clear all; clc; a=imread('resim','bmp');
a1=imcomplement(a); imshow(a1)
```

Orijinal resim	Program sonucu oluşan resim

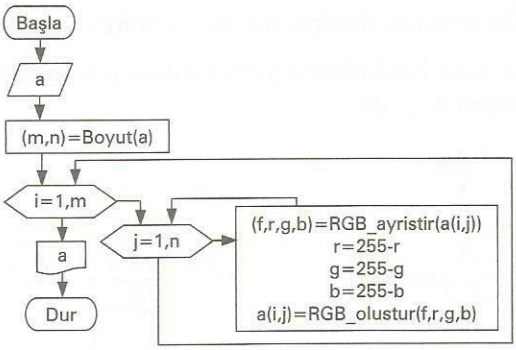
Örnek-8.46: Gri bir resmin/görüntünün negatifini alan program.

8-bitlik gri resim/görüntülerde, siyah-beyazlardan farklı olarak piksel değerleri 0-255 arasındadır. 0 değeri siyah renge karşılık gelirken, 255 de beyaz rengi belirtmektedir.



Örnek-8.47: Renkli bir resmin/görüntünün negatifini alan program.

Renkli resim/görüntülerde piksel renk değerleri üç kısımdan oluşmaktadır: RGB. Yani renkler, üç temel rengin (Kırmızı-R, Yeşil-G, Mavi-B) karışımından/kombinasyonundan meydana gelmektedirler. Dolayısıyla renkli görüntülerde işlem yapılırken önce pikselin RGB renk değerlerinin ayrıştırılması/öğrenilmesi gerekmektedir. 0-255 arasında olan bu değerler elde edildikten sonra Örnek-8.46'daki gibi sonuca kolaylıkla ulaşılabilir.

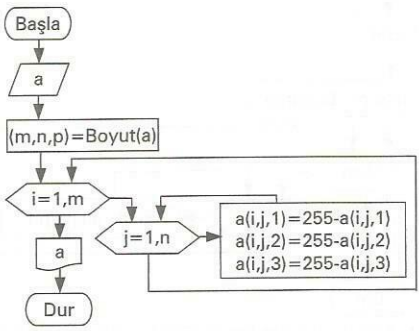


MATLAB

```
% Renkli resmin negatifi
clear all; clc; a=imread('resim','jpg'); [y,g,u]=size(a);
for i=1:y
 for j=1:g
   a(i,j,1)=255-a(i,j,1);a(i,j,2)=255-a(i,j,2);a(i,j,3)=255-a(i,j,3);
 end; end; imshow(a)
--------------------------- veya ----------------------
% Renkli resmin negatifi
clear all; clc; a=imread('resim','jpg'); [y,g,u]=size(a);
for i=1:y
    for j=1:g
        a(i,j,:)=255-a(i,j,:);
    end
end; imshow(a)
--------------------------- veya ----------------------
% Renkli resmin negatifi
clear all; clc; a=imread('resim','jpg');
a(:,:,:)=255-a(:,:,:); imshow(a)
--------------------------- veya ----------------------
% Renkli resmin negatifi
clear all; clc; a=imread('resim','jpg');
a1=imcomplement(a); imshow(a1)
```

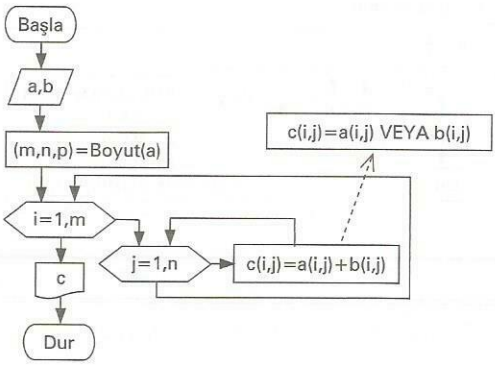
Orijinal resim	Program sonucu oluşan resim

Program kodlarına dikkat edilirse, aslında resim üç boyutlu bir yapıda tutulmaktadır. Bu yapının ilk iki indisi satır/sütun (yükseklik/genişlik) gösterirken üçüncüsü de renk değerlerini (1 - kırmızı, 2 - yeşil, 3 - mavi) göstermektedir. Buna göre çizilmiş akış diyagramı, aşağıda verilmektedir.



Örnek-8.48: İki resmi/görüntüyü üst üste bindiren/yerleştiren program.

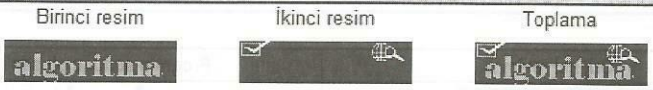
İki resmin üst üste bindirilmesi/yerleştirilmesi/birleştirilmesi, iki görüntü matrisinin toplamından ibarettir.



MATLAB

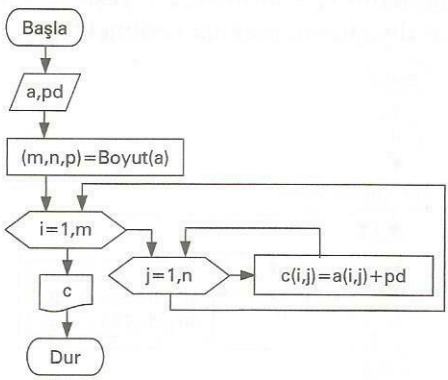
```
% İki resmin üst üste bindirilmesi/yerleştirilmesi/birleştirilmesi
% Matris toplama
clear all; clc; a=imread('r1.bmp'); b=imread('r2.bmp');
c=uint8(double(a)+double(b));
% veya c=imadd(a,b);
% siyah-beyaz resim için c=a | b;
subplot(131);imshow(a);title('Birinci resim')
subplot(132);imshow(b);title('İkinci resim')
subplot(133);imshow(c);title('Toplama')
```

Program çıkıtısı



Örnek-8.49: Bir resmin/görüntünün parlaklığını değiştiren program.

Bir resmin parlaklığının değiştirilmesi (arttırılması/azaltılması), görüntü matrisinin sabit bir pozitif/negatif değerle toplanmasıyla gerçekleştirilmektedir.



MATLAB

```
% Resmin/görüntünün parlaklığının değiştirilmesi
% Matrise sabit sayı ekleme
clear all; clc; a=imread('r5.bmp'); pd=input('Parlaklık değişimi: ');
c=uint8(double(a)+pd);
subplot(121);imshow(a);title('Birinci resim')
subplot(122);imshow(c);title('Parlaklığı değiştirilmiş resim')
```

Program çıkıtısı

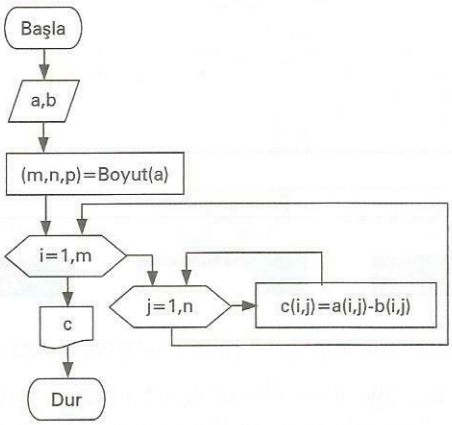
Parlaklık değişimi: 50

Birinci resim

Parlaklığı değiştirilmiş resim

Örnek-8.50: İki resim arasındaki farkları bulan program.

İki resim arasındaki farkların ortaya çıkarılması/algılanması, iki görüntü matrisinin farkından ibarettir.



MATLAB

```
% İki resim arasındaki farkları ortaya çıkarma
% Matris çıkarma
clear all; clc; a=imread('r3.bmp'); b=imread('r1.bmp');
c=uint8(double(a)-double(b)); % veya c=imsubtract(a,b);
subplot(131);imshow(a);title('Birinci resim')
subplot(132);imshow(b);title('İkinci resim')
subplot(133);imshow(c);title('Çıkarma')
```

Program çıkıtısı

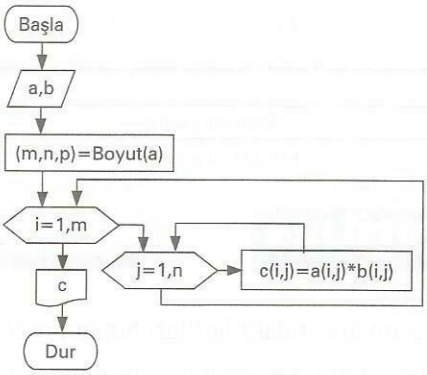
Birinci resim

İkinci resim

Çıkarma

Örnek-8.51: Resim üzerinde maskeleme yapan program.

Bir resim kullanarak, diğeri üzerinde maskeleme yapmak; iki görüntü matrisinin karşılıklı elemanlarının çarpımıyla (elementer çarpma işlemiyle) sağlanmaktadır.



MATLAB

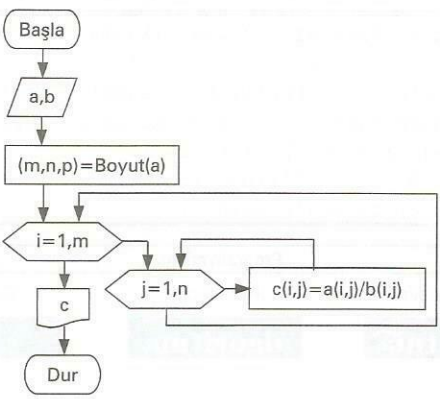
```
% Maskeleme işlemi
% Matrisin karşılıklı elemanlarını çarpma
clear all; clc; a=imread('r1.bmp'); b=imread('r4.bmp');
c=uint8(double(a).*double(b)); % c=immultiply(a,b);
subplot(131);imshow(a);title('Birinci resim')
subplot(132);imshow(b);title('İkinci resim')
subplot(133);imshow(c);title('Çarpma')
```

Program çıkıtısı

Birinci resim İkinci resim Çarpma

Örnek-8.52: Resim üzerinde arka plan bastırması yapan program.

Bir resim üzerinde, diğerini kullanarak arka planın bastırılması; iki görüntü matrisinin karşılıklı elemanlarının bölümüyle (elementer çarpma işlemiyle) yapılabilmektedir.



MATLAB

```
% Arka plan maskeleme işlemi
% Matrisin karşılıklı elemanlarını bölme
clear all; clc; a=imread('r1.bmp'); b=imread('r4.bmp');
c=uint8(double(a)./double(b)); % c=imdivide(a,b);
subplot(131);imshow(a);title('Birinci resim')
subplot(132);imshow(b);title('İkinci resim')
subplot(133);imshow(c);title('Bölme')
```

Program çıktısı

Birinci resim İkinci resim Bölme

Özetle matrislerdeki matematiksel ve mantıksal işlemlerin görüntü işleme açısından genel anlamları, sırasıyla Tablo 8.15-8.16'da verilmektedir.

Tablo 8.15: Matrislerdeki matematiksel işlemlerin genel görüntü işleme karşılıkları

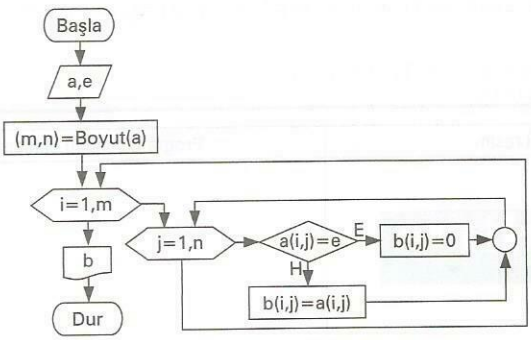
Matris işlemi	Görüntü işlemedeki uygulaması/karşılığı
Toplama	İki resmin üst üste bindirilmesi/yerleştirilmesi/birleştirilmesi
Çıkarma	İki resim arasındaki farkların ortaya çıkarılması
Karşılıklı elemanları çarpma	Maskeleme
Karşılıklı elemanları bölme	Maskeleme/bastırma (arka plan bastırma)
Sabit sayı ile işlem (+,-,*,/)	Resmin parlaklığının/kontrastının değiştirilmesi

Tablo 8.16: Matrislerdeki mantıksal işlemlerin genel görüntü işleme karşılıkları

Matris işlemi	Görüntü işlemedeki uygulaması/karşılığı
DEĞİL	Negatifini alma
VE	İki resmin kesişimi
VEYA	İki resmin birleşimi

Örnek-8.53: Görüntüde "*eşikleme*" işlemi gerçekleştiren program.

Görüntünün/resmin bazı bölgelerini ayırmak için renk değerlerine veya yoğunluklara göre eşikleme işlemi uygulanabilmektedir. Örneğin belirli eşik değerinin altındaki piksel değerleri sıfıra çekilerek, bu işlem gerçekleştirilebilmektedir.



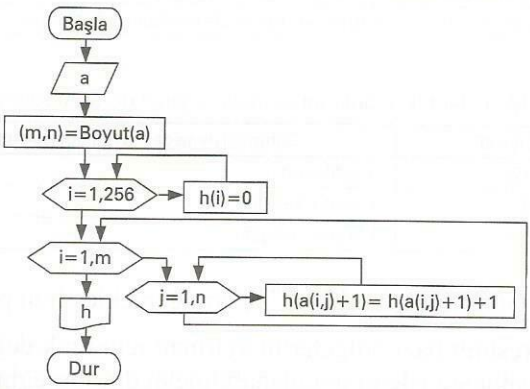
MATLAB

```
% Eşikleme
clear all; clc;
a=imread('resim.bmp');[y,g]=size(a);e=input('Eşik değeri: ');
for i=1:y
    for j=1:g
        if (a(i,j)<e)
            b(i,j)=0;
        else
            b(i,j)=r1(i,j);
    end; end; end;imshow(b)
```

Orijinal resim	Program sonucu oluşan resim
	Eşik değeri: 150

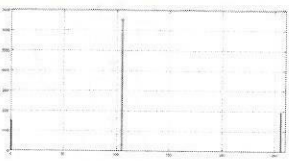
Örnek-8.54: Resmin "*histogram*"ını en genel şekliyle hesaplayan/çizen program.

Histrogram, bir görüntüdeki/resimdeki piksel yoğunluk değerlerinin sıklığını/frekansını gösteren grafiktir. Gri seviyeli bir resmin histogramını elde etmek için 256 elemanlı bir dizi açıp, taranan resimdeki piksel değerlerine göre ilgili dizi elemanında arttırma işlemi gerçekleştirilerek sonuca ulaşılabilmektedir.



MATLAB

```
% Histrogram
clear all; clc; a=imread('resim.bmp'); [y,g]=size(a); h=zeros(256);
for i=1:y
    for j=1:g
        h(a(i,j)+1)=h(a(i,j)+1)+1;
    end; end; bar(h)
```

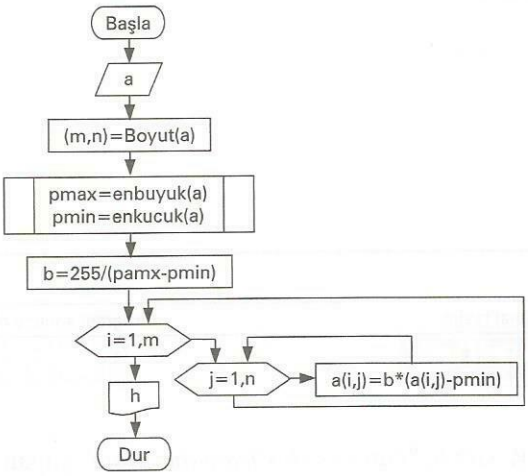
Orijinal resim	Program sonucu oluşan histogram
	

Örnek-8.55: Resme "*otomatik kontrast*" uygulayan program.

Otomatik kontrast yönteminde sonuç resmindeki piksellerin değeri

$$p' = \frac{L-1}{p_{max} - p_{min}}(p - p_{min})$$

ile hesaplanmaktadır. Bu eşitlikte p_{max} en parlak pikselin değeri, p_{min} en koyu pikselin değeri, L orijinal gri resimdeki en yüksek değeri göstermektedir.



Örnek-8.56: Resimde "*kontrast iyileştirme*" yapan program.

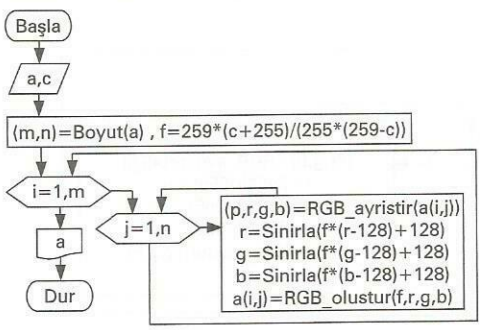
İstenilen C kontrast seviyesine göre öncelikle kontrast düzeltme/iyileştirme faktörü

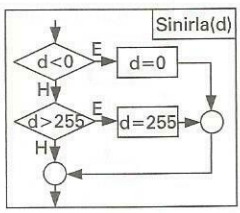
$$F = \frac{259(C+255)}{255(259-C)}$$

ile hesaplanmaktadır. Daha sonra da kontrast ayarı

$$R_{yeni} = F(R_{eski} - 128) + 128$$

ile uygulanmaktadır.





MATLAB

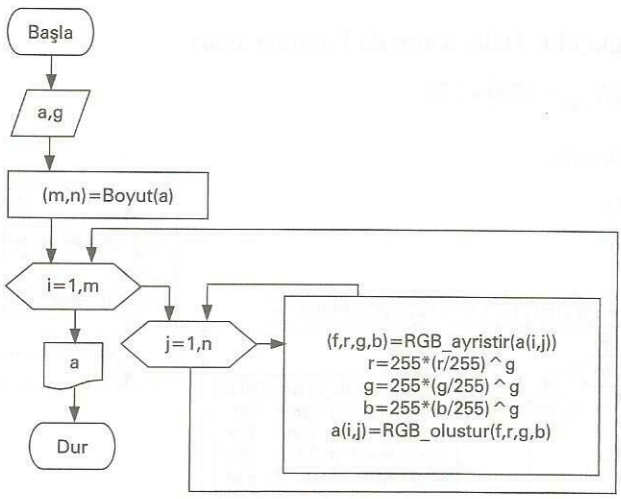
```
% Kontrast iyileştirme
clear all; clc; a=imread('resim','bmp'); [y,g,u]=size(a);
c=input('Kontrast seviyesi: '); f=259*(c+255)/(255*(259-c));
for i=1:y
 for j=1:g
   a(i,j,1)=sinirla(f*(a(i,j,1)-128)+128);
   a(i,j,2)=sinirla(f*(a(i,j,2)-128)+128);
   a(i,j,3)=sinirla(f*(a(i,j,3)-128)+128);
 end; end; imshow(a)
---
function y=sinirla(d)
if d<0
    y=0;
elseif d>255
    y=255;
else
    y=d;
end
```

Orijinal resim	Program sonucu oluşan resim

Örnek-8.57: Resimde "*Gama iyileştirme/düzeltme*" yapan program. İstenilen γ seviyesine göre düzeltme

$$R_{yeni} = 255\left(\frac{R_{eski}}{255}\right)^{\gamma}$$

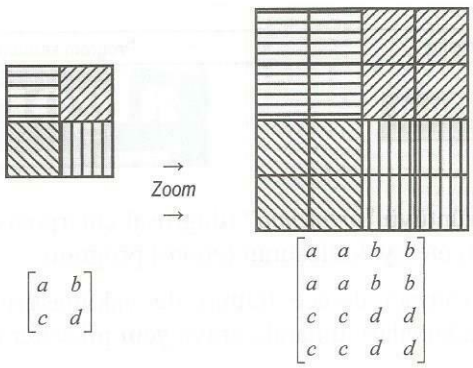
ile yapılmaktadır.



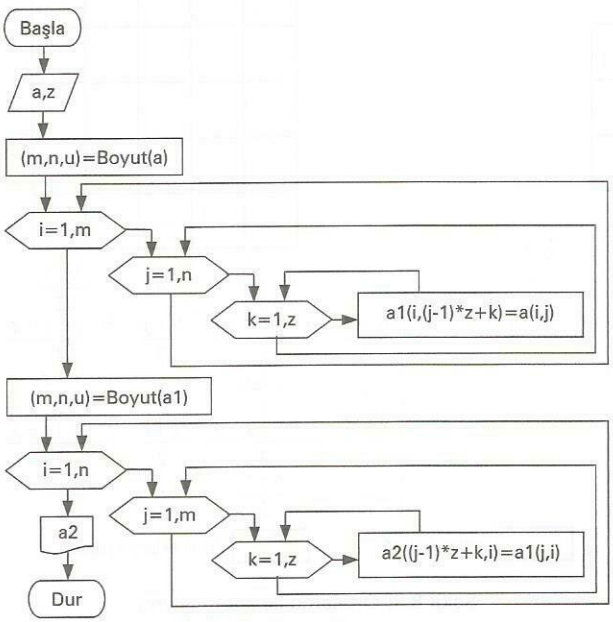
Örnek-8.58: Görüntüyü "*piksel kopyalama*" (piksel çoğaltma, sıfırıncı derece tutma) yöntemiyle büyüten/yakınlaştıran (zoom) program.

Yakınlaştırma (zoom), ayrıntıları daha net ve görünür hale getirmek için resmin genişletilmesi olarak tanımlanabilir. Resmin genişletilmesi de ancak piksellerinin sayısını arttırmakla mümkün olmaktadır. Bu da görüntü matrisinin boyutunun, dolayısıyla elemanlarının sayısının artmasıdır. Yakınlaştırma (zoom) işlemi için birçok farklı yöntemler vardır: piksel kopyalama, enterpolasyonlarla çoğaltma, sıfırıncı derece tutma, K kez yakınlaştırma vb.

Piksel kopyalama (sıfırıncı derece tutma) yönteminde bir önceki piksellerin birer kopyası eklenmektedir. Örneğin 3x3 boyutunda bir görüntünün iki katı yakınlaştırılmasında (zoom), piksellerin durumu Şekil 8.2'de verilmektedir.



Şekil 8.2: Piksel kopyalama ile zoom işlemi



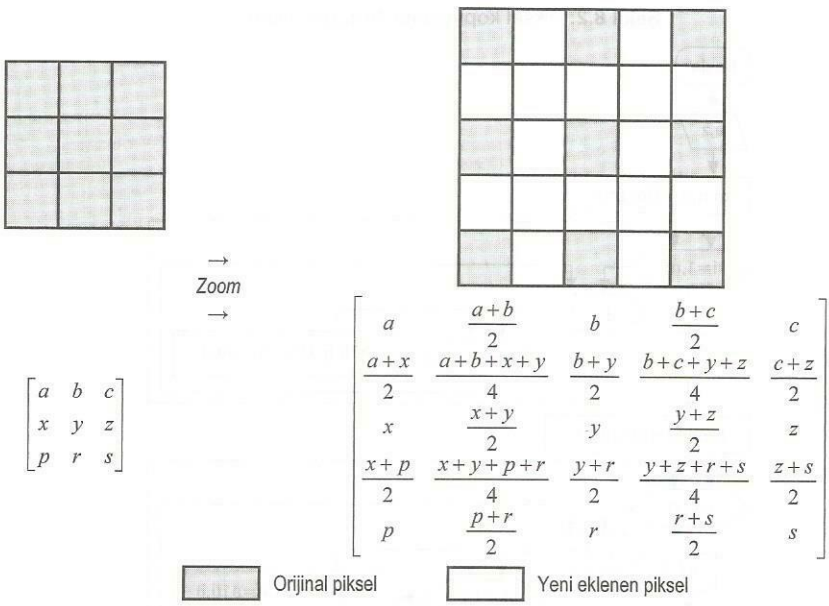
MATLAB

```
% Piksel kopyalama ile zoom
clear all; clc;
a=imread('resim','bmp'); [y,g,u]=size(a); z=input('Yakınlaştırma: ');
for i=1:y
    for j=1:g
        for k=1:z
            a1(i,(j-1)*z+k,:)=a(i,j,:);
        end; end; end; [y,g,u]=size(a1);
for i=1:g
    for j=1:y
        for k=1:z
            a2((j-1)*z+k,i,:)=a1(j,i,:);
        end; end; end; imshow(a2)
```

Orijinal resim	Program sonucu oluşan resim

Örnek-8.59: Görüntüyü "*ortalama*" (doğrusal enterpolasyon, birinci derece tutma) yöntemiyle büyüten/yakınlaştıran (zoom) program.

Ortalama alma (birinci derece tutma) ile yakınlaştırma/zoom işleminde komşu piksellerin ortalamaları alınarak, araya yeni pikseller olarak eklenmektedir (Şekil 8.3).



$$\begin{bmatrix} a & b & c \\ x & y & z \\ p & r & s \end{bmatrix}$$

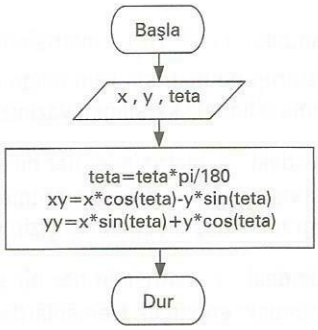
$$\begin{bmatrix} a & \frac{a+b}{2} & b & \frac{b+c}{2} & c \\ \frac{a+x}{2} & \frac{a+b+x+y}{4} & \frac{b+y}{2} & \frac{b+c+y+z}{4} & \frac{c+z}{2} \\ x & \frac{x+y}{2} & y & \frac{y+z}{2} & z \\ \frac{x+p}{2} & \frac{x+y+p+r}{4} & \frac{y+r}{2} & \frac{y+z+r+s}{4} & \frac{z+s}{2} \\ p & \frac{p+r}{2} & r & \frac{r+s}{2} & s \end{bmatrix}$$

Şekil 8.3: Ortalama ile zoom işlemi

Örnek-8.60: Kartezyen koordinat sisteminde herhangi bir (x, y) noktasının (veya görüntüdeki pikselin) saat yönünün tersinde θ açısıyla döndürülmesi

$$\begin{bmatrix} x_{yeni} \\ y_{yeni} \end{bmatrix} = \underbrace{\begin{bmatrix} Cos(\theta) & -Sin(\theta) \\ Sin(\theta) & Cos(\theta) \end{bmatrix}}_{Döndürme\ matrisi} \begin{bmatrix} x \\ y \end{bmatrix}$$

matris çarpımıyla yapılmaktadır. Buna göre klavyeden girilen bir noktayı (pikseli), belirtilen açı kadar döndürerek yeni koordinatlarını ekrana yazdıran program.



SORULAR

1. Klavyeden girilen nxn tipindeki A matrisinin her bir sütun elemanlarını kendi aralarında büyükten küçüğe doğru sıralayan programın akış diyagramını çizip herhangi bir dille kodlayınız.

2. Klavyeden girilen nxn tipindeki A matrisini, transpozesiyle çarpıp B matrisini ($B = A.A^T$) oluşturan ve bu B matrisinin en büyük ve en küçük elemanını bulan programın akış diyagramını çizip herhangi bir programlama diliyle kodlayınız.

3. Klavyeden girilen nxn tipindeki A, B ve C matrislerini kullanarak $D = A.(B^T + C^T)$ işlemiyle D matrisini oluşturup bu matrisin izini bulan programın akış diyagramını çizip herhangi bir programlama dilindeki karşılığını yazınız.

4. Klavyeden girilen nxn tipindeki A matrisinde; her bir sütundaki en büyük elemanları bulup bunları B dizisine aktaran ve B dizisinin aritmetik, geometrik ve harmonik ortalamasını hesaplayan programın akış diyagramını çizip yazılımı yapınız.

5. Klavyeden girilen nxn tipindeki A matrisinin her bir satırındaki en büyük elemanlarından B ve her bir sütunundaki en küçük elemanlardan C dizisini oluşturup bu dizilerin skaler çarpımını yapan programın akış diyagramını çizip herhangi bir programlama diliyle kodlayınız.

6. Klavyeden girilen nxn tipindeki A matrisinin tüm elemanlarının karelerini alarak B matrisini ve tüm elemanlarının karekköklerini alarak C matrisini oluşturup bu matrisleri çarpan ($D = B.C$) programın akış diyagramını çizip yazılımı yapınız.

7. Klavyeden girilen nxn tipindeki A matrisinin belirtilen üssünü alan programın akış diyagramını çizip herhangi bir programlama diliyle kodlayınız.

8. Klavyeden girilen nxn tipindeki A matrisindeki pozitif elemanlarını B ve negatif elemanlarını da C dizisine aktaran programın akış diyagramını çizip yazılımı gerçekleyiniz.

9. Klavyeden girilen nxn tipindeki A ve B matrislerindeki aynı olan elemanlarını C dizisine aktaran programın akış diyagramını çizip yazılımı yapınız.

10. Klavyeden girilen nxn tipindeki A matrisinde hangi elemandan kaç tane olduğunu bulan programın akış diyagramını çizip herhangi bir programlama diliyle kodlayınız.

11. Klavyeden girilen bir kelimenin harflerini; ekranın rastgele yerlerine, rastgele renklerde, rastgele büyüklüklerde yazdıran programın akış diyagramını çizip bilgisayar diline kodlayınız.

12. Klavyeden girilen bir mesajın harflerini; rastgele sırada ve rastgele renkte ekranın altından üstüne doğru 1 s aralıklarla kaydıran programın akış diyagramını çizip yazılımını gerçekleyiniz.

13. Ekranın rastgele koordinatlarına, sürekli, rastgele renklerde 0-9 arası rastgele bir tamsayı yazan programın akış diyagramını çizip herhangi bir programlama diliyle kodlayınız.

14. Klavyeden girilen ve ekranın ortasına yazılan kelimenin yarısını harf harf ortadan yukarı, yarısını da ortadan aşağıya doğru 1 s aralıklarla kaydıran programın akış diyagramını çizip yazılımını yapınız.

15. Klavyeden girilen bir kelimeyi, ekrana dikey olarak yazdıran (harfleri alt alta) programın akış diyagramını çizip programlama diliyle kodlayınız.

16. Klavyeden girilen mxn tipindeki A matrisinde kaç tane birbirinden farklı değer olduğunu veren programın akış diyagramını çizip herhangi bir dille kodlayınız.

17. Klavyeden girilen matrisin, ana ile yedek köşegen üzerindeki elemanlarını karşılıklı olarak değiştiren programın akış diyagramını çizip yazılımını yapınız.

18. Klavyeden girilen kare matris tipine göre aşağıdaki $n = 4$ için örnek matrisleri oluşturan programların akış diyagramlarını ayrı ayrı çizip kodlayınız.

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 3 \\ 3 & 4 & 3 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{bmatrix},$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad E = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \end{bmatrix}, \quad F = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

19. Klavyeden girilen matris tipine göre elemanları, 0-9 arasında rastgele tamsayılardan oluşan ve sütun elemanları toplamı eşit olan programın akış diyagramını çizip yazılımını gerçekleyiniz.

20. M kişilik sınıfın N tane dersine ait notları bilgisayara girilmektedir. Buna göre;
    - Herbir dersten sınıfın aritmetik, geometrik ve harmonik ortalamasını,
    - Herbir öğrencinin not ortalamasını,
    - En yüksek ve en düşük not ortalamasına sahip öğrenci numaralarını ve ortalamaları,
    - Herbir dersten en yüksek ve en düşük not arasındaki farkı,
    - Herbir dersteki en yüksek ve en düşük sınıf ortalamaları arasındaki farkı,

    hesaplayan programın akış diyagramını çizip yazılımını gerçekleyiniz.

21. Klavyeden girilen matrisin tek sıradaki satırlarını kendi arasında (1 ile 3, ...) değiştiren programın akış diyagramını çizip yazılımını yapınız.

22. Klavyeden küçük harflerle girilen mesajın harflerini, sırasıyla ve 1 s aralıklara büyütüp küçülterek yukarıdan aşağıya indiren ve bu işlemi herhangi bir tuşa basıncaya kadar devam ettiren programın akış diyagramını çizip yazılımını yapınız.

24. Klavyeden girilen mxn tipindeki A matrisinin herbir satırındaki negatif değere sahip elemanlarının sayısını B dizisine aktarıp yazdıran programın akış diyagramını çizerek kodlayınız.

25. Klavyeden girilen mxn tipindeki A matrisinde kaç tane pozitif değere sahip eleman olduğunu bulup yazdıran programın akış diyagramını çizerek kodlayınız.

26. Klavyeden girilen kenar uzunluğuna (1-20) göre ekranda "*" karakterlerinden eşkenar üçgen oluşturan programın akış diyagramını çizerek kodlayınız.

27. Klavyeden girilen nxn tipindeki matrisi yatay olarak döndüren/çeviren programın akış diyagramını çizerek kodlayınız.

28. Klavyeden girilen nxn tipindeki matrisi dikey olarak döndüren/çeviren programın akış diyagramını çizerek kodlayınız.

29. Klavyeden girilen nxn tipindeki matrisin her satırındaki en büyük elemanları bulup listeleyen programın akış diyagramını çizerek kodlayınız.

30. Klavyeden girilen nxn tipindeki matrisin her sütunundaki en küçük elemanları bulup listeleyen programın akış diyagramını çizerek kodlayınız.

31. Klavyeden girilen nxn tipindeki (n çift) A matrisini yatay ve dikey olarak ortadan bölerek dört tane $\frac{n}{2} x \frac{n}{2}$ tipinde alt matris oluşturan programın akış diyagramını çizerek kodlayınız.

32. Klavyeden girilen nxn tipindeki matrisin ana köşegenine göre karşılıklı elemanlarının yerlerini değiştiren programın akış diyagramını çizerek kodlayınız.

33. Klavyeden girilen nxn tipindeki matrisin yedek köşegenine göre karşılıklı elemanlarının yerlerini değiştiren programın akış diyagramını çizerek kodlayınız.

BÖLÜM 9

GRAFİK UYGULAMALARI

Bu bölümde

- Bilgisayar grafikleri hakkındaki temel bilgileri,
- Programlama dillerindeki temel çizim komutlarını ve uygulamalarını öğreneceksiniz.

Bölüm sonunda

- Bilgisayar grafiklerini kavrayacak,
- Programlama dillerindeki temel çizim komutlarını öğrenecek,
- Temel çizim işlemleri için programlar yazabileceksiniz.

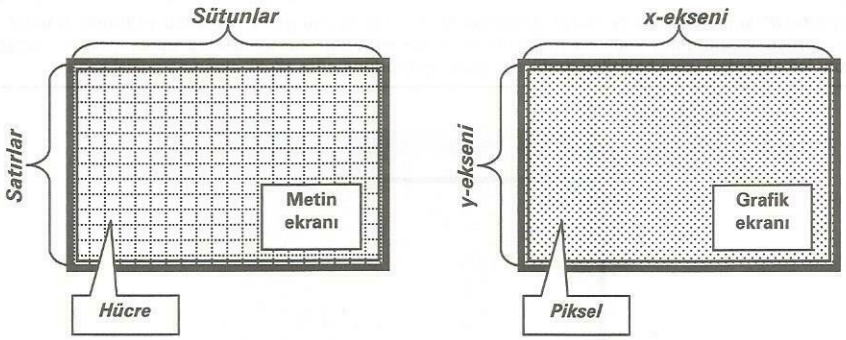
I. GİRİŞ

Bilgisayar ekranı, iki farklı modda kullanılabilmektedir. Bunlar;

i. Metin (text) modu,

ii. Grafik modu.

Metin modunda ekran; genelde 25 satır ve 80 sütundan oluşmakta (25x80 tipinde matris) ve bu ortama, 0-255 arası ASCII koduna sahip karakterler, bir hücre kaplayacak şekilde yazılmaktadırlar. Grafik modunda ise, ekran için satır-sütun kavramı yoktur. Grafik ekranı, "*piksel*" olarak adlandırılan noktacıklardan oluşmakta ve karakterler veya şekiller; belirli piksellerin, verilen renklerde parlamasıyla elde edilirler (Şekil 9.1).



Şekil 9.1 Metin ve grafik ekranı benzetimleri

II. GRAFİK KOMUTLARI

Bu alt bölümde temel grafik komutlarından ve algoritmalarda aşağıdaki isimlendirmeleriyle ifade edilecek olan

- *Çizgi*: Belirtilen iki nokta koordinatları arasına çizgi çizer
- *Daire*: Belirtilen merkez koordinatı ve yarıçap uzunluğuna göre daire çizer
- *Dikdörtgen*: Belirtilen sol üst ile sağ alt köşe koordinatları arasına dikdörtgen çizer
- *Nokta koy*: Belirtilen koordinata, verilen renkte nokta koyar

incelenecektir. Ancak bunlara geçmeden önce; metin ekranından grafik ekranına geçiş sağlayan komutlar özetlenecektir.

Not

Bu bölümde;

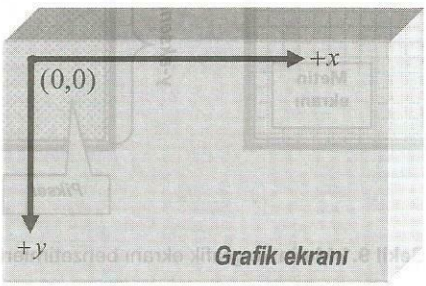
- ✓ Grafik uygulamaları için C ve C++ program kodları ortak verilecektir.
- ✓ C#'taki grafik uygulamaları form üzerinde gerçekleştirilecektir.
- ✓ Java'daki grafik uygulamalar için applet'ler kullanılacaktır.

A. Grafik Ekranına Geçiş Komutları

Metin ekranından grafik ekranına geçişi sağlayan komutlar, toplu olarak Tablo 9.1'de verilmektedir. Grafik modunda ekranın sol üst köşesi "(0,0)" koordinatıdır. Sağa gidildikçe "x" koordinat değerleri, aşağıya da inildikçe "y" koordinat değerleri artar (Şekil 9.2).

Tablo 9.1 Grafik ekranına geçiş sağlayan komutlar ve açıklamaları

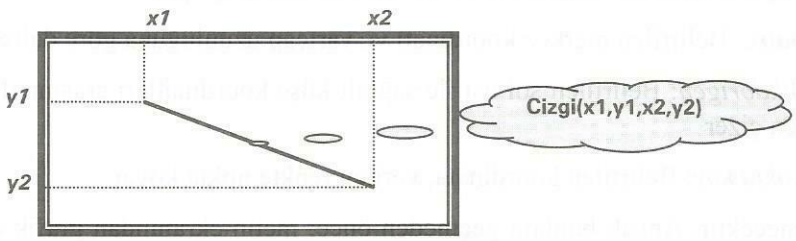
Dil	Komut	Parametre açıklamaları
C / C++	**detectgraph** (&*sürücü*,&*mod*) **initgraph** (&*sürücü*,&*mod*,*'yol'*)	*sürücü*: Grafik sürücünü belirtir. *mod*: Grafik modunu belirtir. *yol*: "*.bgi" dosyalarının bulunduğu yolu (sürücü, dizin...) belirtir.
"detectgraph" komutu otomatik olarak donanımı tarar ve en uygun grafik sürücüsü ve modunu bulur. Eğer grafik sürücü ve modunu kullanıcı belirleyecekse, bu komutun kullanılmasına gerek yoktur. *"initgraph"* komutu ise algılanan veya belirtilen grafik sürücüsü ve moduna göre grafik ekranına geçişi sağlar.		



Şekil 9.2 Grafik modunda koordinatlar

B. Çizgi

"Cizgi (x1,y1,x2,y2)" şeklinde algoritmalarda isimlendirilen komut, belirtilen *(x1,y1)* ile *(x2,y2)* noktaları arasına çizgi çizer (Şekil 9.3-Tablo 9.2).



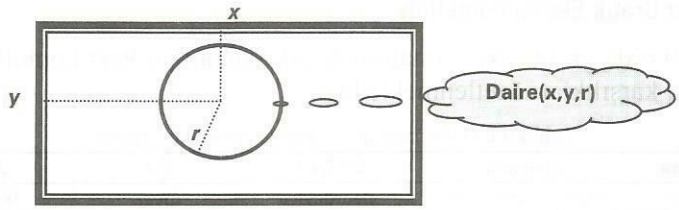
Şekil 9.3 "Cizgi" komutu

Tablo 9.2 "Cizgi(x1,y1,x2,y2)" komutunun 6 dildeki karşılığı

C / C++	C#	Java	MATLAB	Python
line (x1,y1,x2,y2)	DrawLine(çizgi, x1,y1,x2,y2)	drawLine (x1,y1,x2,y2)	line ([x1 x2],[y1 y2]) plot ([x1 x2],[y1 y2])	plot ([x1 x2],[y1 y2])

C. Daire

Algoritmalarda "*Daire(x,y,r)*" şeklinde isimlendirilen komut, verilen merkez noktasına *(x,y)* göre "*r*" yarıçaplı daire çizer (Şekil 9.4-Tablo 9.3).



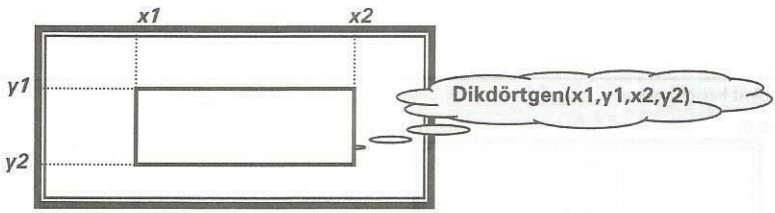
Şekil 9.4 "Daire" komutu

Tablo 9.3 "Daire(x,y,r)" komutunun 5 dildeki karşılığı

C / C++	C#	Java	Python
circle (x,y,r)	DrawEllipse(çizgi,x-r,y-r,2r,2r)	drawOval(x-r,y-r,2r,2r)	Circle ((x,y),r)

D. Dikdörtgen

"*Dikdortgen (x1,y1,x2,y2)*" şeklinde algoritmalarda isimlendirilen komut, verilen sol üst köşe *(x1,y1)* ile sağ alt köşe *(x2,y2)* arasına dikdörtgen çizer (Şekil 9.5-Tablo 9.4).



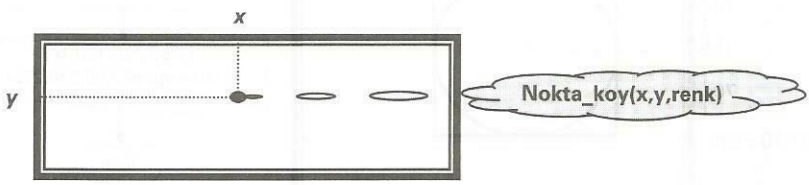
Şekil 9.5 "Dikdortgen" komutu

Tablo 9.4 "Dikdorten(x1,y1,x2,y2)" komutunun 5 dildeki karşılığı

C / C++	C#	Java	Python
rectangle (x1,y1,x2,y2)	DrawRectangle(çizgi, x1,y1,x2-x1,y2-y1)	drawRect (x1,y1,x2-x1,y2-y1)	Rectangle ((x1,y1),x2-x1,y2-y1)

E. Nokta Koy

"*Nokta_koy(x,y,renk)*" şeklinde isimlendirilen temel çizim komutu, verilen koordinata *(x,y)* belirtilen *renk*te nokta koyar(Şekil 9.6-Tablo 9.5).



Şekil 9.6 "Nokta_koy" komutu

Tablo 9.5 Temel çizim komutu olan "Nokta_koy(x,y,renk)" komutunun 6 dildeki karşılığı

C / C++	C#	Java	MATLAB	Python
putpixel(x, y, renk)	SetPixel(x,y,renk)	setPixels	plot(x, y,renk)	plot(x,y)

F. Diğer Bazı Grafik Ekranı Komutları

Tablo 9.6'da grafik uygulamalarında sık kullanılan bazı komutlar ile algoritmalardaki karşılıkları özetlenmektedir.

Tablo 9.6 Diğer bazı grafik ekranı genel komutları

Algoritma	Açıklama	C / C++	C#	Java
Grafik sil	Grafik ekranını siler	**cleardevice**	***Clear***	**fillRect**
Grafik kapat	Grafik modunu kapatır	**closegraph**		
Çizim rengi	Çizim rengini değiştirir	**setcolor**	***Pen***	**setColor**
Zemin rengi	Zemin rengini değiştirir	**setbkcolor**		**setBackground**

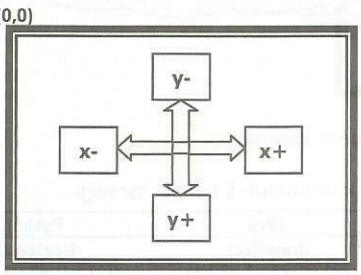
Örnek-9.1: Yandaki şekli çizdiren program.

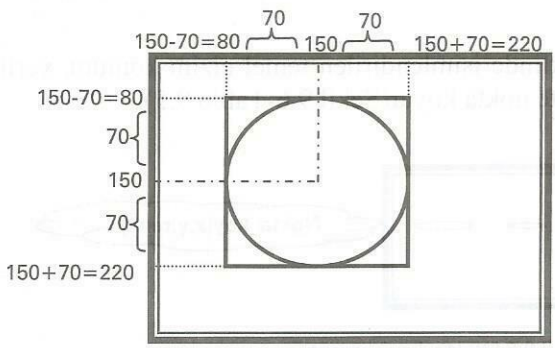
Not: Daire merkezi (150,150) ve yarıçapı 70 birimdir.

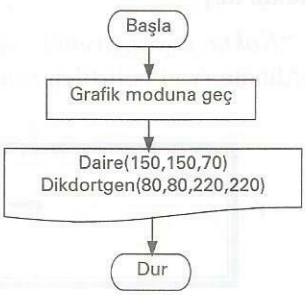
Verilen şekil, bir daire ve kareden oluşmaktadır. Daire merkezi ve yarıçapı soruda verildiğinden; karenin sol üst köşe ile sağ alt köşe koordinatları hesaplanabilir.

Not

Koordinat hesaplamaları







C/C++

```
/* Grafik 1 */
#include <graphics.h>
#include <conio.h>
int graphdriver,graphmode;
main()
{ detectgraph(&graphdriver,&graphmode);
  initgraph(&graphdriver,&graphmode,"");
  circle(150,150,70); rectangle(80,80,220,220);
  getch();closegraph();return 0; }
```

C#

```
// Grafik 1
using System;
using System.Drawing;
using System.Windows.Forms;
namespace ornek91
{ public partial class Form1 : Form
   { public Form1() { InitializeComponent(); }
     private void Form1_Paint(object sender, PaintEventArgs e)
     { Pen cizgi = new Pen(Color.Black, 3);
       e.Graphics.DrawEllipse(cizgi,80,80,140,140);
       e.Graphics.DrawRectangle(cizgi,80,80,140,140); } } }
```

JAVA

```
// Grafik 1
import java.awt.*;
import java.applet.*;
public class grafik1 extends Applet {
  public void paint(Graphics g) {
    g.drawOval(80,80,140,140); g.drawRect(80,80,140,140); }}
```

MATLAB

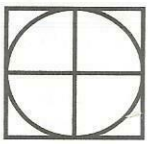
```
% Grafik 1
clear all; clc
a=0:0.01:2*pi; x=70*cos(a); y=70*sin(a); plot(150+x,150+y); hold on
line([80 220],[80 80]); line([220 220],[80 220]);
line([80 220],[220 220]); line([80 80],[220 80]);
```

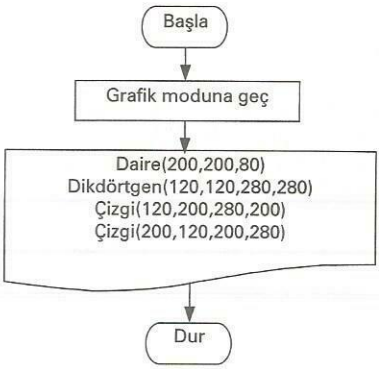
PYTHON

```
# Grafik 1
import matplotlib.pyplot as plt
import matplotlib.patches as patches
fig=plt.figure(); ax=fig.add_subplot(111); s=300
ax.add_patch(patches.Rectangle((80/s,80/s),140/s,140/s,fill=False))
ax.add_patch(patches.Circle((150/s,150/s),70/s,fill=False))
fig.show()
```

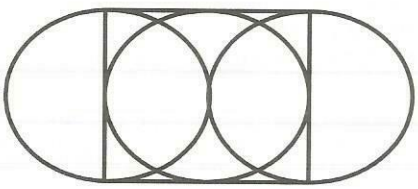
Örnek-9.2: Yandaki şekli çizdiren program.

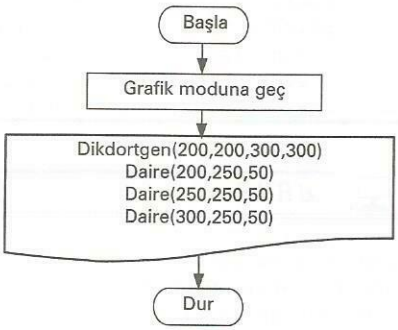
Not: Daire merkezi (200,200) ve yarıçapı 80 birimdir.





Örnek-9.3: Aşağıdaki şekli çizdiren program. *Not:* Karenin sol üst köşesi (200,200) ve bir kenarı 100 birimdir.

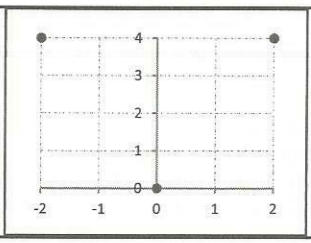


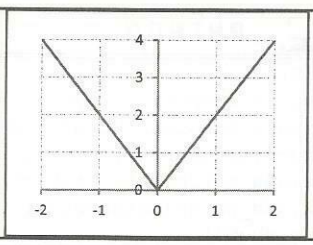


Örnek-9.4: $x \in [-15,15]$ için $y = 10.Sin(5x) + 5.Cos(3x)$ fonksiyonunun grafiğini çizdiren program.

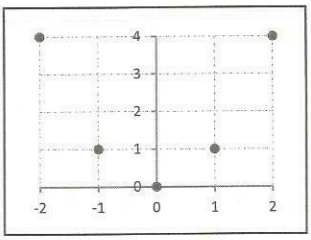
Herhangi bir $f(x)$ fonksiyonunun grafiğini çizdirmek için x değeri, küçük adımlarla arttırılır ve $y = f(x)$ hesaplattırılır. Bu şekilde elde edilen (x, y) koordinatına bir nokta basılır. Bir diğer alternatif de elde edilen iki noktanın çizgiyle birleştirilmesidir. Örneğin $x \in [-2,2]$ için $y = f(x) = x^2$ grafiğini değişik adımlarla çizdirme örnekleri aşağıda verilmektedir.

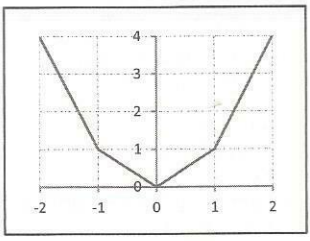
x	f(x)
-2	4
0	0
2	4



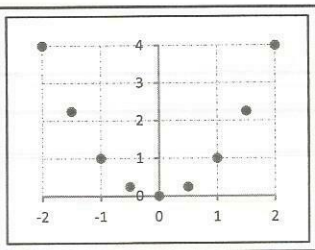


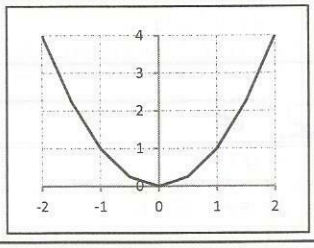
x	f(x)
-2	4
-1	1
0	0
1	1
2	4



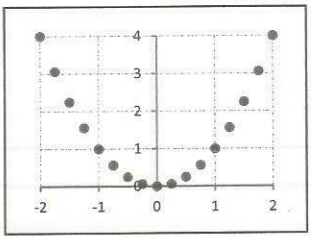


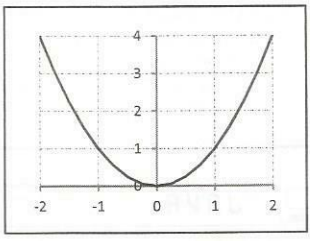
x	f(x)
-2	4
-1,5	2,25
-1	1
-0,5	0,25
0	0
0,5	0,25
1	1
1,5	2,25
2	4



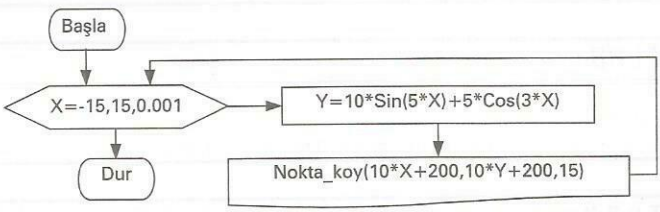


x	f(x)
-2	4
-1,75	3,0625
-1,5	2,25
-1,25	1,5625
-1	1
-0,75	0,5625
-0,5	0,25
-0,25	0,0625
0	0
0,25	0,0625
0,5	0,25
0,75	0,5625
1	1
1,25	1,5625
1,5	2,25
1,75	3,0625
2	4





Görüldüğü gibi, hesaplanan nokta sayısı arttıkça; çizdirilen grafik orijinaline daha yakın (hassas) olmaktadır.



Akış diyagramında "`Nokta_koy (X,Y,15)`" yerine "`Nokta_koy (10*X+200,10*Y+200,15)`" kullanılmıştır. "X" ve "Y" lere 200 eklemenin nedeni orijini, yani (0,0) noktasını ekranın (200,200) noktasına taşımaktır. Ayrıca "X" ve "Y" ler 10 ile çarpılarak grafik ölçeklenmiştir.

C / C++

```
/* Grafik 4 */
#include <graphics.h>
#include <math.h>
#include <conio.h>
int gd,gm;
float x,y;
main()
{ detectgraph(&gd,&gm);initgraph(&gd,&gm,"");x=-15;
  while (x<=15)
   { y=10*sin(5*x)+5*cos(3*x);
     putpixel(10*x+200,10*y+200,15);
     x+=0.001; }
  getch();closegraph();return 0; }
```

C#

```
// Grafik 4
using System;
using System.Drawing;
using System.Windows.Forms;
namespace ornek94
{ public partial class Form1 : Form
   { public Form1()
      { InitializeComponent(); }
  private void Form1_Load(object sender, EventArgs e)
   { double x=-15,y;
     while (x<=15)
      { y=10*Math.Sin(5*x)+5*Math.Cos(3*x);
        chart1.Series[0].Points.AddXY(10*x+200,10*y+200);
        x+=0.001; }}}}
```

JAVA

```
// Grafik 4
import java.awt.*;
import java.applet.*;
public class grafik4 extends Applet {
    public void paint(Graphics g) {
    int x=-15,y;
    while (x<=15)
     { y=(int) Math.round(10*Math.sin(5*x)+5*Math.cos(3*x));
       g.drawLine(10*x+200,10*y+200,10*x+200,10*y+200);  x+=1; }}}
```

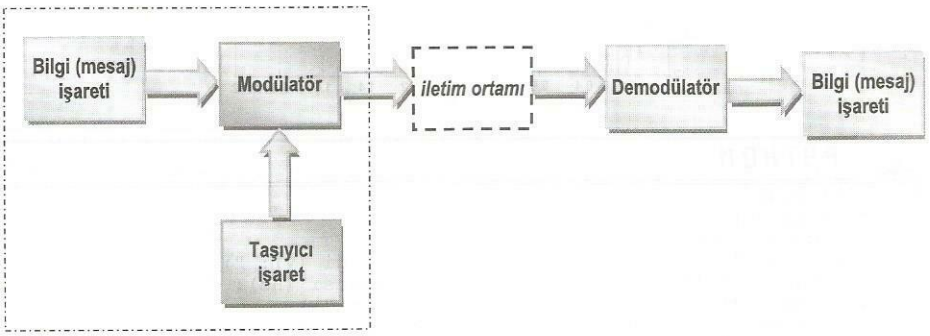
MATLAB

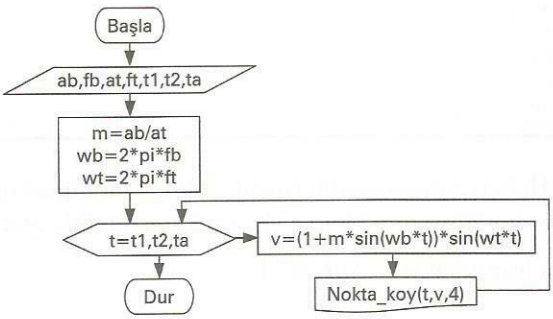
```
% Grafik 4
clear all; clc
x=-15:0.001:15; y=10*sin(5*x)+5*cos(3*x); plot(x,y);
```

PYTHON

```
# Grafik 4
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(-15,15,0.001); y=10*np.sin(5*x)+5*np.cos(3*x)
plt.plot(x,y); plt.show()
```

Örnek-9.5: Haberleşme sistemlerindeki temel modülasyonlardan birisi "*genlik modülasyonu*"dur. Bilgi (mesaj) işareti $v_b(t) = A_b Sin(2\pi f_b t)$, taşıyıcı $v_t(t) = A_t Sin(2\pi f_t t)$ işaretiyle modüle edilerek $v(t) = \left(1 + m.Sin(2\pi f_b t)\right).Sin(2\pi f_t t)$ elde edilmektedir. Burada f_b, bilgi işaretinin frekansı; f_t, taşıyıcı işaretin frekansı ($f_t \gg f_b$) ve $m = \frac{A_b}{A_t}$ modülasyon indeksidir. Buna göre klavyeden girilen bilgi işaretini, yine klavyeden girilen taşıyıcı işaret ile modüle eden vericinin çıkışındaki genlik modülasyonlu işareti belirtilen aralıkta çizdiren program.

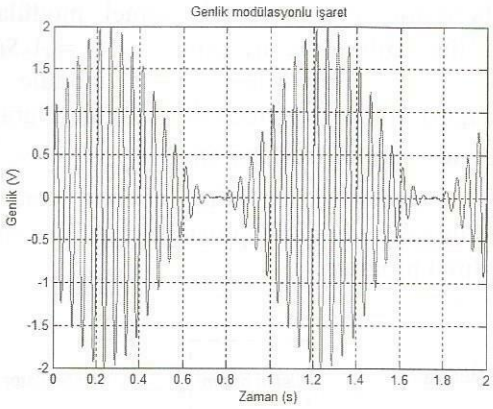




MATLAB

```
% Genlik modülasyonu
clear all; clc; ab=input('Bilgi (mesaj) işaretinin genliği (V): ');
fb=input('Bilgi (mesaj) işaretinin frekansı (Hz): ');
at=input('Taşıyıcı işaretin genliği (V): ');
ft=input('Taşıyıcı işaretin frekansı (Hz): ');
t1=input('Çizimin başlangıç zamanı (s): ');
t2=input('Çizimin bitiş zamanı (s): ');
ta=input('Çizim adım aralığı: '); m=ab/at; wb=2*pi*fb; wt=2*pi*ft;
t=t1:ta:t2; v=(1+m*sin(wb*t)).*sin(wt*t);
plot(t,v); grid on; title('Genlik modülasyonlu işaret');
xlabel('Zaman (s)'); ylabel('Genlik (V)');
```

```
Bilgi (mesaj) işaretinin genliği (V): 1
Bilgi (mesaj) işaretinin frekansı (Hz): 1
Taşıyıcı işaretin genliği (V): 1
Taşıyıcı işaretin frekansı (Hz): 20
Çizimin başlangıç zamanı (s): 0
Çizimin bitiş zamanı (s): 2
Çizim adım aralığı: 0.001
```



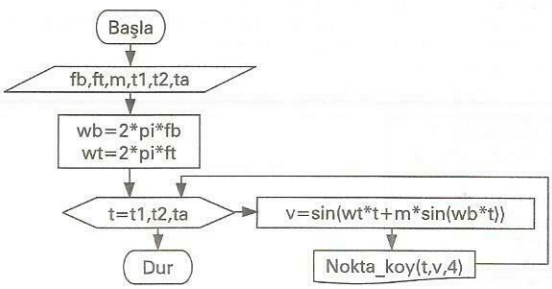
PYTHON

```
# Genlik modülasyonu
import matplotlib.pyplot as plt
import numpy as np
ab=eval(input("Bilgi (mesaj) işaretinin genliği (V): "))
fb=eval(input("Bilgi (mesaj) işaretinin frekansı (Hz): "))
at=eval(input("Taşıyıcı işaretin genliği (V): "))
ft=eval(input("Taşıyıcı işaretin frekansı (Hz): "))
t1=eval(input("Çizimin başlangıç zamanı (s): "))
t2=eval(input("Çizimin bitiş zamanı (s): "))
ta=eval(input("Çizim adım aralığı: "))
m=ab/at; wb=2*np.pi*fb; wt=2*np.pi*ft;
t=np.arange(t1,t2,ta);v=(1+m*np.sin(wb*t))*np.sin(wt*t)
plt.plot(t,v); plt.grid();plt.title("Genlik modülasyonlu işaret")
plt.xlabel("Zaman (s)");plt.ylabel("Genlik (V)"); plt.show()
```

Örnek-9.6: Haberleşme sistemlerindeki temel modülasyon türlerinden bir diğeri de "*frekans modülasyonu*"dur (FM). Frekans modülasyonunda;

- ✓ Bilgi (mesaj) işareti: $v_b(t) = Sin(2\pi f_b t)$
- ✓ Taşıyıcı işaret: $v_t(t) = Sin(2\pi f_t t)$
- ✓ Modülasyonlu işaret: $v(t) = Sin(2\pi f_t t + m.Sin(2\pi f_b t))$

şeklindedir. Buna göre klavyeden girilen bilgi işaretini, yine klavyeden girilen taşıyıcı işaret ile modüle eden vericinin çıkışındaki frekans modülasyonlu işareti belirtilen aralıkta çizdiren program.



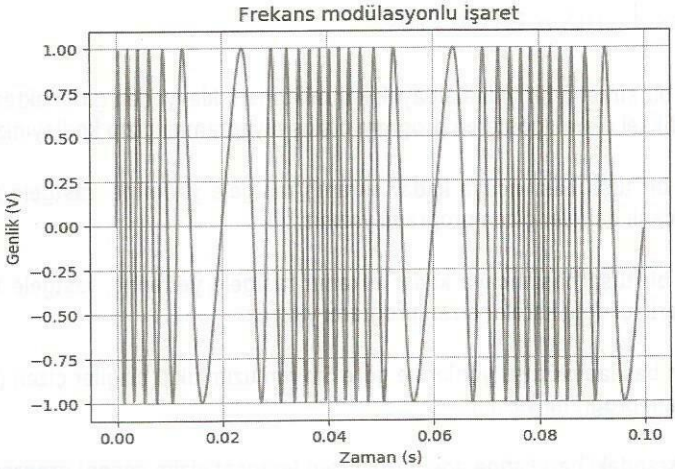
MATLAB

```
% Frekans modülasyonu
clear all; clc; fb=input('Bilgi (mesaj) işaretinin frekansı (Hz): ');
ft=input('Taşıyıcı işaretin frekansı (Hz): ');
m=input('Modülasyon indeksi: ');
t1=input('Çizimin başlangıç zamanı (s): ');
t2=input('Çizimin bitiş zamanı (s): ');
ta=input('Çizim adım aralığı: ');
wb=2*pi*fb; wt=2*pi*ft; t=t1:ta:t2; v=sin(wt*t+m*sin(wb*t));
plot(t,v); grid on; title('Frekans modülasyonlu işaret');
xlabel('Zaman (s)'); ylabel('Genlik (V)');
```

PYTHON

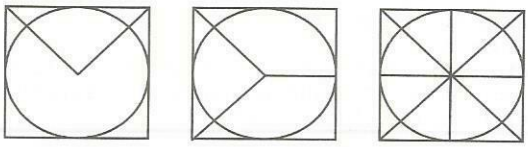
```
# Frekans modülasyonu
import matplotlib.pyplot as plt
import numpy as np
fb=eval(input("Bilgi (mesaj) işaretinin frekansı (Hz): "))
ft=eval(input("Taşıyıcı işaretin frekansı (Hz): "))
m=eval(input("Modülasyon indeksi: "))
t1=eval(input("Çizimin başlangıç zamanı (s): "))
t2=eval(input("Çizimin bitiş zamanı (s): "))
ta=eval(input("Çizim adım aralığı: "))
wb=2*np.pi*fb; wt=2*np.pi*ft;
t=np.arange(t1,t2,ta);v=np.sin(wt*t+m*np.sin(wb*t))
plt.plot(t,v); plt.grid();plt.title("Frekans modülasyonlu işaret")
plt.xlabel("Zaman (s)");plt.ylabel("Genlik (V)"); plt.show()
```

```
Bilgi (mesaj) işaretinin frekansı (Hz):
25
Taşıyıcı işaretin frekansı (Hz): 300
Modülasyon indeksi: 10
Çizimin başlangıç zamanı (s): 0
Çizimin bitiş zamanı (s): 0.1
Çizim adım aralığı: 0.0001
```

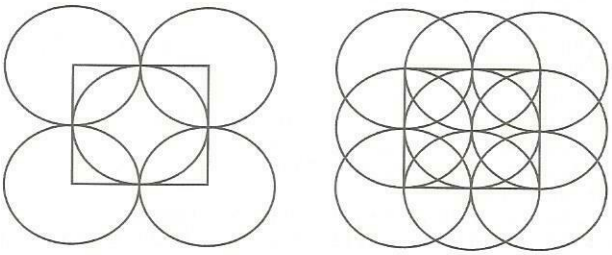


SORULAR

1. Aşağıdaki şekilleri çizdiren programların akış diyagramlarını çizip kodlayınız. Daire merkezi (220, 220) ve yarıçapı 80 birimdir.



2. Aşağıdaki şekilleri çizdiren programların akış diyagramını çizip kodlayınız. Kare içindeki temel dairenin merkezi (200, 200) ve yarıçapı 90 birimdir.



3. $x \in [-20,20]$ için aşağıdaki fonksiyonların grafiklerini çizdiren programların akış diyagramlarını çizip yazılımlarını yapınız.
   - $y = 5.Sin(11x) - 4.Cos(6x)$
   - $y = \left| x^4 - 3x^3 + 2x^2 - 1 \right|$
   - $y = 5e^{x/4} - 10.Sin(4x) + x^2/10$

4. Herhangi bir şirketin bir yılına ait aylık kar miktarları, klavyeden girilmektedir. Kar dağılımını grafiksel olarak gösteren programın akış diyagramını çizip kodlayınız.

5. Herhangi bir tuşa basılıncaya kadar ekranın rastgele yerlerine, rastgele renklerde 33 birim yarıçaplı daireler çizen programı yazınız.

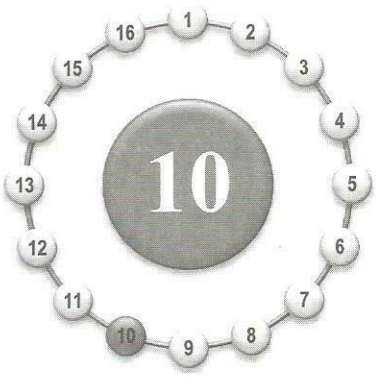
6. Herhangi bir tuşa basılıncaya kadar ekranın rastgele yerlerine, rastgele renklerde 20 birim kenarlı içi dolu kareler çizen programı yazınız.

7. Klavyeden basılan ok tuşu yönlerine göre 5 birim uzunluklu çizgiler çizen (ok tuşlarıyla çizim yapan) programı yazınız.

8. Farenin ekrandaki hareketine göre çizgi çizen (serbest çizim yapan) programı yazınız.

BÖLÜM 10

DOSYALAMA UYGULAMALARI



Bu bölümde

- Dosyalar ve veritabanları hakkındaki temel bilgileri,
- Dosyalardaki temel işlemleri öğrenecekşiniz.

Bölüm sonunda

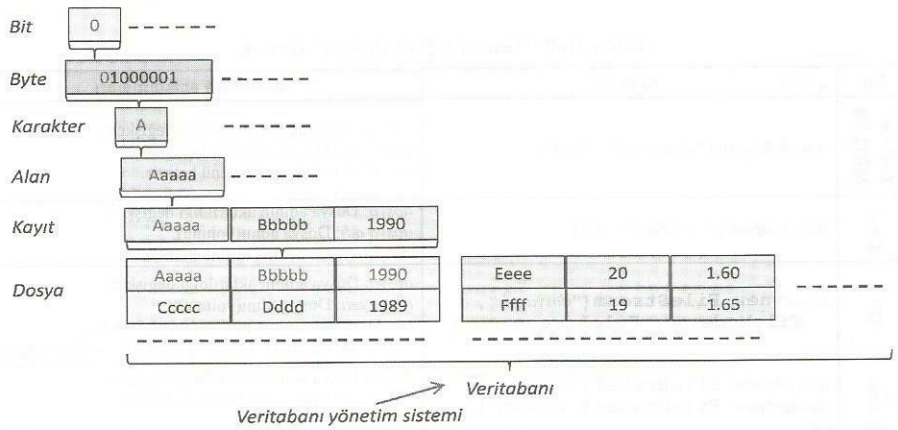
- Dosyalama ve veritabanı terimlerini açıklayabilecek,
- Dosyalar üzerinde temel işlemleri gerçekleştirebilecek,
- Temel dosyalama işlemlerinin akış diyagramlarını çizip kodlayabilecekşiniz.

I. GİRİŞ

Geliştirilen programlarda işlenerek elde edilen ve sayısal ya da alfasayısal tekil veya çoğul (dizi, matris vb.) değişkenlere aktarılan veriler, program çalıştığı sürece bellekte kalmaktadırlar. Program sonlandırıldığında, bu veriler de silinmektedirler. Veriler ve özellikle de çoklu kayıtlar (bilgiler), bilgisayarlarda veya çevre birimlerinde (HDD, CD, DVD, USB bellek vb.) "*dosya*" veya "*kütük*" olarak adlandırılan veri kayıt yapılarında kalıcı olarak tutulurlar/saklanırlar. Veri yapıları ve işlenmesi açısından kullanılan temel terimler, Tablo 10.1'de özetlenmekte ve aralarındaki ilişki Şekil 10.1'de gösterilmektedir.

Tablo 10.1: Veri yapılarındaki temel terimler

Terim	Açıklama
Bit	İkili "0" veya "1" değerlerini alabilen bilgisayardaki en küçük veri parçasıdır.
Byte	Sekiz tane bitin yan yana gelerek oluşturduğu veri grubudur.
Karakter	Verilerin yazımında kullanılan ve 1 Byte'lık harf, rakam ve özel simgelerdir.
Alan	Karakterlerin yan yana gelerek oluşturduğu veri grubudur.
Kayıt	Alanların yan yana gelerek oluşturmuş olduğu yapıdır.
Dosya	Belirli kayıtların tutulduğu topluluk yapısıdır.
Veritabanı	Birçok dosyanın yer aldığı yapıdır.



Şekil 10.1: Veri yapılarındaki terimler arasındaki ilişkiler

Dosyalar; genel olarak verilerin kayıt biçimine göre ikiye, kayıtlara erişim şekline göre de üçe ayrılırlar (Şekil 10.2). Kayıtlara erişim şekillerine göre dosyalar:

- ✓ *Sıralı erişimli dosyalar*: Kayıtlara erişimin sıralı bir şekilde yapıldığı ve kayıt alanı uzunluklarının serbest olduğu dosyalardır.
- ✓ *Rastgele erişimli dosyalar*: Kayıtlara erişimin doğrudan yapıldığı ve kayıt alanı uzunluklarının sabit olduğu dosyalardır.
- ✓ *İndeks erişimli dosyalar*: Kayıtlara erişimin belirli indekslere (terimlere/anahtar kelimelere) göre yapıldığı ve sıralı-rastgele erişimli dosyalama kombinasyonu olan türüdür.

Şekil 10.2: Dosya türleri

Dosyalama programı yazarken

i. Dosyada tutulacak verilerin türüne göre en uygun kayıt alanı oluşturulmalıdır.
ii. Her kayıtta benzersiz bir alan (kayıt anahtarı, birincil anahtar) olmalıdır.
iii. Dosya yoksa oluşturulmalı, varsa da ilgili işlem için açılmalıdır.
iv. Dosya üzerinde işlemler gerçekleştirilmelidir.
v. İşlemler tamamlandıktan sonra dosya kapatılmalıdır.

II. DOSYALAMA KOMUTLARI

A. Dosya Aç

"*Dosya_ac(tür,dosya adı)*" komutu; *adı* belirtilen dosyayı, verilen *tür*de açar (Tablo 10.2).

Tablo 10.2: "Dosya_ac(tür,dosya)" komutu

Dil	Komut	Parametre açıklamaları
C/C++/ MATLAB	*dosya*=**fopen**(*"dosya adı"*, *"tür"*)	*dosya*: Dosya adının aktarıldığı değişkeni, *dosya adı*: Dosya adını(yolunu), *tür*: Dosyanın açılma türünü belirtir.
C++	*dosya*.**open**(*"dosya adı"*, *tür*)	*dosya*: Dosya adının aktarıldığı değişkeni, *dosya adı*: Dosya adını(yolunu), *tür*: Dosyanın açılma türünü belirtir.
C#	*dosya*=**new FileStream**(*"dosya adı"*, **FileMode**.*tür*, **FileAccess**.*işlem*)	*dosya*: Dosya adının aktarıldığı değişkeni, *dosya adı*: Dosya adını(yolunu), *tür*: Dosyanın açılma türünü (aç, oluştur vb.), *erişim*: Dosya işlem (okuma, yazma vb.) şeklini belirtir.
Java	*dosya*=**new FileWriter**(*"dosya adı"*) *dosya*=**new FileReader**(*"dosya adı"*)	*dosya*: Dosya adının aktarıldığı değişkeni, *dosya adı*: Dosya adını(yolunu) belirtir.
Python	*dosya*=**open**(*"dosya adı"*, *"tür"*) ;	*dosya*: Dosya adının aktarıldığı değişkeni, *dosya adı*: Dosya adını(yolunu), *tür*: Dosyanın açılma türünü belirtir.

B. Dosya Kapat

"*Dosya_kapat(dosya adı)*" komutu; adı belirtilen dosyayı kapatır (Tablo 10.3). Üzerinde çalışılan dosya, işlemler bitince mutlaka kapatılmalıdır.

Tablo 10.3: "Dosya_kapat(dosya)" komutu

Dil	Komut	Parametre açıklamaları
C / C++ / MATLAB	**fclose**(*dosya*)	*dosya*: Dosya adının aktarıldığı dosya değişkenini belirtir.
C++ / Java / Python	*dosya*.**close**()	
C#	*dosya*.**Close**()	

Örnek-10.1: Kayıt girme/ekleme işlemi.

Sıralı erişimli dosya	Rastgele erişimli dosya

C

```
/* Sıralı erişimli dosya - Kayıt girişi/ekleme */
#include <stdio.h>
#include <conio.h>
FILE *dosya;
char ad[25], adres[40], tel[12];
void main()
{ clrscr();dosya=fopen("rehber.txt","a+");
  if (dosya==NULL) dosya=fopen("rehber.txt","wt");
  printf("Adı-Soyadı: ");scanf("%s",&ad);
  printf("Adresi: ");scanf("%s",&adres);
  printf("Telefonu: ");scanf("%s",tel);
  fprintf(dosya,"%s\t%s\t%s\n",ad,adres,tel);
  fclose(dosya); getch();}
```

C++

```
// Sıralı erişimli dosya - Kayıt girişi/ekleme
#include <iostream>
#include <fstream>
using namespace std;
int main()
{ ofstream dosya; dosya.open("rehber.txt",ios::app);
  if (!dosya.is_open()) dosya.open("rehber.txt",ios::in);
  char ad[25], adres[40], tel[12];
  system("cls"); cout<<"Adı-Soyadı: "; cin>>ad;
  cout<<"Adresi: "; cin>>adres; cout<<"Telefonu: "; cin>>tel;
  dosya<<ad<<"\t"<<adres<<"\t"<<tel<<"\n";
  dosya.close(); system("pause"); return 0; }
```

C#

```
// Sıralı erişimli dosya - Kayıt girişi/ekleme
using System;
using System.IO;
namespace ornek101
{ class Program
   { static void Main(string[] args)
      { string ad, adres, tel;
```

```
        FileStream dosya=new FileStream("rehber.txt",
                          FileMode.OpenOrCreate, FileAccess.Write);
        StreamWriter sw=new StreamWriter(dosya);
        Console.Write("Adı-Soyadı: "); ad=Console.ReadLine();
        Console.Write("Adresi: "); adres=Console.ReadLine();
        Console.Write("Telefonu: "); tel=Console.ReadLine();
        sw.WriteLine(ad+"\t"+adres+"\t"+tel);
        sw.Flush(); dosya.Close(); Console.ReadLine();    } } }
```

JAVA

```
// Sıralı erişimli dosya - Kayıt girişi/ekleme
import java.util.Scanner;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.io.IOException;
public class ornek101 {
 public static void main(String[] args) {
 try {
  PrintWriter dosya=new PrintWriter(new FileWriter("rehber.txt",true));
  String ad, adres, tel;
  Scanner klavye=new Scanner(System.in);
  System.out.print("Adı-Soyadı: "); ad=klavye.next();
  System.out.print("Adresi: "); adres=klavye.next();
  System.out.print("Telefonu: "); tel=klavye.next();
  dosya.println(ad+"\t"+adres+"\t"+tel); dosya.close();}
 catch(IOException e) {System.out.println("Hata: "+e);}    }}
```

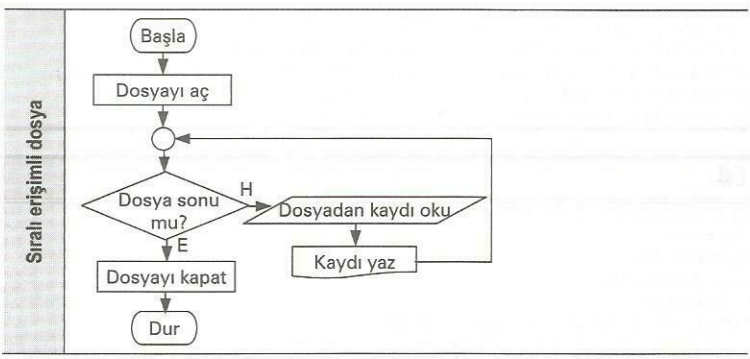
MATLAB

```
% Sıralı erişimli dosya - Kayıt girişi/ekleme
clear all; clc; dosya=fopen('rehber.txt','at');
ad=input('Adı-Soyadı: ','s'); adres=input('Adresi: ','s');
tel=input('Telefonu: ','s');
fprintf(dosya,'%s\t%s\t%s\n',ad,adres,tel); fclose(dosya);
```

PYTHON

```
# Sıralı erişimli dosya - Kayıt girişi/ekleme
dosya=open("rehber.txt","a")
ad=input("Adı-Soyadı: ")
adres=input("Adresi: ")
tel=input("Telefonu: ")
dosya.write(ad+"\t"+adres+"\t"+tel+"\n")
dosya.close()
```

Örnek-10.2: Kayıt listeleme işlemi.



C

```
/* Sıralı erişimli dosya - Kayıt listeleme */
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
FILE *dosya; int k;
main()
{ clrscr();dosya=fopen("rehber.txt","r");
  if (dosya==NULL)
       { printf("Kayıt dosyası yoktur!"); exit(1); }
  while ((k=fgetc(dosya))!=EOF) putchar(k);
  fclose(dosya); getch();return 0; }
```

C++

```
// Sıralı erişimli dosya - Kayıt listeleme
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main()
{ ifstream dosya("rehber.txt");
  if (!dosya.is_open()) { cout<<"Kayıt dosyası yoktur!"; exit(1); }
  string satir;
  while (getline(dosya,satir))
       cout<<satir<<"\n";
  dosya.close(); system("pause"); return 0; }
```

C#

```
// Sıralı erişimli dosya - Kayıt listeleme
using System;
using System.IO;
namespace ornek102
{ class Program
   { static void Main(string[] args)
      { FileStream dosya=new FileStream("rehber.txt",
                         FileMode.Open, FileAccess.Read);
        StreamReader sr=new StreamReader(dosya);
        string satir=sr.ReadLine();
        while (satir!=null)
         { Console.WriteLine(satir); satir=sr.ReadLine(); }
        dosya.Close(); Console.ReadLine();  } } }
```

JAVA

```
// Sıralı erişimli dosya - Kayıt listeleme
import java.util.Scanner;
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
public class ornek102 {
 public static void main(String[] args) {
  try {
 BufferedReader dosya=new BufferedReader(new FileReader("rehber.txt"));
  String satir;
  while ((satir=dosya.readLine())!=null)
        System.out.println(satir);  dosya.close();}
  catch(IOException e) {System.out.println("Hata: "+e); } }}
```

MATLAB

```
% Sıralı erişimli dosya - Kayıt listeleme
clear all; clc; dosya=fopen('rehber.txt','r');
while (feof(dosya)~=1)
    satir=fgetl(dosya); fprintf('%s\n',satir);
end; fclose(dosya);
```

PYTHON

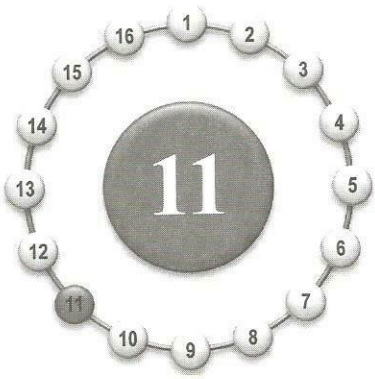
```
# Sıralı erişimli dosya - Kayıt listeleme
dosya=open("rehber.txt","r")
for satir in dosya:
    print(satir,end='')
dosya.close()
```

SORULAR

1. Sıralı erişimli dosyalarda kayıt arama işleminin akış diyagramını çizip kodlayınız.
2. Sıralı erişimli dosyalarda kayıt düzeltme işleminin akış diyagramını çizip kodlayınız.
3. Sıralı erişimli dosyalarda kayıt silme işleminin akış diyagramını çizip kodlayınız.
4. Rastgele erişimli dosyalarda kayıt girişi işleminin akış diyagramını çizip kodlayınız.
5. Rastgele erişimli dosyalarda kayıt listeleme işleminin akış diyagramını çizip kodlayınız.
6. Rastgele erişimli dosyalarda kayıt arama işleminin akış diyagramını çizip kodlayınız.
7. Rastgele erişimli dosyalarda kayıt düzeltme işleminin akış diyagramını çizip kodlayınız.
8. Rastgele erişimli dosyalarda kayıt silme işleminin akış diyagramını çizip kodlayınız.
9. İndeks erişimli dosyaları araştırarak dosyalama uygulamaları gerçekleştiriniz.
10. Bir bölümdeki öğrencilerin kişisel bilgilerini tutan dosya uygulaması programını geliştiriniz.
11. Bir bölümdeki öğrencilerin notlarını tutan dosya uygulaması programını geliştiriniz.
12. Bir mağazadaki ürünlerin bilgilerini tutan dosya uygulaması programını geliştiriniz.
13. Sıralı erişimli dosya yapısı kullanarak adres-telefon rehberi uygulaması geliştiriniz.
14. Rastgele erişimli dosya yapısı kullanarak adres-telefon rehberi uygulaması geliştiriniz.
15. Sıralı erişimli dosya yapısı kullanarak hasta takip uygulaması geliştiriniz.
16. Rastgele erişimli dosya yapısı kullanarak hasta takip uygulaması geliştiriniz.
17. Sıralı erişimli dosya yapısı kullanarak personel takip uygulaması geliştiriniz.
18. Rastgele erişimli dosya yapısı kullanarak personel takip uygulaması geliştiriniz.
19. Sıralı erişimli dosya yapısı kullanarak ürün takip uygulaması geliştiriniz.
20. Rastgele erişimli dosya yapısı kullanarak ürün takip uygulaması geliştiriniz.

BÖLÜM 11

ÖZEL SAYI VE ŞEKİLLER



Bu bölümde

- Bazı özel sayıları ve şekilleri,
- Hesaplanma (elde edilme) yollarını,
- Algoritmalarını hazırlayarak kodlamasını öğreneceksiniz.

Bölüm sonunda

- Bazı özel sayı ve şekilleri tanıyacak,
- Bunları hesaplayabilecek/elde edebilecek,
- Algoritmalarını hazırlayıp kodlayabileceksiniz.

Bilimin birçok alanında farklı özel sabitler, sayılar, şekiller vb. kullanılmaktadır. Bu bölümde farklı alanlara ait bazı özel sayı ve şekillerin hesaplanması/elde edilmesi için kullanılan yöntemler sunularak, akış diyagramları çizilip kodlanacaktır.

I. *e*

Doğal logaritma tabanı olan "*e*" sayısı yaklaşık olarak 2.718281828459 olup farklı şekillerde hesaplanabilmektedir.

Örnek-11.1: e^x fonksiyonunun "*Maclaurin serisi*" açılımı aşağıdadır. Buna göre; klavyeden girilen x değeri için, yine klavyeden girilen terim sayısı (N) kadar seriye açarak e^x 'i hesaplayan program.

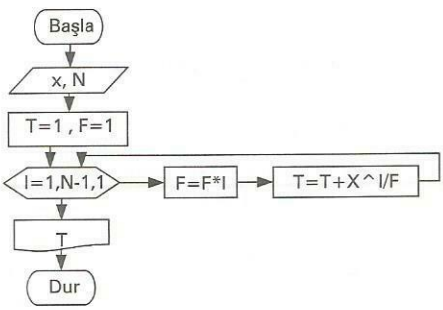
$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots$$

e^x'in açılımına dikkat edilirse; pay kısmında x'in kuvvetleri düzgün bir sırada artarken aynı şekilde paydada da yine aynı değerlerin faktöriyelleri bulunmaktadır. Bu düzgün artımlar k ile gösterilirse

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

şeklinde ardışık toplam ifadesi elde edilir.

Örneğin e^1'i hesaplamak için $x = 1$ alıp ilk üç terim ($N = 3$) ile elde edilen sonuç, Tablo 11.1'de verilmektedir. Terim sayısı arttıkça, doğru sonuca daha da yaklaşılacaktır.



Tablo 11.1: Örnek-11.1 için işlem adımları

I	Eski F	Yeni F	Eski T	Yeni T
1	1	1*1=1	1	$1+1^1/1=2$
2	1	1*2=2	2	$2+1^2/2=2.5$
3	2	2*3=6	2.5	$2.5+1^3/6=2.66$
				e^1 = 2.66

C

```
/* e üzeri */
#include <stdio.h>
#include <conio.h>
#include <math.h>
int N,i; float x,T=1,F=1;
void main()
{ clrscr();printf("Terim sayısını giriniz: ");scanf("%u",&N);
  printf("Hesaplanacak x değerini giriniz: ");scanf("%f",&x);
  for(i=1;i<N;i++) { F*=i;T+=pow(x,i)/F; }
  printf("\nSeri açılımı ile e üzeri x değeri: %0.5f\n",T);
  printf("Komutla e üzeri x değeri: %0.5f\n",exp(x)); getch();}
```

C++

```
// e üzeri
#include <iostream>
#include <math.h>
using namespace std;
int N; float x,T=1,F=1;
int main()
{   system("cls"); cout<<" Terim sayısını giriniz: "; cin>>N;
    cout << "Hesaplanacak x değerini giriniz: "; cin >> x;
    for(int i=1;i<N;i++) { F*=i; T+=pow(x,i)/F; }
    cout<<"\nSeri açılımı ile e üzeri x değeri: "<<T<<endl;
    cout<<"Komutla e üzeri x değeri: "<<exp(x)<<endl;
    system("pause");return 0; }
```

C#

```
// e üzeri
using System;
namespace ornek111
{ class Program
  { static void Main(string[] args)
    { int N; double x,T=1,F=1;
      Console.Write("Terim sayısını giriniz: ");
      N=Convert.ToInt32(Console.ReadLine());
      Console.Write("Hesaplanacak x değerini giriniz: ");
      x=Convert.ToDouble(Console.ReadLine());
      for(int i=1;i<N;i++) { F*=i; T+=Math.Pow(x,i)/F;}
    Console.WriteLine("\nSeri açılımı ile e uzeri x değeri: {0:F5}",T);
    Console.WriteLine("Komutla e üzeri x değeri: {0:F5}",Math.Exp(x));
      Console.ReadLine();  }  }  }
```

JAVA

```
// e üzeri
import java.util.Scanner;
public class ornek111 {
public static void main(String[] args)
{ int N;
  double x,T=1,F=1;
  Scanner klavye=new Scanner(System.in);
  System.out.print("Terim sayısını giriniz: "); N=klavye.nextInt();
  System.out.print("Hesaplanacak x değerini giriniz: ");
  x=klavye.nextDouble();
  for(int i=1;i<N;i++) { F*=i; T+=Math.pow(x,i)/F;}
  System.out.printf("\nSeri açılımı ile e uzeri x değeri: %.5f\n",T);
  System.out.printf("Komutla e üzeri x değeri: %.5f\n",Math.exp(x)); }}
```

```
Terim sayısını giriniz: 10
Hesaplanacak x değerini giriniz: 3

Seri açılımı ile e^x değeri: 20.06339
Komutla e^x değeri: 20.08554
```

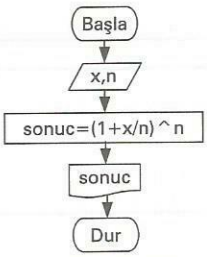
MATLAB

```
% e üzeri
clear all; clc ; T=1; F=1;
N=input(Terim sayısını giriniz: ');
x=input('Hesaplanacak x değerini giriniz: ');
for i=1:N-1
    F=F*i; T=T+x^i/F;
end
fprintf('\nSeri açılımı ile e üzeri x değeri: %0.5f\n',T);
fprintf('Komutla e üzeri x değeri: %0.5f\n',exp(x));
```

PYTHON

```
# e üzeri
import math
N=int(input("Terim sayısını giriniz: "))
x=eval(input("Hesaplanacak x değerini giriniz: ")); T=1; F=1
for i in range(1,N):
   F*=i; T+=x**i/F
print("\nSeri açılımı ile e üzeri x değeri: %0.5f"%T)
print("Komutla e üzeri x değeri: %0.5f"%math.exp(x))
```

Örnek-11.2: e^x değeri $e^x = \underset{n\to\infty}{Lim}\left(1+\frac{x}{n}\right)^n$ "*Binom açılımı*" kullanılarak hesaplanabilmektedir. Buna göre; klavyeden girilen x ve n değerlerine göre e^x 'i hesaplayan program.



C

```
/* e üzeri */
#include <stdio.h>
#include <conio.h>
#include <math.h>
void main()
{ float x,n,sonuc; clrscr();printf("exp(x)=lim(1+x/n)^n\n\n\n");
  printf("x: "); scanf("%f",&x); printf("n: "); scanf("%f",&n);
  sonuc=pow((1+x/n),n); printf("\nexp(%f)=%f\n",x,sonuc);getch();}
```

C++

```
// e üzeri
#include <iostream>
#include <math.h>
using namespace std;
int main()
{ float x,n,sonuc; system("cls"); cout<<"exp(x)=lim(1+x/n)^n\n\n\n";
  cout<<"x: ";cin>>x;cout<<"n: ";cin>>n;
  sonuc=pow((1+x/n),n); cout<<"\nexp("<<x<<")="<<sonuc<<endl;
  system("pause");return 0; }
```

C#

```
// e üzeri
using System;
namespace ornek112
{ class Program
   { static void Main(string[] args)
      { double x,n,sonuc;
        Console.WriteLine("exp(x)=lim(1+x/n)^n\n");
        Console.Write("x: "); x=Convert.ToDouble(Console.ReadLine());
        Console.Write("n: "); n=Convert.ToDouble(Console.ReadLine());
        sonuc=Math.Pow((1+x/n),n);
        Console.WriteLine("\nexp({0})={1}",x,sonuc);
        Console.ReadLine();        }    }    }
```

JAVA

```
// e üzeri
import java.util.Scanner;
public class ornek112 {
  public static void main(String[] args) {
  double x,n,sonuc; Scanner klavye=new Scanner(System.in);
  System.out.printf("exp(x)=lim(1+x/n)^n\n\n\n");
  System.out.print("x: "); x=klavye.nextDouble();
  System.out.print("n: "); n=klavye.nextDouble();
  sonuc=Math.pow((1+x/n),n);
  System.out.printf("\nexp(%f)=%f\n",x,sonuc);      }}
```

MATLAB

```
% e üzeri
clear all; clc; fprintf('exp(x)=Lim(1+x/n)^n\n\n\n');
x=input('x: '); n=input('n: '); fprintf('\ne: %f\n',(1+x/n)^n);
```

PYTHON

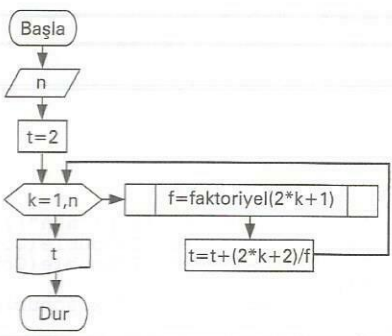
```
# e üzeri
print("exp(x)=Lim(1+x/n)^n\n\n")
x=eval(input("x: ")); n=eval(input("n: "))
sonuc=(1+x/n)**n ; print("\nexp(%f)=%f"%(x,sonuc))
```

```
exp(x)=Lim(1+x/n)^n

x: 1
n: 1000

exp(1.000000)=2.716924
```

Bilgi

$$e \approx \left(1+\frac{1}{n}\right)^{n}\Bigg]_{n\ çok\ büyük} \quad , \quad e \approx (1+n)^{\frac{1}{n}}\Bigg]_{n\ çok\ küçük}$$

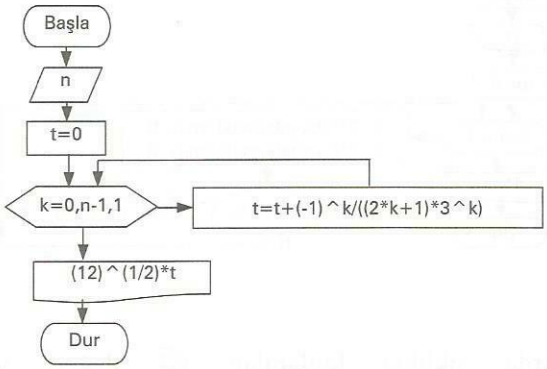
Örnek-11.3: e değeri $e = \sum_{k=0}^{\infty} \frac{2k+2}{(2k+1)!}$ "*Brothers formülü*" kullanılarak hesaplanabilmektedir. Buna göre; klavyeden girilen terim sayısına göre e sayısını hesaplayan program.



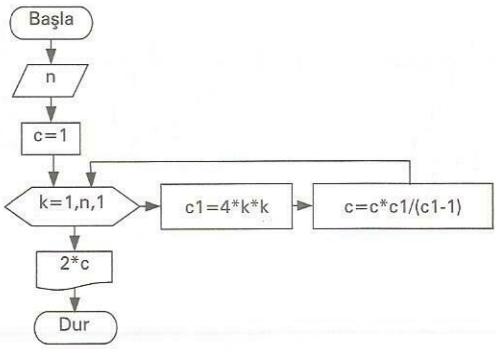
II. π

Birçok alanda kullanılan π sayısının yaklaşık değeri 3.14159265358979323846264338327 95 olup farklı şekillerde hesaplanabilmektedir.

Örnek-11.4: π sayısı $\pi = \sqrt{12}\left\{1 - \frac{1}{3.3} + \frac{1}{5.3^2} - \frac{1}{7.3^3} + \frac{1}{9.3^4} - \frac{1}{11.3^5} + ...\right\}$ yaklaşık olarak *Madhava of Sangamagrama* seri toplamıyla hesaplanabilmektedir. Buna göre klavyeden girilen terim sayısı için π değerini hesaplayıp yazdıran program.



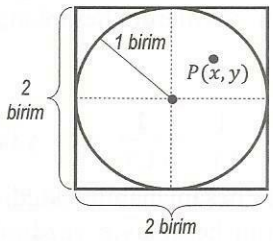
Örnek-11.5: π sayısı $\frac{\pi}{2} = \frac{4}{3}.\frac{16}{15}.\frac{36}{35}.\frac{64}{63}...$ şeklindeki *John Wallis* ardışık çarpımıyla hesaplanabilmektedir. Buna göre klavyeden girilen terim sayısı kadar çarpım yaparak π sayısını hesaplayan ve ekranda gösteren program.



Örnek-11.6: Klavyeden girilen pozitif tamsayıya göre π sayısını "*Monte Carlo yöntemi*"yle hesaplayan program.

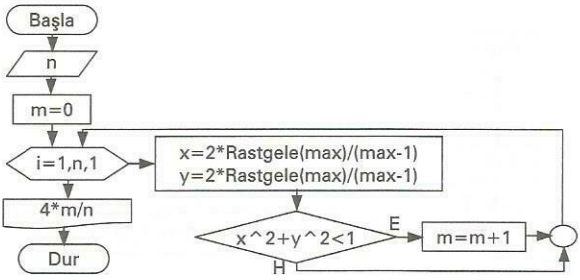
Monte Carlo yöntemleri (Monte Carlo deneyleri); sayısal sonuçlar elde etmek için tekrarlanan tesadüfi (rastgele) değerlere dayanan hesaplama algoritmaları olarak özetlenebilmektedir. Örneğin bir kenarı 2 birim olan karenin kenarla-

rına içten teğet ve yarıçapı 1 birim olan daire çizilirse; rastgele bir noktanın dairenin içine düşme olasılığı



$$P(x^2+y^2<1)=\frac{Dairenin\ alanı}{Karenin\ alanı}=\frac{\pi.1^2}{2.2}=\frac{\pi}{4}$$

dir. Eğer n tane rastgele nokta üretilir ve kaç tanesinin dairenin içinde kaldığı sayılırsa (m) ilgili $\frac{m}{n}=\frac{\pi}{4}$ oranından $\pi=4\frac{m}{n}$ sayısı elde edilebilir.

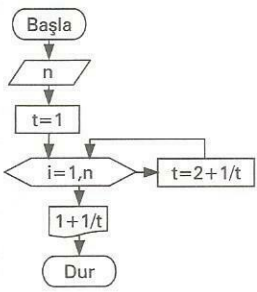


III. $\sqrt{2}$

Hesaplamalarda sıklıkla kullanılan $\sqrt{2}$ değeri yaklaşık olarak 1.41421356237309504880168872420 97 dir.

Örnek-11.7: $\sqrt{2}$ değeri, aşağıdaki sonsuz kesir ile hesaplanabilmektedir. Klavyeden girilen bölüm sayısına göre $\sqrt{2}$ değerini hesaplayıp yazdıran program.

$$\sqrt{2}=1+\cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{2+...}}}}\quad,\quad t=2+\underbrace{\cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{2+...}}}}_{t}$$



C

```
/* Karekök(2) */
#include <stdio.h>
#include <conio.h>
void main()
{ float t=1; int i,n;
  clrscr(); printf("Bölme sayısı: "); scanf("%u",&n);
  for(i=0;i<n;i++) t=2+1/t; printf("Karekök(2)=%f\n",1+1/t);getch();}
```

C++

```
// Karekök(2)
#include <iostream>
using namespace std;
int main()
{   float t=1;int n; system("cls"); cout<<"Bölme sayısı: "; cin>>n;
    for(int i=0;i<n;i++) t=2+1/t;
    cout<<"Karekök(2)="<<1+1/t<<endl; system("pause");return 0; }
```

C#

```
// Karekök(2)
using System;
namespace ornek117
{ class Program
   { static void Main(string[] args)
      { double t=1; Console.Write("Bölme sayısı: ");
        int n=Convert.ToInt32(Console.ReadLine());
        for (int i=0;i<n;i++) t=2+1/t;
        Console.WriteLine("Karekök(2)={0}",1+1/t);
        Console.ReadLine(); }}}
```

JAVA

```
// Karekök(2)
import java.util.Scanner;
public class ornek117 {
  public static void main(String[] args) {
  double t=1; Scanner klavye=new Scanner(System.in);
  System.out.print("Bölme sayısı: "); int n=klavye.nextInt();
  for(int i=0;i<n;i++) t=2+1/t;
  System.out.printf("Karekök(2)=%f\n",1+1/t); }}
```

MATLAB

```
% Karekök(2)
clear all; clc; t=1; n=input('Bölme sayısı: ');
for i=1:n
    t=2+1/t;
end; fprintf('Karekök(2)=%f\n',1+1/t);
```

```
Bölme sayısı: 10
Karekök(2)=1.414214
```

PYTHON

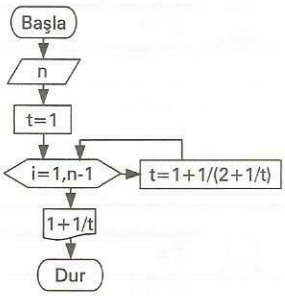
```
# Karekök(2)
t=1;n=eval(input("Bölme sayısı: "))
for i in range(n):
    t=2+1/t
print("Karekök(2)=%f"%(1+1/t))
```

IV. $\sqrt{3}$

Hesaplamalarda sıklıkla kullanılan bir diğer değer $\sqrt{3}$ olup yaklaşık olarak 1.732050807568877293527446341505 9 dir.

Örnek-11.8: $\sqrt{3}$ değeri, aşağıdaki sonsuz kesir ile hesaplanabilmektedir. Klavyeden girilen bölüm sayısına göre $\sqrt{3}$ değerini hesaplayan program.

$$\sqrt{3}=1+\cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{1+\cfrac{1}{2+\ldots}}}} \quad , \quad t=1+\cfrac{1}{2+\cfrac{1}{\underbrace{1+\cfrac{1}{2+\ldots}}_{t}}}$$

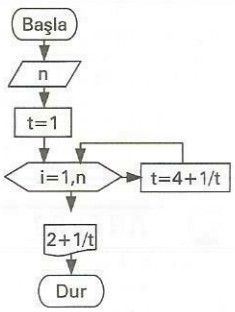


V. $\sqrt{5}$

Hesaplamalarda sıklıkla kullanılan bir diğer değer $\sqrt{5}$ olup yaklaşık olarak 2.2360679774997896964091736687313 dir.

Örnek-11.9: $\sqrt{5}$ değeri, aşağıdaki sonsuz kesir ile hesaplanabilmektedir. Klavyeden girilen bölüm sayısına göre $\sqrt{5}$ değerini hesaplayan program.

$$\sqrt{5}=2+\cfrac{1}{4+\cfrac{1}{4+\cfrac{1}{4+\cfrac{1}{4+\ldots}}}} \quad , \quad t=4+\cfrac{1}{\underbrace{4+\cfrac{1}{4+\cfrac{1}{4+\ldots}}}_{t}}$$



VI. ALTIN ORAN

"*Altın oran*", $\varphi=\dfrac{1+\sqrt{5}}{2}\cong 1.6180339887498948482 1$ şeklindedir.

Bilgi

Altın oran;

- Kendisinden 1 çıkarıldığında, çarpmaya göre kendi tersini veren tek sayıdır.
- Kendisine 1 eklendiğinde, kendi karesini veren tek sayıdır.

Yani $\varphi-1=\dfrac{1}{\varphi}$ dir. "Altın oran", farklı şekillerde hesaplanabilmektedir. Bunlardan bazıları aşağıda verilemektedir:

$$\varphi=\frac{1+\sqrt{5}}{2} \quad , \quad \varphi=1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\ldots}}}} \quad , \quad \varphi=\frac{13}{8}+\sum_{k=0}^{\infty}\frac{(-1)^{k+1}(2k+1)!}{(k+2)!k!4^{2k+3}}$$

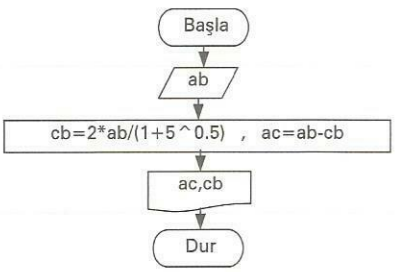
$$\varphi=1+2Sin(\pi/10),\ \varphi=2Cos(\pi/5)=2Cos(36^{\circ}),\ \varphi=2Sin(3\pi/10)=2Sin(54^{\circ})$$

Örnek-11.10: Klavyeden girilen $\overline{AB}$ doğru parçası uzunluğunu, "*altın oran*"a uygun bölerek $\overline{AC}$ ve $\overline{CB}$ uzunluklarını yazdıran program.

Herhangi bir $\overline{AB}$ doğru parçasının "altın oran"a uygun olarak bölünmesi gerektiğinde öyle bir C noktası bulunmalı ki $\frac{\overline{CB}}{\overline{AC}} = \frac{\overline{AB}}{\overline{CB}}$ eşitliği sağlanmalıdır. Yani büyük parçanın, küçük parçaya oranı; tüm doğrunun, büyük parçaya oranına eşit olması gerekir.

$$\frac{\overline{CB}}{\overline{AC}} = \frac{\overline{AB}}{\overline{CB}}$$

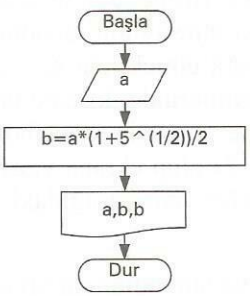
Klavyeden $\overline{AB}$ doğru parçası uzunluğunu girildiğinden $\frac{\overline{CB}}{\overline{AC}} = \frac{\overline{AB}}{\overline{CB}} = \frac{1+\sqrt{5}}{2}$ eşitliği kullanılarak $\overline{CB} = \frac{2.\overline{AB}}{1+\sqrt{5}}$ ve $\overline{AC} = \overline{AB} - \overline{CB}$ bulunabilir.



VII. ALTIN ÜÇGEN

Uzun kenarları eşit ve bu kenarlardan birinin, kısa kenara oranı; altın orana eşit olan ikizkenar üçgenlere "*altın üçgen*" denir.

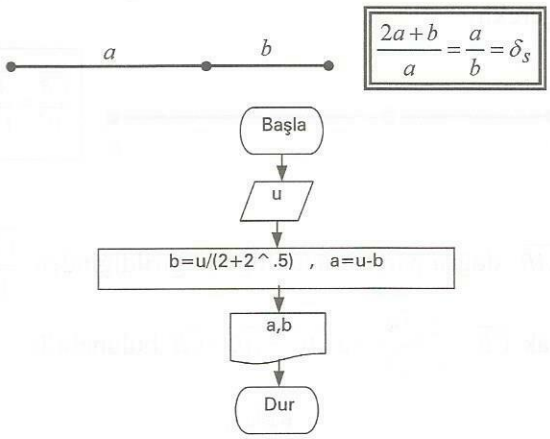
Örnek-11.11: Klavyeden kısa kenarı girilen altın üçgenin tüm kenarlarını yazdıran program.



VIII. GÜMÜŞ ORAN

$(\delta_s - 1)^2 = 2$ denkleminin çözümü olan "*gümüş oran*", $\delta_s = 1 + \sqrt{2} \cong 2.41421356\,2373095$ şeklindedir.

Örnek-11.12: Klavyeden girilen doğru parçası uzunluğunu, "gümüş oran"a uygun bölerek yazdıran program.



$$\delta_s = 2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \ldots}}}$$

IX. ASAL SAYI

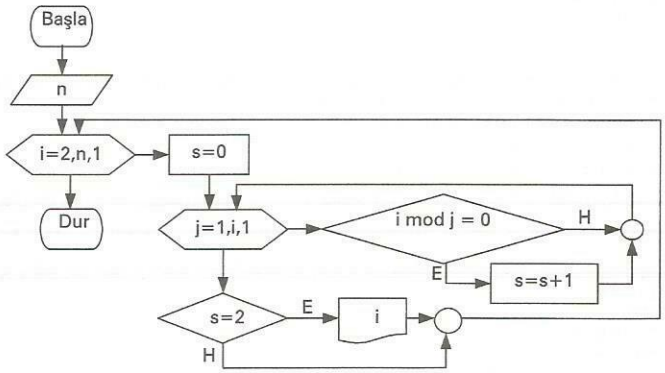
Bir ve kendisinden başka tamsayıya tam bölünemeyen tamsayılara, "*asal sayı*" denir.

Örnek-11.13: Klavyeden girilen üst sınıra kadar olan asal sayıları listeleyen program.

Programda iki döngü açılır. Birinci döngü, 2'den klavyeden girilen üst sınıra kadardır. Birinci döngüdeki tamsayının, kendinden küçük hangi tamsayılara tam olarak bölünebildiğini tespit etmek için de 1'den o sayıya kadar ikinci (iç) döngü açılır. Eğer döngü çevrimlerinde iki tane tam bölme işlemi gerçekleşirse (sayının 1'e tam bölünmesi ve kendisine tam bölünmesi) birinci döngüdeki o anki sayı, asal sayı (Tablo 11.2) olup ekrana yazdırılır. Tüm tamsayılar bire ve kendilerine tam bölünebildikleri için aşağıdaki değişikliklerle, programdaki işlem sayısı azaltılabilir:

- İç döngü 2'den başlatılır ve tam bölünme sayısı 1'e,

- İç döngü 2'den başlatılır, dış döngü de sayının bir eksiğine kadar açılır ve tam bölünme sayısının 0'a eşit olup olmadığı kontrol edilir.



Tablo 11.2: Örnek 11.13 için işlem adımları

Dış döngü	İç döngü	Bölüm	Tam bölünme	Tam bölünme sayısı	Asal sayı mı?
2	1	2	*Var*	2	Evet
2	2	1	*Var*		
3	1	3	*Var*	2	Evet
3	2	1,5	Yok		
3	3	1	*Var*		
4	1	4	*Var*	3	Hayır
4	2	2	*Var*		
4	3	1,333333	Yok		
4	4	1	*Var*		
5	1	5	*Var*	2	Evet
5	2	2,5	Yok		
5	3	1,666667	Yok		
5	4	1,25	Yok		
5	5	1	*Var*		

C

```
/* Asal sayılar */
#include <stdio.h>
#include <conio.h>
void main()
{ int i,j,n,s;
  clrscr();printf("Belirtilen aralıktaki ASAL sayılar\n");
  printf("\nÜst sınır => "); scanf("%d",&n);
  printf("\n+++ ASAL SAYILAR +++\n\n");
  for(i=2;i<=n;i++)
   { s=0; for(j=1;j<=i;j++) if (i%j==0) s++;
     if (s==2) printf("%d\t",i); }          getch(); }
```

```
Belirtilen aralıktaki ASAL sayılar

Üst sınır => 30

+++ ASAL SAYILAR +++

2 3 5 7 11 13 17 19 23 29
```

C++

```
// Asal sayılar
#include <iostream>
using namespace std;
int n,s;
int main()
{ system("cls"); cout << "Belirtilen araliktaki ASAL sayilar\n";
  cout << "\nUst sinir => "; cin >> n;
  cout << "\n+++ ASAL SAYILAR +++\n" << endl;
  for (int i=2;i<=N;i++)
   { s=0; for(int j=1;j<=i;j++) if (i%j==0) s++;
     if (s==2) cout << i << "\t"; }      system("pause");return 0; }
```

C#

```
// Asal sayıları
using System;
namespace ornek1113
{ class Program
   { static void Main(string[] args)
      { int s;
        Console.WriteLine("Belirtilen araliktaki ASAL sayilar");
        Console.Write("\nUst sinir => ");
        int n=Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("\n+++ ASAL SAYILAR +++\n\n");
        for(int i=2;i<=n;i++)
         {  s=0; for(int j=1;j<=i;j++) if (i%j==0) s++;
            if (s==2) Console.Write("{0}\t",i);} Console.ReadLine();}}}
```

JAVA

```
// Asal sayılar
import java.util.Scanner;
public class ornek1113 {
public static void main(String[] args)
{  int n,s;
   Scanner klavye=new Scanner(System.in);
   System.out.println("Belirtilen aralıktaki ASAL sayılar");
   System.out.print("\nÜst sınır => ");n=klavye.nextInt();
   System.out.println("\n+++ ASAL SAYILAR +++\n\n");
   for(int i=2;i<=n;i++)
    {s=0;for(int j=1;j<=i;j++) if (i%j==0) s++;
     if(s==2) System.out.print(i+"\t"); }}}
```

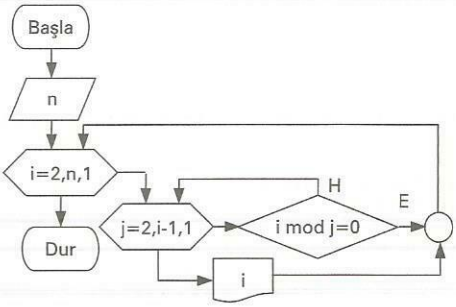
MATLAB

```
% Asal sayılar
clear all; clc ; disp('Belirtilen aralıktaki ASAL sayılar');
n=input('Üst sınır => '); fprintf('\n+++ ASAL SAYILAR +++\n\n');
fprintf('%d\t',primes(n));
```

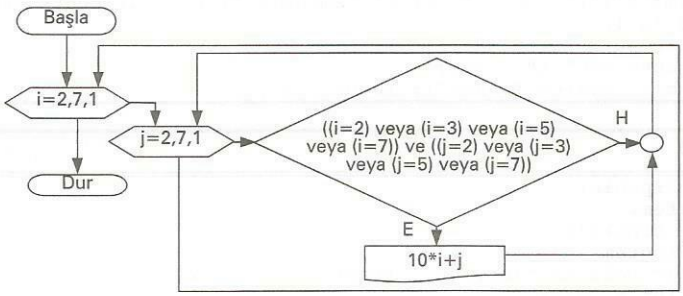
PYTHON

```
# Asal sayılar
print("Belirtilen aralıktaki ASAL sayılar\n")
n=int(input("Üst sınır: ")); print("\n+++ ASAL SAYILAR +++\n");
for i in range(2,n+1):
   s=0
   for j in range(1,i+1):
      if (i%j==0):
         s=s+1
   if (s==2):
      print("%d\t"%i,end='')
```

Alternatif bir çözümün akış diyagramı aşağıda verilmektedir.

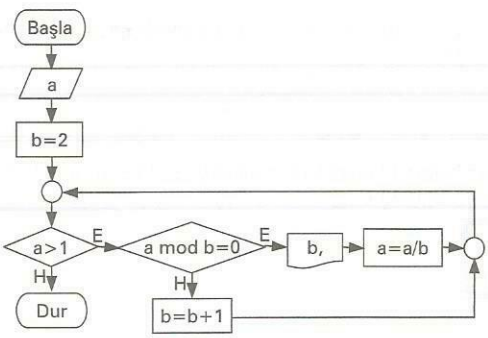


Örnek-11.14: Her iki basamağı da asal sayı olan iki basamaklı pozitif tamsayıları listeleyen program.



Örnek-11.15: Klavyeden girilen 1'den büyük tamsayının "*asal çarpanlar*"ını bulan program.

Problemin çözümünde sayı ilk önce 2'ye bölünmeye çalışılmaktadır. Eğer bölünüyorsa 2 sayısı bir asal çarpandır ve ekrana yazdırılır. Yeni sayı olarak 2'ye bölünmüş hali alınarak aynı işlemler tekrar eder. Eğer 2'ye bölünmezse 3'e (2+1) bölünme durumu, yani 3 asal çarpanının olup olmadığı 2'deki gibi denetlenir. Sonraki aşamalarda 5, 7, 11, 13 gibi asal çarpanlar sırasıyla aranacaktır. 2'ye bölünme kontrol edildiği için 4, 6, 8, 10, 12 gibi çift sayılar; 3'e bölünme kontrol edildiğinden 6, 9, 12 gibi sayılar; 5'ye bölünme kontrol edildiğinden 10, 15, 20 gibi sayılar vb. çarpanlar listesinde yer almayacaktır.



C

```
/* Asal carpanlar */
#include <stdio.h>
#include <conio.h>
void main()
{ int a,b=2;
  clrscr();printf("Pozitif tamsayi: "); scanf("%u",&a);printf("\n");
  while (a>1)
    if (a%b==0)
      { printf("%u\t",b); a/=b; }
    else b++;   getch(); }
```

C++

```
// Asal carpanlar
#include <iostream>
using namespace std;
int main()
{   int a,b=2;
    system("cls"); cout<<"Pozitif tamsayi: "; cin>>a; cout<<endl;
    while (a>1)
    if (a%b==0)
      { cout<<b<<"\t"; a/=b; }
    else b++; system("pause"); return 0; }
```

C#

```
// Asal carpanlar
using System;
namespace ornek1115
{ class Program
   { static void Main(string[] args)
       { int a, b=2;
         Console.Write("Pozitif tamsayi: ");
         a=Convert.ToInt32(Console.ReadLine()); Console.WriteLine();
         while (a > 1)
           if (a % b == 0) { Console.Write("{0}\t", b); a/=b; }
           else b++; Console.ReadLine();   }  }  }
```

JAVA

```
// Asal carpanlar
import java.util.Scanner;
public class ornek1115 {
public static void main(String[] args) {
  int a,b=2;
  Scanner klavye=new Scanner(System.in);
  System.out.print("Pozitif tamsayi: ");a=klavye.nextInt();
  System.out.println();
  while (a > 1)
    if (a % b == 0) { System.out.printf("%d\t", b); a /= b; }
    else b++;     }}
```

MATLAB

```
% Asal çarpanlar
clear all; clc; a=input('Pozitif tamsayı: '); disp(' ');
fprintf('%u\t',factor(a));
```

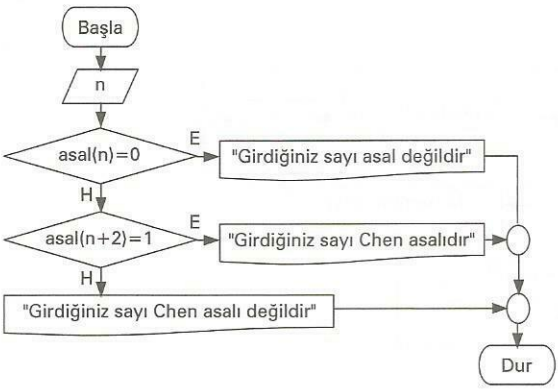
```
Pozitif tamsayı: 2015

5        13        31
```

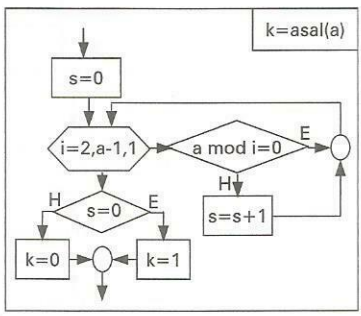
PYTHON

```
# Asal çarpanlar
a=int(input("Pozitif tamsayı: ")); b=2
while (a>1):
   if (a%b==0):
      print("%d\t"%b,end=''); a/=b
   else:
      b+=1
```

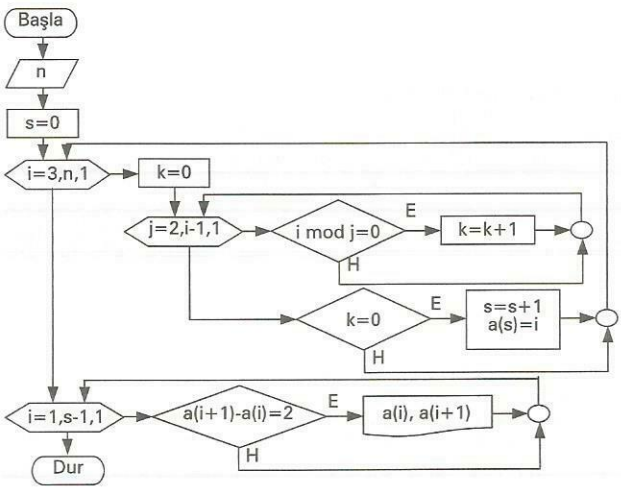
Örnek-11.16: Klavyeden girilen pozitif tamsayının "*Chen asalı*" olup olmadığını test eden program.

Eğer a asal sayısının 2 fazlası $(a + 2)$ da asal ise bu tür sayılar "*Chen asalı*" (ismi Çinli matematikçi Chen Jingrun'dan gelmektedir) olarak adlandırılmaktadır. Örneğin 3, 5, 11, 17, 29...





Örnek-11.17: İki asal sayı arasındaki fark 2 ise bunlara "*ikiz asallar*" denir. Örneğin 3-5, 5-7, 11-13, 17-19 vs. Buna göre klavyeden girilen üst sınıra kadar olan "*ikiz asallar*"ın listesini veren program.



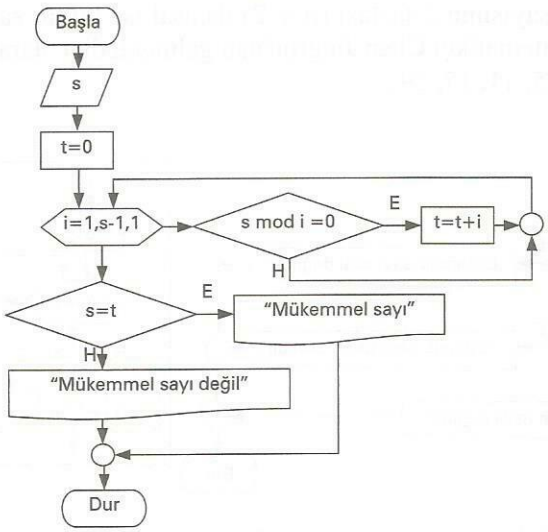
Problemin çözümünde öncelikle üst sınıra kadar olan asal sayılar bulunup bir dizide saklanır. Daha sonra dizinin ardışık elemanları arasındaki fark kontrol edilerek sonuçlar listelenir.

X. MÜKEMMEL SAYI

Kendisi hariç bütün pozitif tamsayı çarpanları (tam bölenleri) toplamı, yine kendisine eşit olan sayılara "*mükemmel sayı*" denir. Örneğin 6=1+2+3 , 28=1+2+4+7+14 birer mükemmel (kusursuz) sayıdır.

Örnek-11.18: Klavyeden girilen bir tamsayının "*mükemmel sayı*" olup olmadığını test eden program.

Tanıma uygun olarak 1'den kontrol edilecek sayının bir eksiğine kadar döngü açılıp tam bölenleri toplanarak girilen sayı ile karşılaştırılır.



C

```
/* Mükemmel sayı */
#include <stdio.h>
#include <conio.h>
unsigned i,s,t=0;
void main()
{  clrscr();printf("Tamsayı: ");scanf("%u",&s);
   for (i=1;i<s;i++) if (s%i==0) t+=i;
   if (s==t) printf("Mükemmel sayı\n");
   else printf("Mükemmel sayı değil\n");        getch();   }
```

C++

```
// Mükemmel sayı
#include <iostream>
using namespace std;
unsigned s,t=0;
int main()
{   system("cls");cout << "Tamsayı: "; cin >> s;
    for(int i=1;i<s;i++) if (s%i==0) t+=i;
    if (s==t) cout << "Mükemmel sayı" << endl;
    else cout << "Mükemmel sayı değil" << endl;
    system("pause");return 0; }
```

C#

```
// Mükemmel sayı
using System;
namespace ornek1118
{ class Program
  { static void Main(string[] args)
    { int s,t=0;
      Console.Write("Tamsayı: ");s=Convert.ToInt32(Console.ReadLine());
      for(int i=1;i<s;i++) if (s%i==0) t+=i;
      if (s==t) Console.WriteLine("Mükemmel sayı");
      else Console.WriteLine("Mükemmel sayı değil");
      Console.ReadLine();          }}}
```

JAVA

```
// Mükemmel sayı
import java.util.Scanner;
public class ornek1118 {
public static void main(String[] args)
{  int s,t=0;
   Scanner klavye=new Scanner(System.in);
   System.out.print("Tamsayı: ");s=klavye.nextInt();
   for(int i=1;i<s;i++) if (s%i==0) t+=i;
   if (s==t) System.out.println("Mükemmel sayı");
   else System.out.println("Mükemmel sayı değil");  }}
```

```
Tamsayı: 8128
Mükemmel sayı
```

MATLAB

```
% Mükemmel sayı
clear all; clc; s=input('Tamsayı: '); t=0;
for i=1:s-1
    if (mod(s,i)==0) t=t+i; end; end;
if (s==t) fprintf('Mükemmel sayı\n');
else fprintf('Mükemmel sayı değil\n'); end
```

PYTHON

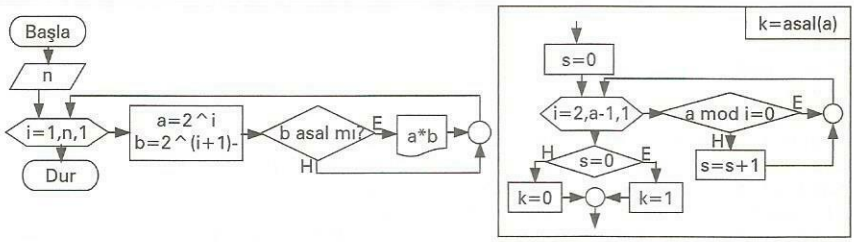
```
# Mükemmel sayı
s=int(input("Tamsayı: ")); t=0
for i in range(1,s):
   if (s%i==0):
      t+=i
if (s==t):
   print("Mükemmel sayı")
else:
   print("Mükemmel sayı değil")
```

Örnek-11.19: Klavyeden girilen pozitif tamsayı üst sınır için aşağıdaki kritere göre "*mükemmel sayı*"ların listesini veren program.

$MS = 2^n.(2^{n+1}-1)$ çarpımıyla elde edilen sayıların $(2^{n+1}-1)$ kısmı asal sayı ise, çarpım sonucu "*mükemmel sayı*"dır (Tablo 11.3).

Tablo 11.3: Örnek-11.19 için sonuçlar

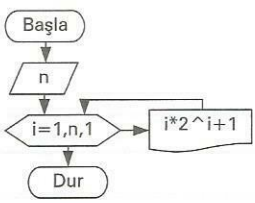
n	2^n	$2^{n+1}-1$	Asal mı?	Mükemmel Sayı
1	2	3	Evet	**6**
2	4	7	Evet	**28**
3	8	15	Hayır	120
4	16	31	Evet	**496**
5	32	63	Hayır	2016
6	64	127	Evet	**8128**
7	128	255	Hayır	32640
8	256	511	Hayır	130816
9	512	1023	Hayır	523776
10	1024	2047	Hayır	2096128



XI. CULLEN SAYISI

$n.2^n+1$ şeklinde yazılabilen doğal sayılara "*Cullen sayıları*" denir.

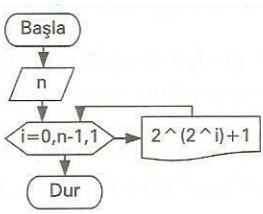
Örnek-11.20: Klavyeden girilen terim sayısı kadar Cullen sayılarını listeleyen program.



XII. FERMAT SAYISI

$2^{2^n}+1$ şeklinde yazılabilen pozitif tamsayılara "*Fermat sayıları*" denir.

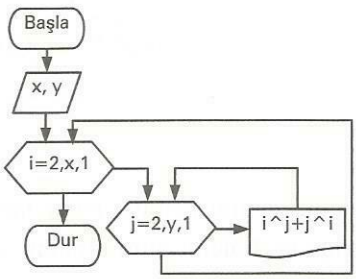
Örnek-11.21: Klavyeden girilen terim sayısı kadar Fermat sayılarını listeleyen program.



XIII. LEYLAND SAYISI

$x>1$ ve $y>1$ tamsayıları için x^y+y^x şeklinde elde edilebilen (oluşturulabilen) sayılara, "*Leyland sayıları*" denir. İlk yedi Leyland sayısı 8, 17, 32, 54, 57, 100 ve 145'tir.

Örnek-11.22: Klavyeden girilen x ve y değerleri için mümkün olan tüm Leyland sayılarını hesaplayıp ekrana yazdıran program.

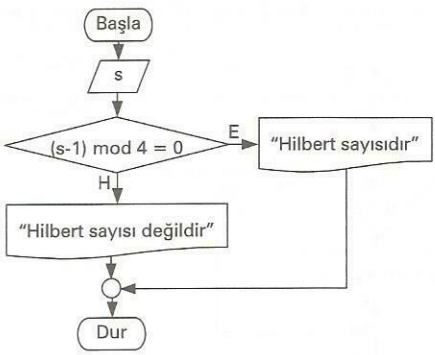


Yukarıdaki akış diyagramında mümkün olan tüm kombinasyonlara karşılık Leyland sayıları listelenmektedir. Tekrarlayan sayıları elemek için akış diygramında nasıl bir değişiklik yapılmalıdır?

XIV. HILBERT SAYISI

Sayılar teorisinde 1, 5, 9, 13, 17, 21, 25, 29, 33 gibi $4k+1$ şeklinde ifade edilebilen pozitif tamsayılar, "*Hilbert sayıları*" olarak adlandırılmaktadır.

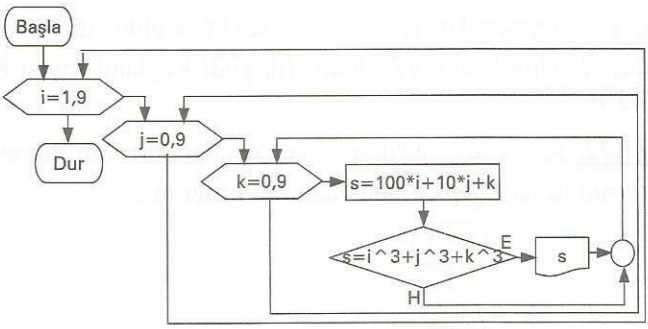
Örnek-11.23: Klavyeden girilen pozitif tamsayının, Hilbert sayısı olup olmadığını test eden program.



XV. ARMSTRONG SAYISI

Herbir hanenin, basamak sayısı kadar üslerinin toplamı yine aynı sayıya eşit olan tamsayılara, "*Armstrong sayıları*" denir. Örneğin $153=1.1.1+5.5.5+3.3.3$ (veya $153=1^3+5^3+3^3$) olduğu için Armstrong sayısı olarak adlandırılır.

Örnek-11.24: 100-999 arasındaki Armstrong sayılarını listeleyen program.

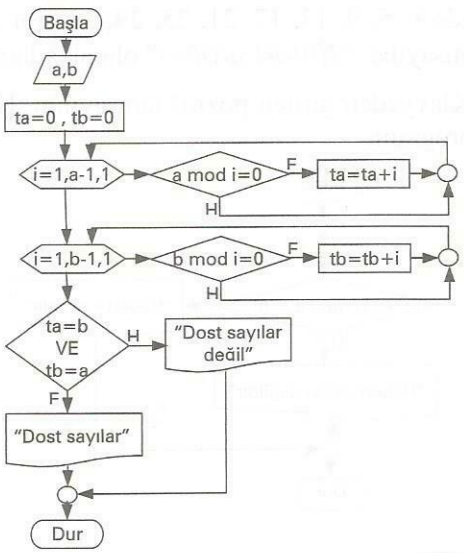


XVI. DOST SAYILAR

(m,n) tamsayı çifti için; m'nin kendisi hariç tam bölenlerinin toplamı n 'ye ve n'nin kendisi hariç tam bölenlerinin toplamı m'ye eşitse, bu iki sayıya "*dost sayılar*" denir.

Örnek-11.25: Klavyeden girilen iki tamsayının, "*dost sayılar*" olup olmadığını tespit eden program.

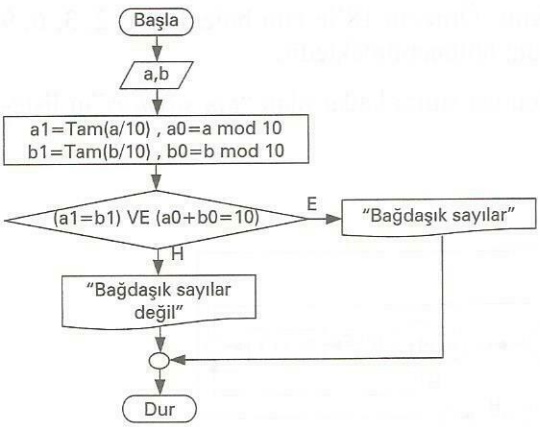
Problemin çözümü için her iki sayının, kendileri hariç tam bölenleri bulunup toplanır. Daha sonra elde edilen toplamlar, sayılarla karşılaştırılır.



XVII. BAĞDAŞIK SAYILAR

İki basamaklı iki doğal sayının birler basamağındaki rakamların toplamı 10 ve onlar basamağındaki rakamları aynı ise bu iki doğal sayıya "*bağdaşık sayılar*" denir.

Örnek-11.26: Klavyeden girilen iki doğal sayının bağdaşık olup olmadığını tespit eden program.



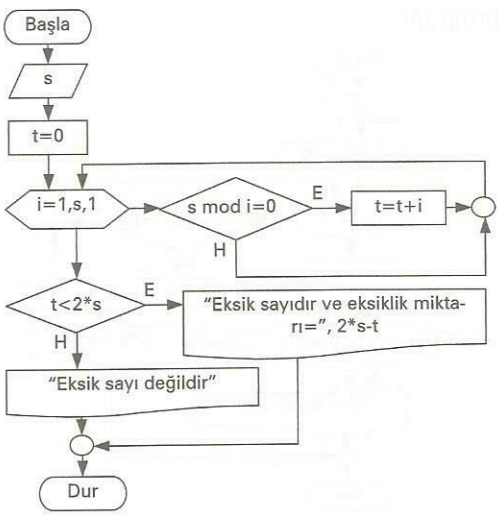
Öncelikle klavyeden girilen iki doğal sayı, basamaklarına ayrıştırılır. Onlar basamaklarının eşit ve birler basamağındakilerin toplamının da 10 olup olmadığı kontrol edilir.

XVIII. EKSİK SAYI

Tam bölenlerinin toplamı, sayının iki katından eksik ise bu tür sayılara "*eksik sayılar*" denir. Örneğin 15 tamsayısının tam bölenleri 1, 3, 5, 15 olup bunların toplamı 1+3+5+15=24 dir. Tam bölenlerin toplamı, sayının iki katından küçük olup (24<30) eksiklik miktarı 30-24=6 dır. İlk 10 eksik sayı 1, 2, 3, 4, 5, 7, 8, 9, 10, 11'dir.

Örnek-11.27: Klavyeden girilen bir tamsayının "*eksik sayı*" olup olmadığını, eksik sayı ise eksiklik miktarını ekrana yazdıran program.

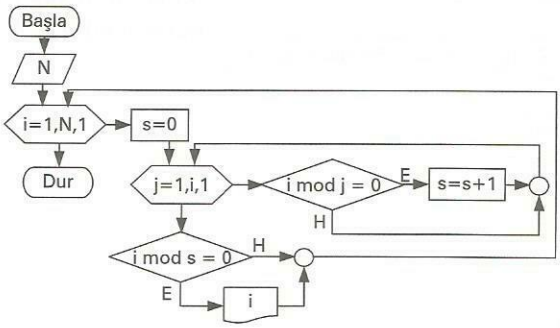
Öncelikle girilen tamsayının tam bölenleri elde edilerek toplanır ve bu toplamın, sayının iki katından küçük olup olmadığı kontrol edilir.



XIX. TAU SAYISI

1, 2, 8, 9, 12, 18, 24, 36, 40, 56 ... gibi tam bölenlerinin sayısına tam bölünebilen tamsayılara "*tau sayısı*" denir. Örneğin 18'in tam bölenleri 1, 2, 3, 6, 9 ve 18 olup altı adettir ve 18, 6'ya tam bölünebilmektedir.

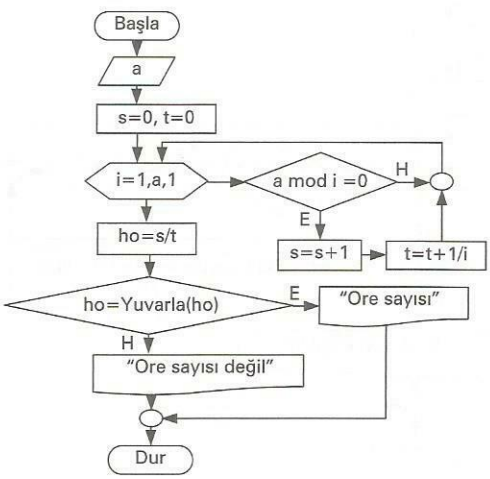
Örnek-11.28: Klavyeden girilen üst sınıra kadar olan "*tau sayıları*"nı listeleyen program.



XX. HARMONİK BÖLÜCÜ (ORE) SAYISI

1, 6, 28, 140, 270, 496 gibi tam bölenlerinin harmonik ortalaması tamsayı olan pozitif tamsayılara "*harmonik bölücü sayısı*" veya "*Ore sayısı*" denir. Örneğin 28 tamsayısının tam bölenleri 1, 2, 4, 7, 14 ve 28 olduğundan bunların harmonik ortalaması $\frac{6}{\frac{1}{1}+\frac{1}{2}+\frac{1}{4}+\frac{1}{7}+\frac{1}{14}+\frac{1}{28}}=3$ olup tamsayıdır. Bu nedenle 28 bir "Ore sayısı"dır.

Örnek-11.29: Klavyeden girilen pozitif tamsayının "*Ore sayısı*" olup olmadığını test eden program.

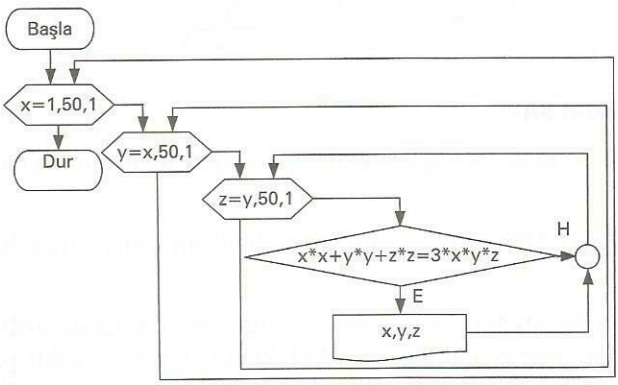


XXI. MARKOV (MARKOFF) SAYISI

"*Markov sayısı*" (Markoff sayısı) "$x^2+y^2+z^2=3xyz$" Markov denkleminin pozitif tamsayı olan x, y ve z şeklindeki çözümleridir. Örneğin (1,1,1), (1,1,2), (1,2,5), (1,5,13) vb.

Örnek-11.30: $x,y,z \in [1,50]$ için "*Markov sayıları*"nı listeleyen program.

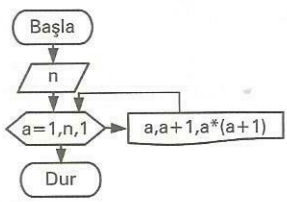
Çözüm için üç tane döngü açılarak, döngü değerlerinin Markov denklemini sağlayıp sağlamadıkları kontrol edilir. Programın aynı sayıların farklı kombinasyonlarını listelememesi için iç döngüler, dış döngüden başlatılır. Böylece (1,1,2), (1,2,1), (2,1,1) kombinasyonlarından sadece (1,1,2) listelenenmektedir.



XXII. HETEROMETRİK SAYI

a pozitif bir tamsayı olmak üzere $a(a+1)$ ile elde edilebilen sayılara "*heterometrik sayı*"lar denir. Örneğin $2 = 1.2, 6 = 2.3, 12 = 3.4$ bu sayılardan bazılarıdır.

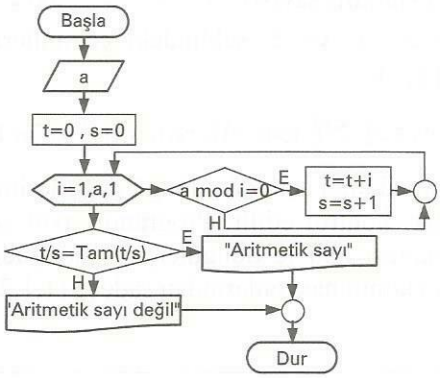
Örnek-11.31: Klavyeden girilen adet sayısınca "*heterometrik sayıları*" oluşturan ve bunları açık yazarak alt alta listeleyen program.



XXIII. ARİTMETİK SAYI

Sayılar teorisinde; herhangi bir pozitif tamsayının pozitif tam bölenlerinin ortalaması, yine tamsayı ise bunlara "*aritmetik sayı*" denir. Örneğin 6 tamsayısı $\frac{1+2+3+6}{4} = 3$ olduğundan bir aritmetik sayıdır. 1, 3, 5, 6, 7, 11... diğer aritmetik sayılardır.

Örnek-11.32: Klavyeden girilen pozitif tamsayının, "*aritmetik sayı*" olup olmadığını test eden program.

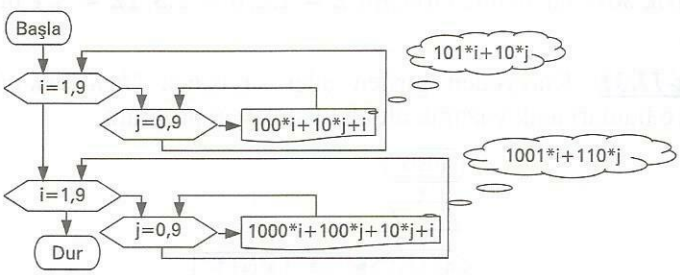


XXIV. PALINDROM SAYI

Baştan ve sondan yazılışları/okunuşları aynı olan tamsayılara "*palindrom sayılar*" denir.

Örnek-11.33: 100-9999 arasındaki palindrom tamsayıları listeleyen program.

Üç ve dört basamaklı tamsayıları oluşturmak için taban aritmetiğine dayanarak iç içe döngülerin açılması gerekmektedir. Üç basamaklı palindrom tamsayılarda yüzler ile birler basamağı aynı olacağından i ve j şeklinde iki döngü yeterlidir. Böylece $100i + 10j + i$ şeklindeki palindrom tamsayıları üretilecektir. Benzer şekilde bu döngülerle $1000i + 100j + 10j + i$ şeklindeki dört basamaklı palindromlar elde edilir.



XXV. EULERIAN SAYISI

"*Eulerian sayıları*"

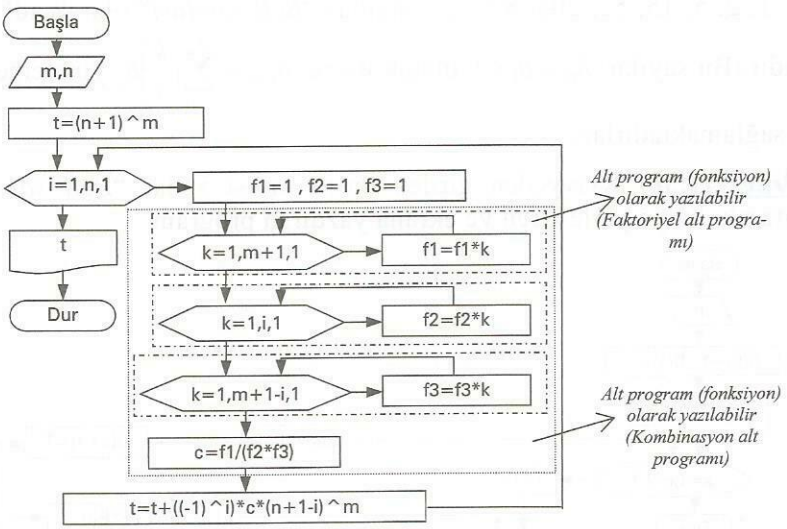
$$E(m,n) = \sum_{i=0}^{n} (-1)^i \binom{m+1}{i} (n+1-i)^m$$

ile hesaplanabilmektedir.

$m \setminus n$	0	1	2	3	4	5
0	1					
1	1					
2	1	1				
3	1	4	1			
4	1	11	11	1		
5	1	26	66	26	1	
6	1	57	302	302	57	1

Örneğin $0 \le m \le 6$ için Eulerian sayılarından oluşan "*Euler üçgeni*" yukarıdaki gibidir. Bu üçgenin her bir satırındaki elemanların toplamı $m!$ dir.

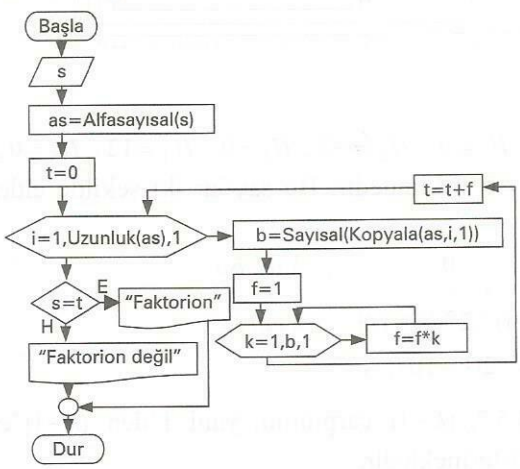
Örnek-11.34: Klavyeden girilen m ve n değerleri için $E(m,n)$ "*Eulerian sayısı*"nı hesaplayan program.



XXVI. FAKTORION

Hanelerinin faktöriyellerinin toplamı, kendisine eşit olan doğal sayılara "*faktorion*" denir. Örneğin $145 = 1!+4!+5! = 1 + 24 + 120$ olduğundan faktoriondur.

Örnek-11.35: Klavyeden girilen sıfırdan farklı pozitif bir tamsayının faktorion olup olmadığını test eden program.



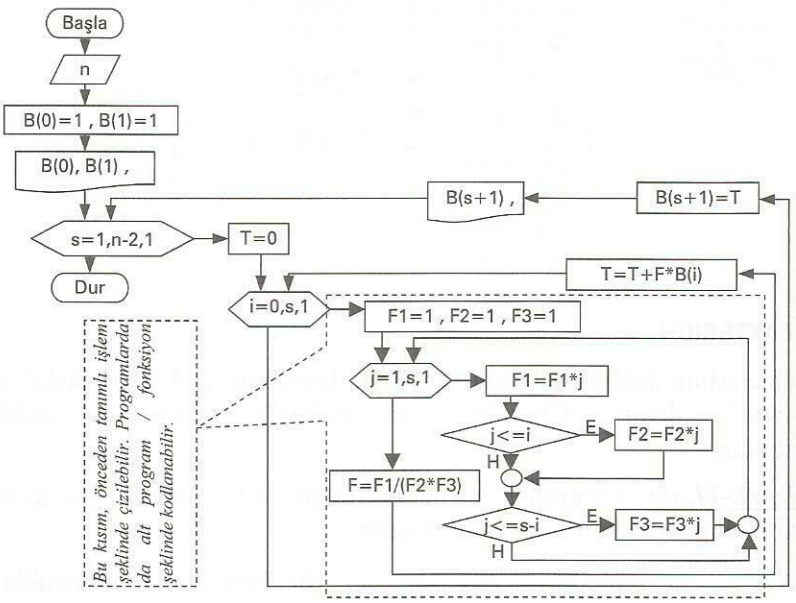
Bir tamsayının basamakları üzerinde işlem gerçekleştirmenin en kolay yollarından birisi, bir önceki örnekte olduğu gibi tamsayıyı alfasayısal veri olarak kabul edip basamaklarına dizi elemanı (kelimenin harflerine ulaşmak gibi) olarak erişmektir. Bu şekilde basamaklar ayrıştırılır ve matematiksel işlem gerçekleştirilmeden önce sayısal veriye dönüştürülür. Programlama dillerinde sayısal ile alfasayısal verileri birbirine dönüştüren

komutlar/fonksiyonlar mevcuttur. Ayrıca komut kullanmadan ASCII kodlardan da dönüşüm yapılabilir. 0'ın ASCII kodu 48 ve 9'un ASCII kodu da 57 olduğundan bunlardan 48 çıkarılırsa, ilgili tamsayı elde edilir.

XXVII. BELL SAYISI

1, 1, 2, 5, 15, 52, 203, 877, ... sayıları "*Bell sayıları*" olarak adlandırılmaktadır. Bu sayılar $B_0 = B_1 = 1$ olmak üzere $B_{k+1} = \sum_{i=0}^{k} \binom{k}{i} B_i$ yinelemeli eşitliğini sağlamaktadırlar.

Örnek-11.36: Klavyeden girilen terim sayısı kadar "*Bell sayıları*"nı oluşturarak diziye yerleştiren ve ekrana yazdıran program.



XXVIII. HERMITE SAYISI

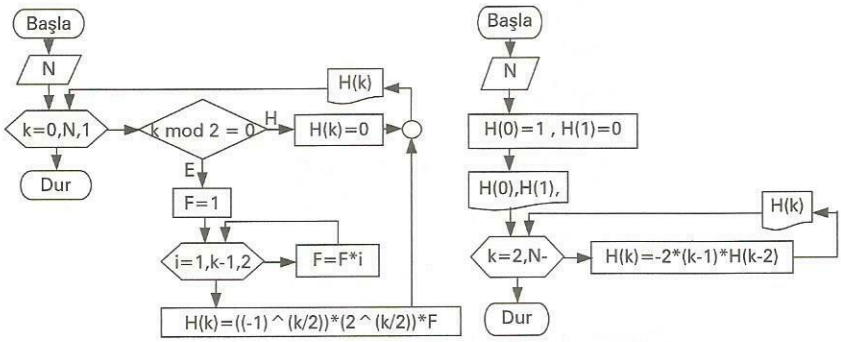
İlk 10 "*Hermite sayısı*" $H_0 = 1, H_1 = 0$, $H_2 = -2$, $H_3 = 0$, $H_4 = 12$, $H_5 = 0$, $H_6 = -120$, $H_7 = 0$, $H_8 = 1680$, $H_9 = 0$ şeklindedir. Bu sayılar iki şekilde elde edilebilmektedirler:

☑ Kapalı formül ile: $H_k = \begin{cases} 0 & , \ k \text{ tek ise} \\ (-1)^{\frac{k}{2}} . 2^{\frac{k}{2}} . (k-1)!! & . \ k \text{ çift ise} \end{cases}$

☑ Yinelemeli eşitlik ile: $H_k = -2(k-1)H_{k-2}$

Kapalı formüldeki $(k-1)!! = 1.3.5.7...(k-1)$ çarpımını, yani 1'den $(k-1)$'e kadar olan tek sayıların çarpımını göstermektedir.

Örnek-11.37: Klavyeden girilen N değeri için $0-N$ arası "*Hermite sayıları*"nı her iki yol ile hesaplayıp diziye yerleştirerek ekrana yazdıran program.



XXIX. FIBONACCI SAYISI

"*Fibonacci serisi*" elemanları (sayıları) aşağıdaki gibidir:

1 1 2 3 5 8 13 21 ...

İlk terim 0 alınarak da 0, 1, 1, 2, 3, 5, 8 ... Fibonacci serisi elde edilir.

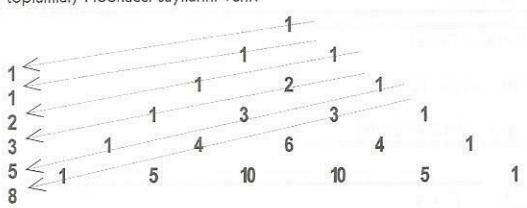
Bilgi

✓ Fibonacci serisinde; ardışık elemanlar arasındaki oran, "altın oran"a yakınsamaktadır. Yani $\lim_{k\to\infty} \frac{F_{k+1}}{F_k} = \varphi$ dir.

✓ $\varphi = \frac{1+\sqrt{5}}{2}$ ve $\psi = \frac{1-\sqrt{5}}{2}$ olmak üzere $F_k = \frac{\varphi^k - \psi^k}{\varphi - \psi} = \frac{\varphi^k - \psi^k}{\sqrt{5}}$ dir.

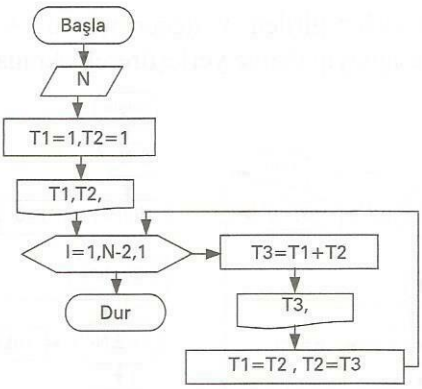
✓ $\begin{bmatrix} F_{k+2} \\ F_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F_{k+1} \\ F_k \end{bmatrix}$ matris şeklinde ifade edilebilir.

✓ Pascal üçgenindeki köşegensel toplamlar, Fibonacci sayılarını verir.



Örnek-11.38: Klavyeden girilen terim sayısı (N) için Fibonacci serisinin ilk N terimini bulup ekrana yazan program.

Fibonacci serisine dikkat edilirse; her terim, kendinden önce gelen iki terimin toplamına eşittir. Bu nedenle üç tane değişken tanımlanmalıdır. Değişken isimleri *T1*, *T2* ve *T3* olarak oluşturulursa terimler arasındaki ilişki *T3=T1+T2* şeklinde olacaktır. Üçüncü terim üretildikten sonra da, bir sonraki iterasyonda yeni *T3*'ün doğru olarak oluşabilmesi için *T1←T2* ve *T2←T3* aktarımları yapılmalıdır. $N=7$ için programın çalışması Tablo 11.4'deki gibidir.



Tablo 11.4: Örnek-11.38 için çalıştırma sonuçları

I	Terim1	Terim2	Terim3	Ekrana yazılan
---	1	1	---	1, 1
1	1	1	1+1=2	2
2	1	2	1+2=3	3
3	2	3	2+3=5	5
4	3	5	3+5=8	8
5	5	8	5+8=13	13

C

```
/* Fibonacci Serisi */
#include <stdio.h>
#include <conio.h>
unsigned N,i,T1,T2,T3;
void main()
{ clrscr();printf("Fibonacci Serisi\n\n");
  printf("Terim sayısı: ");scanf("%u",&N);
  T1=1;T2=1;printf("\n%u\t%u\t",T1,T2);
  for(i=1;i<=N-2;i++) { T3=T1+T2;printf("%u\t",T3);T1=T2;T2=T3; }
  getch();}
```

```
Fibonacci Serisi

Terim sayısı: 10

1 1 2 3 5 8 13 21 34 55
```

C++

```
// Fibonacci Serisi
#include <iostream>
using namespace std;
unsigned N,T1,T2,T3;
int main()
{   system("cls"); cout << "Fibonacci Serisi" << endl<<endl;
    cout << "Terim sayısı: "; cin >> N;
    T1=1;T2=1;cout << "\n" << T1 << "\t"  << T2 <<"\t" ;
    for(int i=1;i<=N-2;i++) { T3=T1+T2; cout<<T3<<"\t"; T1=T2; T2=T3; }
    system("pause");return 0; }
```

C#

```
// Fibonacci serisi
using System;
namespace ornek1138
{class Program
  {static void Main(string[] args)
    {   int i,N,T1=1,T2=1,T3;
        Console.WriteLine("Fibonacci Serisi\n\n");
        Console.Write("Terim sayısı: ");
        N=Convert.ToInt32(Console.ReadLine());
        Console.Write("\n{0}\t{1}\t",T1,T2);
        for(i=1;i<=N-2;i++)
         {T3=T1+T2;Console.Write("{0}\t",T3);T1=T2;T2=T3;}
        Console.ReadLine();    }    }    }
```

JAVA

```
// Fibonacci serisi
import java.util.Scanner;
public class ornek1138 {
public static void main(String[] args)
{int i,N,T1,T2,T3;
 System.out.printf("Fibonacci Serisi\n\n");
 Scanner klavye=new Scanner(System.in);
 System.out.print("Terim sayısı: ");N=klavye.nextInt();
 T1=1;T2=1;System.out.print("\n"+T1+"\t"+T2+"\t");
for(i=1;i<=N-2;i++) {T3=T1+T2;System.out.print(T3+"\t");T1=T2;T2=T3;}}}
```

MATLAB

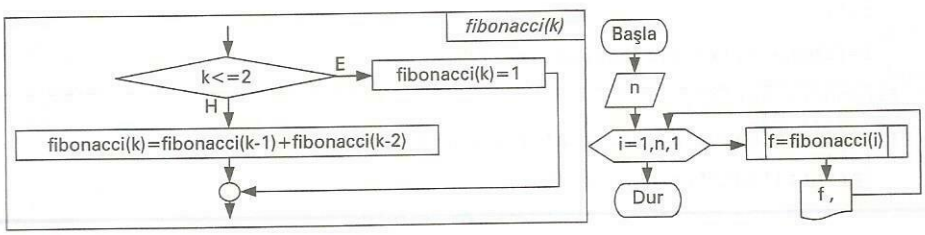
```
% Fibonacci serisi
clear all; clc
fprintf('Fibonacci Serisi\n\n'); N=input('Terim sayısı: ');
T1=1; T2=1; fprintf('\n%d\t%d\t',T1,T2);
for i=1:N-2
    T3=T1+T2; fprintf('%d\t',T3); T1=T2; T2=T3;
end
```

PYTHON

```
# Fibonacci serisi
print("Fibonacci serisi\n")
N=int(input("Terim sayısı: "))
T1=1; T2=1
print("\n%d\t%d\t"%(T1,T2),end='')
for i in range(1,N-1):
   T3=T1+T2; print("%d\t"%T3,end=''); T1=T2; T2=T3
```

Yinelemeli fonksiyon kullanılması durumunda programın akış diyagramı ve kodları, aşağıdaki gibi olacaktır.

$$fibonacci(k) = \begin{cases} fibonacci(k) = 1 & , \quad k = 1 \text{ veya } k = 2 \rightarrow \textit{Temel durum} \\ fibonacci(k-1) + fibonacci(k-2) & , \quad k > 2 \rightarrow \textit{Genel durum} \end{cases}$$



C

```
/* Fibonacci serisi */
#include <stdio.h>
#include <conio.h>
int fibonacci(int k)
{ if (k<=2) return 1; else return fibonacci(k-1)+fibonacci(k-2); }
void main()
{ int i,n; clrscr();printf("Fibonacci Serisi\n\n");
  printf("Terim sayısı: ");scanf("%u",&n);
  for(i=1;i<=n;i++) printf("%u\t",fibonacci(i));        getch();    }
```

C++

```
// Fibonacci serisi
#include <iostream>
using namespace std;
int fibonacci(int k)
{   if (k<=2) return 1; else return fibonacci(k-1)+fibonacci(k-2);   }
int main()
{   int n; system("cls"); cout << "Fibonacci Serisi" << endl<<endl;
    cout << "Terim sayısı: "; cin >> n;
    for (int i=1;i<=n;i++) cout << fibonacci(i) << "\t";
    system("pause");return 0; }
```

C#

```
// Fibonacci serisi
using System;
namespace ornek1138
{class Program
 { public static int fibonacci(int k){
       if (k<=2) return 1; else return fibonacci(k-1)+fibonacci(k-2);}
   static void Main(string[] args)
    {   Console.WriteLine("Fibonacci Serisi\n");
        Console.Write("Terim sayısı: ");
        int N=Convert.ToInt32(Console.ReadLine());
        for(int i=1;i<=N;i++) Console.Write("{0}\t",fibonacci(i));
        Console.ReadLine();     }     }     }
```

JAVA

```
// Fibonacci serisi
import java.util.Scanner;
public class ornek1138 {
public static int fibonacci(int k){
  if (k<=2) return 1; else return fibonacci(k-1)+fibonacci(k-2); }
public static void main(String[] args)
{ int n;
  System.out.printf("Fibonacci Serisi\n\n");
  Scanner klavye=new Scanner(System.in);
  System.out.print("Terim sayısı: "); n=klavye.nextInt();
  for(int i=1;i<=n;i++) System.out.printf("%d\t",fibonacci(i)); }}
```

MATLAB

```
function f=fibonacci(k)
if (k<=2)
    f=1;
else
    f=fibonacci(k-1)+fibonacci(k-2);
end
```

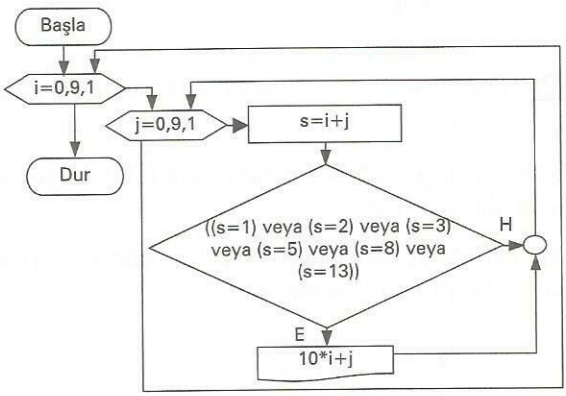
```
% Fibonacci serisi
clear all; clc
fprintf('Fibonacci Serisi\n\n'); n=input('Terim sayısı: ');
for i=1:n
    fprintf('%d\t',fibonacci(i));
end
```

PYTHON

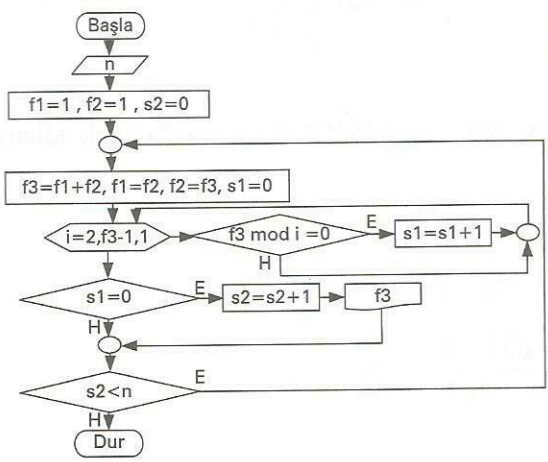
```
# Fibonacci serisi
def fibonacci(k):
    if (k<=2):
        return 1
    else:
        return fibonacci(k-1)+fibonacci(k-2)
print("Fibonacci serisi\n"); N=int(input("Terim sayısı: "))
for i in range(1,N+1):
    print("%d\t"%fibonacci(i),end='')
```

Örnek-11.39: Basamakları toplamı Fibonacci sayısı olan 0-99 arasındaki tamsayıları listeleyen program.

İki basamaklı en büyük tamsayı 99 ve basamakları toplamı da 9+9=18 olduğundan oluşabilecek Fibonacci sayıları 1, 2, 3, 5, 8, 13'tür.



Örnek-11.40: "Fibonacci sayıları" 0, 1, 1, 2, 3, 5, 8, 13, 21, 34 ... şeklindedir. Bu sayılardan 2, 3, 5, 13, 89, 233 ... olanları "*Fibonacci asal sayıları*" olarak adlandırılmaktadır. Buna göre klavyeden girilen terim sayısınca Fibonacci asallarını listeleyen program.

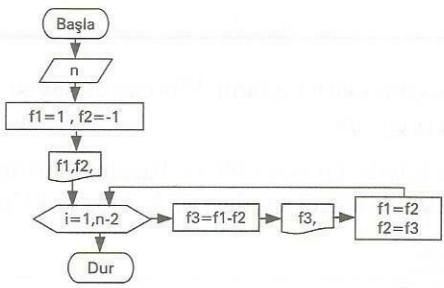


Programda Fibonacci sayılarını üreten iterasyondan sonra asal sayı kontrolü yapan çevrimin açılması gerekmektedir. Yan tarafta programın açık akış diyagramı ve buna göre kodları verilmiştir. Siz de Fibonacci sayılarını veren "fibonacci" ve ilgili sayının asal olup olmadığını denetleyen "asal" fonksiyonlarını kullanarak akış diyagramını çiziniz ve kodlayınız.

Örnek-11.41: *"NegaFibonacci sayıları"*, Fibonacci serisinin negatif indisli elemanlarıdır. NegaFibonacci sayılarının ilk sekiz tanesi

$F_{-1}=1$, $F_{-2}=-1$, $F_{-3}=2$, $F_{-4}=-3$, $F_{-5}=5$, $F_{-6}=-8$, $F_{-7}=13$, $F_{-8}=-21$

şeklindedir. Klavyeden girilen terim sayısına göre, NegaFibonacci serisinin terimlerini yazdıran program.

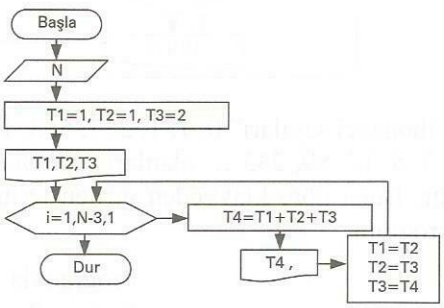


Örnek-11.42: *"Tribonacci serisi"* elemanları aşağıdaki gibidir:

1 1 2 4 7 13 24 44 .....

Buna göre klavyeden girilen terim sayısınca (N) Tribonacci serisinin elemanlarını ekrana yazdıran program.

Tribonacci Serisi'nde de Fibonacci serisine benzer şekilde her terim, kendinden önce gelen üç terimin toplamına eşittir.



XXX. PELL SAYISI

0, 1, 2, 5, 12, 29, 70, 169, 408, 985... sayıları, *"Pell sayıları"* olarak adlandırılmakta ve

☑ Yinelemeli eşitlik: $P(k)=\begin{cases} 0 & , \ k=0 \\ 1 & , \ k=1 \\ P(k-2)+2P(k-1) & , \ k>1 \end{cases}$

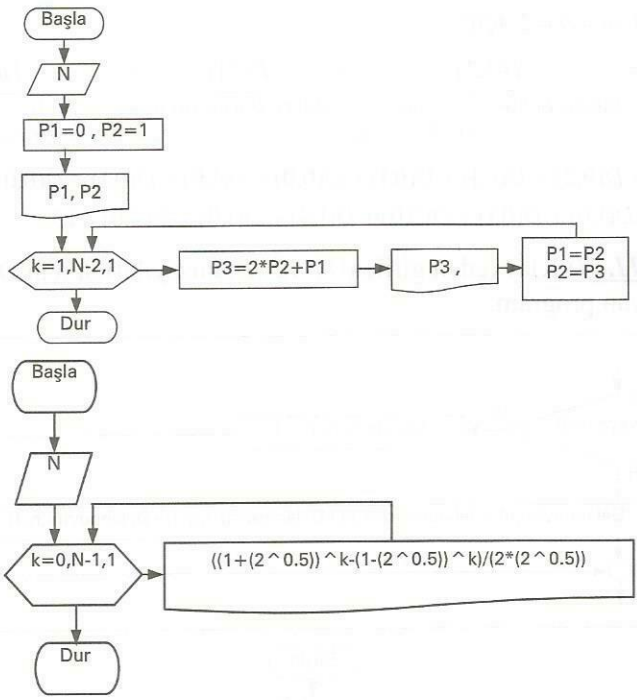
☑ Kapalı formül: $P(k)=\dfrac{\left(1+\sqrt{2}\right)^k-\left(1-\sqrt{2}\right)^k}{2\sqrt{2}}$

hesaplanabilmektedirler.

Bilgi

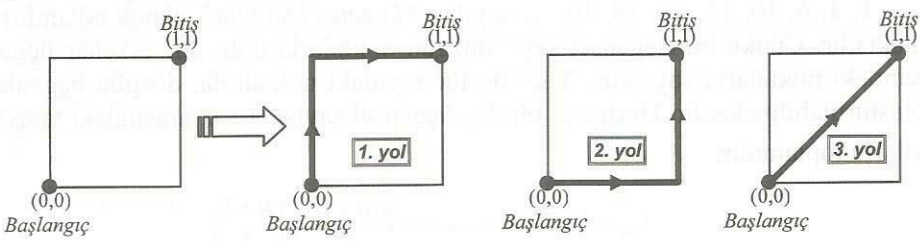
Pell serisinde ardışık iki terimin oranı, gümüş orana ($\delta_s = 1+\sqrt{2}$) yakınsamaktadır.

Örnek-11.43: Klavyeden girilen terim sayısı kadar "*Pell sayılarını*" ekrana yazdıran program.



XXXI. DELANNOY SAYISI

Matematikte "*Delannoy sayısı*"; dikdörtgen ızgaranın (0,0) olarak kabul edilen güneybatı köşesinden (başlangıç noktası) sadece kuzey/kuzeydoğu/doğu yönünde tek adımlar kullanarak (m,n) kuzeydoğu köşesine (bitiş noktası) doğru mümkün olan yolların sayısını verir. Örneğin $m = n = 1$ olması durumundaki $1x1$ tipindeki ızgarada "Delannoy sayısı" 3'tür.



$D(m,n)$ "Delannoy sayısı"

$$D(m,n)=\begin{cases}1 & , \quad m=0 \text{ veya } n=0\\ D(m-1,n)+D(m-1,n-1)+D(m,n-1) & , \quad m\neq 0 \text{ ve } n\neq 0\end{cases}$$

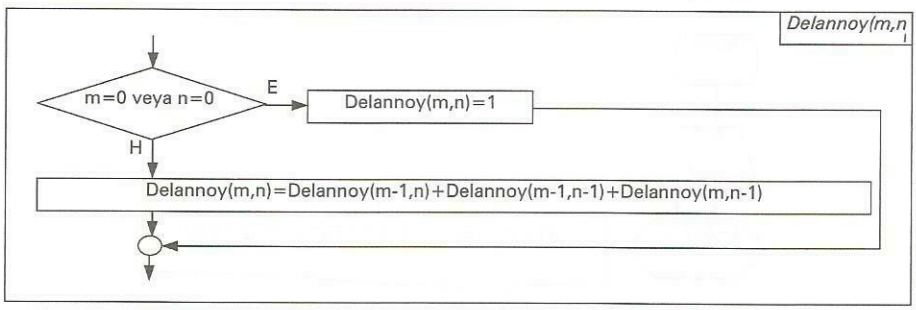
yinelemeli (rekürsif) eşitliğiyle hesaplanabilir. $m=n$ olması durumunda ilk yedi Delannoy sayıları 1, 3, 13, 63, 321, 1683, 8989 şeklindedir.

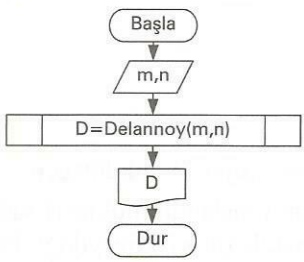
Örneğin $m=n=2$ için

$$D(2,2)=\underbrace{D(1,2)}_{D(0,2)+D(0,1)+\underbrace{D(1,1)}_{D(0,1)+D(0,0)+D(1,0)}}+\underbrace{D(1,1)}_{D(0,1)+D(0,0)+D(1,0)}+\underbrace{D(2,1)}_{\underbrace{D(1,1)}_{D(0,1)+D(0,0)+D(1,0)}+D(1,0)+D(2,0)}$$

$$D(2,2)=D(0,2)+D(0,1)+D(0,1)+D(0,0)+D(1,0)+D(0,1)+D(0,0)+$$
$$D(1,0)+D(0,1)+D(0,0)+D(1,0)+D(1,0)+D(2,0)=13$$

Örnek-11.44: Klavyeden girilen m ve n tamsayılarına için Delannoy sayısını hesaplayan program.

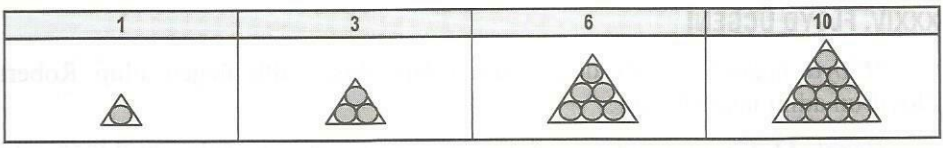




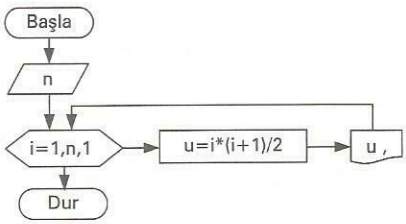
XXXII. ÜÇGENSEL SAYI

1, 3, 6, 10, 15, 21, 28, 36, ... sayıları "*üçgensel sayılar*" olarak adlandırılmaktadır. Çünkü bir üçgensel sayı; düzgün noktalarla dolu, bir eşkenar üçgen içindeki noktaların sayısıdır. Yani bu tür sayıdaki noktalarla, düzgün üçgenler oluşturulabilmektedir. Herhangi bir U_n üçgensel sayısı, $1-n$ arasındaki tamsayıların toplamıdır.

$$U_n=1+2+3+...+n=\frac{n(n+1)}{2}=\binom{n+1}{2}$$

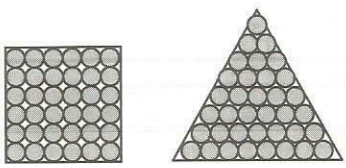


Örnek-11.45: Buna göre klavyeden girilen terim sayısı için üçgensel sayıları veren program.



XXXIII. KARE ÜÇGENSEL (ÜÇGEN KARESEL) SAYI

1, 36, 1225, 41616, 1413721... sayılarına "*kare üçgensel sayılar* (*üçgen karesel sayılar*)" denilmektedir. "Üçgensel sayılar"la düzgün üçgenler oluşturulurken, "kare üçgensel sayılar"la ise hem üçgen hem de kare oluşturulabilmektedir. Örneğin 36 sayısı ile kare ve üçgen oluşturulabilmektedir.

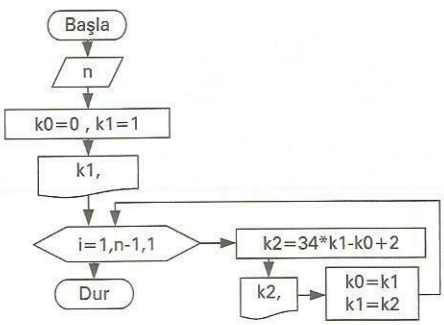


"Kare üçgensel sayılar"

$$\left.\begin{matrix} K_0 = 0 \\ K_1 = 1 \end{matrix}\right\} \Rightarrow K_i = 34K_{i-1} - K_{i-2} + 2 \text{ veya } \left.\begin{matrix} K_0 = 1 \\ K_1 = 36 \end{matrix}\right\} \Rightarrow K_i = \left(6\sqrt{K_{i-1}} - \sqrt{K_{i-2}}\right)^2$$

iterasyon eşitlikleriyle hesaplanabilmektedirler.

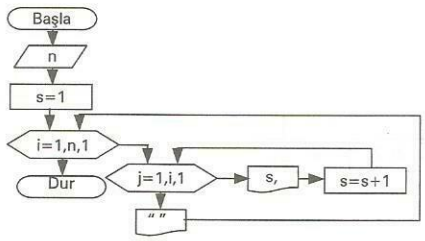
Örnek-11.46: Buna göre klavyeden girilen terim sayısı için kare üçgensel sayıları veren program.



XXXIV. FLOYD ÜÇGENİ

"*Floyd üçgeni*", sıralı doğal sayılardan oluşan dik üçgen olup Robert Floyd'un adını taşımaktadır.

Örnek-11.47: Klavyeden girilen satır sayısı için - aşağıda görüldüğü gibi elemanları arasında tab boşluğu olacak şekilde - Floyd üçgenini ekrana yazdıran program.




```
1
2  3
4  5  6
7  8  9  10
11 12 13 14 15
16 17 18 19 20 21
... ... ... ... ... ... ...
```

C

```
/* Floyd üçgeni */
#include <stdio.h>
#include <conio.h>
void main()
{  int i,j,n,s=1; clrscr();printf("Satır sayısı: ");scanf("%u",&n);
   printf("FLOYD ÜÇGENİ\n");
   for(i=1;i<=n;i++)
     { for(j=1;j<=i;j++)
        { printf("%u\t",s); s++; }
       printf("\n"); } getch();  }
```

C++

```
// Floyd üçgeni
#include <iostream>
using namespace std;
int main()
{   int n,s=1; system("cls"); cout << "Satır sayısı: "; cin >> n;
    cout << "FLOYD ÜÇGENİ\n";
    for(int i=1;i<=n;i++)
     { for(int j=1;j<=i;j++)
        { cout << s <<"\t"; s++; }
       cout <<"\n"; }  system("pause");return 0; }
```

C#

```
// Floyd üçgeni
using System;
namespace ornek1147
{ class Program
   { static void Main(string[] args)
      { int n,s=1; Console.Write("Satır sayısı: ");
        n= Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("FLOYD ÜÇGENİ");
        for (int i = 1; i <= n; i++)
          { for (int j = 1; j <= i; j++)
              { Console.Write("{0}\t",s); s++; }
            Console.WriteLine(); }    Console.ReadLine();  } } }
```

```
Satır sayısı: 5
FLOYD ÜÇGENİ
1
2   3
4   5   6
7   8   9   10
11  12  13  14  15
```

JAVA

```
// Floyd üçgeni
import java.util.Scanner;
public class ornek1147 {
 public static void main(String[] args) {
   int n,s=1; Scanner klavye=new Scanner(System.in);
   System.out.print("Satır sayısı: "); n=klavye.nextInt();
   System.out.println("FLOYD ÜÇGENİ");
   for (int i = 1; i <= n; i++)
     { for (int j = 1; j <= i; j++)
         { System.out.printf("%d\t",s); s++; }
         System.out.println(); }}}
```

MATLAB

```
% Floyd üçgeni
clear all; clc ; s=1;
n=input('Satır sayısı: ');fprintf('FLOYD ÜÇGENİ\n');
for i=1:n
    for j=1:i
        fprintf('%d\t',s);s=s+1;
    end; fprintf('\n'); end;
```

PYTHON

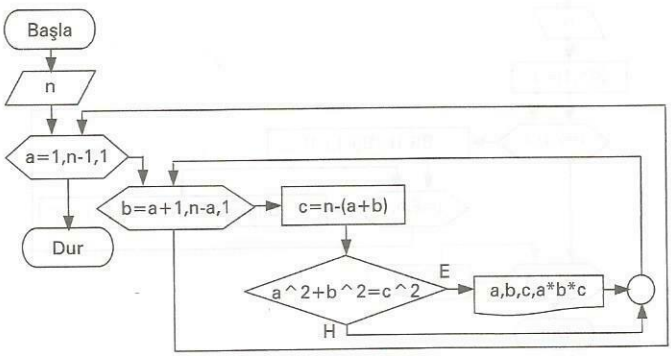
```
# Floyd üçgeni
s=1; n=int(input("Satır sayısı: ")); print("FLOYD ÜÇGENİ")
for i in range(1,n+1):
   for j in range(1,i+1):
      print("%d\t"%s,end=''); s+=1
   print()
```

XXXV. PISAGOR ÜÇLEMESİ

a, b, c, n pozitif tamsayılar olmak üzere $a < b < c$, $a^2 + b^2 = c^2$ ve $a + b + c = n$ özelliklerini (eşitliklerini) sağlayan a, b, c sayıları "*Pisagor üçlemesi*" olarak adlandırılabilmektedir.

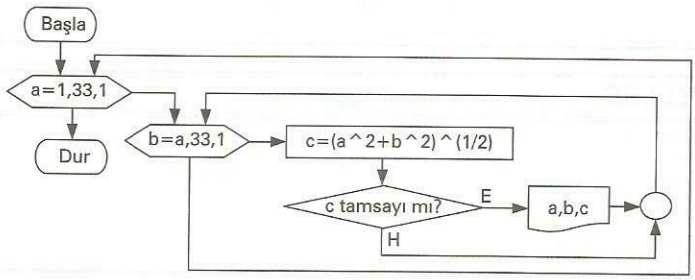
Örnek-11.48: Klavyeden girilen pozitif tamsayı değere göre "*Pisagor üçlemesi*"ni (Pythagorean triplet) hesaplayan program.

Programda klavyeden n değeri girilerek a, b, c ve $a * b * c$ sonuçları elde edilecektir.



Örnek-11.49: Tüm kenarları tamsayı ve dik kenarları 1-33 ($1 \le a,b \le 33$) arasındaki tüm $a-b-c$ "*Pisagor üçlüleri*"ni veren program.

Dik kenarları a ve b olan bir dik üçgende hipotenüs $c=\sqrt{a^2+b^2}$ dir. Dik kenar uzunluk sınırları verildiğinden, bu sınırlar arasında döngüler açılır ve hipotenüs uzunluğunun tamsayı olma koşulu kontrol edilir. Aynı dik üçgenleri oluşturmaması için de iç döngü, dış döngüden başlatılır.

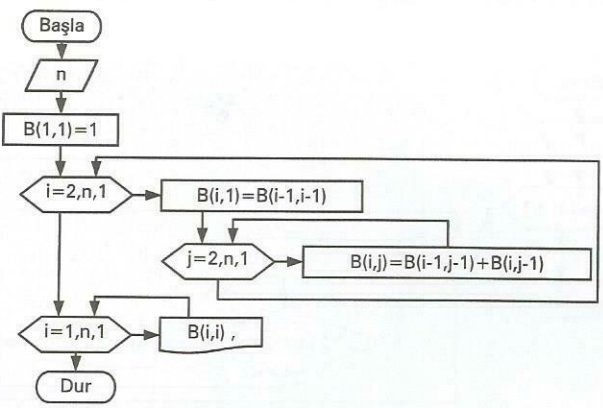


XXXVI. BELL ÜÇGENİ

"*Bell sayıları*", "*Bell üçgeni*" ("*Aitken dizisi*", "*Pierce üçgeni*") kullanılarak da elde edilebilir. Bu üçgende, bulunulan satırdaki eleman ile bir önceki satırdaki eleman karşılıklı toplanarak bulunulan satırın bir sonraki eleman olarak yazılır. Son eleman, yeni satırın ilk elemanı olarak kopyalanmakta ve böylece dik üçgenin hipotenüsünde "*Bell sayıları*" oluşmaktadır.

1					
1	**2**				
2	3	**5**			
5	7	10	**15**		
15	20	27	37	**52**	
52	67	87	114	151	**203**

Örnek-11.50: Klavyeden girilen terim sayısınca Bell sayılarını oluşturup yazdıran program.



C

```
/* Bell sayıları */
#include <stdio.h>
#include <conio.h>
void main()
{ int i,j,n; float B[30][30];
  clrscr();printf("Terim sayısı: ");scanf("%u",&n); B[0][0]=1.0;
  for(i=1;i<n;i++)
    { B[i][0]=B[i-1][i-1];
      for(j=1;j<n;j++) B[i][j]=B[i-1][j-1]+B[i][j-1]; }
  for(i=0;i<n;i++) printf("%0.0f\t",B[i][i]);   getch(); }
```

```
Terim sayısı: 10
1   2   5   15   52   203   877   4140   21147   115975
```

C++

```
// Bell sayıları
#include <iostream>
using namespace std;
int main()
{   int n; system("cls"); cout << "Terim sayisi: "; cin >> n;
    float B[n][n]; B[0][0]=1;
    for(int i=1;i<n;i++)
    { B[i][0]=B[i-1][i-1];
      for(int j=1;j<n;j++) B[i][j]=B[i-1][j-1]+B[i][j-1]; }
    for(int i=0;i<n;i++) cout << B[i][i] << "\t";
    system("pause");return 0; }
```

C#

```
// Bell sayıları
using System;
namespace ornek1150
{ class Program
   { static void Main(string[] args)
      { Console.Write("Terim sayısı: ");
        int n=Convert.ToInt32(Console.ReadLine());
        float[,] B=new float [n,n]; B[0,0]=1;
        for(int i=1;i<n;i++)
         { B[i,0]=B[i-1,i-1];
           for(int j=1;j<n;j++) B[i,j]=B[i-1,j-1]+B[i,j-1]; }
        for(int i=0;i<n;i++) Console.Write("{0:F0}\t",B[i,i]);
        Console.ReadLine();   }   }   }
```

JAVA

```
// Bell sayıları
import java.util.Scanner;
public class ornek1150 {
public static void main(String[] args)
{  int n; Scanner klavye=new Scanner(System.in);
   System.out.print("Terim sayısı: ");n=klavye.nextInt();
   float B[][]=new float [n][n]; B[0][0]=1;
   for(int i=1;i<n;i++)
    { B[i][0]=B[i-1][i-1];
      for(int j=1;j<n;j++) B[i][j]=B[i-1][j-1]+B[i][j-1]; }
    for(int i=0;i<n;i++)
        System.out.printf("%.0f\t",B[i][i]); }}
```

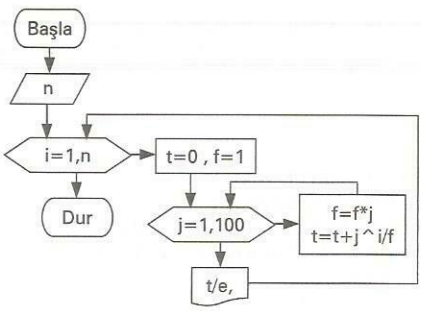
MATLAB

```
% Bell sayıları
clear all; clc; n=input('Terim sayısı: '); B(1,1)=1;
for i=2:n
    B(i,1)=B(i-1,i-1);
    for j=2:n
        B(i,j)=B(i-1,j-1)+B(i,j-1); end; end;
for i=1:n fprintf('%0.0f\t',B(i,i)); end;
```

PYTHON

```
# Bell sayıları
import numpy as np
n=int(input("Terim sayısı: "))
B=np.zeros((n,n)); B[0][0]=1;
for i in range(1,n):
   B[i][0]=B[i-1][i-1]
   for j in range(1,n):
      B[i][j]=B[i-1][j-1]+B[i][j-1]
for i in range(n):
   print("%d\t"%B[i][i],end='')
```

Örnek-11.51: Bell sayıları $B_n = \frac{1}{e}\sum_{k=1}^{\infty}\frac{k^n}{k!}$ şeklindeki "*Dobiński formülü*"yle de hesaplanabilmektedirler. Buna göre klavyeden girilen terim sayısınca Bell sayılarını 100 iterasyon ile oluşturup yazdıran program.



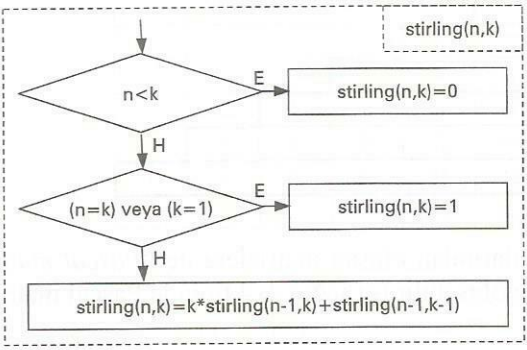
XXXVII. STIRLING ÜÇGENİ

"*Stirling sayıları*"ndan oluşan üçgen aşağıdaki gibidir.

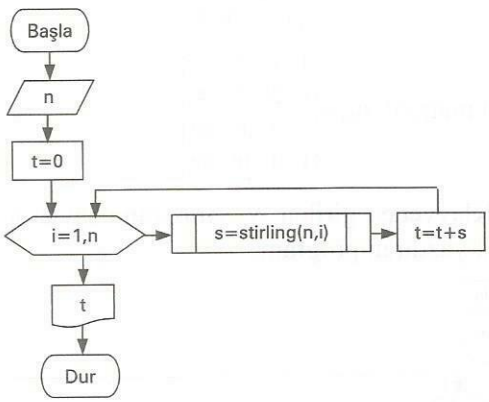
1					
1	1				
1	3	1			
1	7	6	1		
1	15	25	10	1	
1	31	90	65	15	1

Stirling sayıları $S(n,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i\binom{k}{i}(k-i)^n$ ile veya $S(n,k) = kS(n-1,k) + S(n-1,k-1)$ yinelemeli eşitliğiyle hesaplanabilmektedirler.

Örnek-11.52: Klavyeden girilen satır/sütun değerine göre ilgili Stirling sayısını hesaplayan yinelemeli alt program/fonksiyon.

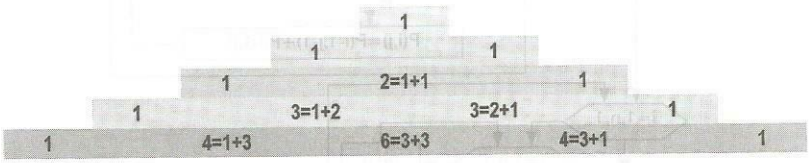


Örnek-11.53: Stirling sayıları ile Bell sayıları arasında $B_n = \sum_{k=1}^{n} S(n,k)$ ilişkisi vardır. Klavyeden girilen n değerine göre n. Bell sayısını hesaplayan program.



XXXVIII. PASCAL ÜÇGENİ

"*Pascal üçgeni*", "*Binom katsayıları*"ndan oluşan bir üçgendir. Yani bu üçgenin satırları, Binom katsayıları dizilerinden oluşur. Yan kenarları "1", taban kenarları da bir üst taban kenarın/dizinin iki elemanının toplamıdır.



Pascal üçgeni'ndeki satırlar sola hizalanırsa; elemanların köşegensel (matrislerdeki yedek köşegen) toplamı, ardışık Fibonacci sayılarını verir.

1	1	2	3	5	8	13	21
1	1						
1	2	1					
1	3	3	1				
1	4	6	4	1			
1	5	10	10	5	1		
1	6	15	20	15	6	1	
1	7	21	35	35	21	7	1
...	...	...	...	...	...	...	...

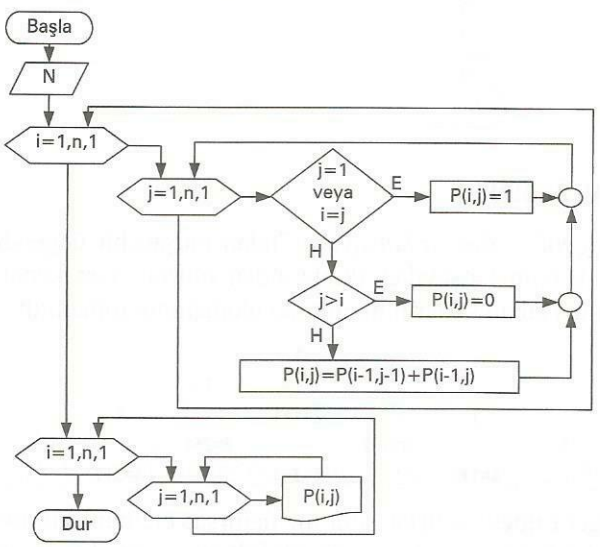
Binom katsayılarından oluşan matrislere de "*Pascal matrisi*" denir. Pascal matrisleri 3 şekilde oluşabilir (örneğin 4x4 tipinde Pascal matrisleri):

a. Üst üçgen Pascal matrisi: $P_{U4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

b. Alt üçgen Pascal matrisi: $P_{A4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 3 & 3 & 1 \end{bmatrix}$

c. Simetrik Pascal matrisi: $P_{S4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 \\ 1 & 3 & 6 & 10 \\ 1 & 4 & 10 & 20 \end{bmatrix}$

Örnek-11.54: Klavyeden girilen N satır/sütun sayısı için alt üçgen Pascal matrisini oluşturarak yazdıran program.



C

```
/* Pascal matrisi */
#include <stdio.h>
#include <conio.h>
void main()
{ int i,j,n,P[30][30];
  clrscr();printf("Satır/sütun sayısı: ");scanf("%u",&n);
  for(i=0;i<n;i++)
    { for(j=0;j<n;j++)
        if ((j==0) || (i==j)) P[i][j]=1;
        else if (j>i) P[i][j]=0;
        else P[i][j]=P[i-1][j-1]+P[i-1][j]; }
  for(i=0;i<n;i++)
   {for(j=0;j<n;j++) printf("%d\t",P[i][j]);printf("\n");} getch(); }
```

C++

```
// Pascal matrisi
#include <iostream>
using namespace std;
int main()
{   int i,j,n; system("cls"); cout << "Satir/sutun sayisi: "; cin >> n;
    int P[n][n];;
    for(i=0;i<n;i++)
    { for(j=0;j<n;j++)
        if ((j==0) || (i==j)) P[i][j]=1;
        else if (j>i) P[i][j]=0;
        else P[i][j]=P[i-1][j-1]+P[i-1][j]; }
    for(i=0;i<n;i++)
      { for(j=0;j<n;j++) cout<<P[i][j]<<"\t"; cout<<"\n"; }
    system("pause");return 0; }
```

```
Satir/sutun sayisi: 5
1   0   0   0   0
1   1   0   0   0
1   2   1   0   0
1   3   3   1   0
1   4   6   4   1
```

C#

```
// Pascal matrisi
using System;
namespace ornek1154
{ class Program
   { static void Main(string[] args)
      { Console.Write("Satır/sütun sayısı: ");
        int n=Convert.ToInt32(Console.ReadLine());
        int[,] P=new int[n,n];
        for(int i=0;i<n;i++)
         { for(int j=0;j<n;j++)
           if ((j==0) || (i==j)) P[i,j]=1;
           else if (j>i) P[i,j]=0;
           else P[i,j]=P[i-1,j-1]+P[i-1,j]; }
        for(int i=0;i<n;i++)
         { for(int j=0;j<n;j++) Console.Write("{0}\t",P[i,j]);
             Console.WriteLine();     }   Console.ReadLine(); } }   }
```

JAVA

```
// Pascal matrisi
import java.util.Scanner;
public class ornek1154 {
public static void main(String[] args)
{  int i,j,n; Scanner klavye=new Scanner(System.in);
   System.out.print("Satır/sütun sayısı: "); n=klavye.nextInt();
   int P[][]=new int [n][n];
   for(i=0;i<n;i++)
    { for(j=0;j<n;j++)
        if ((j==0) || (i==j)) P[i][j]=1;
        else if (j>i) P[i][j]=0;
        else P[i][j]=P[i-1][j-1]+P[i-1][j]; }
   for(i=0;i<n;i++)
    { for(j=0;j<n;j++) System.out.printf("%d\t",P[i][j]);
      System.out.printf("\n"); } }}
```

MATLAB

```
% Pascal matrisi
clear all; clc; n=input('Satır/sütun sayısı: ');
for i=1:n
    for j=1:n
        if ((j==1) || (i==j)) P(i,j)=1;
        elseif (j>i) P(i,j)=0;
        else P(i,j)=P(i-1,j-1)+P(i-1,j);
end;end;end; disp(P)
```

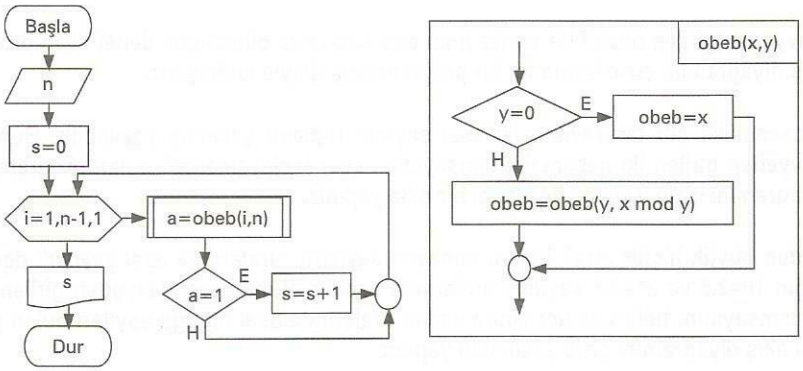
PYTHON

```
# Pascal matrisi
import numpy as np
n=int(input("Satır/sütun sayısı: ")); P=np.zeros((n,n));
for i in range(n):
   for j in range(n):
      if ((j==0) | (i==j)):
          P[i][j]=1
      elif (j>i):
          P[i][j]=0
      else:
          P[i][j]=P[i-1][j-1]+P[i-1][j]
print(P)
```

XXXIX. TOTIENT

Sayılar teorisinde "*totient*" (*Euler totient fonksiyonu*) olarak adlandırılan ve $\varphi(n)$ veya $\phi(n)$ ile gösterilen bu fonksiyon, 1763 yılında İsveçli matematikçi ve fizikçi tarafından Leonhard Euler tarafından ortaya atılmıştır. Şifreleme (kripto) uygulamalarında da sıklıkla kullanılan bu fonksiyon; herhangi bir tamsayının, kendisinden küçük ve ilgili sayı ile aralarında asal olan sayı adedini vermektedir. Örneğin 11 tamsayısının totienti $\varphi(9) = 6$ dır. Çünkü 9'dan küçük ve 1'den başka ortak böleni olmayan (aralarında asal olan) 1, 2, 4, 5, 7 ve 8 olup bunların sayısı 6 tanedir.

Örnek-11.55: Klavyeden girilen pozitif tamsayının totientini hesaplayan program.



SORULAR

1. **Klavyeden girilen pozitif bir tamsayının asal sayı olup olmadığını denetleyen programın akış diyagramını çizip herhangi bir programlama diliyle kodlayınız.**

2. **İki basamaklı çift tamsayılar, iki asal sayının toplamı şeklinde yazılabilir. Buna göre klavyeden girilen iki basamaklı tamsayının *"asal toplayanlarını"* bulan programın akış diyagramını çizip yazılımı herhangi bir dille yapınız.**

3. **Birden büyük hiçbir ortak böleni olmayan sayılara *"aralarında asal sayılar"* denir. Örneğin 10=2.5 ve 21=3.7 sayıları, aralarında asaldır. Buna göre klavyeden girilen pozitif bir tamsayının; belirtilen üst sınıra kadar, aralarında asal olduğu sayıları bulan programın akış diyagramını çizip yazılımını yapınız.**

4. **π sayısı değişik şekillerde hesaplanabilir. Bunlardan bazıları aşağıdadır. Buna göre klavyeden girilen terim sayısı için π değerini hesaplayan programların akış diyagramlarını çizip kodlayınız.**

Yöntem	İfade
Leibnitz-Gregory-Madhava serisi	$\pi = 4\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1} = 4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots\right)$
Bailey-Borwein-Plouffle formülü	$\pi = \sum_{k=0}^{\infty}\left(\frac{4}{8k+1}-\frac{2}{8k+4}-\frac{1}{8k+5}-\frac{1}{8k+6}\right)\left(\frac{1}{16}\right)^k$
Bellard formülü	$\pi = \frac{1}{2^6}\sum_{k=0}^{\infty}\frac{(-1)^k}{2^{10k}}\left(-\frac{2^5}{4k+1}-\frac{1}{4k+3}+\frac{2^8}{10k+1}-\frac{2^6}{10k+3}-\frac{2^2}{10k+5}-\frac{2^2}{10k+7}+\frac{1}{10k+9}\right)$
Chudnovsky algoritması	$\frac{1}{\pi} = 12\sum_{k=0}^{\infty}\frac{(-1)^k(6k)!(13591409+545140134k)}{(3k)!(k!)^3 640320^{3k+3/2}}$
Wallis çarpımı	$\frac{\pi}{2} = \prod_{k=1}^{\infty}\left(\frac{2k}{2k-1}\cdot\frac{2k}{2k+1}\right) = \frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdots$
Aritmetik-Geometrik ortalama	✓ $a_0 = 1,\ b_0 = \frac{1}{\sqrt{2}},\ t_0 = \frac{1}{4},\ p_0 = 1$ ✓ $a_{k+1} = \frac{a_k+b_k}{2},\ b_{k+1} = \sqrt{a_k b_k},\ t_{k+1} = t_k - p_k(a_k-a_{k+1})^2,\ p_{k+1} = 2p_k$ ✓ $\pi \cong \frac{(a_k+b_k)^2}{4t_k}$

François Viète	$\frac{2}{\pi}=\frac{\sqrt{2}}{2}.\frac{\sqrt{2+\sqrt{2}}}{2}.\frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2}...$
William Brouncker	$\frac{4}{\pi}=1+\cfrac{1^2}{2+\cfrac{3^2}{2+\cfrac{5^2}{2+\cfrac{7^2}{2+...}}}}$
Isaac Newton	$\frac{\pi}{6}=\frac{1}{2}+\frac{1}{2}.\frac{1}{3.2^3}+\frac{1}{2}.\frac{3}{4}.\frac{1}{5.2^5}+\frac{1}{2}.\frac{3}{4}.\frac{5}{6}.\frac{1}{7.2^7}+...$
Nilakantha Somayaji	$\frac{\pi-3}{4}=\frac{1}{2.3.4}-\frac{1}{4.5.6}+\frac{1}{6.7.8}-...$
Euler	$\frac{\pi}{2}=1+\frac{1}{3}+\frac{1.2}{3.5}+\frac{1.2.3}{3.5.7}+\frac{1.2.3.4}{3.5.7.9}+...$
Euler	$\frac{\pi^2}{6}=1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+...$
Euler	$\frac{\pi^3}{32}=1-\frac{1}{3^3}+\frac{1}{5^3}-\frac{1}{7^3}+...$
Euler	$\frac{\pi^4}{90}=1+\frac{1}{2^4}+\frac{1}{3^4}+\frac{1}{4^4}+...$
Euler	$\frac{\pi}{4}=\frac{3}{4}.\frac{5}{4}.\frac{7}{8}.\frac{11}{12}.\frac{13}{12}.\frac{17}{16}....$
Euler	$\frac{\pi^2}{6}=\frac{2^2}{2^2-1}.\frac{3^2}{3^2-1}.\frac{5^2}{5^2-1}....$
	$\frac{\pi\sqrt{2}}{4}=1+\frac{1}{3}-\frac{1}{5}-\frac{1}{7}+\frac{1}{9}+...$
	$\frac{\pi^2}{8}=1+\frac{1}{3^2}+\frac{1}{5^2}+\frac{1}{7^2}+...$
	$\frac{\pi^2}{24}=\frac{1}{2^2}+\frac{1}{4^2}+\frac{1}{6^2}+...$
	$\frac{2}{\pi}=\left(1-\frac{1}{2^2}\right)\left(1-\frac{1}{4^2}\right)\left(1-\frac{1}{6^2}\right)...$

5. Aşağıdaki değerleri, ilgili açılımlarla/kesirlerle hesaplayan programların akış diyagramlarını çizip yazılımını yapınız.

$\sqrt{2} = \left(1+\frac{1}{1}\right)\left(1-\frac{1}{3}\right)\left(1+\frac{1}{5}\right)\ldots$
$\sqrt{2} = 1+\frac{1}{2}-\frac{1}{2.4}+\frac{1.3}{2.4.6}\ldots$
$\sqrt{2} = \frac{1}{2}+\frac{3}{8}+\frac{15}{64}+\frac{35}{256}\ldots$
$\varphi = 1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\ldots}}}}$ *Altın oran:* $\varphi=\frac{1+\sqrt{5}}{2}$
$\delta_S = 2+\cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{2+\cfrac{1}{2+\ldots}}}}$ *Gümüş oran:* $\delta_S = 1+\sqrt{2}$

6. *"Tetranacci Serisi"* elemanları 1, 1, 2, 4, 8, 15, 29, 56, 108, ... şeklindedir. Buna göre klavyeden girilen terim sayısı kadar *Tetranacci Serisi*'nin elemanlarını hesaplayıp yazdıran programın akış diyagramını çizip kodlayınız.

7. 2, 1, 3, 4, 7, 11, 18, 29, 47, 76, 123 ... sayıları, *"Lucas serisi"*nin elemanları olup *"Lucas sayıları"* olarak adlandırılmaktadır. Lucas sayıları matematiksel olarak

$$L(k) = \begin{cases} 2 & , \ k = 0 \\ 1 & , \ k = 1 \\ L(k-2)+L(k-1) & , \ k > 1 \end{cases}$$

şeklinde tanımlanabilir. Fibonacci serisinde olduğu gibi Lucas serisinde de her eleman, kendinden önceki iki elemanın toplamından ibarettir. Başlangıç değerleri farklı olduğundan dolayı, serilerin elemanları farklı oluşmaktadır. Ayrıca Lucas sayıları; $\varphi = \frac{1+\sqrt{5}}{2}$ altın oran olmak üzere $L(k) = \varphi^k + (1-\varphi)^k = \left(\frac{1+\sqrt{5}}{2}\right)^k + \left(\frac{1-\sqrt{5}}{2}\right)^k$ kapalı formülü ile de hesaplanabilir. Buna göre klavyeden girilen terim sayısı kadar Lucas sayılarını hem yinelemeli eşitlikle hem de kapalı formülle oluşturup ekrana yazdıran programın akış diyagramlarını ayrı ayrı çiziniz ve herhangi bir programlama diliyle kodlayınız.

8. Klavyeden girilen tamsayı üst sınırına kadar olan mükemmel sayıların listesini veren programın akış diyagramını çizip kodlayınız.

9. Klavyeden girilen terim sayısına göre "*Cullen asal sayıları*"nı listeleyen programın akış diyagramını çizip kodlayınız.

10. $n.2^n - 1$ şeklinde yazılabilen doğal sayılara "*Woodall sayıları*" (Cullen sayılarının ikinci türü) denir. Buna göre, klavyeden girilen terim sayısı için "*Woodall sayıları*"nı listeleyen programın akış diyagramını çizip kodlayınız.

11. k tek ve n pozitif tamsayı olmak üzere $k < 2^n$ koşulu altında $k.2^n + 1$ şeklinde yazılabilen sayılara "*Proth sayıları*" denir. Buna göre, klavyeden girilen terim sayısınca "*Proth sayıları*" listeleyen programın akış diyagramını çizip kodlayınız.

12. 2'in kuvvetinden 1 eksik olan $(2^n - 1)$ pozitif tamsayılara, "*Mersenne sayıları*" denir. Buna göre, klavyeden girilen terim sayısınca "*Mersenne sayıları*" listeleyen programın akış diyagramını çizip kodlayınız.

13. Haneleri toplamına tam bölünebilen tamsayılara "*Harshad sayıları*" denir. Buna göre 1-999 arasındaki "*Harshad sayıları*"nı listeleyen programın akış diyagramını çizip kodlayınız.

14. Hanelerine dört temel matematiksel işlem (toplama, çıkarma, çarpma ve bölme) ve bazen de üs alma işlemleri uygulanarak aynı tamsayı elde edilebiliyorsa, bu tür tamsayılara "*Friedman sayıları*" denir. Örneğin $347=7^3+4$, $25=5^2$ gibi... Buna göre klavyeden girilen tamsayının, "*Friedman sayısı*" olup olmadığını test eden programın akış diyagramını çizip kodlayınız.

15. $\sqrt{2}$ sayısı $a_0 = 1$ başlangıç değeri kullanılarak $a_{k+1} = \frac{a_k}{2} + \frac{1}{a_k}$ iterasyonuyla hesaplanabilir. Buna göre klavyeden girilen iterasyon sayısına göre $\sqrt{2}$ sayısını hesaplayıp yazdıran programın akış diyagramını çizip kodlayınız.

16. Bir dairedeki "*altın açı*" değeri derece cinsinden $\frac{360}{\varphi^2}$ ve radyan cinsinden $\frac{2\pi}{\varphi^2}$ ile hesaplanır. Buna göre bir daire için "*altın açı*" değerini hesaplayan programın akış diyagramını çizip kodlayınız.

17. Klavyeden kısa kenarı girilen ve aşağıda tanımlanan üçgenlerin diğer kenarlarını hesaplayıp yazdıran programların akış diyagramlarını çizip kodlayınız.

Üçgen adı	Birim kenar oranları
$45^\circ - 45^\circ - 90^\circ$	$1:1:\sqrt{2}$
$30^\circ - 60^\circ - 90^\circ$	$1:\sqrt{3}:2$
$3-4-5$	$3:4:5$
"Kepler üçgeni"	$1:\sqrt{\varphi}:\varphi$

18. $4^n - 2^{n+1} - 1$ şeklinde veya eşdeğeri olan $\left(2^n - 1\right)^2 - 2$ şeklinde yazılabilen/oluşturulabilen tamsayılara, "*Carol sayıları*" denir. -1, 7, 47, 223, 959, 3967, 16127, ... Carol sayılarının bazılarıdır. Buna göre klavyeden girilen terim sayısı kadar Carol sayılarını oluşturup ekrana yazdıran programın akış diyagramını çizip kodlayınız.

19. "*Worpitzky sayıları*" $W_{n,k} = \sum_{i=0}^{k} (-1)^{i+k} . (i+1)^n \binom{k}{i}$ ile verilmektedir. Buna göre klavyeden girilen n ve k değeri için ilgili $W_{n,k}$ Worpitzky sayısını hesaplayıp ekrana yazdıran programın akış diyagramını çizerek kodlayınız.

20. $n \geq 0$ için $C_n = \frac{(2n)!}{n!(n+1)!}$ şeklindeki sayılara, n. "*Catalan sayıları*" denir. Örneğin 1, 2, 5, 14, 42, 132, 429, ... bu sayılardan bazılarıdır. Klavyeden girilen n değeri için C_n Catalan sayısını hesaplayıp ekrana yazdıran programın akış diyagramını çiziniz ve herhangi bir programlama diliyle kodlayınız.

21. 1'den ve kendisinden başka pozitif böleni olan pozitif tamsayılara "*bileşik sayı*" denir. Bu tür sayılardan bazıları 4, 6, 8, 9, 10, 12, 14, 15, 16, 18, 20, 21, 22, 24, 25 ... dir. Buna göre klavyeden girilen pozitif tamsayının "bileşik sayı" olup olmadığını test eden programın akış diyagramını çizerek kodlayınız.

22. Klavyeden girilen N tamsayısı kadar "*Hilbert sayılarını*" oluşturup ekrana listeleyen programın akış diyagramını çiziniz ve herhangi bir programlama dilinde kodlayınız.

23. a, b ve c farklı pozitif tamsayılar (veya $c > b > a > 0$) olmak üzere $ab + bc + ac$ şeklinde yazılamayan (elde edilemeyen) pozitif tamsayılara, matematikte "*idoneal sayılar*" denir. Örneğin 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15, 16, 18, 21, 22, 24, 25, 28, 30, 33 gibi tamsayılar "idoneal sayılar"dır. Buna göre klavyeden girilen pozitif bir tamsayının, idoneal sayısı olup olmadığını test eden programın akış diyagramını çizip kodlayınız.

24. 7, 23, 79, 287, 1087, 4223 gibi $4^k + 2^{k+1} - 1$ veya bunun eşdeğeri olan $\left(2^k + 1\right)^2 - 2$ şeklinde yazılabilen tamsayılara "*Kynea sayıları*" denir. Buna göre klavyeden girilen tamsayının, Kynea sayısı olup olmadığını test eden programın akış diyagramını çizip herhangi bir dilde kodlayınız.

25. 1, 4, 8, 9, 16, 25, 27, 32, 36, 49, 64, 72, 81, 100, 108 ... gibi bir kare ve bir küpün çarpımıyla elde edilebilen tamsayılara, "*güçlü sayılar*" denir. Yani a ve b pozitif tamsayılar olmak üzere $g = a^2 b^3$ şeklinde ifade edilebilen tamsayılardır. Buna göre klavyeden girilen N pozitif tamsayı değeri için ilk N tane güçlü sayıyı listeleyen programın akış diyagramını çizip kodlayınız.

26. Sayılar teorisinde 2, 5, 11, 23, 47, 95, 191, 383, 767, 1535, 3071 gibi $k \geq 0$ için $3.2^k - 1$ formunda elde edilebilen/yazılabilen tamsayılar "*Thabit sayısı*" veya "*321 sayısı*" olarak adlandırılmaktadır. Buna göre klavyeden girilen N değerine göre $0 \leq k \leq N$ için Thabit sayılarının listesini ekrana yazdıran programın akış diyagramını çizerek kodlayınız.

27. "*Fibonacci sayıları*" { 1 , 1 , 2 , 3 , 5 , 8 , 13 , 21 , 34 , 55 , 89 , 144 , ... } şeklindedir. Bu sayılara göre "*Fibonacci üçgen*"leri şu şekilde oluşmaktadır:

- Başlangıç (ilk) üçgen : 5 – 4 – 3
- "8" terimi atlanarak ikinci üçgen : 13 – (5+4+3) – (8-3) -> 13–12–5
- "21" terimi atlanarak üçüncü üçgen : 34 – (13+12+5) – (21-5) -> 34–30–16
- .....

Klavyeden girilen üçgen sayısına göre, "*Fibonacci üçgen*"lerini listeleyen programın akış diyagramını çizip kodlayınız.

28. 2, 3, 6, 20, 168, 7581, 7828354 ... ilk 7 "*Dedekind sayıları*"dır. Bu sayılar $n \geq 0$ için $a_k^i = \left\lfloor \frac{i}{2^k} \right\rfloor - 2\left\lfloor \frac{i}{2^{k+1}} \right\rfloor$ olmak üzere

$$D(n) = \sum_{i=1}^{2^{2^n}} \prod_{j=1}^{2^n-1} \prod_{k=0}^{j-1} \left(1 - a_k^i . a_i^j \cdot \prod_{m=0}^{\log_2 k} \left(1 - a_m^k + a_m^k a_m^j\right)\right)$$

Kisielewicz tarafından bulunan toplam formülü ile hesaplanmaktadır. Buna göre klavyeden girilen n değeri için $D(n)$ Dedekind sayısını hesaplayıp ekrana yazdıran programın akış diyagramını çizip kodlayınız.

29. Aşağıdaki tabloda bazı "*poligonsal sayılar*"ın matematiksel eşitlikleri verilmektedir. Bu tür sayılarla, düzgün poligon şekiller oluşturulabilmektedir. Buna göre her bir sayı türü için klavyeden girilen terim sayısı kadar oluşturup ekrana yazdıran programların akış diyagramlarını çizerek kodlayınız.

Sayı	Formülü	Örnek sayılar
Karesel	n^2	1, 4, 9, 16, 25, 36, 49, ...
Üçgensel	$\frac{n(n+1)}{2}$	1, 3, 6, 10, 15, 21, 28, ...
Beşgensel (pentagonal)	$\frac{n(3n-1)}{2}$	1, 5, 12, 22, 35, 51, 70, ...
Altıgensel (hexagonal)	$n(2n-1)$	1, 6, 15, 28, 45, 66, 91, ...
Yedigensel (heptagonal)	$\frac{n(5n-3)}{2}$	1, 7, 18, 34, 55, 81, 112, ...
Sekizgensel (octagonal)	$n(3n-2)$	1, 8, 21, 40, 65, 96, 133, ...

30. İki asal sayının çarpımından elde edilen tamsayılar, "*yarıasal sayı*" (semiprime, biprime vb.) olarak isimlendirilmektedirler. Örneğin 4, 6, 9, 10, 14, 15, 21, 22, 25 ... yarıasal sayılardır. Buna göre klavyeden girilen pozitif tamsayının "yarıasal" olup olmadığını test eden programın akış diyagramını çizip kodlayınız.

31. Aşağıdaki tabloda bazı özel asal sayıların tanımlamaları verilmektedir. Buna göre klavyeden girilen terim sayısınca ilgili asal sayıları hesaplayıp listeleyen programların akış diyagramlarını çizip kodlayınız.

İsim	Parametre	Biçim
Mersenne asalları	p *asal sayı*	$2^p - 1$
Solinas asalları	$0 < b < a$	$2^a \pm 2^b \pm 1$
Wagstaff asalları	p *asal sayı*	$\frac{2^p + 1}{3}$

32. Bir ve kendisinden başka tamsayıya kalansız bölünemeyen pozitif tamsayılara, "*asal sayı*" denir. "a" herhangi bir asal sayı olmak üzere "a", "a+2" ve "a+6" doğal sayıları da asal sayı ise bunlara "*asal üçüz*" denir. Buna göre klavyeden girilen asal sayının, ilgili "*asal üçüz*" de yer alıp almadığını tespit eden programın akış diyagramını çizip yazılımını yapınız.

Örneğin; 3,5,9 için 3 asal üçüzde yer almaz.
5,7,11 için 5 asal üçüzde yer alır.
7,9,13 için 7 asal üçüzde yer almaz.

33. Klavyeden girilen "A" doğal sayısına kadar olan asal üçüzlerin listesini ekrana yazdıran programın akış diyagramını çizip yazılımını yapınız.

34. Tersten yazılışı yine kendisine eşit olan tamsayılara "*palindrom*" denir. Klavyeden girilen basamak sayısına göre palindromların listesini veren programın akış diyagramını çizip yazılımını yapınız.

35. 10-99 arasındaki "*bağdaşık sayılar*"ın listesini yazdıran programın akış diyagramını çizip kodlayınız.

36. Herhangi bir tamsayının rakamları toplamı, yine bu tamsayının asal çarpanlarının rakamlarının toplamına eşitse "*Smith sayısı*" denir. Örneğin 121=11*11 yani 1+2+1=1+1+1+1 veya 166=2*83 (1+6+6=2+8+3) ya da 728=2*2*2*7*13 (7+2+8=2+2+2+7+1+3) olduklarından Smith sayılarıdır. Klavyeden girilen bir tamsayının "*Smith sayısı*" olup olmadığını test eden programın akış diyagramını çizip kodlayınız.

37. 1-10000 arasındaki "*Smith sayıları*"nın listesini veren programın akış diyagramını çizip kodlayınız.

38. Eğer çift basamaklı bir tamsayı; kendisini oluşturan rakamların yarılarıyla elde edilen iki tamsayının çarpımına eşitse "*Vampir sayı*" olarak adlandırılır. Örneğin 1435=41*35, 6880=86*80, 2187=27*81 gibi. Buna göre klavyeden girilen çift basamaklı tamsayının "*Vampir sayı*" olup olmadığını test eden, olması durumunda çarpanlarını da yazdıran programın akış diyagramını çizip kodlayınız.

39. 1000-9999 arasındaki "*Vampir sayıları*"nın listesini veren programın akış diyagramını çizip kodlayınız.

40. 'n' basamaklı bir 's' tamsayısının karesinden oluşan tamsayının (s^2) sağındaki 'n' basamaklı tamsayı ile solunda kalan 'n' veya 'n-1' basamaklı tamsayının toplamı yine 's' tamsayısını veriyorsa buna "*Kaprekar sayısı*" denir. Örneğin 99 tamsayısının karesi 99^2=9801 ve 01+98=99 olduğundan "*Kaprekar sayısı*"dır. Benzer şekilde 2728 (2728^2=7441984 ve 1984+744=2728) de bu türden sayıdır. Buna göre klavyeden girilen tamsayının, "*Kaprekar sayısı*" olup olmadığını test eden programın akış diyagramını çizip kodlayınız.

41. 1^2=1 , 5^2=25 , 6^2=36 , 25^2=625 , 76^2=5776 , 376^2=141376 , 625^2=390625 gibi sayılara (1, 5, 6, 25, 76, 376, 625 ...) yani karelerinin son haneleri kendileri olanlara "*otomorf sayılar*" denir. Buna göre klavyeden girilen tamsayının, "otomorf sayı" olup olmadığını test eden programın akış diyagramını çizip kodlayınız.

42. 1^3=1 , 4^3=64 , 5^3=125 , 6^3=216 , 9^3=729 , 24^3=13824 , 25^3=15625 , 49^3=117649 , 51^3=132651 gibi sayılara, yani küplerinin son haneleri kendileri olanlara "*trimorf sayılar*" denir. Buna göre klavyeden girilen adet kadar "*trimorf sayıların*" listesini veren programın akış diyagramını çizip kodlayınız.

43. $4k+3$ biçimindeki a ve b farklı asal sayıların çarpımından elde edilen yarı-asal sayıya, "*Blum tamsayısı*" denir. Bunlardan birkaçı 21, 33, 57, 69, 77, 93, 129, 133, ... dir. Buna göre klavyeden girilen tamsayının, Blum tamsayısı olup olmadığını test eden programın akış diyagramını çizerek kodlayınız.

44. 1'den başka hiçbir Hilbert sayısına tam bölünemeyen Hilbert sayılarına, "*Hilbert asalları*" denir. Buna göre klavyeden girilen pozitif tamsayının, Hilbert asalı olup olmadığını test eden programın akış diyagramını çizip kodlayınız.

45. 4, 6, 9, 12, 15, 18, 21, 26, 30, 34 gibi; ardışık iki tek asal sayının ortalaması olan sayılara, matematikse "*ara asal sayı*" (interprime) denir. Örneğin $12=\frac{11+13}{2}$, $15=\frac{13+17}{2}$, $18=\frac{17+19}{2}$ olduklarından "ara asal sayı"dırlar. Buna göre klavyeden girilen N tamsayısı için ilk N tane ara asal sayıların listesini yazdıran programın akış diyagramını çizip kodlayınız.

46. (3,7), (7,11), (13,17), (19,23), (37,41) gibi aralarındaki farkın 4 olduğu asal sayılara, "*kuzen asallar*" denir. Buna göre klavyeden girilen N tamsayı değeri için ilk N tane kuzen asal sayıları listeleyen programın akış diyagramını çiziniz ve herhangi bir dilde kodlayınız.

47. Sayılar teorisinde; 1, 2, 4, 6, 8, 12, 16, 18, 20, 24, 28 ... gibi farklı tam bölenlerinin toplamıyla kendisinden küçük tüm pozitif tamsayıların elde edilebildiği sayılara, "*pratik sayılar*" denir. Örneğin 8 sayısının tam bölenleri 1, 2, 4 olup 1-7 arasındaki pozitif tamsayılar 1, 2, 1+2, 4, 4+1, 4+2, 4+2+1 şeklinde elde edilebilmektedir. Buna göre klavyeden girilen bir tamsayının, "*pratik sayı*" olup olmadığını test eden programın akış diyagramını çizerek kodlayınız.

48. 1, 1, 3, 5, 9, 15, 25, 41, 67, 109, 177 ... sayıları, "*Leonardo sayıları*" olarak adlandırılmaktadır. Matematiksel olarak Leonardo sayıları

$$L(k)=\begin{cases}1 & , \ k=0\\ 1 & , \ k=1\\ L(k-2)+L(k-1)+1 & , \ k>1\end{cases}$$

ile hesaplanabilmektedir. Ayrıca $k \geq 0$ için Fibonacci sayıları ile aralarında $L(k)=2F(k)-1$ ilişkisi mevcuttur. Buna göre klavyeden girilen tamsayı değer (terim sayısı) kadar Leonardo sayılarını bir diziye aktarıp ekranda listeleyen programın akış diyagramını çiziniz ve herhangi bir programlama dilinde kodlayınız.

49. 0, 1, 1, 3, 5, 11, 21, 43, 85, 171, 341, 683, 1365 ... sayıları matematikte "*Jacobsthal sayıları*" olarak adlandırmaktadırlar. Bu sayılar;

☑ $$J_k=\begin{cases}0 & , \ k=0\\ 1 & , \ k=1\\ J_{k-1}+2J_{k-2} & , \ k>1\end{cases}$$

☑ $J_{k+1}=2J_k+(-1)^k$

☑ $J_{k+1}=2^k-J_k$

☑ $J_k=\dfrac{2^k-(-1)^k}{3}$, $k>1$

eşitliklerinin herhangi birisiyle elde edilebilir. Buna göre klavyeden girilen tamsayı kadar Jacobsthal sayılarını diziye aktarıp listeyen programların akış diyagramlarını, herbir eşitlik için ayrı ayrı çizip kodlayınız.

50. 2, 1, 5, 7, 17, 31, 65, 127, 257, 511, 1025, 2047, 4097 ... sayıları matematikte "*Jacobsthal-Lucas sayıları*" olarak adlandırmaktadırlar. Bu sayılar;

☑ $$JL_k=\begin{cases}2 & , \ k=0\\ 1 & , \ k=1\\ JL_{k-1}+2JL_{k-2} & , \ k>1\end{cases}$$

☑ $JL_{k+1}=2JL_k+3(-1)^k$

☑ $JL_{k+1}=2^k+(-1)^k$, $k>1$

eşitliklerinin herhangi birisiyle elde edilebilir. Buna göre klavyeden girilen tamsayı kadar Jacobsthal sayılarını diziye aktarıp listeyen programların akış diyagramlarını, herbir eşitlik için ayrı ayrı çizip kodlayınız.

51. 6174 tamsayısı, "*Kaprekar sabiti*" olarak bilinmektedir. Aşağıdaki adımlar izlendiğinde mutlaka bu sabite ulaşılmaktadır:

i. En az iki farklı rakam kullanarak, dört haneli/basamaklı tamsayı oluşturunuz.

ii. Bu tamsayının hanelerini küçükten büyüğe ve büyükten küçüğe doğru sıralayarak iki tane tamsayı elde ediniz. Gerekirse sıfır/sıfırlar ilave ediniz.

iii. Elde edilen büyük sayıdan küçük sayıyı çıkarınız.

iv. Elde edilen fark sayısı için ii. adımına gidiniz.

Örneğin 2011 tamsayısı için bu adımlar

$$\begin{array}{cccccccc} 2110 & & 9981 & & 8820 & & 8532 \\ \underline{0112} & \rightarrow & \underline{1899} & \rightarrow & \underline{0288} & \rightarrow & \underline{2358} \\ 1998 & & 8082 & & 8532 & & 6174 \end{array}$$

şeklinde olmaktadır. Buna göre klavyeden girilen dört basamaklı tamsayı için "Kaprekar sabiti"ni elde edinceye kadar gerçekleştirilen işlem adımlarını ekrana yazan programın akış diyagramını çiziniz ve herhangi bir programlama diliyle kodlayınız.

52. 2618 tamsayısının 11 ile çarpımı 28798 dir. 2618 tamsayını ters çevirip 11 ile çarpılırsa 8162x11=89782 elde edilir. Dikkat edilirse sonuç, yine bir önceki sonucun ters çevrilmiş halidir. Klavyeden girilen pozitif tamsayının bu özelliği sağlayıp sağlamadığını test eden programın akış diyagramını çizip kodlayınız.

53. Klavyeden girilen satır/sütun sayısına göre üst üçgen ve simetrik Pascal matrislerini ayrı ayrı oluşturup yazdıran programların akış diyagramlarını çiziniz ve herhangi bir dil ile kodlayınız.

54. Matematikte

☑ **İşaretsiz "Lah sayıları":** $L(m,n)=\binom{m-1}{n-1}\frac{m!}{n!}$

☑ **İşaretli "Lah sayıları":** $L'(m,n)=(-1)^m\binom{m-1}{n-1}\frac{m!}{n!}$

şeklinde hesaplanabilmektedir. Aşağıda bazı Lah sayıları verilmektedir.

$m \setminus n$	1	2	3	4	5	6
1	1					
2	2	1				
3	6	6	1			
4	24	36	12	1		
5	120	240	120	20	1	
6	720	1800	1200	300	30	1

Buna göre klavyeden girilen m ve n değerleri için işaretsiz ve işaretli $L(m,n)$ Lah sayısını hesaplayıp ekrana yazdıran programın akış diyagramını çiziniz ve herhangi bir programlama dilinde kodlayınız.

55. Klavyeden girilen n basamaklı pozitif tamsayıyı, n basamaklı en büyük asal sayıdan çıkaran programın akış diyagramını çizinip kodlayınız.

56. 3, 0, 2, 3, 2, 5, 5, 7, 10, 12, 17... sayıları, "*Perrin sayıları*" olarak adlandırılmakta ve

$$P(k)=\begin{cases} 3 & , k=0 \\ 0 & , k=1 \\ 2 & , k=2 \\ P(k-3)+P(k-2) & , k>2 \end{cases}$$

yinelemeli eşitliği veya

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}^k \begin{bmatrix} 3 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} P(k) \\ P(k+1) \\ P(k+2) \end{bmatrix}$$

matris çarpımıyla hesaplanabilir. Buna göre klavyeden girilen değer kadar Perrin sayısı oluşturan programın akış diyagramlarını hem yinelemeli eşitlik hem de matris çarpımı için çizip kodlayınız.

57. $n>2$ olmak satır, sütun ve köşegen elemanları toplamı "*sihirli sabit*"e eşit olan NxN tipindeki kare matrislere "*sihirli kare*" denir. Matrisin elemanları, birbirinden farklı olmak üzere $\{1,2,3,...,n^2\}$ kümesinden olup "*sihirli sabit*" $\frac{n(n^2+1)}{2}$ ile hesaplanır. Örneğin

$$\begin{bmatrix} 17 & 24 & 1 & 8 & 15 \\ 23 & 5 & 7 & 14 & 16 \\ 4 & 6 & 13 & 20 & 22 \\ 10 & 12 & 19 & 21 & 3 \\ 11 & 18 & 25 & 2 & 9 \end{bmatrix}$$

bir "sihrili kare"dir. Satır, sütun ve ana-yedek köşegenler toplamı daima bu matris için "sihirli sabit" olan 65'e eşittir. "Sihirli kare" problemini araştırarak klavyeden girilen n değerine göre bu türden matris oluşturan programın akış diyagramını çizip kodlayınız.

58. Diğer özel sayı ve şekilleri araştırarak akış diyagramlarını çiziniz ve herhangi bir dil ile kodlayınız.

BÖLÜM 12

SIRALAMA ALGORİTMALARI

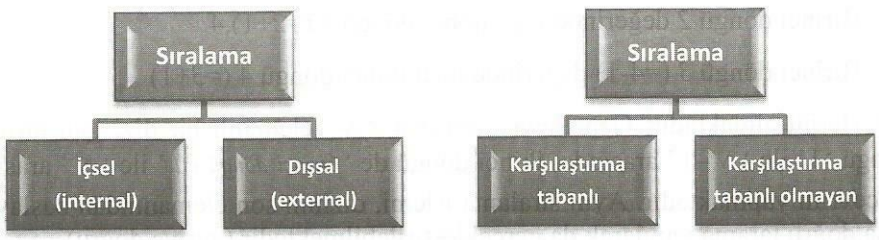
Bu bölümde

- Sıralama algoritmalarını tanıyacak,
- Akış diyagramlarını çizerek kodlamasını öğreneceksiniz.

Bölüm sonunda

- Sıralama algoritmalarını kavrayacak,
- Performanslarını karşılaştırabilecek,
- Akış diyagramlarını çizip kodlayabileceksiniz.

Uygulamalarda verilerin belirli bir düzende (küçükten büyüğe doğru sıralı, alfabetik olarak sıralı vb.) olması, hem işlem/işleme hızını arttırmakta hem de büyük kolaylıklar sağlamaktadır. Verileri sıralamak için çalışma yöntemleri ve hızları farklı birçok algoritma mevcuttur. Tüm sıralama algoritmalarında temel amaç; hafızayı verimli bir şekilde kullanarak veriyi en kısa sürede, en doğru şekilde sıralamaktır. Sıralanacak verilerin bulunduğu ortama/hafıza durumuna göre sıralama algoritmaları, temel olarak ikiye ayrılabilir: içsel ve dışsal. Eğer verilerin tümü dahili bellekte ise içsel, verilerin bazıları ikincil/harici bellek konumlarında ise dışsal sıralama olarak isimlendirilmektedir. Sıralama algoritmaları için bir diğer sınıflandırma karşılaştırma tabanlı ve karşılaştırma tabanlı olmayan şeklinde yapılmaktadır (Şekil 12.1). Ayrıca bu algoritmalarda hesaplama karmaşıklığına, yer değiştirmelerin karmaşıklığına, bellek kullanımına, durağanlığına, yineleme vb. özelliklerine de bakılarak sınıflandırmalar veya performans karşılaştırmaları yapılabilmektedir.



Şekil 12.1: Sıralama türleri

Bu bölümde bazı temel sıralama algoritmaları tanıtılacaktır. İlgili sıralama algoritmaları sunulurken, daha açıklayıcı olması açısından az elemanlı sabit bir dizi ele alınacaktır. Bu algoritmaları fonksiyon/alt program olarak düzenleyerek, diğer uygulamalarınızda doğrudan kullanabilirsiniz.

I. SEÇME SIRALAMA (SELECTION SORT)

"*Seçme sıralama*" *(selection sort)* algoritmasını kullanarak küçükten büyüğe doğru yapılan sıralama işleminde, her taramada en küçük eleman bulunarak/seçilerek baş tarafa aktarılmaktadır. Yani birinci taramada bulunan en küçük eleman birinci sıraya; ikinci elemandan başlayarak yapılan ikinci taramada bulunan en küçük eleman ikinci sıraya; üçüncü elemandan başlayarak yapılan üçüncü taramada bulunan en küçük eleman üçüncü sıraya vb. yerleştirilerek sıralama işlemi tamamlanmaktadır.

Örneğin A dizisinin elemanları şu şekilde olsun: $A=(2,1,4,3)$. Birinci adımda dizinin ilk elemanı olan '2' alınıp ikinci, üçüncü ve dördüncü elemanla karşılaştırılmaya başlanacaktır. Eğer alınan (ilk) elemandan daha küçük dizi elemanı varsa, bunların yerleri değiştirilir. Dolayısıyla '2' ile '1' yer değiştirir ve A dizisi $A=(1,2,4,3)$ şeklini alır. Bu işlem son elemana kadar devam edece-

ğinden ilk eleman - artık '1' oldu - üçüncü ve dördüncü elemanla karşılaştırılır ve büyüklük-küçüklük durumuna göre herhangi bir değişiklik olmaz. Böylece dizi elemanlarının birinci taranmasında; en küçük eleman, en başa gelmektedir. Daha sonra ikinci eleman (artık ikinci eleman '2') alınıp dizi sonuna kadar diğer elemanlarla kıyaslanır. Yani '2' değeri '4' ve '3' ile karşılaştırılır. Daha küçük bir değer olmadığı için dizide herhangi bir değişiklik olmaz. En son adımda da dizinin üçüncü elemanı ('4'), dizinin son elemanı ile kıyaslanır, yer değişikliği yapılır ve dizi $A=(1,2,3,4)$ şeklini alır (Tablo 12.1). Buradaki dizi 4 elemanlı olduğu için sırasıyla 1. eleman alınıp 2., 3. ve 4. ile; 2. eleman alınıp 3. ve 4. ile; 3. eleman alınıp 4. ile karşılaştırılmaktadır. Yani birinci döngü (kıyaslanacak olan değer), 1 ile 3 (4-1) arasında çevrim yapmaktadır. Karşılaştırılan değerler ise alınan değerin bir sonrakiyle başlayıp dizi sonuna kadar gitmektedir. Yani;

Birinci döngü *1* değerinde iken ikinci döngü 2 (=*1*+1),3,4

Birinci döngü *2* değerinde iken ikinci döngü 3 (=*2*+1),4

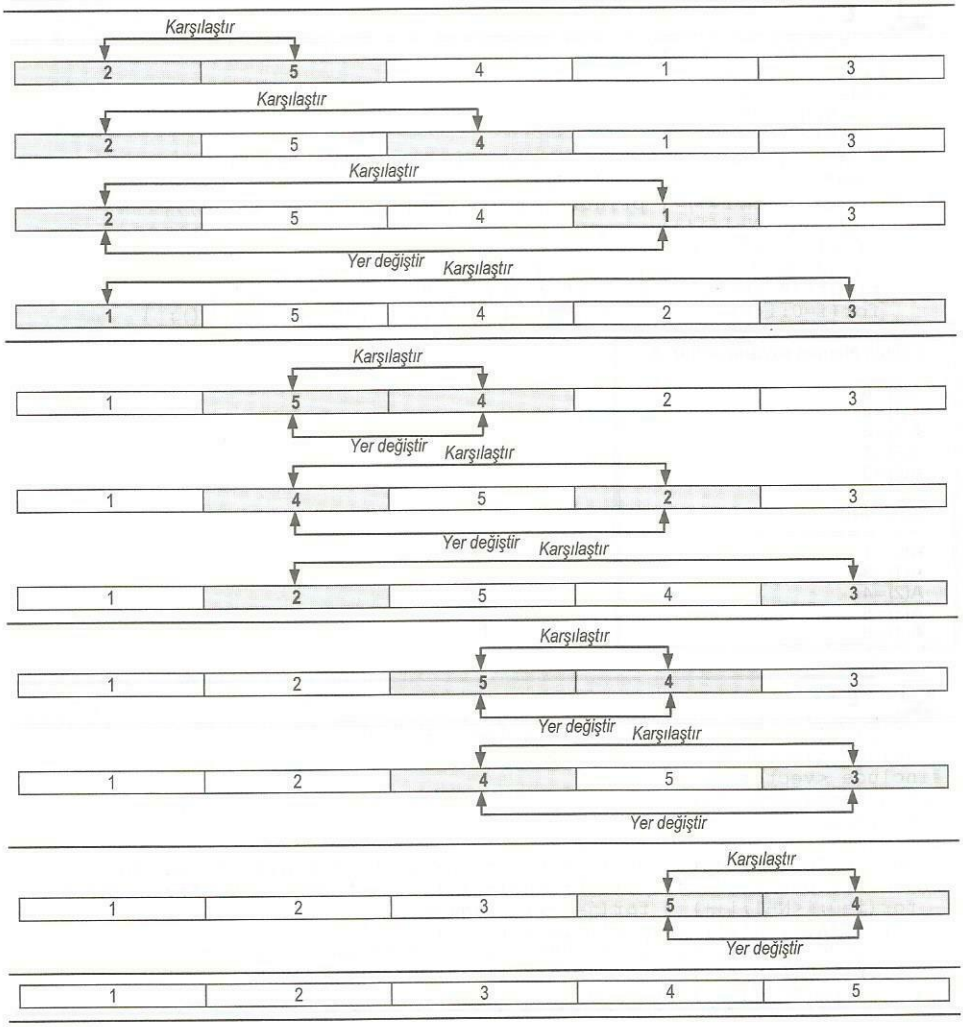
Birinci döngü *3* (=4-1) değerinde iken ikinci döngü 4 (=*3*+1)

değerlerini almaktadır. Genelleme yapılırsa, 'N' elemanlı bir dizi için birinci döngü '1' ile '$N-1$' arasında; ikinci döngü de '*birinci döngü*$+1$' ile 'N' arasında çevrim yapmaktadır. Aynı sıralama işlemi, dizinin son elemanından başlayıp başa doğru tarama yapılarak da gerçekleştirilebilmektedir (geriye doğru).

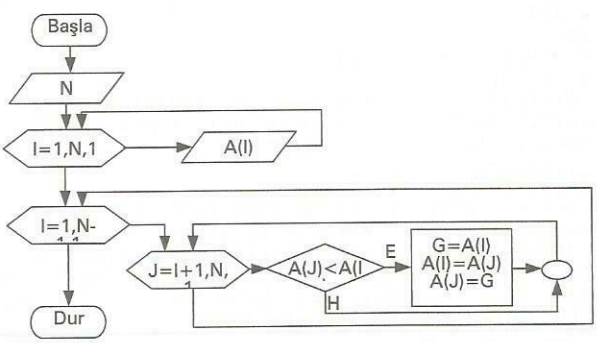
Tablo 12.1: Örnek için seçme sıralama algoritmasındaki işlem adımları

I	J	Koşul	Değişiklik	Dizi
1	2	A(2)<A(1) 1<2	A(1)↔A(2)	A=(1,2,4,3)
	3	A(3)<A(1) 4<1	-	A=(1,2,4,3)
	4	A(4)<A(1) 3<1	-	A=(1,2,4,3)
2	3	A(3)<A(2) 4<2	-	A=(1,2,4,3)
	4	A(4)<A(2) 3<2	-	A=(1,2,4,3)
3	4	A(4)<A(3) 3<4	A(3)↔A(4)	A=(1,2,3,4)

`A=[2,5,4,1,3]` dizisinin "*ileriye doğru seçme sıralama yöntemi*"yle sıralanması, adım adım aşağıda verilmektedir.



Örnek-12.1: Klavyeden girilen N elemanlı A dizisini küçükten büyüğe doğru "*seçme sıralama (Selection sort)*" algoritmasıyla sıralayan program.



C

```
/* Dizi elemanlarını sıralama - Selection sort */
#include <stdio.h>
#include <conio.h>
int i,j,N,g,A[100];
void main()
{   clrscr();printf("Dizinin eleman sayısını giriniz: ");
    scanf("%u",&N);printf("\n");
    for(i=0;i<N;i++) { printf("A(%u)=",i);scanf("%d",&A[i]);}
    for(i=0;i<N-1;i++)
     {for(j=i+1;j<N;j++)
       { if (A[j]<A[i]) { g=A[i];A[i]=A[j];A[j]=g; }}}
     printf("\nSıralı dizi:\n"); printf("============\n");
     for(i=0;i<N;i++) printf("A(%u)=%d\n",i+1,A[i]); getch(); }
```

```
Dizinin eleman sayısını giriniz: 5

A(0)=6
A(1)=8
A(2)=4
A(3)=1
A(4)=3

Sıralı dizi:
============
A(0)=1
A(1)=3
A(2)=4
A(3)=6
A(4)=8
```

C++

```
// Dizi elemanlarını sıralama - Selection sort
#include <iostream>
#include <vector>
using namespace std;
int i,j,N;
int main()
{ system("cls");cout<<"Dizinin eleman sayısını giriniz: ";cin>>N;
int A[N];cout << "\n";for(i=0;i<N;i++) {cout<<"A("<<i<<")=";cin>>A[i];}
  for(i=0;i<N-1;i++) { for(j=i+1;j<N;j++)
    { if(A[j]<A[i]) swap(A[i],A[j]); } }
    cout << "\nSıralı dizi: \n"; cout << "============ \n";
    for(i=0;i<N;i++) cout<<"A("<<i+1<<")="<<A[i]<<"\n";
    system("pause");return 0; }
```

C#

```
// Dizi elemanlarını sıralama - Selection sort
using System;
namespace ornek121
{ class Program
   { static void Main(string[] args)
    { int i, j, N, g;
      Console.Write("Dizinin eleman sayısını giriniz: ");
      N = Convert.ToInt32(Console.ReadLine()); Console.WriteLine();
      int[] A=new int[N];
      for (i=0;i<N;i++)
       {Console.Write("A({0})= ",i);
        A[i]=Convert.ToInt32(Console.ReadLine());}
      for (i=0;i<N-1;i++) { for (j=i+1;j<N;j++)
          if (A[j]<A[i]) { g=A[i]; A[i]=A[j]; A[j]=g; }}
Console.WriteLine("\nSıralı dizi:");Console.WriteLine("============");
      for (i=0; i<N;i++) Console.WriteLine("A({0})={1}",i+1,A[i]);
      Console.ReadLine();       }     }     }
```

JAVA

```
// Dizi elemanlarını sıralama - Selection sort
import java.util.Scanner;
public class ornek121 {
public static void main(String[] args)
{  int i,j,N,g;
   Scanner klavye=new Scanner(System.in);
   System.out.print("Dizinin eleman sayısını giriniz: ");
   N=klavye.nextInt();System.out.println(""); int A[]=new int[N];
   for(i=0;i<N;i++)
     { System.out.print("A("+i+")="); A[i]=klavye.nextInt();}
   for(i=0;i<N-1;i++) { for(j=i+1;j<N;j++)
            if(A[j]<A[i]) { g=A[i];A[i]=A[j];A[j]=g;}}
   System.out.println("\nSıralı dizi:");
   System.out.println("============");
   for (i=0;i<N;i++) System.out.println("A("+(i+1)+")="+A[i]); }}
```

MATLAB

```
% Dizi elemanlarını sıralama - Selection sort
clear all; clc; N=input('Dizinin eleman sayısını giriniz: ');disp(' ');
for i=1:N fprintf('A(%d)= ',i);A(i)=input(''); end;
for i=1:N-1
    for j=i+1:N
        if (A(j)<A(i))
            g=A(i);A(i)=A(j);A(j)=g;
        end;end;end;fprintf('\nSıralı dizi\n==========\n');
for i=1:N fprintf('A(%d)=%d\n',i,A(i)); end;
```

PYTHON

```
# Dizi elemanlarını sıralama - Selection sort
A=[]
N=int(input("Dizinin eleman sayısını giriniz: "))
for i in range(N):
   print("A(%d)= "%(i+1),end='')
   A.append(int(input()))
for i in range(N-1):
    for j in range(i+1,N):
        if (A[j]<A[i]):
            g=A[i]
            A[i]=A[j]
            A[j]=g
print("\nSıralı dizi\n==========")
for i in range(N):
    print("A(%d)=%d"%(i+1,A[i]))
```

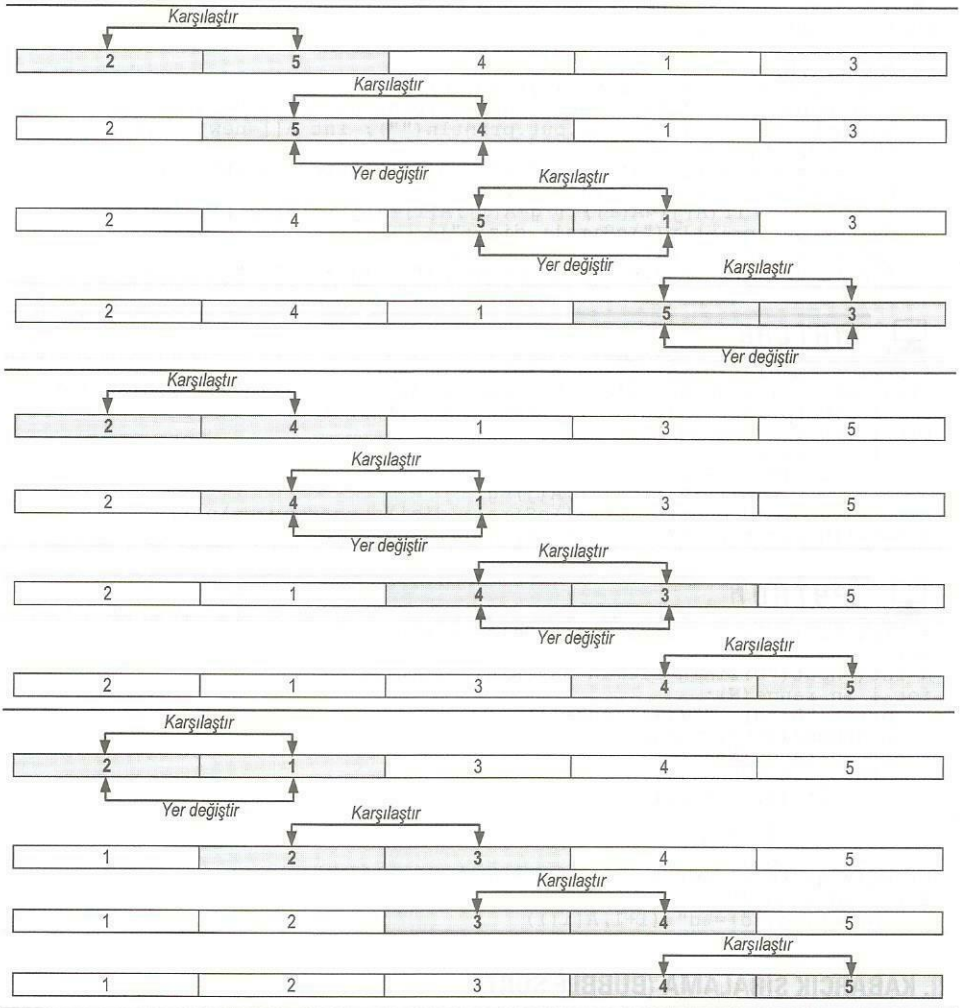
II. KABARCIK SIRALAMA (BUBBLE SORT)

Kabarcık sıralamada baştan sona veya sondan başa doğru komşu elemanlar karşılaştırılarak büyüklük-küçüklük durumlarına göre yerleri değiştirilir. Komşu elemanların bu karşılaştırmalı taraması, dizinin eleman sayısınca tekrarlanır. $A = (3,1,2)$ için sıralama adımları Tablo 12.2'de verilmektedir.

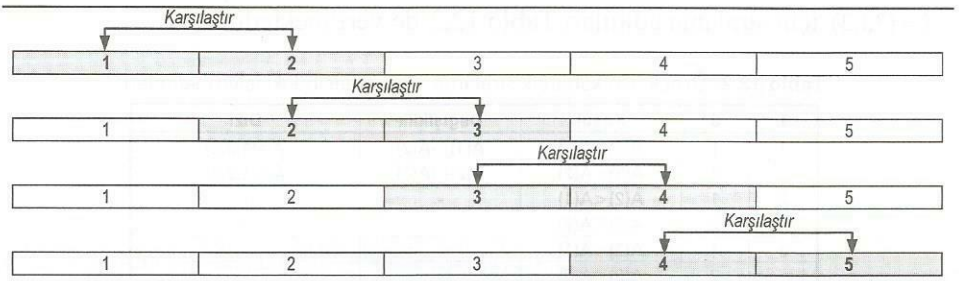
Tablo 12.2: Örnek için kabarcık sıralama algoritmasındaki işlem adımları

I	J	Koşul	Değişiklik	Dizi
1	1	A(2)<A(1)	A(1)↔A(2)	A=(1,3,2)
	2	A(3)<A(2)	A(2)↔A(3)	A=(1,2,3)
2	1	A(2)<A(1)	-	-
	2	A(3)<A(2)	-	-
3	1	A(2)<A(1)	-	-
	2	A(3)<A(2)	-	-

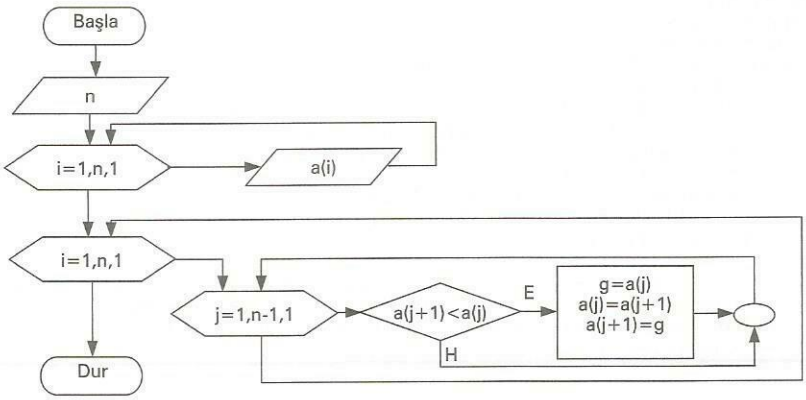
`A=[2,5,4,1,3]` dizisinin "*ileriye doğru kabarcık sıralama yöntemi*"yle sıralanması, adım adım aşağıda verilmektedir.



Dizinin sıralanması tamamlanmış olmasına rağmen 1. döngü daha bitmediğinden aşağıdaki dört adım 2(=5-3) kere tekrarlanacaktır.



Örnek-12.2: Klavyeden girilen n elemanlı a dizisini küçükten büyüğe doğru "*kabarcık sıralama (Bubble sort)*" algoritmasıyla sıralayan program.



C

```
/* Bubble sort */
#include <stdio.h>
#include <conio.h>
int i,j,n,g,a[100];
void main()
{ clrscr();printf("Dizinin eleman sayısı: ");
  scanf("%u",&n);printf("\n");
  for(i=0;i<n;i++) { printf("A(%u): ",i+1);scanf("%d",&a[i]);}
  for(i=0;i<n;i++)
   {for(j=0;j<n-1;j++)
     { if (a[j+1]<a[j]) { g=a[j];a[j]=a[j+1];a[j+1]=g; }}}
  printf("\nSıralı dizi:\n"); printf("------------\n");
  for(i=0;i<n;i++) printf("%d\t",a[i]); getch(); }
```

C++

```
// Bubble sort
#include <iostream>
using namespace std;
int main()
{ int i,j,n,g; system("cls"); cout<<"Dizinin eleman sayısı: ";
  cin>>n; int a[n]; cout<<"\n";
  for(i=0;i<n;i++) { cout<<"A("<<i+1<<"): "; cin>>a[i]; }
  for(i=0;i<n;i++)
   {for(j=0;j<n-1;j++)
     { if (a[j+1]<a[j]) { g=a[j];a[j]=a[j+1];a[j+1]=g; }}}
  cout<<"\nSıralı dizi:\n------------\n";
  for(i=0;i<n;i++) cout<<a[i]<<"\t"; system("pause"); return 0; }
```

```
Dizinin eleman sayısı: 5

A(1): 5
A(2): 1
A(3): 4
A(4): 2
A(5): 3

Sıralı dizi:
------------
1   2   4   5   3
```

C#

```
// Bubble sort
using System;
namespace ornek122
{ class Program
   { static void Main(string[] args)
      { int i,j,n,g;
        Console.Write("Dizinin eleman sayısı: ");
        n=Convert.ToInt32(Console.ReadLine());
        int[] a=new int[n]; Console.WriteLine();
        for (i=0;i<n;i++)
         { Console.Write("A({0}): ",i+1);
           a[i]=Convert.ToInt32(Console.ReadLine()); }
        for (i=0;i<n;i++)
         { for (j=0;j<n-1;j++)
            { if (a[j+1]<a[j]) { g=a[j]; a[j]=a[j+1]; a[j+1]=g; }}}
        Console.WriteLine("\nSıralı dizi:\n------------");
        for (i=0;i<n;i++) Console.Write("{0}\t",a[i]);
        Console.ReadLine();  }    }    }
```

JAVA

```
// Bubble sort
import java.util.Scanner;
public class ornek122 {
public static void main(String[] args)
{ int i,j,n,g; Scanner klavye=new Scanner(System.in);
  System.out.print("Dizinin eleman sayısı: ");n=klavye.nextInt();
  int[] a=new int[n]; System.out.println();
  for(i=0;i<n;i++)
   { System.out.printf("A("+(i+1)+"): "); a[i]=klavye.nextInt(); }
  for (i=0;i<n;i++)
   { for (j=0;j<n-1;j++)
      { if (a[j+1]<a[j]) { g=a[j]; a[j]=a[j+1]; a[j+1]=g; }}}
  System.out.printf("\nSıralı dizi:\n------------\n");
  for(i=0;i<n;i++) System.out.printf("%d\t",a[i]); }}
```

MATLAB

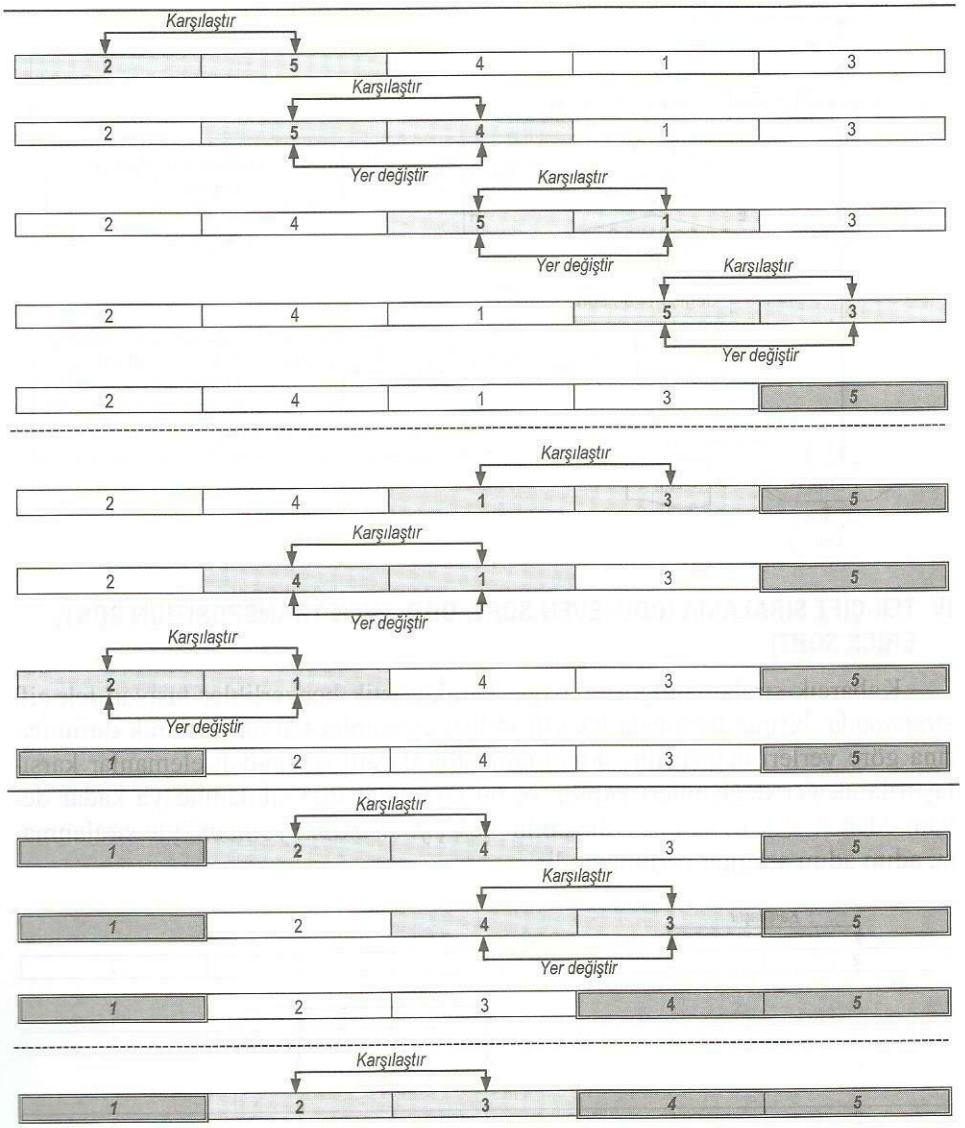
```
% Bubble sort
clear all; clc; fprintf('Diziyi [a b c d] şeklinde giriniz!\n')
a=input(''); n=length(a);
for i=1:n
    for j=1:n-1
        if a(j+1)<a(j)
            g=a(j);a(j)=a(j+1);a(j+1)=g; end;end;end;
fprintf('\nSıralı dizi:\n------------\n'); disp(a);
```

PYTHON

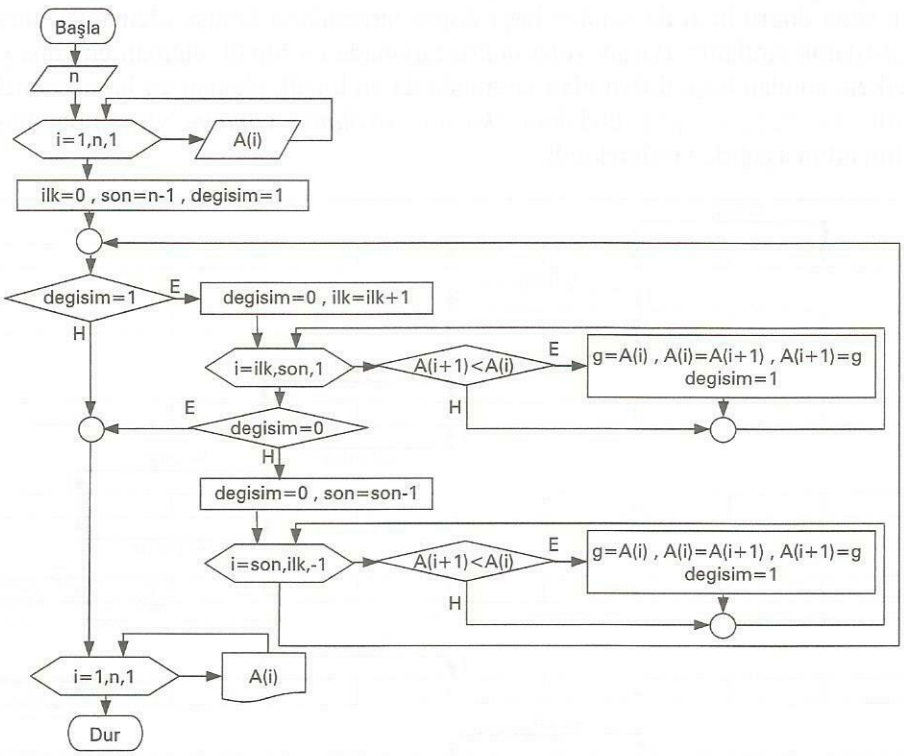
```
# Bubble sort
print("Diziyi [a b c d] şeklinde giriniz!")
a=[int(i) for i in input().split()]
n=len(a)
for i in range(n):
    for j in range(n-1):
        if a[j+1]<a[j]:
            a[j],a[j+1]=a[j+1],a[j]
print("\nSıralı dizi:\n------------")
for i in range(n):
    print("%d\t"%a[i],end='')
```

III. KOKTEYL SIRALAMA (COCKTAIL SORT, BIDIRECTIONAL BUBBLE SORT, COCKTAIL SHAKER SORT, SHAKER SORT, RIPPLE SORT, SHUFFLE SORT, SHUTTLE SORT)

Çift yönlü kabarcık sıralama algoritması olan kokteyl sıralamada hem baştan sona doğru hem de sondan başa doğru taramalarla komşu elemanlar karşılaştırılarak sıralanır. Baştan sona doğru taramada en büyük eleman en sona giderken, sondan başa doğru olan taramada da en küçük eleman en başa gelmektedir. `A=[2,5,4,1,3]` dizisinin *"kokteyl sıralama yöntemi"*yle sıralanması, adım adım aşağıda verilmektedir.

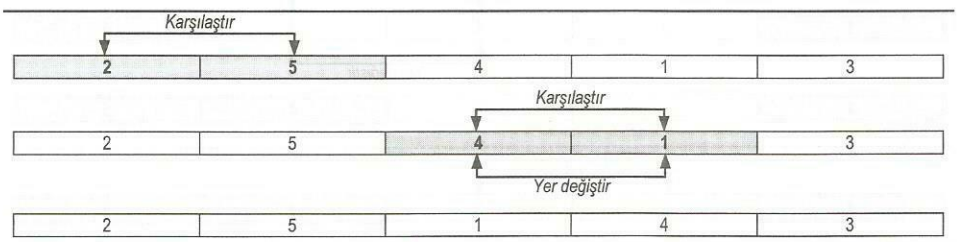


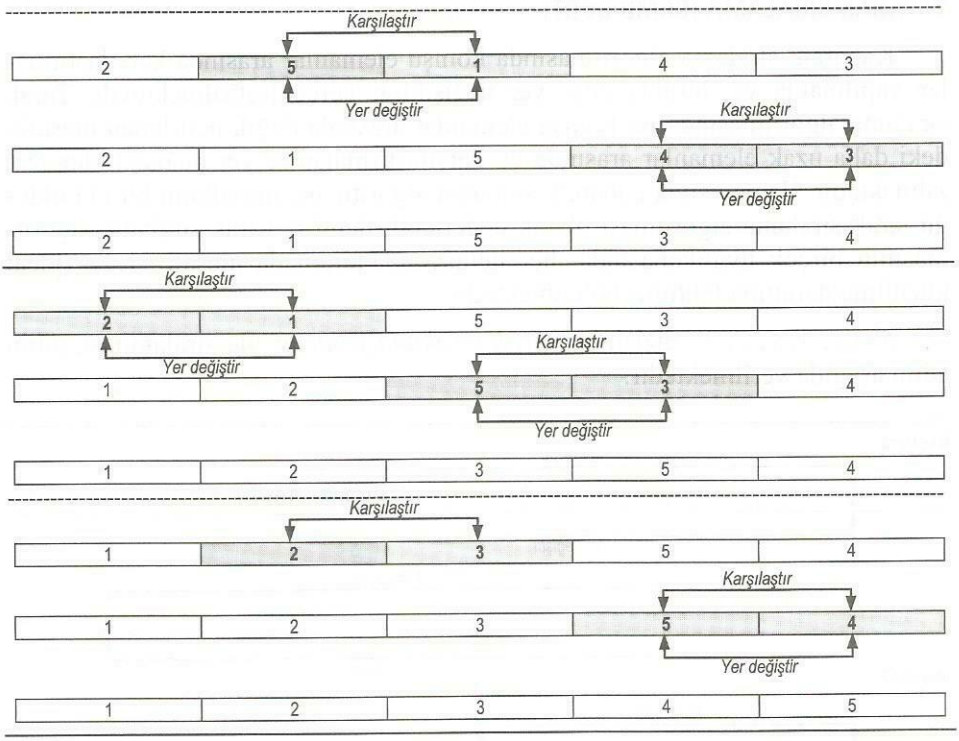
Örnek-12.3: Klavyeden girilen n elemanlı A dizisini küçükten büyüğe doğru "*kokteyl sıralama (Cocktail sort, bidirectional bubble sort, cocktail shaker sort, shaker sort, ripple sort, shuffle sort, shuttle sort)*" algoritmasıyla sıralayan program.



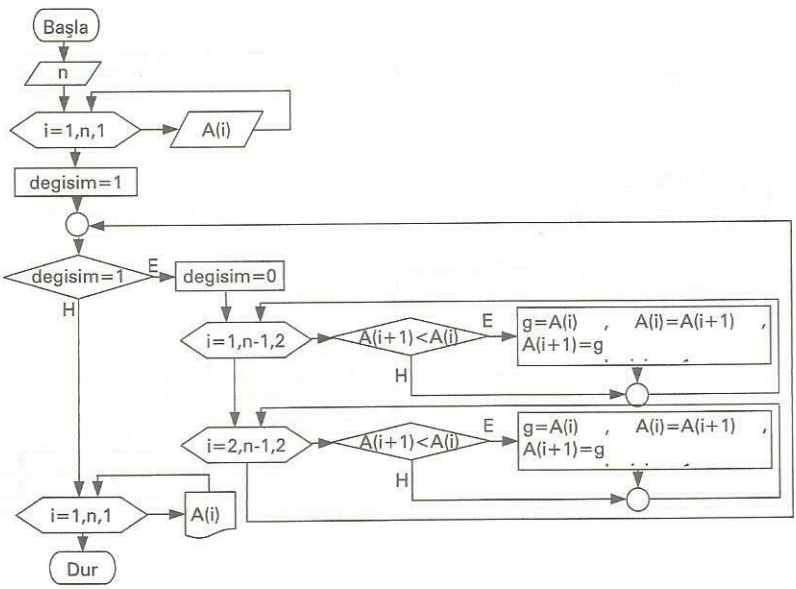
IV. TEK-ÇİFT SIRALAMA (ODD–EVEN SORT, ODD–EVEN TRANSPOSITION SORT, BRICK SORT)

Kabarcık sıralama algoritmasıyla karakteristik benzerlikler taşıyan tek-çift sıralamada; birinci taramada tek/çift indisli elemanlar karşılaştırılarak durumlarına göre yerleri değiştirilir, ikinci taramada da çift/tek indisli elemanlar karşılaştırılarak yer değişimleri yapılır ve bu taramalar dizi sıralanıncaya kadar devam eder. `A=[2,5,4,1,3]` dizisinin "*tek-çift sıralama yöntemi*"yle sıralanması, adım adım aşağıda verilmektedir.





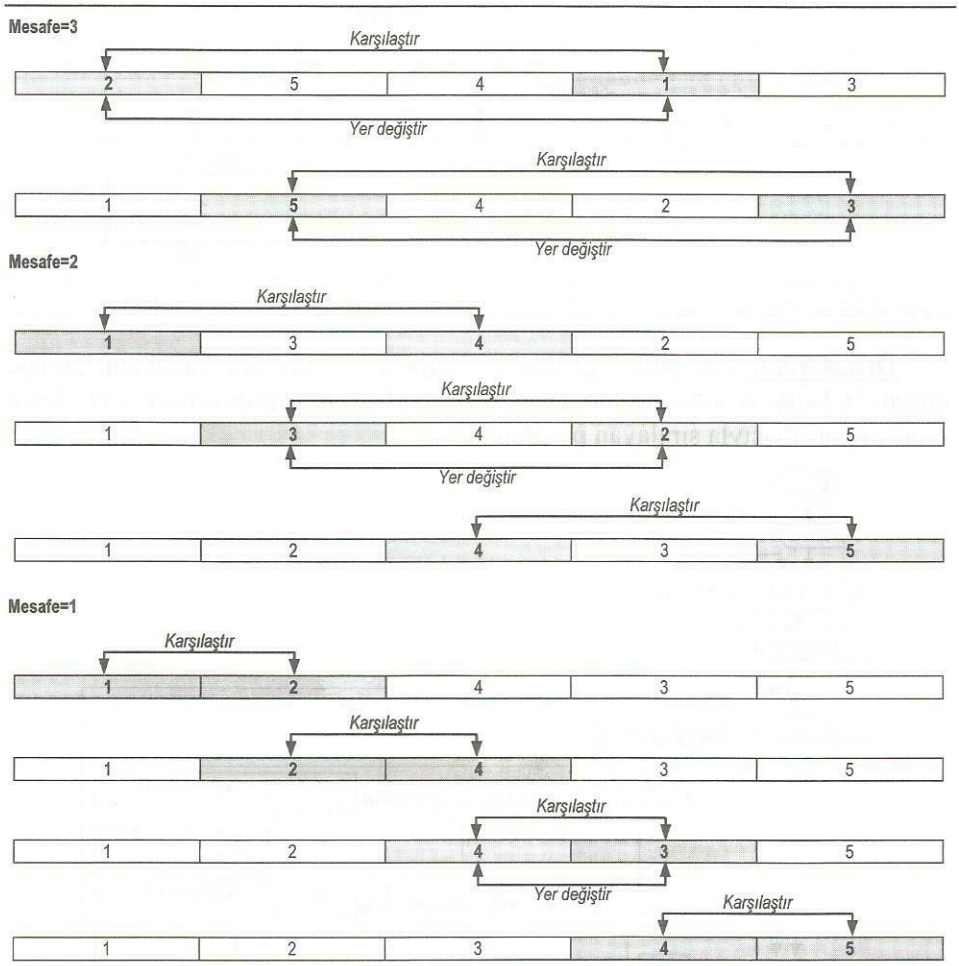
Örnek-12.4: Klavyeden girilen n elemanlı A dizisini küçükten büyüğe doğru "*tek-çift sıralama (odd–even sort, odd–even transposition sort, brick sort)*" algoritmasıyla sıralayan program.



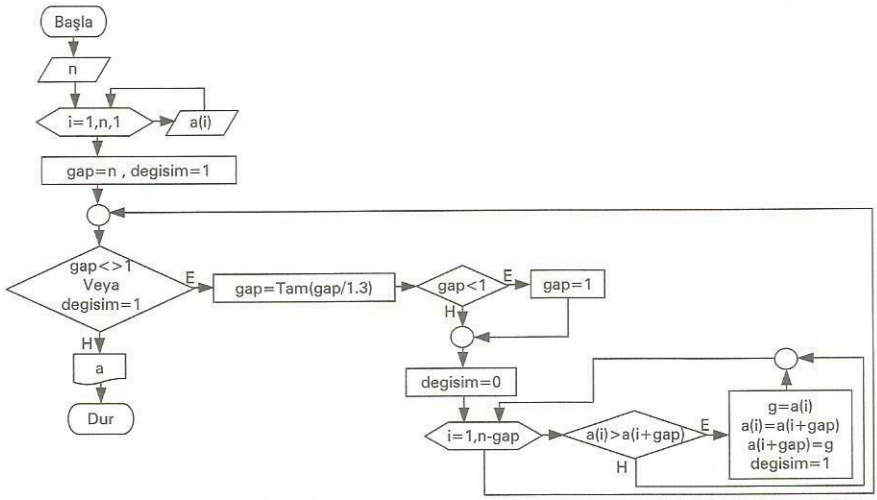
V. TARAK SIRALAMA (COMB SORT)

Kabarcık sıralama algoritmasında komşu elemanlar arasında karşılaştırmalar yapılmakta ve duruma göre yer değişikliği gerçekleştirilmekteydi. Tarak sıralama algoritmasında ise komşu elemanlar arasında değil, belirlenen mesafedeki daha uzak elemanlar arasında bu karşılaştırmalar ve yer değişiklikleri yapılmaktadır. Dolayısıyla kabarcık sıralama algoritması, mesafenin bir (1) olduğu tarak sıralama algoritması olarak değerlendirilebilir. Tarak sıralama algoritmasının birçok uygulamasında; her iterasyonda/taramada mesafe, 1.3'e (ideal küçültme/daraltma faktörü) bölünmektedir.

`A=[2,5,4,1,3]` dizisinin "*tarak sıralama yöntemi*"yle sıralanması, adım adım aşağıda verilmektedir.



Örnek-12.5: Klavyeden girilen *n* elemanlı *a* dizisini küçükten büyüğe doğru *tarak sıralama* algoritmasıyla sıralayan program.



C

```
/* Comb sort */
#include <stdio.h>
#include <conio.h>
int i,n,a[100];
void tarak_siralama()
{ int gap,degisim=1,g; gap=n;
  while ((gap!=1) || (degisim==1))
   { gap=(int) gap/1.3; if (gap<1) gap=1; degisim=0;
     for (i=0;i<n-gap;i++)
       if (a[i]>a[i+gap])
        { g=a[i]; a[i]=a[i+gap];a[i+gap]=g; degisim=1; } } }
void main()
{ clrscr();printf("Dizinin eleman sayısı: "); scanf("%u",&n);
  printf("\n");
  for(i=0;i<n;i++)
    { printf("Dizinin %d. elemanı: ",i+1); scanf("%d",&a[i]); }
  tarak_siralama(); printf("\nSıralı dizi:\n----------\n");
  for(i=0;i<n;i++) printf("%d\t",a[i]); getch(); }
```

C++

```
// Comb sort
#include <iostream>
using namespace std;
int i,n;
void tarak_siralama(int a[])
{ int gap,degisim=1,g; gap=n;
  while ((gap!=1) || (degisim==1))
   { gap=(int) gap/1.3; if (gap<1) gap=1; degisim=0;
     for (i=0;i<n-gap;i++)
       if (a[i]>a[i+gap])
        { g=a[i]; a[i]=a[i+gap];a[i+gap]=g; degisim=1; } } }
int main()
{ system("cls"); cout<<"Dizinin eleman sayısı: "; cin>>n; int a[n];
  cout<<"\n";
  for(i=0;i<n;i++)
  { cout<<"Dizinin "<<i+1<<". elemanı: "; cin>>a[i]; }
  tarak_siralama(a); cout<<"\nSıralı dizi:\n----------\n";
  for(i=0;i<n;i++) cout<<a[i]<<"\t"; system("pause"); return 0; }
```

C#

```
// Comb sort
using System;
namespace ornek125
{class Program {
  static void tarak_siralama(int[] a, int n)
   { int degisim=1,g,i,gap=n;
     while ((gap!=1) || (degisim==1))
      { gap=(int) (gap/1.3); if (gap<1) gap=1; degisim=0;
        for (i=0;i<n-gap;i++)
         if (a[i]>a[i+gap])
          { g=a[i]; a[i]=a[i+gap]; a[i+gap]=g; degisim=1; }}}
  static void Main(string[] args)
   { int i,n; Console.Write("Dizinin eleman sayısı: ");
     n=Convert.ToInt32(Console.ReadLine());
     int[] a=new int[n]; Console.WriteLine();
     for (i=0;i<n;i++)
      { Console.Write("Dizinin {0}. elemanı: ",i+1);
        a[i]=Convert.ToInt32(Console.ReadLine());}
     tarak_siralama(a,n);
     Console.WriteLine("\nSıralı dizi:\n------------");
     for (i=0;i<n;i++) Console.Write("{0}\t",a[i]);
     Console.ReadLine();  } } }
```

```
Dizinin eleman sayısı: 5

Dizinin 1. elemanı: 2
Dizinin 2. elemanı: 4
Dizinin 3. elemanı: 3
Dizinin 4. elemanı: 1
Dizinin 5. elemanı: 5

Sıralı dizi:
------------
1   2   3   4   5
```

JAVA

```
// Comb sort
import java.util.Scanner;
public class ornek125 {
  public static void tarak_siralama(int[] a, int n)
   { int degisim=1,g,i,gap=n;
     while ((gap!=1) || (degisim==1))
      { gap=(int) (gap/1.3); if (gap<1) gap=1; degisim=0;
        for (i=0;i<n-gap;i++)
         if (a[i]>a[i+gap])
          { g=a[i]; a[i]=a[i+gap]; a[i+gap]=g; degisim=1; }}}
  public static void main(String[] args) {
    int i,n; Scanner klavye=new Scanner(System.in);
    System.out.print("Dizinin eleman sayısı: ");n=klavye.nextInt();
    int[] a=new int[n]; System.out.println();
    for(i=0;i<n;i++)
    { System.out.printf("Dizinin "+(i+1)+". elemanı: ");
      a[i]=klavye.nextInt(); } tarak_siralama(a,n);
      System.out.printf("\nSıralı dizi:\n------------\n");
    for(i=0;i<n;i++) System.out.printf("%d\t",a[i]);     }}
```

MATLAB

```
% Comb sort
clear all; clc; fprintf('Diziyi [a b c d] şeklinde giriniz!\n')
a=input('');n=length(a); a=tarak_siralama(a);
fprintf('\nSıralı dizi:\n----------\n'); disp(a);
---
function a=tarak_siralama(a)
n=length(a); gap=n; degisim=1;
while ((gap~=1) || (degisim==1))
  gap=floor(gap/1.3);
  if (gap<1) gap=1; end; degisim=0;
  for i=1:n-gap
    if (a(i)>a(i+gap))
       g=a(i); a(i)=a(i+gap); a(i+gap)=g; degisim=1; end; end; end;
```

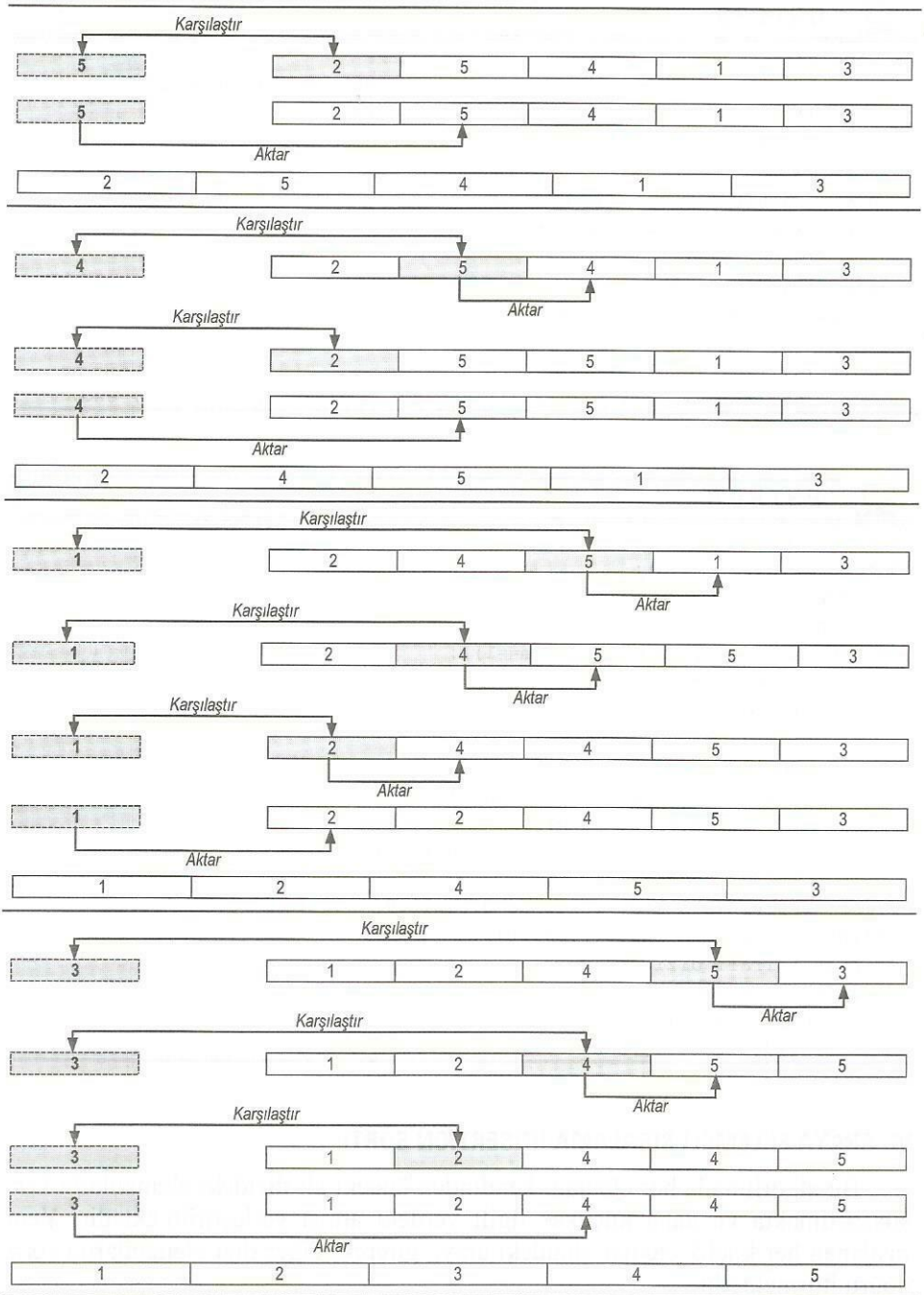
PYTHON

```
# Comb sort
def tarak_siralama(a):
    gap=len(a)
    degisim=1
    while ((gap!=1) or (degisim==1)):
        gap=int(gap/1.3)
        if (gap<1):
            gap=1
        degisim=0
        for i in range(0,len(a)-gap):
            if (a[i]>a[i+gap]):
                a[i],a[i+gap]=a[i+gap],a[i]
                degisim=1
print("Diziyi [a b c d] şeklinde giriniz!")
a=[int(i) for i in input().split()]
print("\nSıralı dizi:\n----------")
tarak_siralama(a)
for i in range(len(a)):
    print("%d\t"%a[i],end='')
```

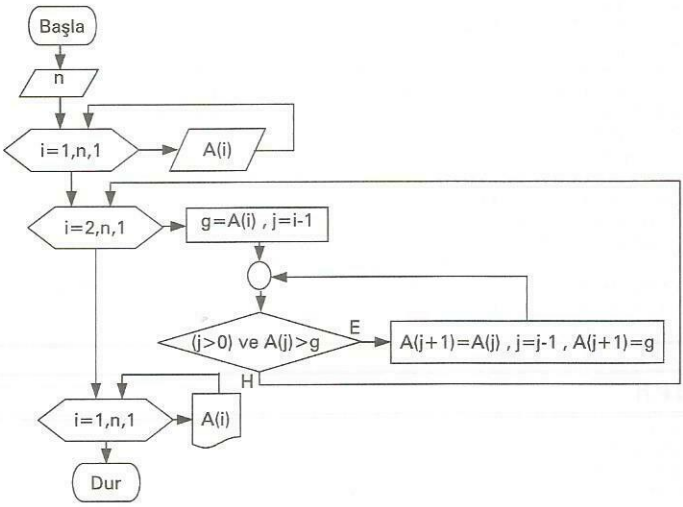
VI. ARAYA EKLEMELİ SIRALAMA (INSERTION SORT)

Bu algoritmada her eleman, kendinden önceki alt dizideki elemanlarla karşılaştırılmakta ve daha küçükse ilgili yerdeki araya yerleştirilmektedir. Yani rastlanan her küçük eleman dizideki araya girerek, diğer dizi elemanlarını sona doğru ittirmektedir.

Örneğin `A=[2,5,4,1,3]` dizisinin "*araya eklemeli sıralama yöntemi*"yle sıralanması, adım adım aşağıda verilmektedir.


Karşılaştır
5 2 5 4 1 3
5 2 5 4 1 3
Aktar
2 5 4 1 3
Karşılaştır
4 2 5 4 1 3
Aktar
Karşılaştır
4 2 5 5 1 3
4 2 5 5 1 3
Aktar
2 4 5 1 3
Karşılaştır
1 2 4 5 1 3
Aktar
Karşılaştır
1 2 4 5 5 3
Aktar
Karşılaştır
1 2 4 4 5 3
Aktar
1 2 2 4 5 3
Aktar
1 2 4 5 3
Karşılaştır
3 1 2 4 5 3
Aktar
Karşılaştır
3 1 2 4 5 5
Aktar
Karşılaştır
3 1 2 4 4 5
3 1 2 4 4 5
Aktar
1 2 3 4 5

Örnek-12.6: Klavyeden girilen n elemanlı A dizisini küçükten büyüğe doğru "*eklemeli (araya eklemeli, yerleştirme) sıralama (Insertion sort)*" algoritmasıyla sıralayan program.



C

```
/* Insertion sort */
#include <stdio.h>
#include <conio.h>
int i,j,n,g,a[100];
void main()
{ clrscr();printf("Dizinin eleman sayısı: ");
  scanf("%u",&n);printf("\n");
  for(i=0;i<n;i++) { printf("A(%u): ",i+1);scanf("%d",&a[i]);}
  for(i=1;i<n;i++)
   { g=a[i]; j=i-1;
     while ((j>-1) && (a[j]>g))
      { a[j+1]=a[j]; j--; a[j+1]=g; }}
printf("\nSıralı dizi:\n"); printf("------------\n");
for(i=0;i<n;i++) printf("%d\t",a[i]); getch(); }
```

C++

```
// Insertion sort
#include <iostream>
using namespace std;
int main()
{ int i,j,n,g;
  system("cls"); cout<<"Dizinin eleman sayısı: "; cin>>n; int a[n];
  cout<<"\n";
  for(i=0;i<n;i++)
  { cout<<"A("<<i+1<<"): "; cin>>a[i]; }
  for(i=1;i<n;i++)
   { g=a[i]; j=i-1;
     while ((j>-1) && (a[j]>g))
      { a[j+1]=a[j]; j--; a[j+1]=g; }}
  cout<<"\nSıralı dizi:\n------------\n";
  for(i=0;i<n;i++) cout<<a[i]<<"\t";
  system("pause"); return 0; }
```

C#

```
// Insertion sort
using System;
namespace ornek126
{ class Program
   { static void Main(string[] args)
      { int i,j,n,g;
        Console.Write("Dizinin eleman sayısı: ");
        n=Convert.ToInt32(Console.ReadLine());
        int[] a = new int[n]; Console.WriteLine();
        for (i=0;i<n;i++)
         { Console.Write("A({0}): ",i+1);
           a[i]=Convert.ToInt32(Console.ReadLine()); }
        for (i=1;i<n;i++)
         { g=a[i];j=i-1;
           while ((j>-1) && (a[j]>g))
            { a[j+1]=a[j]; j--; a[j+1]=g; }}
        Console.WriteLine("\nSıralı dizi:\n------------");
        for (i=0;i<n;i++) Console.Write("{0}\t", a[i]);
        Console.ReadLine(); }}}
```

JAVA

```
// Insertion sort
import java.util.Scanner;
public class ornek126 {
 public static void main(String[] args) {
  int i,j,n,g; Scanner klavye=new Scanner(System.in);
  System.out.print("Dizinin eleman sayısı: ");n=klavye.nextInt();
  int[] a=new int[n]; System.out.println();
  for(i=0;i<n;i++)
   { System.out.printf("A("+(i+1)+"): "); a[i]=klavye.nextInt(); }
  for (i=1;i<n;i++)
   { g=a[i];j=i-1;
     while ((j>-1) && (a[j]>g))
      { a[j+1]=a[j]; j--; a[j+1]=g; }}
  System.out.printf("\nSıralı dizi:\n------------\n");
  for(i=0;i<n;i++) System.out.printf("%d\t",a[i]); }}
```

```
Dizinin eleman sayısı: 6

A(1): 6
A(2): 5
A(3): 4
A(4): 1
A(5): 3
A(6): 2

Sıralı dizi:
------------
1  2  3  4  5  6
```

MATLAB

```
% Insertion sort
clear all; clc; fprintf('Diziyi [a b c d] şeklinde giriniz!\n')
a=input(''); n=length(a);
for i=2:n
    g=a(i);j=i-1;
    while ((j>0) && (a(j)>g))
        a(j+1)=a(j); j=j-1; a(j+1)=g;
    end;
end;
fprintf('\nSıralı dizi:\n------------\n'); disp(a);
```

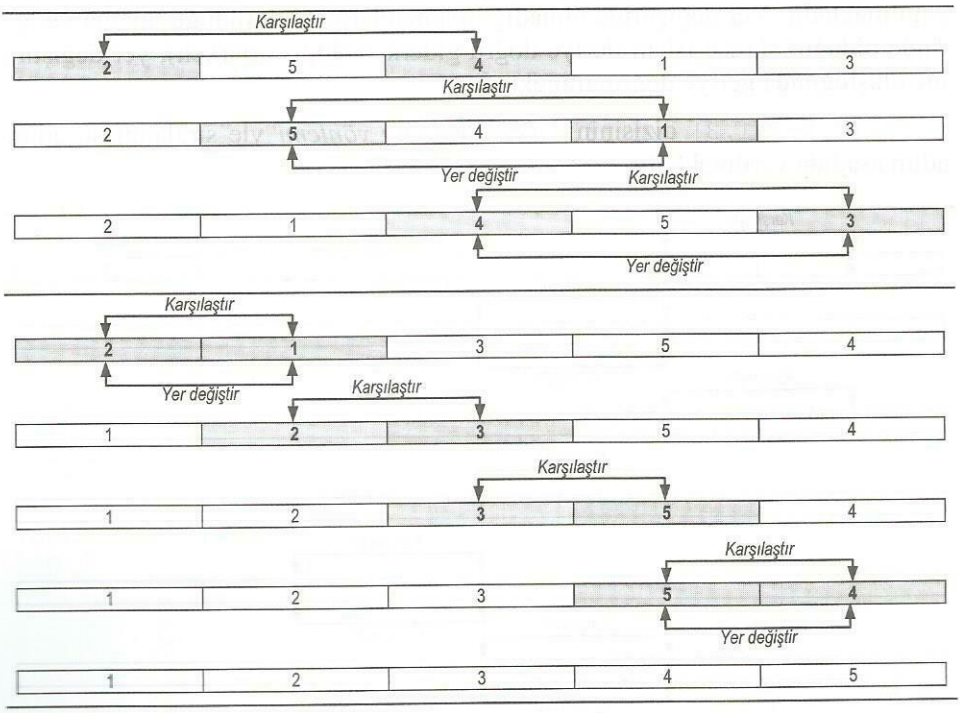
PYTHON

```
# Insertion sort
print("Diziyi [a b c d] şeklinde giriniz!")
a=[int(i) for i in input().split()]
n=len(a)
for i in range(1,n):
    g=a[i]
    j=i-1
    while ((j>-1) & (a[j]>g)):
        a[j+1]=a[j]
        j=j-1
        a[j+1]=g
print("\nSıralı dizi:\n------------")
for i in range(n):
    print("%d\t"%a[i],end='')
```

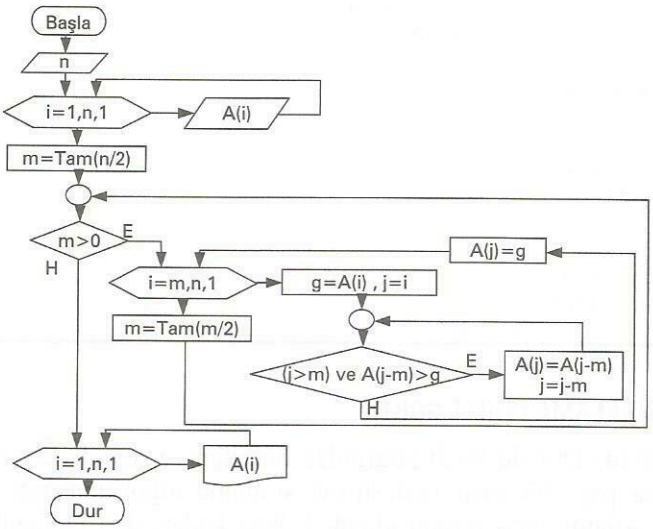
VII. KABUK SIRALAMA (SHELL SORT)

1959 yılında Donald Shell tarafından önerilen ve soyadıyla anılan bu algoritma, atlama payı bir olan yerleştirme sıralama algoritmasının iyileştirilmiş durumudur. Atlama payı birden küçük kalana kadar, dizi elemanlarını atlama payına göre karşılaştırarak sıralamaktadır.

`A=[2,5,4,1,3]` dizisinin "*kabuk sıralama yöntemi*"yle sıralanması, adım adım aşağıda verilmektedir.



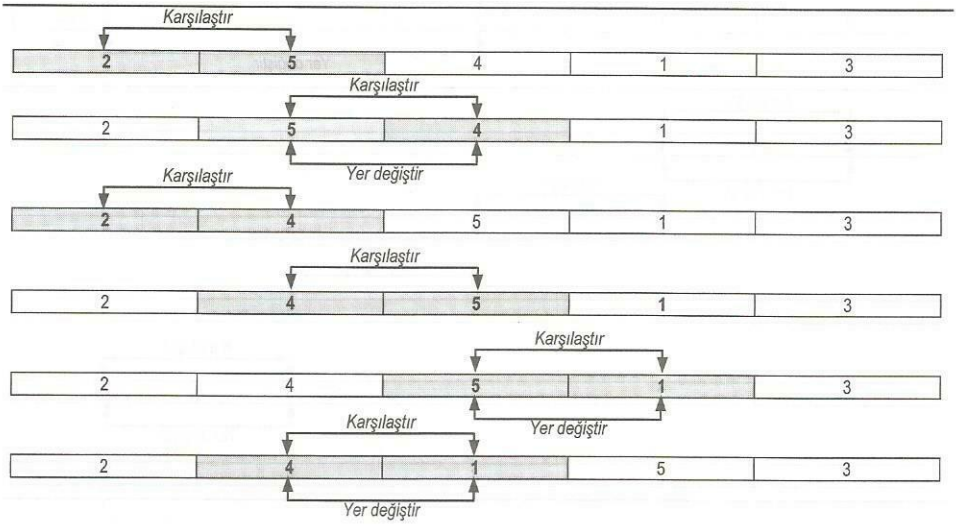
Örnek-12.7: Klavyeden girilen *n* elemanlı *A* dizisini küçükten büyüğe doğru *kabuk sıralama (Shell sort)* algoritmasıyla sıralayan program.

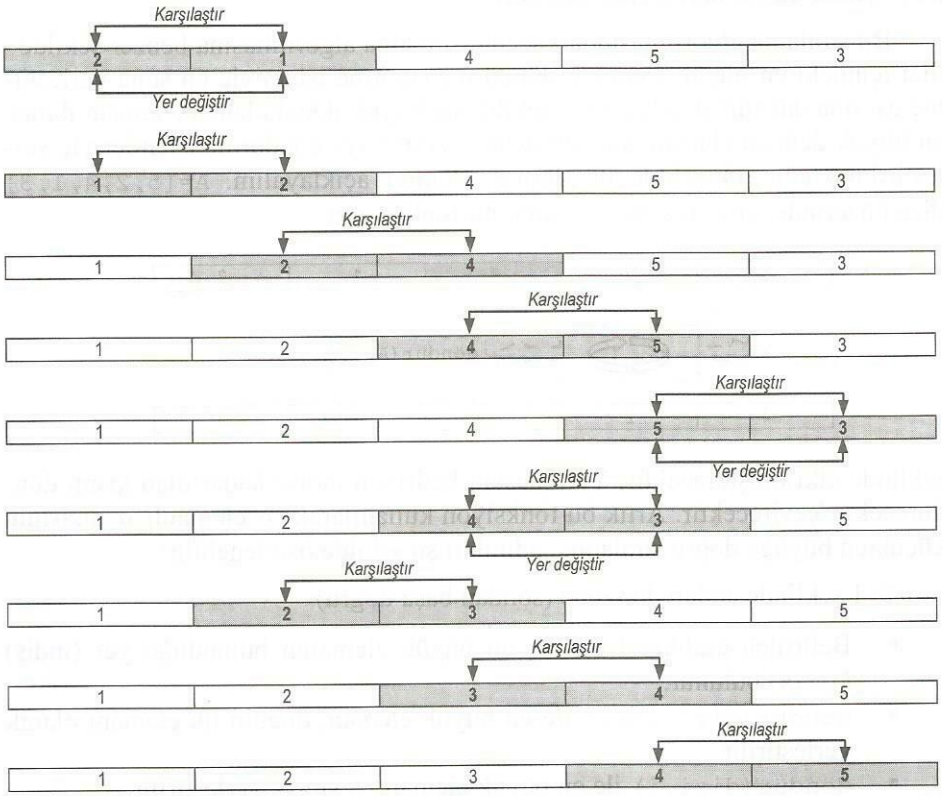


VIII. CÜCE SIRALAMA (GNOME SORT, STUPID SORT)

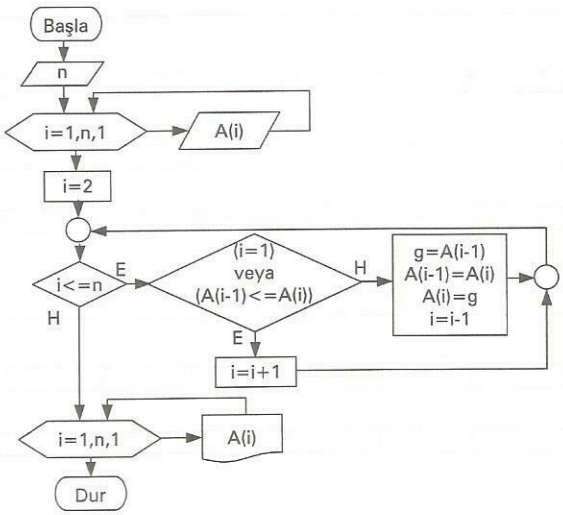
Araya eklemeli sıralama algoritmasına benzeyen cüce sıralamada; eleman yerleştirmeleri, kabarcık sıralamadaki gibi birçok yer değiştirme sonucunda yapılmaktadır. Yer değiştirme olmadığı (elemanların sıralı olduğu, uygun yerlerinde olduğu) sürece işlem ileriye doğru giderken (indis artarken), yer değiştirme oluştuğunda geriye doğru gitmektedir.

`A=[2,5,4,1,3]` dizisinin "*cüce sıralama yöntemi*"yle sıralanması, adım adım aşağıda verilmektedir.



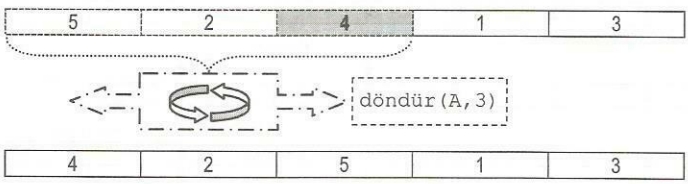


Örnek-12.8: Klavyeden girilen *n* elemanlı *A* dizisini küçükten büyüğe doğru *cüce sıralama (Gnome sort, stupid sort)* algoritmasıyla sıralayan program.



IX. PANKEK SIRALAMA (PANCAKE SORT)

Bu sıralama algoritmasında - seçme sıralama algoritmasına benzer şekilde - dizi içindeki en büyük elemanı, döndürme/çevirme işlemiyle en sona yerleştirme esasına dayanmaktadır. Aynı şekilde en büyük ikinci eleman, sondan ikinci; en büyük üçüncü eleman, sondan üçüncü sıraya vb. döndürme işlemleriyle yerleştirilmektedir. Öncelikle döndürme işlemini açıklayalım. `A=[5,2,4,1,3]` dizisi üzerinde "`döndür(A,3)`" komutu/fonksiyonu

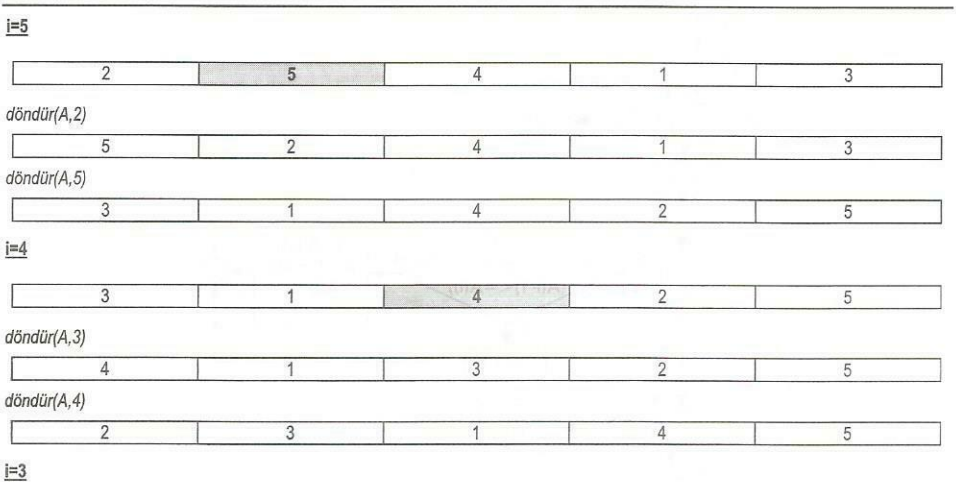


şeklinde etki oluşturacaktır. Yani baştan belirtilen indise kadar olan kısmı döndürecektir/çevirecektir. Artık bu fonksiyon kullanılarak n elemanlı a dizisinin küçükten büyüğe doğru sıralanma adımları şu şekilde özetlenebilir:

$i=n,2,-1$ şeklinde açılan döngüde (sondan başa doğru)

- Belirtilen aralıktaki dizide, en büyük elemanın bulunduğu yer (indis) (`imax`) bulunur.
- `döndür(dizi,imax)` ile en büyük eleman, dizinin ilk elemanı olarak yerleştirilir.
- `döndür(dizi,i)` ile en büyük eleman, `i`. sıraya yerleştirilir.

`A=[2,5,4,1,3]` dizisinin "*pankek sıralama*" algoritmasıyla sıralanması, adım adım aşağıda verilmektedir.



2	3	1	4	5

döndür(A,2)

3	2	1	4	5

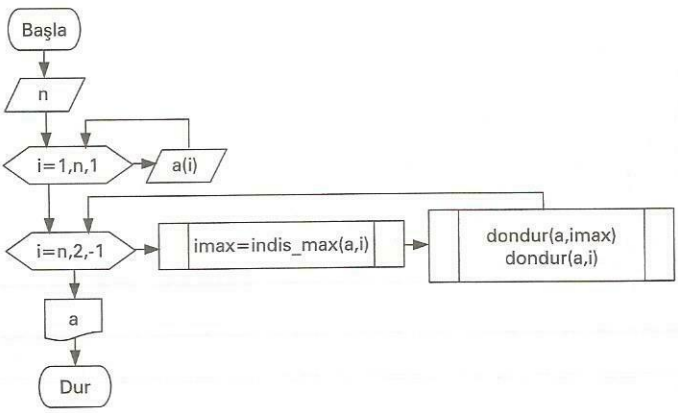
döndür(A,3)

1	2	3	4	5

i=2

1	2	3	4	5

Örnek-12.9: Klavyeden girilen *n* elemanlı *a* dizisini küçükten büyüğe doğru *pancake sort* algoritmasıyla sıralayan program.



C

```
/* Pancake sort */
#include <stdio.h>
#include <conio.h>
int a[100];
void dondur(int a[],int k)
{ int j=0,g;
  while (j<k)
    {   g=a[j]; a[j]=a[k]; a[k]=g; j++; k--; }}
int indis_max(int a[],int k)
{ int i,max=0;
  for (i=1;i<=k;i++) if (a[max]<a[i]) max=i; return max; }
void pancake_sort(int a[],int n)
{ int i,imax;
  for (i=n-1;i>0;i--)
   { imax=indis_max(a,i);
     if (imax!=i)
      { if (imax!=0) dondur(a,imax); dondur(a,i); }}}
void main()
{ int i,n;
  clrscr();printf("Dizinin eleman sayısı: "); scanf("%u",&n);
  for (i=0;i<n;i++)
   { printf("A(%d): ",i+1); scanf("%d",&a[i]); }
  pancake_sort(a,n);
  printf("\nSıralı dizi:\n");
  for (i=0;i<n;i++) printf("%d\t",a[i]);
  getch(); }
```

C++

```
// Pancake sort
#include <iostream>
using namespace std;
void dondur(int a[],int k)
{ int j=0,g;
  while (j<k)
   {   g=a[j]; a[j]=a[k]; a[k]=g; j++; k--; }}
int indis_max(int a[],int k)
{ int i,max=0;
  for (i=1;i<=k;i++) if (a[max]<a[i]) max=i; return max; }
void pancake_sort(int a[],int n)
{ int i,imax;
  for (i=n-1;i>0;i--)
   { imax=indis_max(a,i);
     if (imax!=i)
      { if (imax!=0) dondur(a,imax); dondur(a,i); }}}
int main()
{ int i,n;
  system("cls"); cout<<"Dizinin eleman sayısı: "; cin>>n; int a[n];
  cout<<"\n";
  for(i=0;i<n;i++) { cout<<"A("<<i+1<<"): "; cin>>a[i]; }
  pancake_sort(a,n);
  cout<<"\nSıralı dizi:\n------------\n";
  for(i=0;i<n;i++) cout<<a[i]<<"\t";
  system("pause"); return 0; }
```

C#

```
// Pancake sort
using System;
namespace ornek129
{ class Program
   { static void dondur(int[] a,int k)
      { int j=0,g;
        while (j<k)
         { g=a[j]; a[j]=a[k]; a[k]=g; j++; k--; }}
     static int indis_max(int[] a,int k)
      { int i,max=0;
        for (i=1;i<=k;i++) if (a[max]<a[i]) max=i; return max; }
     static void pancake_sort(int[] a,int n)
      { int i,imax;
        for (i=n-1;i>0;i--)
         { imax=indis_max(a,i);
           if (imax!=i)
            { if (imax!=0) dondur(a,imax); dondur(a,i); }}}
     static void Main(string[] args)
      { int i,n; Console.Write("Dizinin eleman sayısı: ");
        n=Convert.ToInt32(Console.ReadLine());
        int[] a=new int[n]; Console.WriteLine();
        for (i=0;i<n;i++)
         { Console.Write("A({0}): ",i+1);
           a[i]=Convert.ToInt32(Console.ReadLine()); }
        pancake_sort(a, n);
        Console.WriteLine("\nSıralı dizi:\n------------");
        for (i=0;i<n;i++) Console.Write("{0}\t",a[i]);
        Console.ReadLine(); }}}
```

JAVA

```
// Pancake sort
import java.util.Scanner;
public class ornek129 {
 static void dondur(int[] a,int k)
  { int j=0,g;
    while (j<k)
     { g=a[j]; a[j]=a[k]; a[k]=g; j++; k--; }}
 static int indis_max(int[] a,int k)
  { int i,max=0;
    for (i=1;i<=k;i++) if (a[max]<a[i]) max=i; return max; }
  static void pancake_sort(int[] a,int n)
      { int i,imax;
        for (i=n-1;i>0;i--)
         { imax=indis_max(a,i);
           if (imax!=i)
            { if (imax!=0) dondur(a,imax); dondur(a,i); }}}
  public static void main(String[] args) {
    int i,n;
    Scanner klavye=new Scanner(System.in);
    System.out.print("Dizinin eleman sayısı: ");n=klavye.nextInt();
    int[] a=new int[n]; System.out.println();
    for(i=0;i<n;i++)
    { System.out.printf("A("+(i+1)+"): ");
      a[i]=klavye.nextInt(); }
    pancake_sort(a,n);
    System.out.printf("\nSıralı dizi:\n------------\n");
    for(i=0;i<n;i++) System.out.printf("%d\t",a[i]);  }}
```

MATLAB

```
% Pancake sort
clear all; clc; fprintf('Diziyi [a b c d] şeklinde giriniz!\n')
a=input(''); n=length(a); a=pancake_sort(a,n);
fprintf('\nSıralı dizi:\n------------\n'); disp(a);
---
function a=pancake_sort(a,n)
for i=n:-1:2
    imax=i;
    for j=i:-1:1
        if a(j)>=a(imax)
            imax=j; end; end;
    if i~=imax
        a(imax:-1:1)=a(1:imax); a(i:-1:1)=a(1:i); end; end;
```

```
Diziyi [a b c d] şeklinde giriniz!
[9 7 5 3 1 8 6 4 2]

Sıralı dizi:
------------
   1   2   3   4   5   6   7   8   9
```

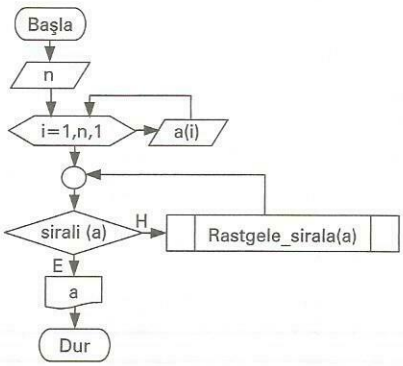
PYTHON

```
# Pancake sort
def pancake_sort(a,n):
    for i in range(n,1,-1):
        imax=max(range(i),key=a.__getitem__)
        if imax+1!=i:
            if imax!=0:
                a[:imax+1]=reversed(a[:imax+1])
            a[:i]=reversed(a[:i])
print("Diziyi [a b c d] şeklinde giriniz!")
a=[int(i) for i in input().split()]
print("\nSıralı dizi:\n------------")
n=len(a)
pancake_sort(a,n)
for i in range(n):
    print("%d\t"%a[i],end='')
```

X. RASTGELE SIRALAMA (RANDOM SORT, BOGO SORT, MONKEY SORT, SHOTGUN SORT)

Eğitim amaçlı kullanılan bu algoritmada; sıralanacak veri dizisi, rastgele dizilir ve sıralı olup olmadığı kontrol edilir. Bu rastgele dizilim ve kontrol, dizi sıralı hale gelinceye kadar devam eder.

Örnek-12.10: Klavyeden girilen *n* elemanlı *a* dizisini küçükten büyüğe doğru "*rastgele sıralama (Random sort)*" algoritmasıyla sıralayan program.

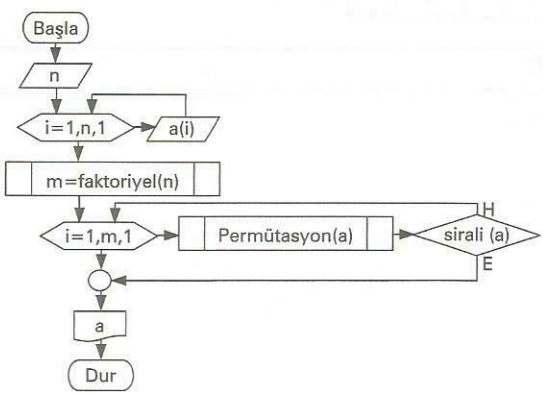


XI. PERMÜTASYON SIRALAMA (PERMUTATION SORT)

Bu sıralama algoritmasında, dizinin tüm permütasyonları - sıralı hale gelinceye kadar - denenmektedir. Örneğin `A=[5,2,4]` dizisinin "*permütasyon sıralama*" yöntemiyle sıralanması, adım adım aşağıda verilmektedir.

Adım	Dizi	Sıralı mı?	Devam mı?
1	5, 2, 4	Hayır	Evet
2	5, 4, 2	Hayır	Evet
3	2, 5, 4	Hayır	Evet
4	2, 4, 5	Evet	Hayır
5	-	-	-

Örnek-12.11: Klavyeden girilen *n* elemanlı *a* dizisini küçükten büyüğe doğru "*permütasyon sıralama (Permutation sort)*" algoritmasıyla sıralayan program.



XII. BİRLEŞTİRME/BİRLEŞTİRMELİ SIRALAMA (MERGE SORT)

Bu algoritmada dizi, alt dizilere ayrıştırılarak herbiri ayrı ayrı sıralanıp birleştirilmektedir. Yinelemeli olan bu algoritmanın çalışması aşağıdaki gibi özetlenebilir:

i. Diziyi eşit iki alt diziye ayır.

ii. Bu ayırma işlemini, alt dizilerde en fazla 2 eleman kalıncaya kadar sürdür.

iii. Alt dizileri kendi içinde küçükten büyüğe doğru sırala.

iv. Sıralı alt dizileri birleştir.

Sıralı iki alt (1. ve 2.) dizi birleştirilirken;

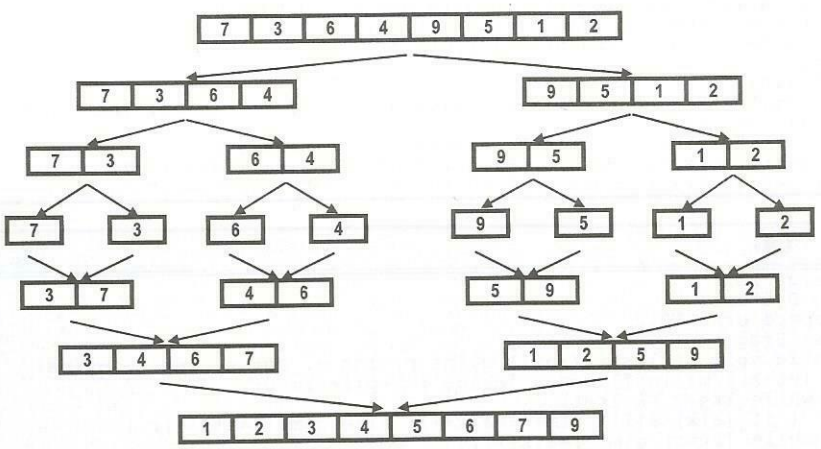
a. Her iki alt dizi elemanları bitinceye kadar, sıradaki elemanları alınarak karşılaştırılır.

a.1. Eğer 1. alt dizinin sıradaki elemanı, 2. alt dizinin sıradaki elemanından küçükse, oluşan birleştirilmiş dizinin sıradaki elemanı olarak atanır. 1. dizi için sıradaki elemanı gösteren indis ve oluşan birleştirilmiş dizinin indisi bir arttırılır.

a.2. Eğer 1. alt dizinin sıradaki elemanı, 2. dizinin sıradaki elemanından büyükse, 2. alt dizinin sıradaki elemanı; oluşan birleştirilmiş dizinin sıradaki elemanı olarak atanır. 2. dizi için sıradaki elemanı gösteren indis ve oluşan birleştirilmiş dizinin indisi bir arttırılır.

b. Eğer herhangi bir alt dizide sona ulaşılmamışsa; bu dizinin elemanları, oluşan birleştirilmiş diziye eklenir.

`A=(7,3,6,4,9,5,1,2)` örnek dizisinin "Merge sort" algoritması ile küçükten büyüğe doğru sıralanması, Şekil 12.2'de verilmektedir.



Şekil 12.2 "Merge sort" ile sıralama işlemi

Örnek-12.12: Klavyeden girilen *n* elemanlı *a* dizisini küçükten büyüğe doğru "*birleştirme sıralama (Merge sort)*" algoritmasıyla sıralayan program.

C

```
/* Merge sort */
#include <stdio.h>
#include <conio.h>
void birlestir(int a[],int p, int r, int s, int t)
{ int k,l,m,g[100]; k=p;l=s;m=0;
  while (k<=r && l<=t)
   {   if (a[k]<a[l]) g[m++]=a[k++]; else g[m++]=a[l++]; }
  while (k<=r) g[m++]=a[k++];
  while (l<=t) g[m++]=a[l++];
  for (k=p,l=0;k<=t;k++,l++) a[k]=g[l]; }
void birlestirme_siralama(int a[],int x, int y)
{ int orta;
  if (x<y)
   { orta=(x+y)/2; birlestirme_siralama(a,x,orta);
     birlestirme_siralama(a,orta+1,y);birlestir(a,x,orta,orta+1,y);}}
void main()
{ int i,n,a[100];
  clrscr();printf("Dizinin eleman sayısı: "); scanf("%u",&n);
  for(i=0;i<n;i++)
      {      printf("Dizinin %d. elemanı: ",i+1); scanf("%d",&a[i]);
}
  birlestirme_siralama(a,0,n-1); printf("\nSıralı dizi:\n");
  for(i=0;i<n;i++) printf("%d\t",a[i]); getch(); }
```

C++

```
// Merge sort
#include <iostream>
using namespace std;
void birlestir(int a[],int p, int r, int s, int t,int es)
{ int k,l,m,g[es]; k=p;l=s;m=0;
  while (k<=r && l<=t)
   {   if (a[k]<a[l]) g[m++]=a[k++]; else g[m++]=a[l++]; }
  while (k<=r) g[m++]=a[k++];
  while (l<=t) g[m++]=a[l++];
  for (k=p,l=0;k<=t;k++,l++) a[k]=g[l]; }
void birlestirme_siralama(int a[],int x, int y,int es)
{ int orta;
  if (x<y)
   { orta=(x+y)/2; birlestirme_siralama(a,x,orta,es);
     birlestirme_siralama(a,orta+1,y,es);
     birlestir(a,x,orta,orta+1,y,es);} }
int main()
{ int i,n;
  system("cls"); cout<<"Dizinin eleman sayısı: "; cin>>n; int a[n];
  for(i=0;i<n;i++)
  { cout<<"Dizinin "<<i+1<<". elemanı: "; cin>>a[i]; }
  birlestirme_siralama(a,0,n-1,n); cout<<"\nSıralı dizi:\n";
  for(i=0;i<n;i++) cout<<a[i]<<"\t"; system("pause"); return 0; }
```

C#

```
// Merge sort
using System;
namespace ornek1212
{class Program {
  static void birlestir(int[] a,int p, int r, int s, int t,int es)
   { int k,l,m; int[] g=new int[es]; k=p;l=s;m=0;
     while (k<=r && l<=t)
      { if (a[k]<a[l]) g[m++]=a[k++]; else g[m++]=a[l++]; }
     while (k<=r) g[m++]=a[k++];
     while (l<=t) g[m++]=a[l++];
     for (k=p,l=0;k<=t;k++,l++) a[k]=g[l];}
  static void birlestirme_siralama(int[] a,int x, int y,int es)
   { int orta;
```

```
    if (x<y)
     { orta=(x+y)/2; birlestirme_siralama(a,x,orta,es);
       birlestirme_siralama(a,orta+1,y,es);
       birlestir(a,x,orta,orta+1,y,es);}}
  static void Main(string[] args)
   { int i,n;
     Console.Write("Dizinin eleman sayısı: ");
     n=Convert.ToInt32(Console.ReadLine()); int[] a=new int[n];
     for (i=0;i<n;i++)
      { Console.Write("Dizinin {0}. elemanı: ",i+1);
        a[i]=Convert.ToInt32(Console.ReadLine());}
     birlestirme_siralama(a,0,n-1,n);
     Console.WriteLine("\nSıralı dizi:");
     for (i=0;i<n;i++) Console.Write("{0}\t", a[i]);
     Console.ReadLine();}}}
```

JAVA

```
// Merge sort
import java.util.Scanner;
public class ornek1212 {
public static void birlestir(int[] a,int p, int r, int s, int t,int es)
 { int k,l,m; int[] g=new int[es]; k=p;l=s;m=0;
   while (k<=r && l<=t)
    { if (a[k]<a[l]) g[m++]=a[k++]; else g[m++]=a[l++]; }
   while (k<=r) g[m++]=a[k++];
   while (l<=t) g[m++]=a[l++];
   for (k=p,l=0;k<=t;k++,l++) a[k]=g[l];}
public static void birlestirme_siralama(int[] a,int x, int y,int es)
 { int orta;
   if (x<y)
    { orta=(x+y)/2; birlestirme_siralama(a,x,orta,es);
      birlestirme_siralama(a,orta+1,y,es);
      birlestir(a,x,orta,orta+1,y,es);}}
public static void main(String[] args) {
 int i,n; Scanner klavye=new Scanner(System.in);
 System.out.print("Dizinin eleman sayısı: ");n=klavye.nextInt();
 int[] a=new int[n];
 for(i=0;i<n;i++)
  { System.out.printf("Dizinin "+(i+1)+". elemanı: ");
    a[i]=klavye.nextInt(); } birlestirme_siralama(a,0,n-1,n);
 System.out.printf("\nSıralı dizi:\n");
 for(i=0;i<n;i++) System.out.printf("%d\t",a[i]); }}
```

MATLAB

```
% Merge sort
clear all; clc; fprintf('Diziyi [a b c d] şeklinde giriniz!\n')
a=input('');n=length(a);
b=birlestirme_siralama(a);
fprintf('Sıralı dizi:\n'); disp(b);
---
function z=birlestir(x,y)
n=length(x); m=length(y);z=zeros(1,n+m); i=1;j=1;
for k=1:(n+m)
if i>n
    z(k)=y(j); j=j+1;
elseif j>m
    z(k)=x(i); i=i+1;
elseif x(i)<=y(j)
    z(k)=x(i); i=i+1;
else
    z(k)=y(j); j=j+1;
end; end
---
function y=birlestirme_siralama(x)
n=length(x);
if n==1
    y=x;
else
    m=floor(n/2); y1=birlestirme_siralama(x(1:m));
    y2=birlestirme_siralama(x(m+1:n)); y=birlestir(y1,y2);
end
```

PYTHON

```
# Merge sort
def birlestirme_siralama(a):
    if len(a)>1:
        orta=len(a)//2; sol=a[:orta]; sag=a[orta:]
        birlestirme_siralama(sol); birlestirme_siralama(sag)
        i=0; j=0; k=0
        while ((i<len(sol)) and (j<len(sag))):
            if (sol[i]<sag[j]):
                a[k]=sol[i]; i=i+1
            else:
                a[k]=sag[j]; j=j+1
            k=k+1
        while (i<len(sol)):
            a[k]=sol[i]; i=i+1; k=k+1
        while (j<len(sag)):
            a[k]=sag[j]; j=j+1; k=k+1
print("Diziyi [a b c d] şeklinde giriniz!")
a=[int(i) for i in input().split()]
birlestirme_siralama(a); print("\nSıralı dizi:")
for i in range(len(a)):
    print("%d\t"%a[i],end='')
```

```
Diziyi [a b c d] şeklinde giriniz!
8 6 4 2 9 7 5 3 1

Sıralı dizi:
1       2       3       4       5       6       7       8       9
```

XIII. HIZLI SIRALAMA (QUICK SORT)

Yinelemeli olan ve sık kullanılan bu algoritmanın çalışması aşağıdaki gibi özetlenebilir:

i. Belirli kriterlere (ilk eleman, son eleman, ortadaki eleman, rastgele, vb.) göre eksen eleman (mihenk, pivot) seç.

ii. Diziyi üç alt diziye ayrıştır:
   - ✓ eksen elemandan küçük olan elemanları birinci (soldaki) diziye,
   - ✓ eksen elemanı ikinci (ortadaki) diziye,
   - ✓ eksen elemandan büyük olanları da üçüncü (sağdaki) diziye yerleştir.

iii. Aynı işlemi birinci (soldaki) ve üçüncü (sağdaki) alt dizilere, tek elemanlı kalıncaya kadar uygula.

iv. Sıralı alt dizileri birleştir.

Örnek-12.13: Klavyeden girilen n elemanlı A dizisini küçükten büyüğe doğru "*hızlı sıralama (Quick sort)*" algoritmasıyla sıralayan program.

C

```
/* Quick sort */
#include <stdio.h>
#include <conio.h>
void hizli_siralama(int a[],int sol, int sag)
{ int i=sol,j=sag,g,pivot=a[(sol+sag)/2];
  while (i<=j)
   { while (a[i]<pivot) i++;
     while (a[j]>pivot) j--;
```

```
      if (i<=j) { g=a[i]; a[i]=a[j]; a[j]=g; i++; j--; } }
   if (sol<j) hizli_siralama(a,sol,j);
   if (i<sag) hizli_siralama(a,i,sag);  }
void main()
{ int i,n,a[100];
  clrscr();printf("Dizinin eleman sayısı: "); scanf("%u",&n);
  printf("\n");
  for(i=0;i<n;i++)
    {  printf("Dizinin %d. elemanı: ",i+1); scanf("%d",&a[i]); }
  hizli_siralama(a,0,n-1); printf("\nSıralı dizi:\n------------\n");
  for(i=0;i<n;i++) printf("%d\t",a[i]); getch(); }
```

```
Dizinin eleman sayısı: 4

Dizinin 1. elemanı: 7
Dizinin 2. elemanı: 1
Dizinin 3. elemanı: 0
Dizinin 4. elemanı: 2

Sıralı dizi:
------------
0   1   2   7
```

C++

```
// Quick sort
#include <iostream>
using namespace std;
void hizli_siralama(int a[],int sol, int sag)
{ int i=sol,j=sag,g,pivot=a[(sol+sag)/2];
  while (i<=j)
   { while (a[i]<pivot) i++;
     while (a[j]>pivot) j--;
     if (i<=j) { g=a[i]; a[i]=a[j]; a[j]=g; i++; j--; } }
  if (sol<j) hizli_siralama(a,sol,j);
  if (i<sag) hizli_siralama(a,i,sag);  }
int main()
{ int i,n;
  system("cls"); cout<<"Dizinin eleman sayısı: "; cin>>n; int a[n];
  cout<<"\n";
  for(i=0;i<n;i++)
  { cout<<"Dizinin "<<i+1<<". elemanı: "; cin>>a[i]; }
  hizli_siralama(a,0,n-1); cout<<"\nSıralı dizi:\n------------\n";
  for(i=0;i<n;i++) cout<<a[i]<<"\t";   system("pause"); return 0; }
```

C#

```
// Quick sort
using System;
namespace ornek1213
{class Program {
  static void hizli_siralama(int[] a,int sol, int sag)
   { int i=sol,j=sag,g,pivot=a[(sol+sag)/2];
     while (i<=j)
      { while (a[i]<pivot) i++;
        while (a[j]>pivot) j--;
        if (i<=j) { g=a[i]; a[i]=a[j]; a[j]=g; i++; j--; } }
     if (sol<j) hizli_siralama(a,sol,j);
     if (i<sag) hizli_siralama(a,i,sag);  }
  static void Main(string[] args)
   { int i,n; Console.Write("Dizinin eleman sayısı: ");
     n=Convert.ToInt32(Console.ReadLine());
     int[] a=new int[n]; Console.WriteLine();
     for (i=0;i<n;i++)
      { Console.Write("Dizinin {0}. elemanı: ",i+1);
        a[i]=Convert.ToInt32(Console.ReadLine()); }
     hizli_siralama(a,0,n-1);
     Console.WriteLine("\nSıralı dizi:\n------------");
     for (i=0;i<n;i++) Console.Write("{0}\t", a[i]);
     Console.ReadLine();     } } }
```

JAVA

```
// Quick sort
import java.util.Scanner;
public class ornek1213 {
  public static void hizli_siralama(int[] a,int sol, int sag)
   { int i=sol,j=sag,g,pivot=a[(sol+sag)/2];
     while (i<=j)
      { while (a[i]<pivot) i++;
        while (a[j]>pivot) j--;
        if (i<=j) { g=a[i]; a[i]=a[j]; a[j]=g; i++; j--; } }
     if (sol<j) hizli_siralama(a,sol,j);
     if (i<sag) hizli_siralama(a,i,sag);  }
  public static void main(String[] args) {
    int i,n; Scanner klavye=new Scanner(System.in);
    System.out.print("Dizinin eleman sayısı: ");n=klavye.nextInt();
    int[] a=new int[n]; System.out.println();
    for(i=0;i<n;i++)
    { System.out.printf("Dizinin "+(i+1)+". elemanı: ");
      a[i]=klavye.nextInt(); } hizli_siralama(a,0,n-1);
    System.out.printf("\nSıralı dizi:\n------------\n");
    for(i=0;i<n;i++) System.out.printf("%d\t",a[i]);     }}
```

MATLAB

```
% Quick sort
clear all; clc; fprintf('Diziyi [a b c d] şeklinde giriniz!\n')
a=input('');n=length(a); a=hizli_siralama(a,1,n);
fprintf('Sıralı dizi:\n------------\n'); disp(a);
---
function a=hizli_siralama(a,sol,sag)
i=sol;j=sag;pivot=a(floor((sol+sag)/2));
while (i<=j)
 while (a(i)<pivot) i=i+1; end;
 while (a(j)>pivot) j=j-1; end;
 if (i<=j)
  g=a(i); a(i)=a(j); a(j)=g; i=i+1; j=j-1; end; end;
if (sol<j) a=hizli_siralama(a,sol,j); end;
if (i<sag) a=hizli_siralama(a,i,sag);  end;
```

PYTHON

```
# Quick sort
import math
def hizli_siralama(a,sol,sag):
    i=sol; j=sag; pivot=a[math.floor((sol+sag)/2)]
    while (i<=j):
        while (a[i]<pivot):
            i+=1
        while (a[j]>pivot):
            j-=1
        if (i<=j):
            g=a[i]; a[i]=a[j]; a[j]=g; i+=1; j-=1
    if (sol<j):
        hizli_siralama(a,sol,j)
    if (i<sag):
        hizli_siralama(a,i,sag)
print("Diziyi [a b c d] şeklinde giriniz!")
a=[int(i) for i in input().split()]
print("\nSıralı dizi:\n------------")
hizli_siralama(a,0,len(a)-1)
for i in range(len(a)):
    print("%d\t"%a[i],end='')
```

XIV. SAYMA SIRALAMA (COUNTING SORT)

Bu algoritma ile pozitif tamsayılardan oluşan ve eleman değerleri çok büyük olmayan diziler sıralanabilir. Algoritmanın işlem adımları aşağıdaki gibi özetlenebilir:

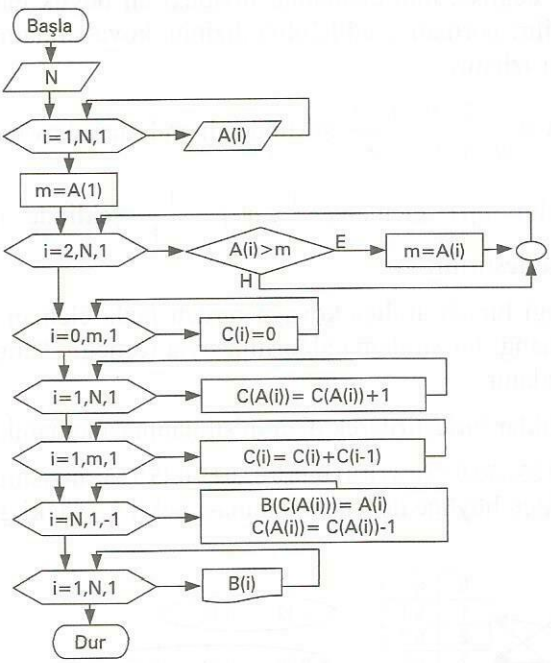
i. Sıralanacak olan N elemanlı A dizisinin elemanları $[0..m]$ aralığında ise m elemanlı yedek bir C dizisi tanımlanır.

ii. Başlangıç değerleri olarak C dizisinin tüm elemanları sıfırlanır.

iii. A dizisi bir kere taranarak herbir elemandan kaç tane olduğu sayılır. Yani her $A[i]$ için $C[A[i]]$ değeri bir arttırılır.

iv. Oluşmuş olan C dizisinin ardışık iki elemanı toplanarak, A dizisinin elemanlarının sıralanmış dizideki yerleri bulunarak yerleştirilir ve C dizisindeki ilgili değer, bir azaltılır.

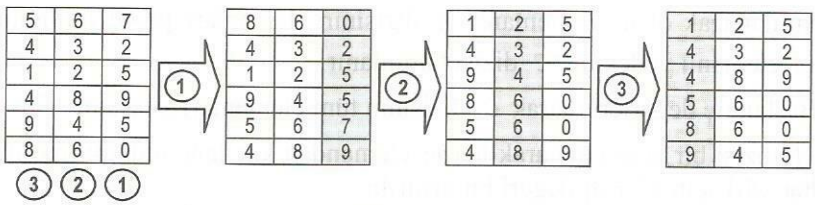
Örnek-12.14: Klavyeden girilen N elemanlı A dizisini küçükten büyüğe doğru "*sayma sıralama (Counting sort)*" algoritmasıyla sıralayan program.



XV. HANE/BASAMAK SIRALAMA (RADIX SORT)

Tamsayıların sıralanabildiği bu algoritma; birler basamağından (en düşük değerlikli basamaktan) itibaren sayı hanelerinin kendi aralarında herhangi bir durağın sıralamayla, yani elemanların bağıl konumlarını değiştirmeyen bir yöntemle (örneğin sayma sıralama algoritması) sıralanması esasına dayanır. Bu algoritma ile sıralanacak tamsayıların basamak sayısı aynı olmalı, aksi durumda

önüne sıfırlar eklenerek en yüksek basamaklı sayının hane sayısına tüm tamsayılar tamamlanmalıdır. Örneğin üç haneli tamsayılardan oluşan `A=(567,432,125,489,945,860)` dizisinin büyükten küçüğe doğru sıralanması Şekil 12.3'teki gibi olur.



Şekil 12.3 "Radix sort" ile örnek sıralama işlemi

XVI. KOVA SIRALAMA (BUCKET SORT)

Elemanları [0..1) aralığında düzgün dağılmış n adet gerçel sayıdan oluşan (eğer bu şekilde değilse, tüm elemanlar dizideki en büyük elemana bölünerek bu hale getirilebilir; normalize edilebilir) dizinin, kova sıralama algoritmasında aşağıdaki adımlar izlenir:

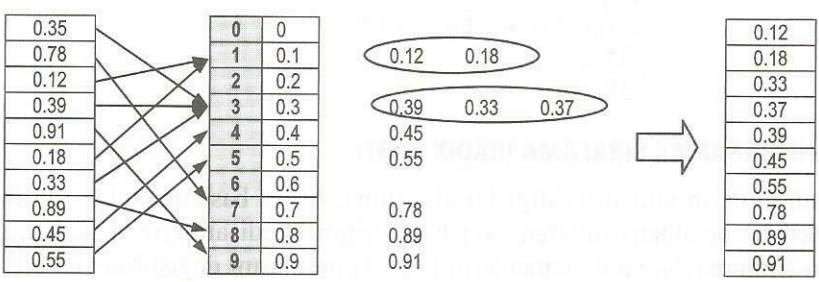
i. [0..1) aralığı $0, \frac{1}{n}, \frac{2}{n}, \frac{3}{n} \ldots \frac{n-1}{n}$ şeklinde alt aralıklara/kovalara bölünür.

ii. Dizinin herbir $a[i]$ elemanı, $\frac{1}{i} \le a[i] \le \frac{1}{i+1}$ şeklinde uygun alt aralığa/kovaya yerleştirilir.

iii. Eğer herhangi bir alt aralığa/kovaya birden fazla eleman düşerse; bu elemanlar, herhangi bir sıralama algoritmasıyla (örneğin "araya eklemeyle sıralama") sıralanır.

iv. Tüm alt aralıklar birleştirilerek dizinin sıralanması tamamlanır.

Örneğin a=(0.35,0.78,0.12,0.39,0.91,0.18,0.33,0.89,0.45,0.55) dizisinin bahsedilen algoritma ile küçükten büyüğe doğru sıralanması Şekil 12.4'teki gibidir.



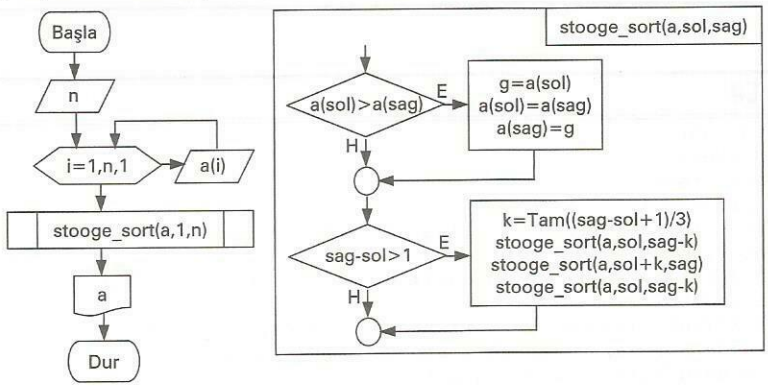
Şekil 12.4 "Bucket sort" ile örnek sıralama işlemi

XVII. PARÇALI SIRALAMA (STOOGE SORT)

Yinelemeli olan bu sıralama algoritmasının çalışması şu şekilde özetlenebilmektedir:

- Eğer dizinin başındaki değer, sonundaki değerden büyükse yerleri değiştirilir.
- Eğer dizi üç veya daha fazla elemanlı ise
  - Dizinin başlangıcındaki 2/3 elemanlarına algoritma uygulanır.
  - Dizinin sonundaki 2/3 elemanlarına algoritma uygulanır.
  - Dizinin başlangıcındaki 2/3 elemanlarına algoritma tekrar uygulanır.

Örnek-12.15: Klavyeden girilen *n* elemanlı *a* dizisini küçükten büyüğe doğru "*parçalı sıralama (Stooge sort)*" algoritmasıyla sıralayan program.



C

```
/* Stooge sort */
#include <stdio.h>
#include <conio.h>
int i,n,a[100];
void stooge_sort(int a[],int sol,int sag)
{ int k,g;
  if(a[sol]>a[sag])
   { g=a[sol]; a[sol]=a[sag]; a[sag]=g;   }
  if(sag-sol>1)
   { k=(int)((sag-sol+1)/3); stooge_sort(a,sol,sag-k);
     stooge_sort(a,sol+k,sag); stooge_sort(a,sol,sag-k);  }}
void main()
{ clrscr();printf("Dizinin eleman sayısı: "); scanf("%u",&n);
  printf("\n");
  for(i=0;i<n;i++)
    { printf("Dizinin %d. elemanı: ",i+1); scanf("%d",&a[i]); }
  stooge_sort(a,0,n-1);
  printf("\nSıralı dizi:\n----------\n");
  for(i=0;i<n;i++) printf("%d\t",a[i]); getch(); }
```

C++

```
// Stooge sort
#include <iostream>
using namespace std;
void stooge_sort(int a[],int sol,int sag)
{ int k,g;
  if(a[sol]>a[sag])
   { g=a[sol]; a[sol]=a[sag]; a[sag]=g;   }
  if (sag-sol>1)
   { k=(int)((sag-sol+1)/3); stooge_sort(a,sol,sag-k);
     stooge_sort(a,sol+k,sag); stooge_sort(a,sol,sag-k); }}
int main()
{ int i,n;
  system("cls"); cout<<"Dizinin eleman sayısı: "; cin>>n; int a[n];
  cout<<"\n";
  for(i=0;i<n;i++)
  { cout<<"Dizinin "<<i+1<<". elemanı: "; cin>>a[i]; }
  stooge_sort(a,0,n-1);
  cout<<"\nSıralı dizi:\n-----------\n";
  for(i=0;i<n;i++) cout<<a[i]<<"\t";
  system("pause"); return 0; }
```

C#

```
// Stooge sort
using System;
namespace ornek1215
{class Program {
  static void stooge_sort(int[] a,int sol,int sag)
   { int k,g;
     if(a[sol]>a[sag]) { g=a[sol]; a[sol]=a[sag]; a[sag]=g;   }
     if(sag-sol>1)
      { k=(int)((sag-sol+1)/3); stooge_sort(a,sol,sag-k);
     stooge_sort(a,sol+k,sag); stooge_sort(a,sol,sag-k);  }}
  static void Main(string[] args)
   { int i, n; Console.Write("Dizinin eleman sayısı: ");
     n=Convert.ToInt32(Console.ReadLine());
     int[] a=new int[n]; Console.WriteLine();
     for (i=0;i<n;i++)
      { Console.Write("Dizinin {0}. elemanı: ",i+1);
        a[i]=Convert.ToInt32(Console.ReadLine()); }
     stooge_sort(a,0,n-1);
     Console.WriteLine("\nSıralı dizi:\n-----------");
     for (i=0;i<n;i++) Console.Write("{0}\t", a[i]);
     Console.ReadLine(); } } }
```

```
Dizinin eleman sayısı: 5

Dizinin 1. elemanı: 2
Dizinin 2. elemanı: 4
Dizinin 3. elemanı: 3
Dizinin 4. elemanı: 1
Dizinin 5. elemanı: 5

Sıralı dizi:
-----------
1   2   3   4   5
```

JAVA

```
// Stooge sort
import java.util.Scanner;
public class ornek1215 {
  public static void stooge_sort(int[] a,int sol,int sag)
   { int k,g;
     if(a[sol]>a[sag]) { g=a[sol]; a[sol]=a[sag]; a[sag]=g;   }
     if(sag-sol>1)
      { k=(int)((sag-sol+1)/3); stooge_sort(a,sol,sag-k);
        stooge_sort(a,sol+k,sag); stooge_sort(a,sol,sag-k);  }}
  public static void main(String[] args) {
    int i,n; Scanner klavye=new Scanner(System.in);
    System.out.print("Dizinin eleman sayısı: ");n=klavye.nextInt();
    int[] a=new int[n]; System.out.println();
    for(i=0;i<n;i++)
    { System.out.printf("Dizinin "+(i+1)+". elemanı: ");
      a[i]=klavye.nextInt(); }
    stooge_sort(a,0,n-1);
    System.out.printf("\nSıralı dizi:\n----------\n");
    for(i=0;i<n;i++) System.out.printf("%d\t",a[i]);     }}
```

MATLAB

```
% Stooge sort
clear all; clc; fprintf('Diziyi [a b c d] şeklinde giriniz!\n')
a=input(''); n=length(a);
a=stooge_sort(a,1,n);
fprintf('Sıralı dizi:\n----------\n'); disp(a);
---
function a=stooge_sort(a,sol,sag)
if(a(sol)>a(sag))
    g=a(sol); a(sol)=a(sag); a(sag)=g;
end
if(sag-sol>1)
    k=round((sag-sol+1)/3); a=stooge_sort(a,sol,sag-k);
    a=stooge_sort(a,sol+k,sag); a=stooge_sort(a,sol,sag-k);
end
```

PYTHON

```
# Stooge sort
def stooge_sort(a,sol,sag):
    if(a[sol]>a[sag]):
        g=a[sol]; a[sol]=a[sag]; a[sag]=g
    if(sag-sol>1):
        k=int((sag-sol+1)/3); stooge_sort(a,sol,sag-k)
        stooge_sort(a,sol+k,sag); stooge_sort(a,sol,sag-k)
print("Diziyi [a b c d] şeklinde giriniz!")
a=[int(i) for i in input().split()]
print("\nSıralı dizi:\n----------")
stooge_sort(a,0,len(a)-1)
for i in range(len(a)):
    print("%d\t"%a[i],end='')
```

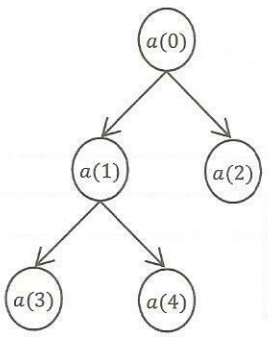
XVIII. YIĞIN SIRALAMA (HEAP SORT)

Karşılaştırma tabanlı olan yığın sıralama, iki aşamalı yinelemeli bir algoritmadır:

i. Birinci aşamada sıralı olmayan dizi, yığın düzeni (heap) haline getirilir. Yığın düzeni, iki şekilde yapılabilir: maksimum ve minimum. Maksimum ta-

banlı yığın düzeninde; hiç bir düğüm, ebeveyn düğümünden büyük değer taşıyamaz. Yani ebeveyn düğümün değeri, çocuk düğümlerinden büyük olmak zorundadır. Aynı şekilde minimum tabanlı yığın düzeninde de; hiçbir düğüm, ebeveyn düğümünden küçük değer taşıyamamaktadır (ebeveyn düğümünün değeri, çocuk düğümlerinden küçük olmalıdır) (Şekil 12.5).

ii. İkinci aşamada; yığın düzeni haline getirilmiş dizinin kök/ilk elemanı, son elemanla yer değiştirir ve sonuncu elemanı içermeyen yeni dizi, yinemeli olarak aynı işlemlere - dizi tamamlanıncaya kadar - tabi tutulur.



Dizi: $A = (a_0, a_1, a_2, a_3, a_4)$

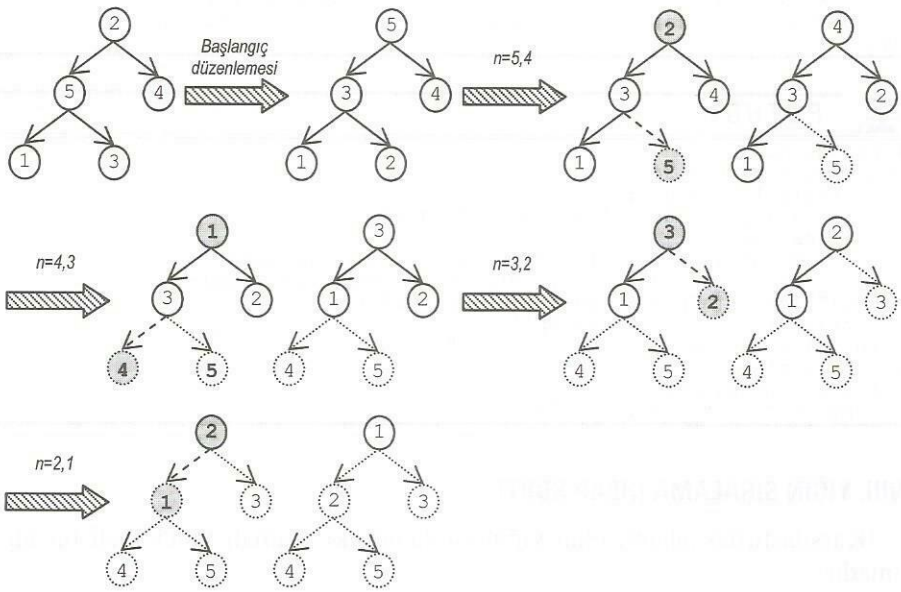
a_i düğümü için:

- ✓ Ebeveyn (parent) indisi: $\frac{i-1}{2}$
- ✓ Sol taraftaki çocuk (child) indisi: $2i + 1$
- ✓ Sağ taraftaki çocuk (child) indisi: $2i + 2$

- Maksimum tabanlı yığın düzeninde: $a_i \geq a_{2i+1}$ ve $a_i \geq a_{2i+2}$
- Minimum tabanlı yığın düzeninde: $a_i \leq a_{2i+1}$ ve $a_i \leq a_{2i+2}$

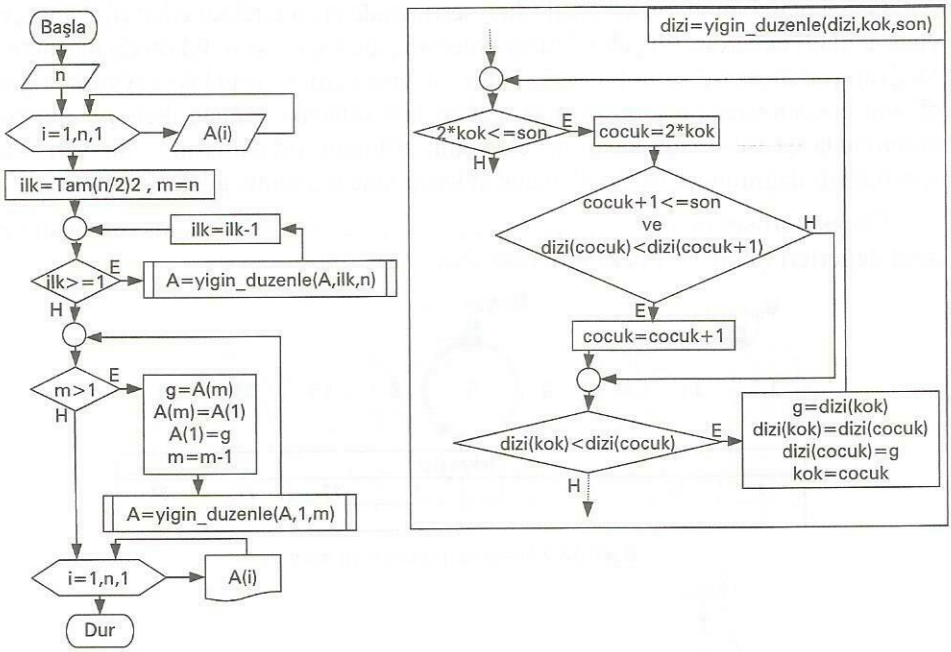
Şekil 12.5 Yığın düzeni (ağacı) ve özellikleri

`A=[2,5,4,1,3]` dizisinin "*yığın sıralama yöntemi*"yle sıralanması, Şekil 12.6'da verilmektedir.

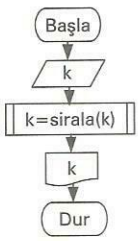


Şekil 12.6 "Heap sort" ile örnek sıralama işlemi

Örnek-12.16: Klavyeden girilen *n* elemanlı *A* dizisini küçükten büyüğe doğru "*yığın sıralama (Heap sort)*" algoritmasıyla sıralayan program.

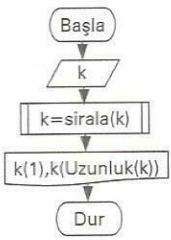


Örnek-12.17: Klavyeden büyük harflerle girilen kelimenin harflerini alfabetik olarak sıralayıp yeniden yazdıran program ("BURSA" girildiğinde "ABRSU" oluşacaktır).



Daha önce anlatılan sıralama algoritmaları sadece sayısal veriler üzerinde değil, alfasayısal verilerde de çalışmaktadırlar. Çünkü alfasayısal veriler de nihayetinde ASCII kod tablosundaki sayısal değerlere dayanmaktadır. Ayrıca alfasayısal verilerin de birer dizi olduğunu hatırlayınız. Bu nedenle problemin çözümü için herhangi bir sıralama algoritması kullanılabilir.

Örnek-12.18: Klavyeden büyük harflerle girilen bir kelimedeki alfabetik olarak en önde ve en sonda olan karakteri bulup yazdıran program.

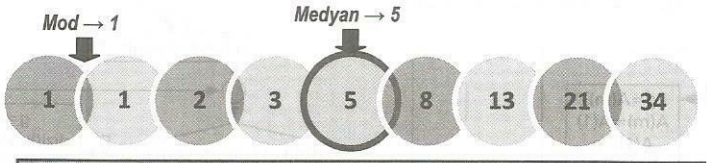


Bir önceki örnekte olduğu gibi problemin çözümünde ya kelimenin harfleri alfabetik olarak sıralanıp ilk ve son harfleri yazdırılır ya da maksimum-minimum problemlerinde olduğu gibi ilk harf en önde-en sonda kabul edilip diğer harflerle karşılaştırılır ve sonuçlarda buna uygun değişiklikler yapılır.

Örnek-12.19: Klavyeden girilen n elemanlı A dizisinin medyan ve modunu bulan program.

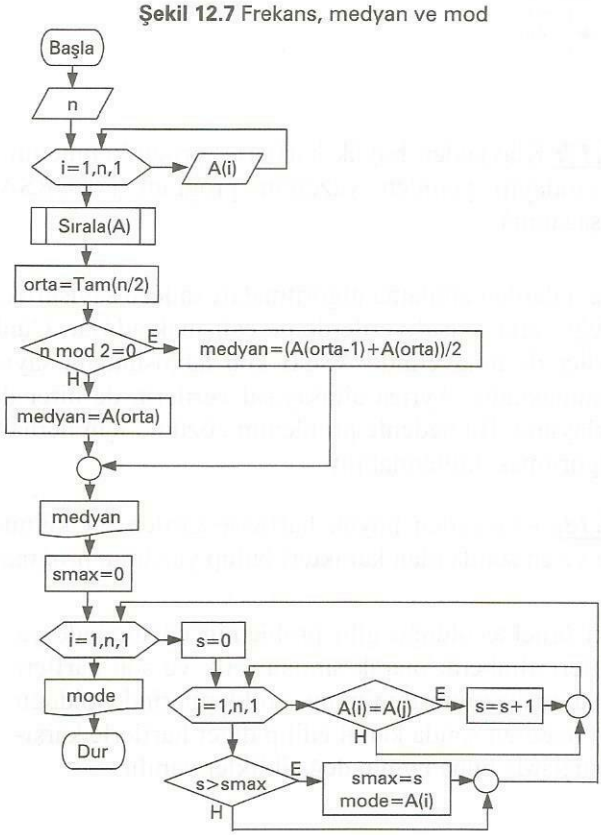
Genel olarak mod, bir veri seti (dizi, seri) içinde en çok tekrar eden (frekansı en yüksek olan) değerdir. En çok tekrar edenler iki tane ise bu setin iki modu olacaktır. Medyan (ortanca) ise sıralı bir veri setinin (dizinin) tam ortasındaki değerdir. Eğer dizinin eleman sayısı tek ise medyan değeri tam ortadaki eleman iken çift olması durumunda da ise ortada kalan iki elemanın aritmetik ortalamasıdır. Bir veri seti için frekans dağılımı ise herbir elemandan kaçar tane olduğunu gösterir.

Örnek veri setimiz `A=(1,1,2,3,5,8,13,21,34)` ise frekans, medyan ve mod değerleri Şekil 12.7'deki gibi olacaktır.



Frekanslar							
1	*2*	*3*	*5*	*8*	*13*	*21*	*34*
2	1	1	1	1	1	1	1

Şekil 12.7 Frekans, medyan ve mod



SORULAR

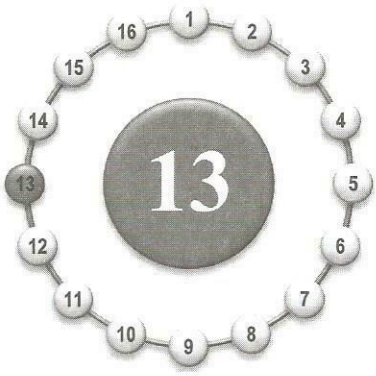
1. Diğer sıralama algoritmalarını araştırarak akış diyagramlarını çiziniz ve kodlayınız.

2. Programlama dillerinde mevcut/hazır fonksiyon/komut olarak tanımlanmış sıralama algoritmalarını araştırarak karşılaştırınız.

3. Sıralama algoritmalarında performans analizlerinin nasıl gerçekleştirildiğini araştırınız.

BÖLÜM 13

ARAMA ALGORİTMALARI



Bu bölümde

- Arama algoritmalarını tanıyacak,
- Akış diyagramlarını çizerek kodlamasını öğreneceksiniz.

Bölüm sonunda

- Arama algoritmalarını kavrayacak,
- Bunların performanslarını karşılaştırabilecek,
- Akış diyagramlarını çizip kodlayabileceksiniz.

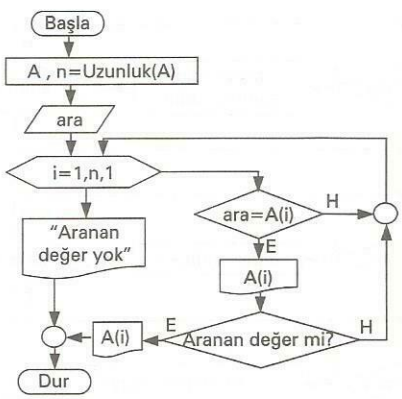
Birçok verinin/bilginin içinde; doğru (aranan) bilgiye, en kısa sürede ulaşmak için farklı arama algoritmaları geliştirilmiştir. Bu bölümde temel arama algoritmalarından bahsedilecektir.

I. DOĞRUSAL/SIRALI ARAMA (LINEAR/SEQUENTIAL SEARCH)

Yavaş olmasına rağmen en basit olan bu algoritmada; veri yığınındaki tüm bilgiler (kayıtlar), sırasıyla çağırılır ve aranan değerle karşılaştırılır. Çağırılan değer ile aranan değer aynı ise arama işlemi sona erdirilir. Aynı bilgiden (aynı isimli olan, aynı şehirde oturan vb.) birden fazla olma ihtimaline karşın; ilgili mesajla kullanıcıya sorulabilir ve aranan bilgi değilse, arama işlemi devam ettirilebilir. Örnek dizimiz `A=(13,5,2,1,34,8,3,1,21)` ve aranan değer "8" olsun. Sıralı arama işlemindeki adımlar, Şekil 13.1'de verilmektedir.

Şekil 13.1 "Sıralı arama algoritması" ile arama işlemi

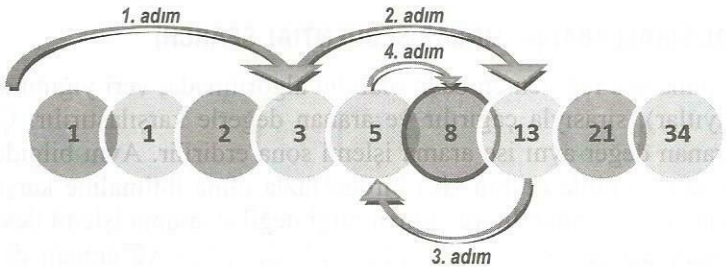
Örnek-13.1: *n* elemanlı *A* dizisi içinde (veri yığınında, kayıt dosyasında) klavyeden girilen değeri, *sıralı (doğrusal) (sequential, linear) arama algoritması* ile arayan program.



II. ATLAMALI/BLOK ARAMA (JUMP/BLOCK SEARCH)

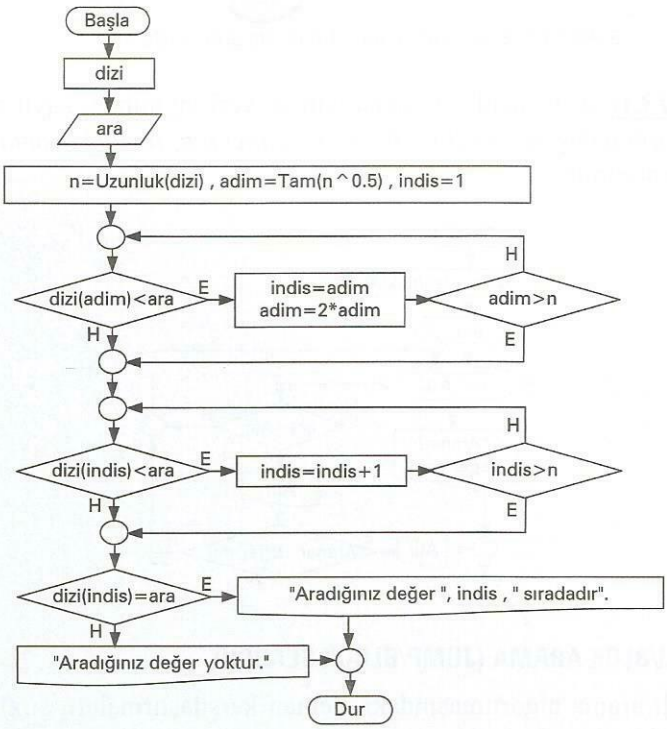
Atlamalı arama algoritmasındaki eleman karşılaştırmaları, sıralı arama algoritmasındaki gibi ardışık (birer birer, atlama adımı 1) olarak yapılmamaktadır. Belirli adımlarla atlanarak erişim sağlanmakta ve karşılaştırma gerçekleştirilmektedir. Küçükten büyüğe doğru sıralı bir listede (dizide) ulaşılan indisteki veri, aranandan küçükse ileriye; büyükse de geriye doğru gidilip tekrar ileriye doğru devam etmektedir. Bu algoritmada da genellikle başlangıç atlama adımı olarak dizinin eleman sayısının karekökü seçilmektedir (Neden?). Örnek dizi-

miz `A=(1,1,2,3,5,8,13,21,34)` ve aranan değer "`8`" olsun. Veri dizisi 9 elemanlı olduğu için adım değeri 3 seçilmekte ve karşılaştırma/arama adımları Şekil 13.2'de verilmektedir.



Şekil 13.2 "Jump search" ile arama işlemi

Örnek-13.2: *n* elemanlı *A* dizisi içinde (veri yığınında, kayıt dosyasında) klavyeden girilen değeri, *atlamalı (blok) (jump, block) arama algoritması* ile arayan program.

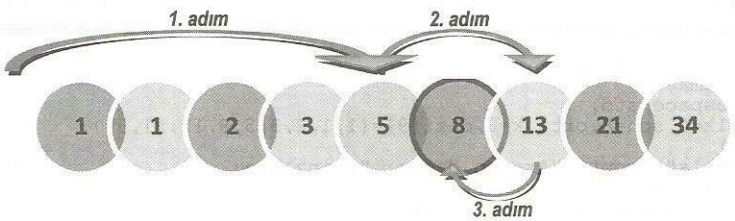


III. İKİLİ ARAMA (BİNARY SEARCH)

İkili (binary) arama algoritmasında karşılaştırma işlemine, küçükten büyüğe sıralı olan dizinin ortasında bulunan elemanla başlanır. Dizi, küçükten büyüğe doğru sıralı olduğundan; eğer aranan değer, dizinin orta elemanından küçük-

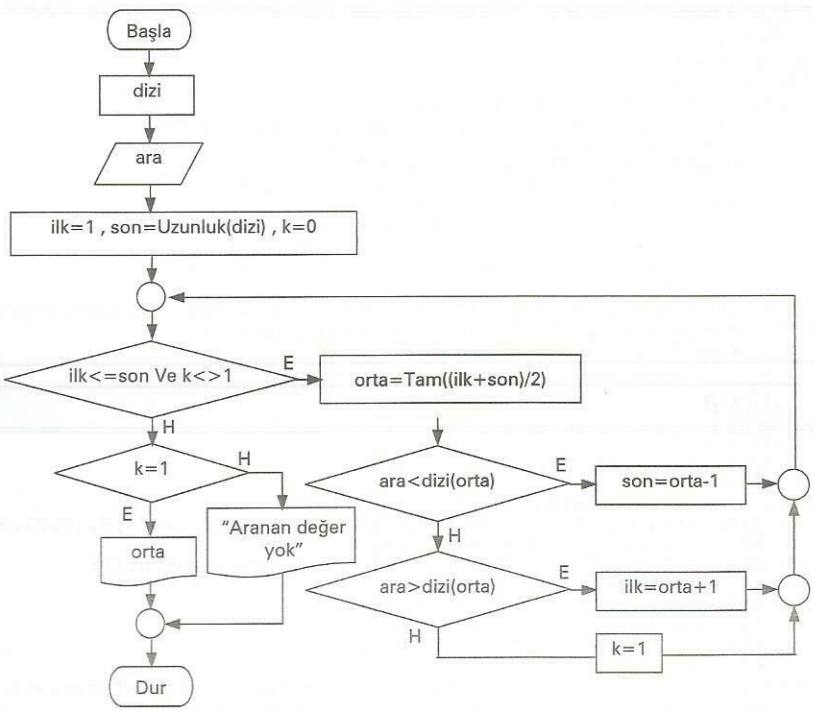
se, dizinin ortadan sonraki kısmı (üst kısmı) atılır. Eğer aranan değer, dizinin orta elemanından büyükse, dizinin alt kısmı atılır. Bu şekilde dizi yarılanmış olur. Bir sonraki aşamada karşılaştırma, dizinin kalan yarım parçasının orta elemanıyla yapılır. Böylece dizi bölünerek/yarılanarak, eleman bulununcaya kadar karşılaştırma işlemine devam edilir.

Örnek dizimiz `A=(1,1,2,3,5,8,13,21,34)` ve aranan değer "8" olsun. Veri dizisi 9 elemanlı olduğu için karşılaştırmaya (aramaya) ortadaki, yani 5. sıradaki ($\frac{1+9}{2} = 5$) elemanla başlanmakta ve arama adımları Şekil 13.3'te gösterilmektedir.



Şekil 13.3 "Binary search" ile arama işlemi

Örnek-13.3: *n* elemanlı *A* dizisi içinde (veri yığınında, kayıt dosyasında) klavyeden girilen değeri, *ikili (binary) arama algoritması* ile arayan program.



C

```
/* Binary search */
#include <stdio.h>
#include <conio.h>
#include <math.h>
int ara,ilk=0,son=8,orta,k=0,dizi[9]={1,1,2,3,5,8,13,21,34};
void main()
{ clrscr();printf("Aranan değer: "); scanf("%d",&ara);
  while ((ilk<=son) && (k!=1))
   { orta=floor((ilk+son)/2);
     if (ara<dizi[orta]) son=orta-1;
     else if (ara>dizi[orta]) ilk=orta+1;
     else k=1; }
  if (k==1) printf("Aradığınız değer %d. sırada\n",orta+1);
  else printf("Aradığınız değer yoktur\n");   getch(); }
```

C++

```
// Binary search
#include <iostream>
#include <cmath>
using namespace std;
int ara,ilk=0,son=8,orta,k=0,dizi[9]={1,1,2,3,5,8,13,21,34};
int main()
{ system("cls");cout<<"Aranan değer: ";cin>>ara;
  while ((ilk<=son) && (k!=1))
  { orta=(int) floor((ilk+son)/2);
    if (ara<dizi[orta]) son=orta-1;
    else if (ara>dizi[orta]) ilk=orta+1;
    else k=1; }
  if (k==1) cout<<"\nAradığınız değer "<<orta+1<<" sırada\n";
  else cout<<"\nAradığınız değer yoktur\n"; system("pause");return 0;}
```

C#

```
// Binary search
using System;
namespace ornek133
{ class Program
    { static void Main(string[] args)
        { int ara, orta=0,ilk=0,son=8,k=0; double g;
          int[] dizi={1,1,2,3,5,8,13,21,34};
          Console.Write("Aranan değer: ");
          ara=Convert.ToInt32(Console.ReadLine());
          while ((ilk <= son) && (k != 1))
            { g = (ilk + son) / 2; orta = (int) Math.Floor(g);
              if (ara < dizi[orta]) son = orta - 1;
              else if (ara > dizi[orta]) ilk = orta + 1;
              else k = 1; }
 if (k == 1) Console.WriteLine("\nAradığınız değer {0} sırada",++orta);
 else Console.WriteLine("\nAradığınız değer yoktur");
 Console.ReadLine(); }    }   }
```

JAVA

```
// Binary search
import java.util.Scanner;
public class ornek133 {
  public static void main(String[] args) {
    int ara, orta=0,ilk=0,son=8,k=0; int[] dizi={1,1,2,3,5,8,13,21,34};
    Scanner klavye=new Scanner(System.in);
    System.out.print("Aranan değer: "); ara=klavye.nextInt();
    while ((ilk <= son) && (k != 1))
      { orta=(int) Math.floor((ilk+son)/2);
        if (ara<dizi[orta]) son=orta-1;
        else if (ara>dizi[orta]) ilk=orta+1;
        else k=1; }
 if (k==1) System.out.printf("\nAradığınız değer %d sırada",orta+1);
 else System.out.printf("\nAradığınız değer yoktur"); }}
```

MATLAB

```
% Binary search
clear all; clc; dizi=[1 1 2 3 5 8 13 21 34]; % Veriler
ara=input('Aranan değer: '); ilk=1; son=length(dizi); k=0;
while ((ilk<=son) && (k~=1))
    orta=floor((ilk+son)/2);
    if (ara<dizi(orta)) son=orta-1;
    elseif (ara>dizi(orta)) ilk=orta+1;
    else k=1;      end;   end;
if (k==1) fprintf('\nAradığınız değer %d. sırada\n',orta)
else fprintf('\nAradığınız değer yoktur\n'); end;
```

```
Aranan değer: 8

Aradığınız değer 6. sırada
```

PYTHON

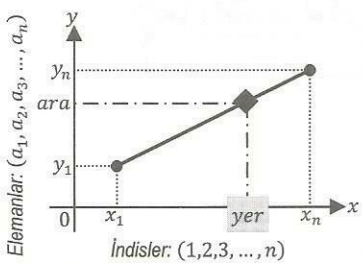
```
# Binary search
import math
dizi=[1,1,2,3,5,8,13,21,34]
ara=int(input("Aranan değer: "))
ilk=1; son=len(dizi); k=0
while ((ilk<=son) & (k!=1)):
    orta=math.floor((ilk+son)/2)
    if (ara<dizi[orta]):
        son=orta-1
    elif (ara>dizi[orta]):
        ilk=orta+1
    else:
        k=1
if (k==1):
    print("\nAradığınız değer %d. sırada"%(orta+1))
else:
    print("\nAradığınız değer yoktur")
```

Tablo 13.1: İkili arama algoritması

C	C++	C#	Java	MATLAB	Python
	binary_search	BinarySearch	binarySearch		bisect

IV. ENTERPOLASYON ARAMA (INTERPOLATION SEARCH)

Enterpolasyon arama algoritması, sıralı ve düzgün dağılımlı veriler için ikili arama algoritmasının geliştirilmiş hali olarak değerlendirilebilir. Bu algoritmada aranan değerin konumu, aralık daraltılarak doğrusal enterpolasyonla (doğru denklemiyle) hesaplanmaktadır (tahmin edilmektedir).

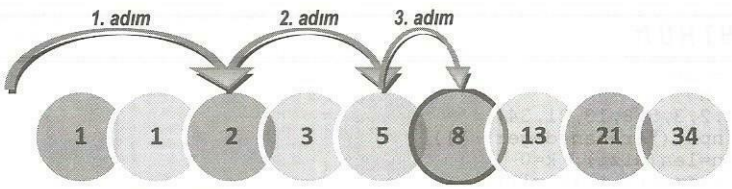


Şekil 13.4 Doğrusal enterpolasyon

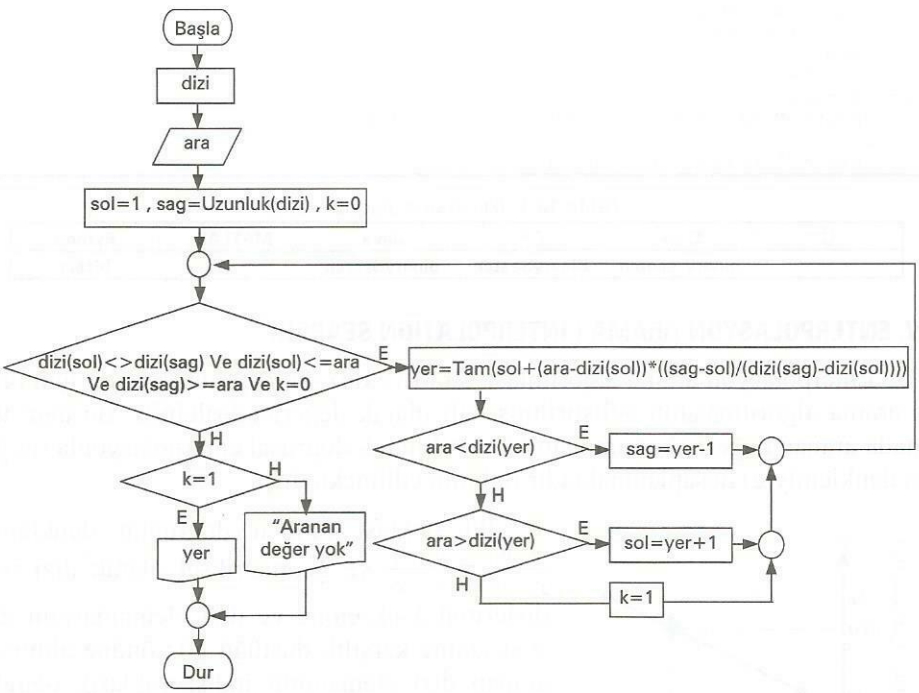
İki noktası bilinen doğrunun denklemi $\frac{y-y_1}{y_2-y_1} = \frac{x-x_1}{x_2-x_1}$ ve geometriksel olarak dizi indislerinin x-eksenine ve dizi elemanlarının da y-eksenine karşılık düştüğü göz önüne alınırsa aranan dizi elemanının indisi yaklaşık olarak $yer = x_1 + (ara - y_1)\frac{x_n - x_1}{y_n - y_1}$ ile bulunur. Aranan değer bu indiste değilse, alt ya da üst aralık atılarak yeni indis hesaplanır(Şekil 13.4).

Örnek-13.4: *n* elemanlı *A* dizisi içinde (veri yığınında, kayıt dosyasında) klavyeden girilen değeri, *enterpolasyon (interpolation) arama algoritması* ile arayan program.

Örnek dizimiz `A=(1,1,2,3,5,8,13,21,34)` ve aranan değer "8" olsun. İlk hesaplanan tahmini indis $yer = 1 + (8-1)\frac{9-1}{34-1} = 2.697 \cong 3$ olup aranan "8" değeri üçüncü indiste yer alan "2"den büyük olduğundan alt kısım atılmakta ve yeni arama dizisi `A=(1,1,2,3,5,8,13,21,34)` şekline dönüşmektedir. Bu durumda yeni tahmini indis $yer = 4 + (8-3)\frac{9-4}{34-3} = 4.8065 \cong 5$ olarak hesaplanmakta ve aramaya devam edilmektedir (Şekil 13.5).



Şekil 13.5 "Interpolation search" ile arama işlemi



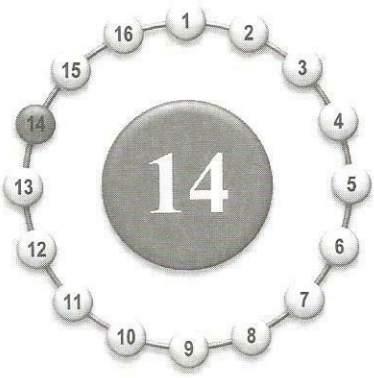
SORULAR

1. Diğer arama algoritmalarını araştırarak akış diyagramlarını çiziniz ve kodlayınız.

2. Programlama dillerinde mevcut/hazır fonksiyon/komut olarak tanımlanmış arama algoritmalarını araştırarak karşılaştırınız.

BÖLÜM 14

KODLAMA/ŞİFRELEME UYGULAMALARI



Bu bölümde

- Kodlama ve şifreleme türlerini tanıyacak,
- Algoritmalarını hazırlayarak kodlamasını öğreneceksiniz.

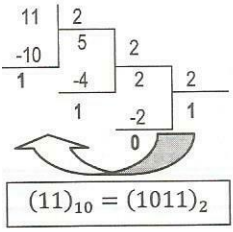
Bölüm sonunda

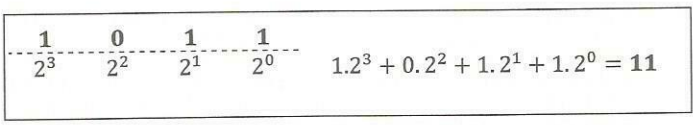
- Kodlama ve şifreleme türlerini kavrayacak,
- Algoritmalarını hazırlayıp kodlayabileceksiniz.

I. TABAN DÖNÜŞÜMLERİ

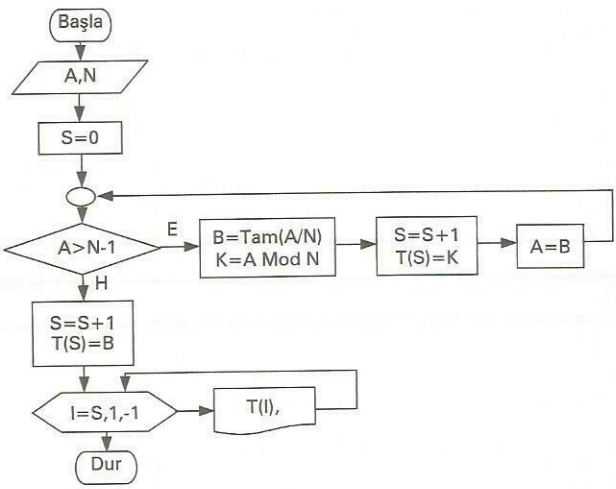
Bilindiği üzere 10 tabanındaki bir tamsayının herhangi tabandaki karşılığı; bu tamsayının, taban değerine sürekli bölünmesiyle elde edilen kalanların ters sırada yazılmasıyla hesaplanır. Benzer şekilde n tabanındaki sayının 10 tabanına dönüşümü $(abcd)_n = d.n^0 + c.n^1 + b.n^2 + a.n^3$ şeklinde yapılır. Dolayısıyla girilen sayının basamakları ayrıştırılır ve tabanın kuvvetleriyle çarpılıp toplanır. Ayrıca bazı programlama dillerinde tabanlar arası dönüşüm yapan hazır komutlar vardır.

Örneğin $(11)_{10}$ tamsayısının 2 tabanına ve tekrar 10 tabanına dönüşümleri aşağıda verilmektedir.





Örnek-14.1: Klavyeden girilen 10 tabanındaki pozitif tamsayının, belirtilen tabandaki karşılığını hesaplayan program.



C

```
/* Taban aritmetiği */
#include <stdio.h>
#include <conio.h>
int A,N,B,K,T[99],i,s=0;
void main()
```

```
{ clrscr();printf("    TABAN ARİTMETİĞİ   \n");
  printf("\nPozitif bir tamsayı giriniz -> ");scanf("%u",&A);
  printf("Hangi tabana dönüştürülecek -> ");scanf("%u",&N);
  while (A>N-1) { B=(int) A/N;K=A % N;s++;T[s-1]=K;A=B; }
  s++;T[s-1]=B; printf("\n%u tabanındaki karşılığı: ",N);
  for(i=s-1;i>=0;i--) printf("%u",T[i]); getch();  }
```

```
TABAN ARİTMETİĞİ

Pozitif bir tamsayı giriniz -> 99
Hangi tabana dönüştürülecek -> 2

2 tabanındaki karşılığı: 1 1 0 0 0 1 1
```

C++

```
// Taban aritmetiği
#include <iostream>
using namespace std;
int A,N,B,K,T[99],s=0;
int main()
{   system("cls"); cout << "    TABAN ARİTMETİĞİ\n";
    cout << "Pozitif bir tamsayı giriniz -> "; cin >> A;
    cout <<"Hangi tabana dönüştürülecek -> "; cin>>N;
    while (A>N-1) { B=(int) A/N;K=A % N;s++;T[s-1]=K;A=B; }
    s++;T[s-1]=B; cout << "\n" << N <<" tabanındaki karşılığı: ";
    for(int i=s-1;i>=0;i--) cout << T[i];  system("pause");return 0; }
```

C#

```
// Taban aritmetiği
using System;
namespace ornek141
{ class Program
   { static void Main(string[] args)
      { int A,N,B,K,s=0; int[] T=new int[99];
        Console.WriteLine("    TABAN ARİTMETİĞİ\n");
        Console.Write("Pozitif bir tamsayı giriniz -> ");
        A=Convert.ToInt32(Console.ReadLine());
        Console.Write("Hangi tabana dönüştürülecek -> ");
        N=Convert.ToInt32(Console.ReadLine());
        while(A>N-1) { B=(int) (A/N); K=A%N; s++; T[s-1]=K; A=B; }
        s++;T[s-1]=A; Console.Write("\n{0} tabanındaki karşılığı: ",N);
        for(int i=s-1;i>=0;i--) Console.Write(T[i]);
        Console.ReadLine();      }    }    }
```

JAVA

```
// Taban aritmetiği
import java.util.Scanner;
public class ornek141 {
public static void main(String[] args)
{  int A,N,B,K,s=0; int T[]=new int[99];
   Scanner klavye=new Scanner(System.in);
   System.out.println("    TABAN ARİTMETİĞİ\n");
   System.out.print("Pozitif bir tamsayı giriniz -> ");
   A=klavye.nextInt();
   System.out.print("Hangi tabana dönüştürülecek -> ");
   N=klavye.nextInt();
   while(A>N-1) { B=(int) (A/N); K=A%N; s++; T[s-1]=K; A=B; }
   s++;T[s-1]=A; System.out.print("\n"+N+" tabanındaki karşılığı:");
   for(int i=s-1;i>=0;i--) System.out.print(T[i]);   }}
```

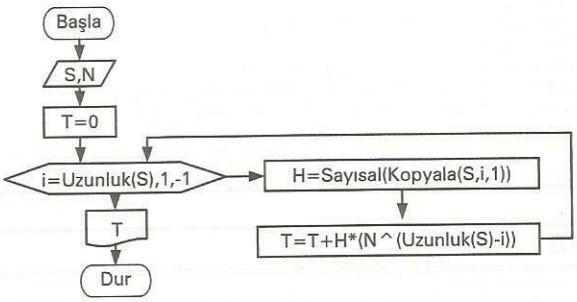
MATLAB

```
% Taban aritmetiği
clear all; clc; fprintf('       TABAN ARİTMETİĞİ\n\n');
A=input('Pozitif bir tamsayı giriniz -> ');
N=input('Hangi tabana dönüştürülecek -> ');
fprintf('\n%d tabanındaki karşılığı: %s\n',N,dec2base(A,N));
```

PYTHON

```
# Taban aritmetiği
import numpy as np
print("TABAN ARİTMETİĞİ")
A=int(input("Pozitif bir tamsayı giriniz -> "))
N=int(input("Hangi tabana dönüştürülecek -> "))
print("\n%d tabanındaki karşılığı: %s"%(N,np.base_repr(A,N)))
```

Örnek-14.2: $n=2-9$ olmak üzere n tabanında girilen sayının 10 tabanındaki karşılığını bulan program.



C

```
/* 10 tabanına dönüştürme */
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>
char s[100], *h1; int i,h,N; float T=0;
void main()
{ clrscr();printf("Sayı: ");gets(s); printf("Taban: ");scanf("%u",&N);
  for(i=strlen(s)-1;i>=0;i--)
    { *h1=s[i];h=atoi(h1); T=T+h*(pow(N,strlen(s)-1-i)); }
  printf("\n10 tabanındaki değer: %0.0f",T);   getch(); }
```

C++

```
// 10 tabanına dönüştürme
#include <iostream>
#include <string.h>
#include <math.h>
using namespace std;
char s[100]; int N,ns; float T=0;
int main()
{ system("cls"); cout<<"Sayi: ";cin>>s; cout<<"Taban: "; cin>>N;
  for(int i=strlen(s)-1;i>=0;i--)
    {ns=(int)s[i]-48;T=T+ns*(pow(N,strlen(s)-1-i));}
  cout<<"\n10 tabanindaki deger: "<<T<<endl;system("pause");return 0;}
```

```
Sayı: 101101101
Taban:2

10 tabanındaki değer: 365
```

C#

```
// 10 tabanına dönüştürme
using System;
namespace ornek142
{ class Program
    { static void Main(string[] args)
      { int N,ns; String s; double T = 0;
        Console.Write("Sayı: ");s=Console.ReadLine();
        Console.Write("Taban: ");N=Convert.ToInt32(Console.ReadLine());
        for(int i=s.Length-1;i>=0;i--)
          { ns=(int)s[i]-48;T=T+ns*(Math.Pow(N,s.Length-1-i)); }
        Console.WriteLine("\n10 tabanındaki değer: "+(int)T);
        Console.ReadLine();       }    }    }
```

JAVA

```
// 10 tabanına donusturme
import java.util.Scanner;
public class ornek142 {
public static void main(String[] args)
{  int N,ns; String s; double T = 0;
   Scanner klavye=new Scanner(System.in);
   System.out.print("Sayi: ");s=klavye.nextLine();
   System.out.print("Taban: ");N=klavye.nextInt();
   for(int i=s.length()-1;i>=0;i--)
    {ns=(int)s.charAt(i)-48;T=T+ns*(Math.pow(N,s.length()-1-i));}
   System.out.print("\n10 tabanındaki değer: "+(int)T);   }}
```

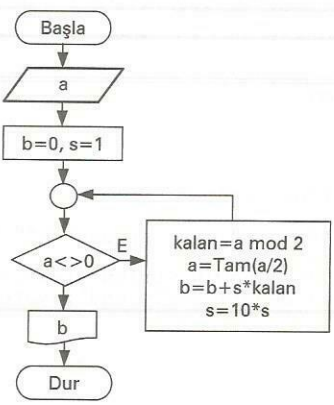
MATLAB

```
% 10 tabanına dönüştürme
clear all; clc; S=input('Sayı: ','s');N=input('Taban: ');
fprintf('\n10 tabanındaki değer: %d\n',base2dec(S,N));
```

PYTHON

```
# 10 tabanına dönüştürme
S=input("Sayı: "); N=int(input("Taban: "))
print("\n10 tabanındaki değer: %d"%(int(S,N)))
```

Örnek-14.3: 10 tabanındaki pozitif tamsayıyı, 2 tabanına dönüştüren program.



C

```
/* Decimal-Binary */
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void main()
{ int a; char b[100];
  clrscr();printf("10 tabanında pozitif tamsayı: "); scanf("%u",&a);
  itoa(a,b,2); printf("2 tabanındaki karşılığı: %s\n ",b);  getch(); }
```

C++

```
// Decimal-Binary
#include <iostream>
#include <string>
#include <stdlib.h>
using namespace std;
int main()
{ int a; char b[100]; system("cls");
  cout<<"10 tabanında pozitif tamsayı: ";cin>>a;
  itoa(a,b,2); cout<<"2 tabanındaki karşılığı: "<<b<<endl;
  system("pause");return 0;}
```

C#

```
// Decimal-Binary
using System;
namespace ornek143
{class Program {
  static void Main(string[] args) {
    int a; string b;
    Console.Write("10 tabanında pozitif tamsayı: ");
    a = Convert.ToInt32(Console.ReadLine());
    b = Convert.ToString(a, 2);
    Console.WriteLine("2 tabanındaki karşılığı: {0}", b);
    Console.ReadLine();    } } }
```

```
10 tabanında pozitif tamsayı: 2017
2 tabanındaki karşılığı: 11111100001
```

JAVA

```
// Decimal-Binary
import java.util.Scanner;
public class ornek143 {
  public static void main(String[] args) {
    int a; String b; Scanner klavye=new Scanner(System.in);
    System.out.print("10 tabanında pozitif tamsayı: ");
    a=klavye.nextInt(); b=Integer.toString(a,2);
    // b=Integer.toBinaryString(a);
    System.out.printf("2 tabanındaki karşılığı: "+b); }}
```

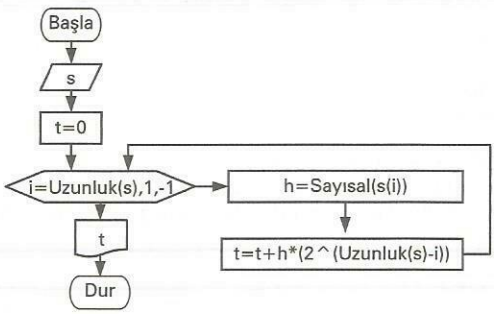
MATLAB

```
% Decimal-Binary
clear all; clc; a=input('10 tabanında pozitif tamsayı: ');
b=dec2bin(a); fprintf('2 tabanındaki karşılığı: %s\n',b);
```

PYTHON

```
# Decimal-Binary
a=int(input("10 tabanında pozitif tamsayı: "))
b=bin(a)[2:]; print("2 tabanındaki karşılığı: %s"%b)
---------------- veya ----------------------
# Decimal-Binary
import numpy as np
a=int(input("10 tabanında pozitif tamsayı: "))
b=np.binary_repr(a); print("2 tabanındaki karşılığı: %s"%b)
```

Örnek-14.4: 2 tabanındaki pozitif tamsayıyı, 10 tabanına dönüştüren program.



C

```
/* Binary-Decimal */
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void main()
{ char a[100],*p; long b;
  clrscr();printf("2 tabanında sayı: "); gets(a);
  b=strtol(a,&p,2);; printf("10 tabanındaki karşılığı: %ld\n ",b);
  getch(); }
```

C++

```
// Binary-Decimal
#include <iostream>
#include <stdlib.h>
using namespace std;
int main()
{ char a[100],*p; long b;
  system("cls"); cout<<"2 tabanında sayı: ";cin>>a;
  b=strtol(a,&p,2); cout<<"10 tabanındaki karşılığı: "<<b<<endl;
  system("pause");return 0;}
```

C#

```
// Binary-Decimal
using System;
namespace ornek144
{ class Program
   { static void Main(string[] args)
      { Console.Write("2 tabanında sayı: ");
        String s = Console.ReadLine();
        Console.WriteLine("\n10 tabanındaki karşılığı: " +
                          Convert.ToInt32(s,2)); Console.ReadLine(); }}}
```

JAVA

```
// Binary-Decimal
import java.util.Scanner;
public class ornek144 {
 public static void main(String[] args) {
 Scanner klavye=new Scanner(System.in);
 System.out.print("2 tabanında sayı: "); String s=klavye.nextLine();
 System.out.printf("\n10 tabanındaki karşılığı: "+
                   Integer.parseInt(s,2)); }}
```

```
2 tabanında sayı: 111000

10 tabanındaki karşılığı: 56
```

MATLAB

```
% Binary-Decimal
s=input('2 tabanında sayı: ','s');
fprintf('\n10 tabanındaki karşılığı: %d\n',bin2dec(s));
```

PYTHON

```
# Binary-Decimal
s=input("2 tabanında sayı: ")
print("\n10 tabanındaki karşılığı: %d"%(int(s,2)))
```

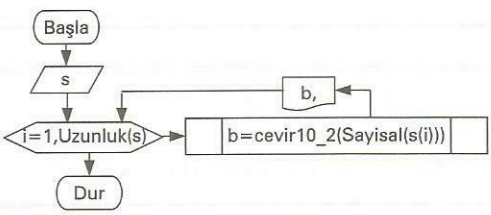
II. SAYISAL KODLAR

A. BCD (8421) kodu

10 tabanındaki herhangi bir pozitif tamsayı, BCD (Binary Coded Decimal) koduna dönüştürülürken; her basamak 2 tabanında 4 bit olarak yazılmaktadır.

$$\left(\underbrace{2}_{0010}\ \underbrace{0}_{0000}\ \underbrace{1}_{0001}\ \underbrace{7}_{0111}\right)_{10} = (0010\ 0000\ 0001\ 0111)_{BCD}$$

Örnek-14.5: Klavyeden girilen pozitif tamsayıyı, BCD koduna dönüştürüp yazdıran program.



C

```
/* Decimal-BCD */
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
void main()
{ char s[100],s1[4],*p; int i,b;
  clrscr();printf("Sayı: ");gets(s); printf("\nBCD kodu: ");
  for(i=0;i<strlen(s);i++)
   { b=(int)s[i]-48; itoa(b,s1,2); b=strtol(s1,&p,10);
     printf("%04d ",b); }  getch(); }
```

C++

```
// Decimal-BCD
#include <iostream>
#include <iomanip>
#include <string>
#include <stdlib.h>
using namespace std;
int main()
{ char s[100],s1[4],*p; int b;
  system("cls");cout<<"Sayı: ";cin>>s; cout<<"\nBCD kodu: ";
  for(int i=0;i<strlen(s);i++)
   { b=(int)s[i]-48; itoa(b,s1,2); b=strtol(s1,&p,10);
     cout<<setfill('0')<<setw(4)<<b<<' '; }
  system("pause");return 0;}
```

C#

```
// Decimal-BCD
using System;
namespace ornek145
{class Program {
  static void Main(string[] args) {
    Console.Write("Sayı: "); String s=Console.ReadLine();
    Console.Write("\nBCD kodu: ");
    for (int i=0;i<s.Length;i++)
Console.Write("{0} ",Convert.ToString((int)s[i]-48, 2).PadLeft(4,'0'));
    Console.ReadLine(); } } }
```

JAVA

```
// Decimal-BCD
import java.util.Scanner;
public class ornek145 {
 public static void main(String[] args) {
  Scanner klavye=new Scanner(System.in);
  System.out.print("Sayı: ");
  String s=klavye.nextLine();
  System.out.printf("\nBCD kodu: ");
  for (int i=0;i<s.length();i++)
   System.out.printf("%04d ",
    Integer.parseInt(Integer.toBinaryString((int)s.charAt(i)-48))); }}
```

MATLAB

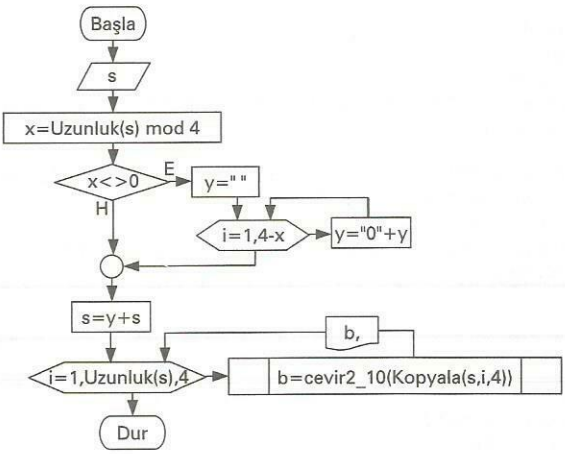
```
% Decimal-BCD
clear all; clc; s=input('Sayı: ','s')
fprintf('\nBCD kodu: ');
for i=1:length(s)
    fprintf('%04s ',dec2bin(str2num(s(i))));
end
```

PYTHON

```
# Decimal-BCD
s=input("Sayı: ")
print("\nBCD kodu: ",end='')
for i in range(len(s)):
    print("%s "%bin(int(s[i]))[2:].zfill(4),end='')
```

```
Sayı: 2017

BCD kodu: 0010 0000 0001 0111
```

Örnek-14.6: Klavyeden BCD kodunda girilen sayıyı, 10 tabanına dönüştürüp yazdıran program.



C

```
/* BCD-Decimal */
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
void main()
{  char s[100],g[100],ss[100],y[99],*p; int i,j,x,b;
       clrscr();printf("BCD kodu: ");gets(s); x=strlen(s)%4;
   if (x!=0)
       { y[0]='0';for(i=1;i<4-x;i++) y[i]='0';    y[i]='\0';
   strcpy(g,y); strcat(g,s); strcpy(s,g); } printf("\nDeğer: ");
   for(i=0;i<strlen(s);i+=4)
       { for(j=0;j<4;j++) ss[j]=s[i+j]; ss[j]='\0'; b=strtol(ss,&p,2);
         printf("%d",b); }   getch(); }
```

```
BCD kodu: 10000000010111

Değer: 2017
```

C++

```
// BCD-Decimal
#include <iostream>
#include <string.h>
#include <math.h>
using namespace std;
double cevir2_10(string s1)
{ int i,n,h; double t=0; n=s1.length();
  for(i=n-1;i>=0;i--) {h=(int)s1[i]-48;t+=h*(pow(2,n-1-i));} return t;
}
int main()
{ string s; system("cls"); cout<<"BCD kodu: "; cin>>s;
  int x=s.length()%4;
  if (x!=0) {   string y=""; for(int i=0;i<4-x;i++) y+='0'; s=y+s; }
  cout<<"\nDeğer: ";
  for(int i=0;i<s.length();i+=4) cout<<cevir2_10(s.substr(i,4));
  system("pause"); return 0; }
```

C#

```
// BCD-Decimal
using System;
namespace ornek146
{ class Program
   { static void Main(string[] args)
      { Console.Write("BCD kodu: "); String s=Console.ReadLine();
        int x=s.Length % 4;
        if (x!=0)
         { String y=""; for (int i=0;i<4-x;i++) y+='0'; s=y+s; }
        Console.Write("\nDeğer: ");
        for (int i=0;i<s.Length;i+=4)
         Console.Write("{0}",Convert.ToInt32(s.Substring(i,4), 2));
        Console.ReadLine();    }    }    }
```

JAVA

```
// BCD-Decimal
import java.util.Scanner;
public class ornek146 {
 public static void main(String[] args) {
  Scanner klavye=new Scanner(System.in);
  System.out.print("BCD kodu: "); String s=klavye.nextLine();
  int x=s.length()%4;
  if (x!=0)
   { String y=""; for (int i=0;i<4-x;i++) y=y+'0'; s=y+s; }
  System.out.printf("\nBCD değer: ");
  for (int i=0;i<s.length();i+=4)
   System.out.printf("%d",Integer.parseInt(s.substring(i,i+4),2));}}
```

MATLAB

```
% BCD-Decimal
clear all; clc; s=input('BCD kodu: ','s'); x=mod(length(s),4);
if (x~=0)
  y=''; for i=1:4-x
          y=strcat('0',y);
  end; s=strcat(y,s); end; fprintf('\nDeğer: ');
for i=1:4:length(s)
    fprintf('%d',bin2dec(s(i:i+3)))
end
```

PYTHON

```
# BCD-Decimal
s=input("BCD kodu: "); x=len(s)%4
if (x!=0):
    y=""
    for i in range(4-x):
        y+="0";
    s=y+s;
print("\nDeğer: ",end='');
for i in range(0,len(s),4):
    print(int(s[i:i+4],2),end='')
```

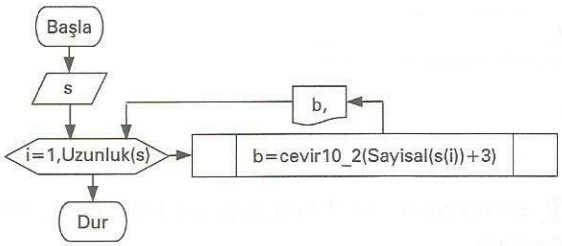
B. +3 kodu

10 tabanındaki herhangi bir pozitif tamsayı, +3 (Excess-3) koduna dönüştürülürken; her basamağa ayrı ayrı 3 eklenmekte ve elde edilen yeni basamak değerleri 2 tabanında 4 bit olarak yazılmaktadır.

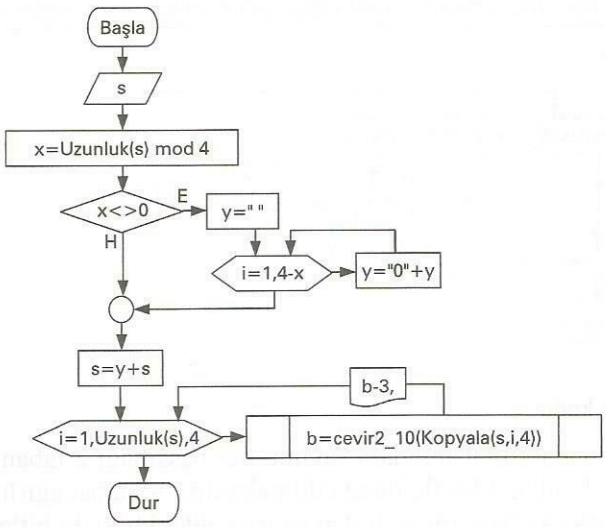
$$(2017)_{10} \rightarrow \left(\underbrace{2}_{+3}\ \underbrace{0}_{+3}\ \underbrace{1}_{+3}\ \underbrace{7}_{+3}\right) \rightarrow \left(\underbrace{5}_{0101}\ \underbrace{3}_{0011}\ \underbrace{4}_{0100}\ \underbrace{10}_{1010}\right)$$

$$(2017)_{10} = (0101\ 0011\ 0100\ 1010)_{+3}$$

Örnek-14.7: Klavyeden girilen pozitif tamsayıyı, +3 koduna dönüştürüp yazdıran program.



Örnek-14.8: Klavyeden +3 kodunda girilen sayıyı, 10 tabanına dönüştürüp yazdıran program.



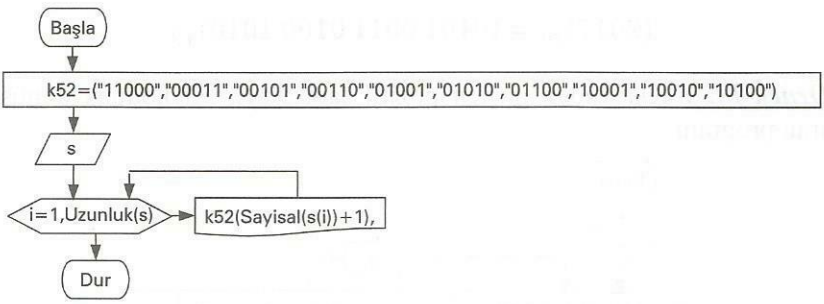
C. 5'te 2 (74210) kodu

5'te 2 kodunda 10 tabanındaki sayının her basamağı 2 tabanında basamak ağırlıkları "74210" olan 5 bit ile ifade edilmekte ve bu 5 bitin içinde kesinlikle iki tane '1' bulunmaktadır. Basamak ağırlıklarına dikkat edilirse 1-9 arası ra-

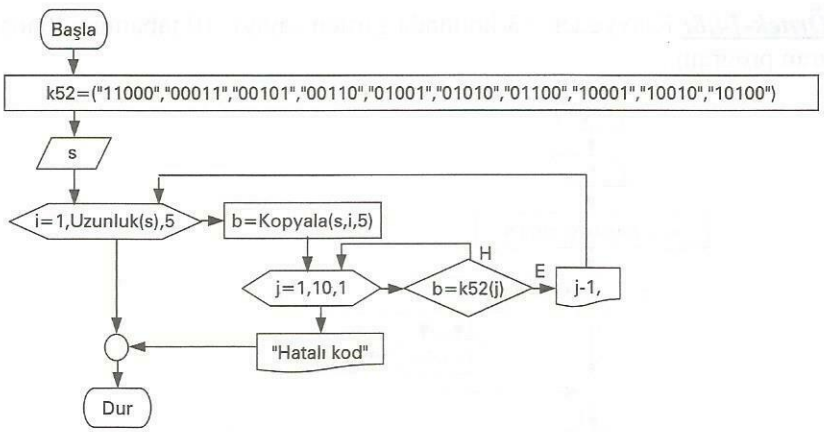
kamlar, 2 tane '1' biti kullanılarak kolaylıkla yazılabilmektedir. Özel olarak 0 rakamı için "11000" kodlaması kabul edilmektedir.

$$\left(\underbrace{2}_{00101}\ \underbrace{0}_{11000}\ \underbrace{1}_{00011}\ \underbrace{7}_{10001}\right) \rightarrow (2017)_{10} = (00101\ 11000\ 00011\ 10001)_{5'te2}$$

Örnek-14.9: Klavyeden girilen pozitif tamsayıyı, 5'te 2 koduna dönüştürüp yazdıran program.



Örnek-14.10: Klavyeden 5'te 2 kodunda girilen sayıyı, 10 tabanına dönüştürüp yazdıran program.

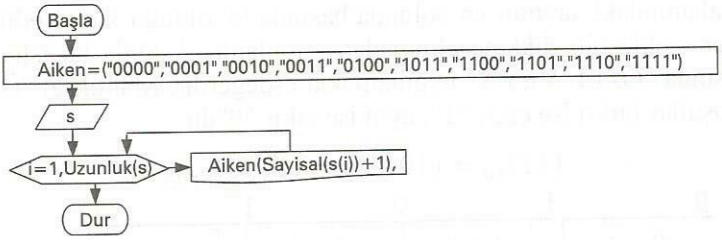


D. Aiken (2421) kodu

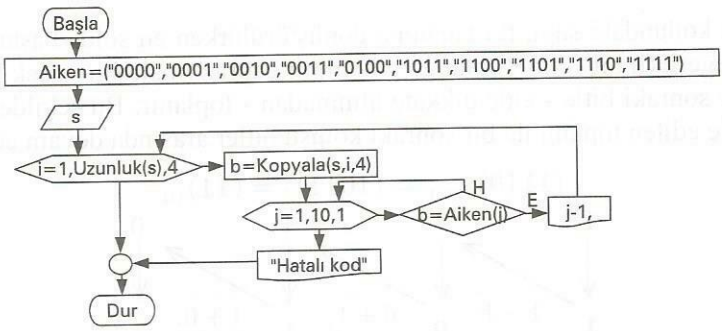
Aiken kodunda 10 tabanındaki sayının her basamağı 2 tabanında, basamak ağırlıkları "2421" olan 4 bit ile ifade edilmektedir. 0-5 arası ağırlıklı olarak sağdaki/sondaki bitler kullanılırken, 6-9 arası ise soldaki/baştaki bitler kullanılmaktadır.

$$\left(\underbrace{2}_{0010}\ \underbrace{0}_{0000}\ \underbrace{1}_{0001}\ \underbrace{7}_{1101}\right) \rightarrow (2017)_{10} = (0010\ 0000\ 0001\ 1101)_{Aiken}$$

Örnek-14.11: Klavyeden girilen pozitif tamsayıyı, Aiken koduna dönüştürüp yazdıran program.



Örnek-14.12: Klavyeden Aiken kodunda girilen sayıyı, 10 tabanına dönüştürüp yazdıran program.

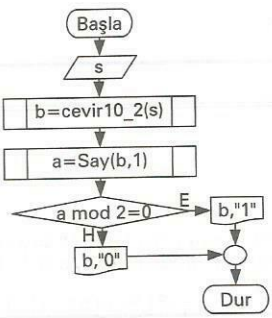


E. Parity (eşlik) kodu

Tek veya çift parity (eşlik) biti olarak düzenlenebilen bu kodlamada; 10 tabanındaki sayı, 2 tabanına dönüştürülmekte ve ardına parity (eşlik) biti eklenmektedir. Bu ekleme tek parity'de, içindeki '1' lerin sayısı tek; çift parity'de ise çift olacak şekilde '0' veya '1' ilave edilerek yapılmaktadır.

$$(15)_{10} = (0001111)_2 \rightarrow \begin{cases} Tek\ parity & \rightarrow \quad \mathbf{00011111} \\ Çift\ parity & \rightarrow \quad \mathbf{00011110} \end{cases}$$

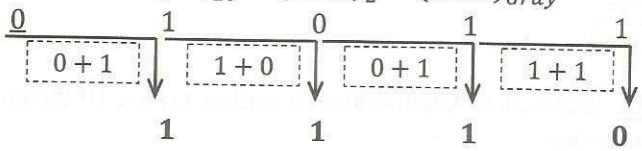
Örnek-14.13: Klavyeden girilen pozitif tamsayıya, tek parity (eşlik) biti ekleyen program.



F. Gray kodu

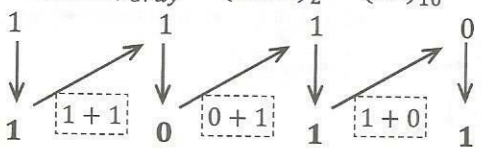
10 tabanındaki bir sayı Gray koduna çevrilmeden önce, 2 tabanına dönüştürülür. 2 tabanındaki sayının en solunda/başında '0' olduğu kabul edilerek, yan yana bitler - elde biti dikkate alınmadan - toplanır. Aslında bu işlem, komşu bitler arasında "ÖZEL VEYA" uygulamakla eşdeğerdir. Mantıksal "ÖZEL VEYA"da koşullar farklı ise çıkış "1", aynı ise çıkış "0"dır.

$$(11)_{10} = (1011)_2 = (1110)_{Gray}$$

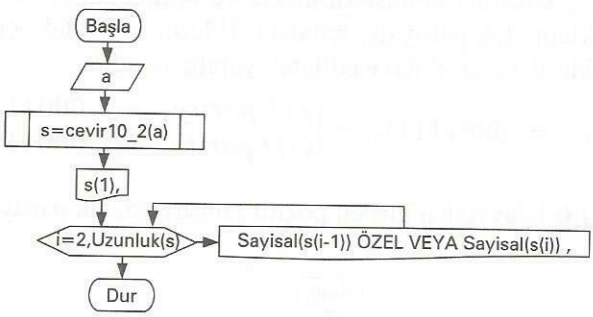


Gray kodundaki sayı, iki tabanına dönüştürülürken en solda/baştaki bit (en yüksek değerlikli bit - MSB) olduğu gibi alınır çapraz (matrisin yedek köşegeni) olarak bir sonraki bitle - elde dikkate alınmadan - toplanır. Bu şekildeki çapraz işlem, elde edilen toplam ile bir sonraki komşu bitler arasında devam eder.

$$(1110)_{Gray} = (1011)_2 = (11)_{10}$$



Örnek-14.14: Klavyeden girilen pozitif tamsayıyı, Gray koduna dönüştüren program.



C

```
/* Decimal-Gray */
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
void main()
{  int a,i; char s[100]; clrscr();printf("Sayı: ");scanf("%d",&a);
   itoa(a,s,2); printf("\nGray kodu: %c",s[0]);
   for(i=1;i<strlen(s);i++)
       printf("%d",((int)s[i-1]-48)^((int)s[i]-48)); getch(); }
```

C++

```
// Decimal-Gray
#include <iostream>
#include <string>
#include <stdlib.h>
using namespace std;
int main()
{ int a,b; char s[100]; system("cls"); cout<<"Sayi: ";cin>>a;
  itoa(a,s,2); cout<<"\nGray kodu: "<<s[0];
  for(int i=1;i<strlen(s);i++)
    { b=((int)s[i-1]-48)^((int)s[i]-48);
      cout<<b;} system("pause");return 0;}
```

```
Sayi: 11

Gray kodu: 1110
```

C#

```
// Decimal-Gray
using System;
namespace ornek1414
{class Program {
  static void Main(string[] args) {
   Console.Write("Sayı: "); int a=Convert.ToInt32(Console.ReadLine());
   String s=Convert.ToString(a,2);
   Console.Write("\nGray kodu: {0}",s[0]);
   for (int i=1;i<s.Length; i++)
    Console.Write("{0}",Convert.ToInt32(s[i-1])^Convert.ToInt32(s[i]));
   Console.ReadLine();    }    }    }
```

JAVA

```
// Decimal-BCD
import java.util.Scanner;
public class ornek1414 {
 public static void main(String[] args) {
  Scanner klavye=new Scanner(System.in);
  System.out.print("Sayı: "); int a=klavye.nextInt();
  String s=Integer.toString(a,2);
  System.out.printf("\nGray kodu: %c",s.charAt(0));
  for (int i=1;i<s.length();i++)
   System.out.printf("%d",((int)s.charAt(i-1)-48)^
                          ((int)s.charAt(i)-48)); }}
```

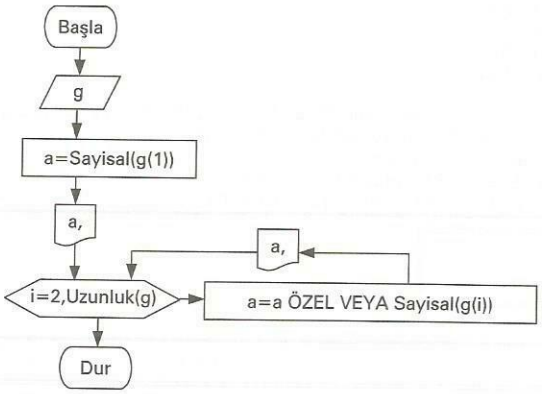
MATLAB

```
% Decimal-Gray
clear all; clc;
s=input('Sayı: '); s=dec2bin(s); fprintf('\nGray kodu: %c',s(1));
for i=2:length(s)
   fprintf('%d',xor(str2num(s(i-1)),str2num(s(i))));
end
```

PYTHON

```
# Decimal-Gray
a=int(input("Sayı: ")); s=bin(a)[2:];
print("\nGray kodu: %c"%s[0],end='')
for i in range(1,len(s)):
    print(int(s[i-1])^int(s[i]),end='')
```

Örnek-14.15: Klavyeden girilen Gray kodundaki sayıyı, 2 tabanına dönüştüren program.



C

```
/* Gray-Binary */
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{  char g[100];int a,i; clrscr();printf("Gray kodunda sayı: ");gets(g);
   a=(int)g[0]-48; printf("\n2 tabanındaki karşılığı: %d",a);
   for(i=1;i<strlen(g);i++)
       { a=a^(int)g[i]-48; printf("%d",a); }   getch(); }
```

C++

```
// Gray-Binary
#include <iostream>
#include <string.h>
using namespace std;
int main()
{ string g; system("cls"); cout<<"Gray kodunda sayı: "; cin>>g;
  int a=(int)g[0]-48; cout<<"\n2 tabanındaki karşılığı: "<<a;
  for(int i=1;i<g.length();i++)
       { a=a^(int)g[i]-48;  cout<<a; } system("pause");return 0;}
```

C#

```
// Gray-Binary
using System;
namespace ornek1415
{class Program {
  static void Main(string[] args) {
   Console.Write("Gray kodunda sayı: "); String g=Console.ReadLine();
   int a=Convert.ToInt32(g[0]-48);
   Console.Write("\n2 tabanındaki karşılığı: {0}",a);
   for (int i=1;i<g.Length; i++)
     { a=a^Convert.ToInt32(g[i]-48); Console.Write("{0}", a); }
   Console.ReadLine();    }    }    }
```

```
Gray kodunda sayı: 1110

2 tabanındaki karşılığı: 1011
```

JAVA

```
// Gray-Binary
import java.util.Scanner;
public class ornek1415 {
 public static void main(String[] args) {
  Scanner klavye=new Scanner(System.in);
   System.out.print("Gray kodunda sayı: "); String g=klavye.nextLine();
   int a=(int)(g.charAt(0)-48);
   System.out.printf("\n2 tabanındaki karşılığı: %d",a);
   for (int i=1;i<g.length();i++)
       { a=a^((int)g.charAt(i)-48); System.out.printf("%d",a);  }}}
```

MATLAB

```
% Gray-Binary
clear all; clc; g=input('Gray kodunda sayı: ','s'); a=str2num(g(1));
fprintf('\n2 tabanındaki karşılığı: %d',a);
for i=2:length(g) a=xor(a,str2num(g(i))); fprintf('%d',a); end
```

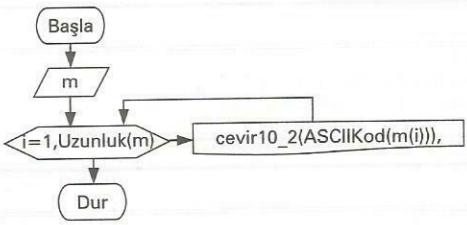
PYTHON

```
# Gray-Binary
g=input("Gray kodunda sayı: ")
a=int(g[0]); print("\n2 tabanındaki karşılığı: %d"%a,end='')
for i in range(1,len(g)):
    a=a^int(g[i]); print("%d"%a,end='')
```

III. ALFASAYISAL KODLAR

Kullanılan bütün rakam, harf, simge vb. karakterleri, tamsayıların iki tabanındaki karşılıklarıyla kodlamak mümkündür. En çok kullanılan alfasyısal kodlama sistemleri ASCII (American Standard Code for Information Interchange) (Ek-A) ve EBCDIC (Extended Binary Coded Desimal Interchange Code)'dir .

Örnek-14.16: Klavyeden girilen mesajı ASCII kod tablosuna göre ikili kodlayan program.



C

```
/* Mesaj-Binary (ASCII) */
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <stdlib.h>
void main()
{ char m[100],m1[7]; int i; clrscr();printf("Mesaj: ");gets(m);
  printf("\nİkili (ASCII) kodlama: ");
  for(i=0;i<strlen(m);i++)
    { itoa((int) m[i],m1,2); printf("%s",m1); }   getch(); }
```

C++

```
// Mesaj-Binary (ASCII)
#include <iostream>
#include <string.h>
#include <bitset>
using namespace std;
int main()
{ string m; system("cls"); cout<<"Mesaj: "; cin>>m;
  cout<<"\nİkili (ASCII) kodlama: ";
  for(int i=0;i<m.length();i++)
   { bitset<7>x((int) m[i]); cout<<x; }  system("pause");return 0;}
```

C#

```
// Mesaj-Binary (ASCII)
using System;
namespace ornek1416
{class Program {
  static void Main(string[] args) {
   Console.Write("Mesaj: "); String m=Console.ReadLine();
   Console.Write("\nİkili (ASCII) kodlama: ");
   for (int i=0;i<m.Length;i++)
    Console.Write("{0}",Convert.ToString((int) m[i],2));
   Console.ReadLine();     }     }     }
```

JAVA

```
// Mesaj-Binary (ASCII)
import java.util.Scanner;
public class ornek1416 {
 public static void main(String[] args) {
  Scanner klavye=new Scanner(System.in);
  System.out.print("Mesaj: "); String m=klavye.nextLine();
  System.out.printf("\nİkili (ASCII) kodlama: ");
  for (int i=0;i<m.length();i++)
   System.out.printf("%s",Integer.toBinaryString((int)m.charAt(i)));}}
```

```
Mesaj: Bursa

İkili (ASCII) kodlama: 1000010111010111100101110011110001
```

MATLAB

```
% Mesaj-Binary (ASCII)
clear all; clc; m=input('Mesaj: ','s');
fprintf('\nİkili (ASCII) kodlama: ');
for i=1:length(m) fprintf(dec2bin(m(i))); end
```

PYTHON

```
# Mesaj-Binary (ASCII)
m=input("Mesaj: "); print("\nİkili (ASCII) kodlama: ",end='')
for i in range(0,len(m)):
    print("%s"%bin(ord(m[i]))[2:],end='')
```

IV. ROMEN RAKAMLARI

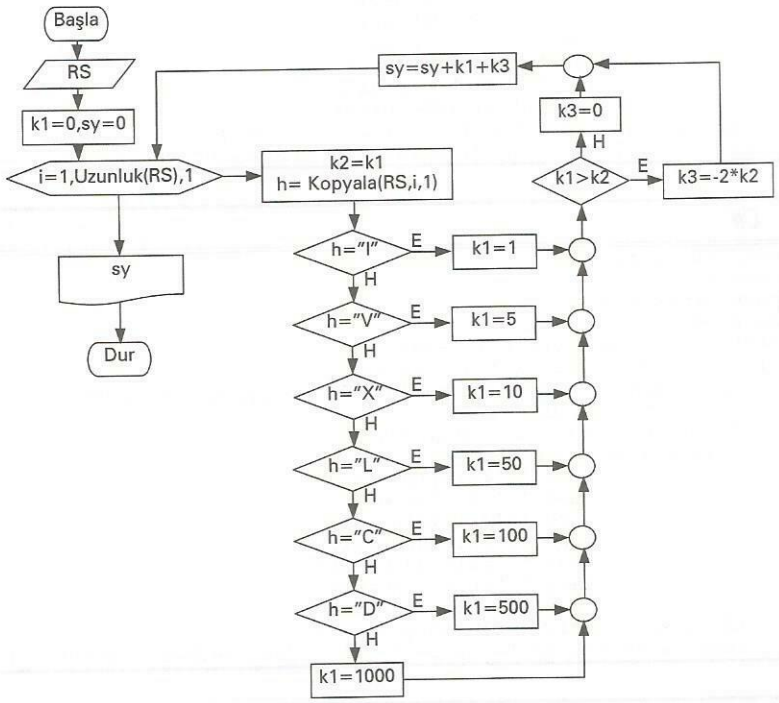
Antik Roma'da ortaya çıkan Romen (Roma) rakamları, 7 rakamdan/harften (I, V, X, L, C, D, M) oluşmaktadırlar. Romen rakamlarıyla sayılar yazılırken/oluşturulurken bazı kurallara uyulması gerekmektedir:

- Bir rakam (harf), yan yana en fazla üç kere yazılabilir.

- Kendisinden daha küçük değerlikli bir rakam;
  - ✓ Bir rakamın sağına/arkasına gelirse toplanır,
  - ✓ Bir rakamın soluna/önüne gelirse çıkarılır.

Romen rakamı	I	V	X	L	C	D	M
Karşılığı	1	5	10	50	100	500	1000

Örnek-14.17: Romen rakamlarıyla girilen sayıyı dönüştüren program.




```
/* Romen rakamları */
#include <stdio.h>
#include <conio.h>
#include <string.h>
char rs[255]; int sy=0,k1=0,k2,k3,i;
void main()
{ clrscr();printf("Romen rakamlarıyla sayıyı giriniz: ");gets(rs);
  for(i=0;i<strlen(rs);i++)
   { k2=k1; switch (rs[i])
            { case 'I':k1=1;break;
              case 'V':k1=5;break;
              case 'X':k1=10;break;
              case 'L':k1=50;break;
              case 'C':k1=100;break;
              case 'D':k1=500;break;
              case 'M':k1=1000;break; }
     if (k1>k2) k3=-2*k2; else k3=0;sy=sy+k1+k3; }
  printf("\nSayı: %d",sy); getch(); }
```

C++

```
// Romen rakamları
#include <iostream>
#include <cstring>
using namespace std;
char rs[255]; int sy=0,k1=0,k2,k3;
int main()
{ system("cls");cout<<"Romen rakamlariyla sayiyi giriniz: ";cin>>rs;
  for(int i=0;i<strlen(rs);i++)
    { k2=k1; switch (rs[i])
                { case 'I':k1=1;break;
                  case 'V':k1=5;break;
                  case 'X':k1=10;break;
                  case 'L':k1=50;break;
                  case 'C':k1=100;break;
                  case 'D':k1=500;break;
                  case 'M':k1=1000;break; }
       k3=(k1>k2)? k3=-2*k2 : k3=0; sy=sy+k1+k3; }
    cout<<"\nSayi: "<<sy<<endl; system("pause");return 0; }
```

C#

```
// Romen rakamları
using System;
namespace ornek1417
{ class Program
   { static void Main(string[] args)
      { int sy=0,k1=0,k2,k3; String rs;
        Console.Write("Romen rakamlarıyla sayıyı giriniz: ");
        rs=Console.ReadLine();
        for (int i=0;i<rs.Length;i++)
        { k2=k1; switch(rs[i])
                  { case 'I':k1=1;break;
                    case 'V':k1=5;break;
                    case 'X':k1=10;break;
                    case 'L':k1=50;break;
                    case 'C':k1=100;break;
                    case 'D':k1=500;break;
                    case 'M':k1=1000;break; }
       k3=k1>k2 ? -2*k2 : 0; sy=sy+k1+k3; }
       Console.WriteLine("\nSayı: "+sy);  Console.ReadLine(); } } }
```

JAVA

```
// Romen rakamları
import java.util.Scanner;
public class ornek1417 {
public static void main(String[] args)
{  int sy=0,k1=0,k2,k3; String rs;
   Scanner klavye=new Scanner(System.in);
   System.out.print("Romen rakamlarıyla sayıyı giriniz: ");
   rs=klavye.nextLine();
   for (int i=0;i<rs.length();i++)
       { k2=k1; switch(rs.charAt(i))
                  { case 'I':k1=1;break;
                    case 'V':k1=5;break;
                    case 'X':k1=10;break;
                    case 'L':k1=50;break;
                    case 'C':k1=100;break;
                    case 'D':k1=500;break;
                    case 'M':k1=1000;break; }
        if(k1>k2) k3=-2*k2; else k3=0; sy=sy+k1+k3; }
    System.out.print("\nSayı: "+sy); }}
```

MATLAB

```
% Romen rakamları
clear all; clc;
rs=input('Romen rakamlarıyla sayıyı giriniz: ','s'); sy=0; k1=0;
for i=1:length(rs)
    k2=k1; switch (rs(i))
        case 'I'
            k1=1;
        case 'V'
            k1=5;
        case 'X'
            k1=10;
        case 'L'
            k1=50;
        case 'C'
            k1=100;
        case 'D'
            k1=500;
        case 'M'
            k1=1000;
    end; if (k1>k2) k3=-2*k2; else k3=0; end; sy=sy+k1+k3; end;
fprintf('\nSayı: %d\n',sy);
```

```
Romen rakamlarıyla sayıyı giriniz: MMXVII

Sayı: 2017
```

PYTHON

```
# Romen rakamları
rs=input("Romen rakamlarıyla sayıyı giriniz: "); sy=0; k1=0
for i in range(len(rs)):
    k2=k1
    if rs[i]=="I":
        k1=1
    elif rs[i]=="V":
        k1=5
    elif rs[i]=="X":
        k1=10
    elif rs[i]=="L":
        k1=50
    elif rs[i]=="C":
        k1=100
    elif rs[i]=="D":
        k1=500
    else:
        k1=1000
    if k1>k2:
        k3=(-2)*k2
    else:
        k3=0
    sy+=(k1+k3)
print("\nSayı: %d"%sy)
```

V. MORSE ALFABESİ

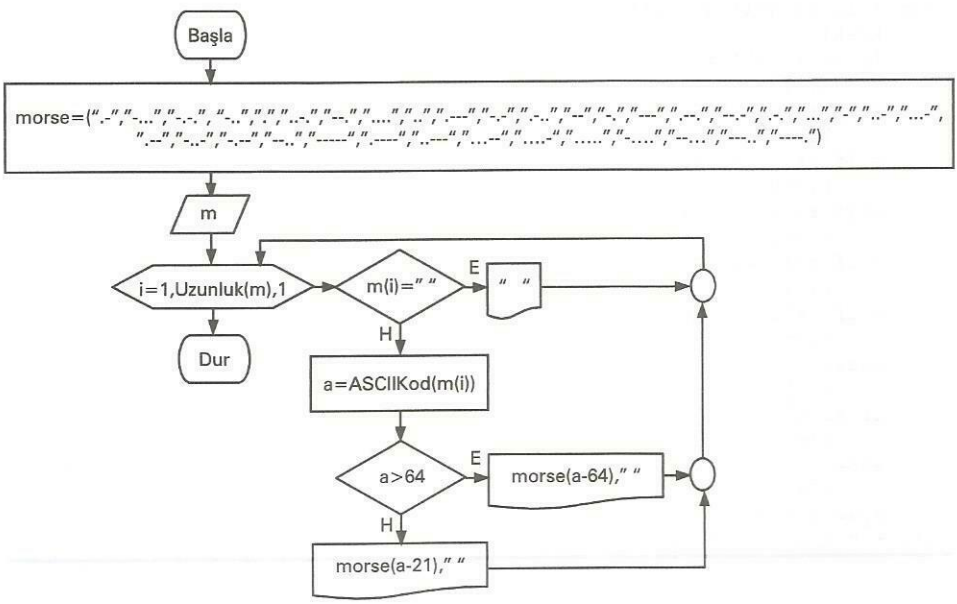
Samuel Morse (1791-1872) tarafından ortaya atılan Morse alfabesi, kısa (nokta) ve uzun (tire) işaretlerden oluşmaktadır. Karakter ve rakamların Morse alfabesindeki karşılıkları, Tablo 14.1'de verilmektedir.

Tablo 14.1: Morse alfabesi (kodları)

A	.-	J	.---	S	...	1	.----
B	-...	K	-.-	T	-	2	..---
C	-.-.	L	.-..	U	..-	3	...--
D	-..	M	--	V	...-	4	....-
E	.	N	-.	W	.--	5	.....
F	..-.	O	---	X	-..-	6	-....
G	..-	P	.--.	Y	-.--	7	--...
H	....	Q	--.-	Z	--..	8	---..
I	..	R	.-.	0	-----	9	----.

Örnek-14.18: Noktalama işaretleri kullanılmadan klavyeden büyük harflerle girilen mesajı "*Morse alfabesi*"yle kodlayan program.

Programda "A-Z" arasındaki harflerin Morse alfabesindeki karşılıkları (Tablo 14.1) "morse" dizisinin 1-26 indisli elemanları, "0-9" arasındaki rakamların karşılıkları da 27-36 indisli elemanları olarak atanırsa; mesajın ilk karakterinden son karakterine kadar açılan döngüyle çağırılan karakterlerin durumuna göre "morse" dizisindeki karşılıkları yazdırılır.



C

```
/* Mesaj - Morse */
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
void main()
{ char *morse[]={".-","-...","-.-.", "-..",".","..-.","--.",
   "....","..",".---","-.-",".-..","--","-.","---",".--.","--.-",".-.",
   "...","-","..-","...-",".--","-..-","-.--","--..","-----",".----",
   "..---","…--","….-","…..","-….","--…","---..","----."};
  char m[255]; int i,a;
  clrscr();printf("Mesaj: ");gets(m); printf("\nMorse kodlaması: ");
  for(i=0;i<strlen(m);i++)
   { a=(int) toupper(m[i]);
     if (a==32) printf(" ");
     else { if (a>64) printf("%s ",morse[a-65]);
            else printf("%s ",morse[a-22]);}}   getch(); }
```

C++

```
// Mesaj-Morse
#include <iostream>
#include <string.h>
#include <ctype.h>
using namespace std;
int main()
{ char *morse[]={".-","-...","-.-.", "-..",".","..-.","--.",
   "....","..",".---","-.-",".-..","--","-.","---",".--.","--.-",".-.",
   "...","-","..-","...-",".--","-..-","-.--","--..","-----",".----",
   "..---","…--","….-","…..","-….","--…","---..","----."};
  string m; int i,a; system("cls"); cout<<"Mesaj: "; getline(cin,m);
  cout<<"\nMorse kodlaması: ";
  for(int i=0;i<m.length();i++)
   { a=(int) toupper(m[i]); cout<<a;
     if (a==32) cout<<" ";
     else { if (a>64) cout<<morse[a-65];<<" "
            else cout<<morse[a-22]<<" "; }}
  system("pause");return 0;}
```

C#

```
// Mesaj - Morse
using System;
namespace ornek1418
{ class Program
   { static void Main(string[] args)
      { String[] morse=new String[36] {".-","-...","-.-.", "-..",".",
                "..-.","--.","....","..",".---","-.-",".-..","--",
                "-.","---",".--.","--.-",".-.","...","-","..-",
                "...-",".--","-..-","-.--","--..","-----",".----",
                "..---","…--","….-","…..","-….","--…","---..","----."};
        Console.Write("Mesaj: ");
        String m=Console.ReadLine().ToUpper();
        Console.Write("\nMorse kodlaması: ");
        for (int i=0;i<m.Length; i++)
         { int a=(int) m[i];
           if (a==32) Console.Write(" ");
           else
            { if (a>64) Console.Write(morse[a-65]);
              else Console.Write(morse[a-22]); }}
        Console.ReadLine(); } } }
```

JAVA

```
// Mesaj - Morse
import java.util.Scanner;
public class ornek1418 {
public static void main(String[] args)
{  String[] morse=new String[] {".-","-...","-.-.", "-..",".",
                "..-.","--.","....","..",".---","-.-",".-..","--",
                "-.","---",".--.","--.-",".-.","...","-",
                "...-",".--","-..-","-.--","--..","-----",".----",
                "..---","...--","....-",".....","-....","--...","---..","----."};
   Scanner klavye=new Scanner(System.in);
   System.out.print("Mesaj: ");
   String m=klavye.nextLine().toUpperCase();
   System.out.printf("\nMorse kodlaması: ");
   for (int i=0;i<m.length();i++)
    { int a=(int) m.charAt(i);
      if (a==32) System.out.print(" ");
      else { if (a>64) System.out.printf(morse[a-65]);
             else System.out.printf(morse[a-22]);}}}}
```

MATLAB

```
% Mesaj - Morse
clear all; clc;
morse={'.-','-...','-.-.', '-..','.','..-.','--.','....','..','.---',
'-.-','.-..','--','-.','---','.--.','--.-','.-.','...','-','..-',
'...-','.--','-..-','-.--','--..','-----','.----','..---','...--',
'....-','.....','-....','--...','---..','----.'};
m=input('Mesaj: ','s'); m=upper(m); fprintf('\nMorse kodlaması: ');
for i=1:length(m)
  a=m(i); if (a==32) fprintf(' ');
          else
            if (a>64) fprintf('%s ',morse{a-64});
            else fprintf('%s ',morse{a-21}); end;end;end
```

PYTHON

```
# Mesaj-Binary (ASCII)
morse=['.-','-...','-.-.', '-..','.','..-.','--.','....','..','.---',
'-.-','.-..','--','-.','---','.--.','--.-','.-.','...','-','..-',
'...-','.--','-..-','-.--','--..','-----','.----','..---','...--',
'....-','.....','-....','--...','---..','----.'];
m=input("Mesaj: "); m=m.upper(); print("\nMorse kodlaması: ",end='')
for i in range(0,len(m)):
    a=ord(m[i])
    if (a==32):
        print(" ",end='')
    else:
        if (a>64):
            print("%s "%morse[a-65],end='')
        else:
            print("%s "%morse[a-22],end='')
```

```
Mesaj: Bursa 16

Morse kodlaması: -... ..- .-. ... .- .---- -....
```

VI. BAR (ÇUBUK) KODU

Rakamların, farklı şekillerde düzenlenmiş çubuklarla gösterilmesidir. Bu gösterim, iki farklı şekilde yapılabilmektedir: iki seviyeli ve çok seviyeli.

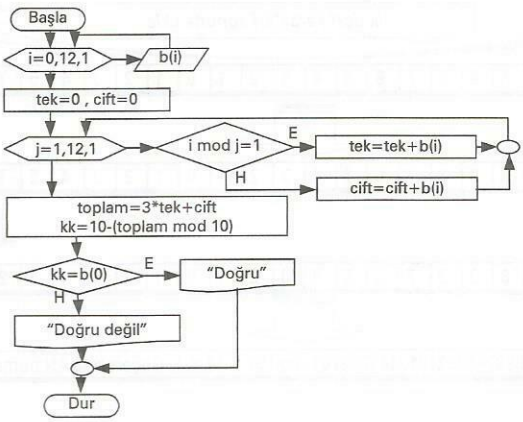
Örnek-14.19: Klavyeden bir ürün için girilen "*barkod*" numarasının doğru olup olmadığını kontrol eden program.

Ürünlerin referans numarasını içeren, ince-kalın siyah çizgi ve boşluklardan (beyaz çizgilerden) oluşan bu etiketlerin yapısı Şekil 14.1'deki gibidir. Etikette siyah çizgiler mantıksal 1 seviyesini, boşluklar da mantıksal 0 seviyesini temsil eder. En ince siyah çizgi "1" iken en kalın siyah çizgi de "1111"dir. Aynı şekilde en ince boşluk, "0" iken en kalın boşluk da "0000"dır.

Şekil 14.1 Örnek barkod şekli

Barkod okuyucusu, okuduğu 13 basamaklı numaranın doğru olup olmadığını şu şekilde test eder:

- Kontrol kodu göz ardı edilip ilk rakam tek olacak şekilde tek-çift şeklinde ayrıştırılır.
- Tek (*tek*) ve çift (*cift*) sıradaki rakamlar ayrı ayrı toplanır.
- Tek sıradakilerin toplamının üç katına çift sıradakilerin toplamı ilave edilir ($toplam = 3tek + cift$).
- 10'a tam bölünebilen tamsayıya eşit olacak şekilde ilave edilecek rakam bulunur. Kontrol kodu ile bu rakam aynı ise, barkod doğru okunmuş demektir.



VII. IBAN/KREDİ KARTI/IMEI KODLAMA

IBAN (International Bank Account Number) bir uluslararası hesap numarası standardıdır. Ülkeler için IBAN numarası uzunlukları (en fazla 34 basamaklı olabilir) ve kodları (her ülke için harflerden oluşmuş ülke kodları belirlenmiştir) arasında farklılıklar olsa da doğruluğunu kontrol etmek için ISO7064 mod 97 (mod 97-10) yöntemi kullanılabilir. Ülkemizdeki hesaplar için IBAN uzunluğu 26 karakter olup yapısı aşağıdaki gibidir.

T	R																								

Ülke kodu	*Kontrol kodu*	*Banka kodu*	*Rezerv*	*Hesap numarası*
2 hane	*2 hane*	*5 hane*	*1 hane*	*16 hane*

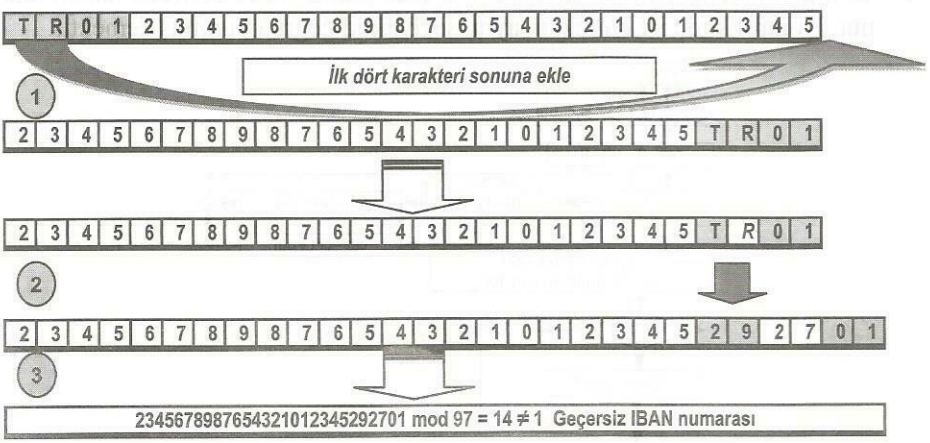
Mod 97'ye göre IBAN kontrol/doğrulama adımları genel olarak aşağıdaki gibi özetlenebilir:

i. İlgili ülke için IBAN karakter uzunluğu ve ülke kodu kontrol edilir. Karakter (basamak) uzunluğu ve/veya ülke kodu doğru değilse "hata/yanlış" mesajı verilerek işlem sonlandırılır.

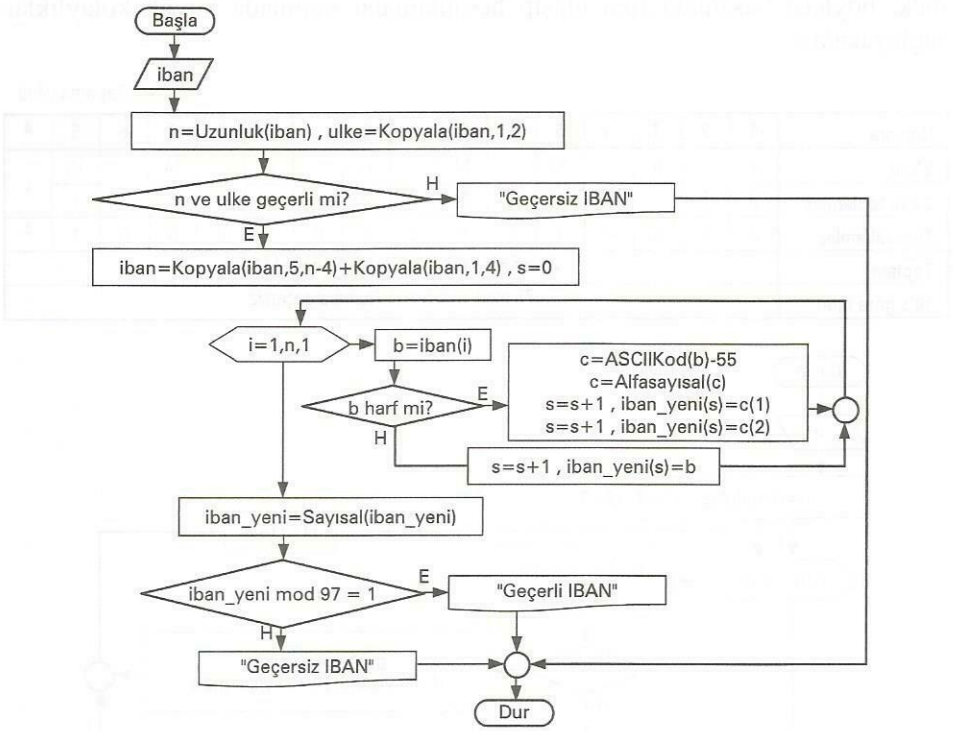
ii. IBAN numarasının ilk 4 karakteri (basamağı) alınarak sonuna eklenir.

iii. IBAN numarasındaki tüm harfler, sayılarla değiştirilir. Bu sayılar "A" için 10, "B" için 11, "C" için 12 şeklinde sıralı gitmektedir.

iv. Sayısallaştırılmış IBAN numarasının 97'e göre modu alınır. Eğer mod işlemi sonucu 1 ise IBAN numarası doğru/geçerli, değilse yanlış/geçersizdir.



Örnek-14.20: Klavyeden girilen IBAN numarasının geçerli olup olmadığını (doğruluğunu) kontrol eden program.



Örnek-14.21: Klavyeden girilen kredi kartı numarasının geçerli olup olmadığını (doğruluğunu) kontrol eden program.

IBM bilim adamlarından Hans Peter Luhn (1896-1964) tarafından geliştirilen "*Luhn algoritması*" (*Luhn formülü, mod-10 algoritması*); kredi kartı, IMEI gibi kimlik numaralarını doğrulamak (geçerliliği test etmek) için kullanılan basit bir yöntemdir. Test edilecek numaranın geçerliliği aşağıdaki adımlar izlenerek bulunur:

i. Numaranın en sağındaki (sonundaki) ikinci rakamdan başlanarak ve birer atlanarak (yani ikişer ikişer gidilerek) alınan rakamların iki katı alınır.

ii. İki katı alma işlemi sonucunda oluşan sayılar birbiriyle toplanır. Eğer 2 ile çarpma sonucunda iki basamaklı sayı oluştuysa, haneleri toplanır.

iii. Elde edilen toplama mumaradaki işleme alınmayan, yani iki katı alınmayan diğer rakamlar da ilave edilir.

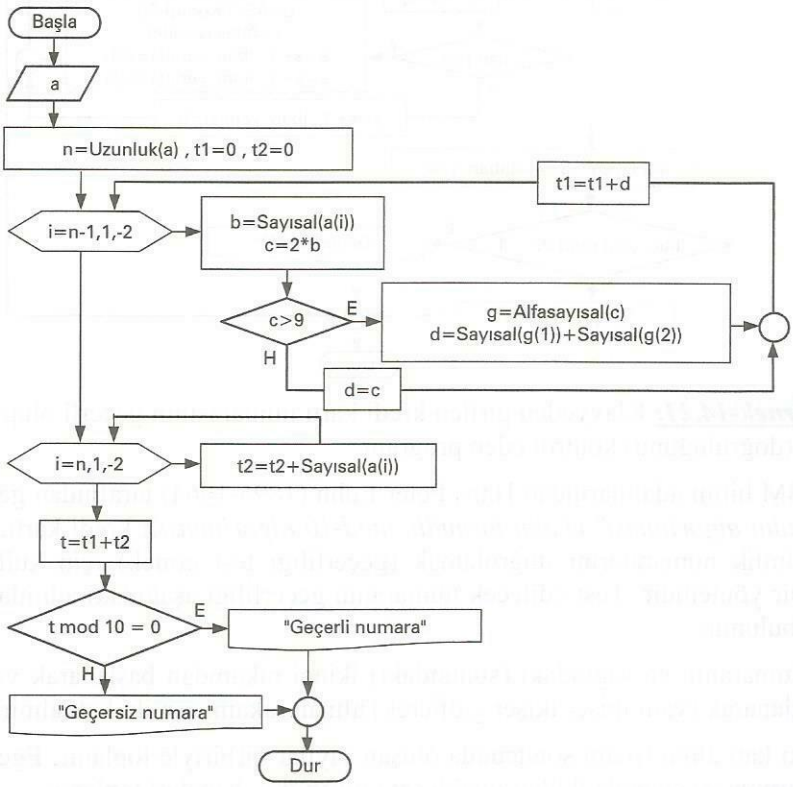
iv. Elde edilen toplam değerinin 10'a göre modu alınır, yani 10'a tam bölünüp bölünmediği kontrol edilir.

v. Eğer 10'a göre modu 0 ise test edilen numara geçerlidir, değilse geçersizdir.

Aşağıda "1234567890987654" numarasının geçerlilik testinin adım adım uygulanması gösterilmektedir. Programda girilen numarayı kelime olarak işlemek, böylece basamaklarına ulaşıp hesaplamalar yapmada büyük kolaylıklar sağlayacaktır.

← *Başlama yönü*

Numara	1	2	3	4	5	6	7	8	9	0	9	8	7	6	5	4
2 katı	2	↓	6	↓	10	↓	14	↓	18	↓	18	↓	14	↓	10	↓
2 katı toplamlar	2		6		1		5		9		9		5		1	
Tüm rakamlar	2	2	6	4	1	6	5	8	9	0	9	8	5	6	1	4
Toplam	2+2+6+4+1+6+5+8+9+0+9+8+5+6+1+4=75															
10'a göre mod	75 mod 10 = 5 ≠ 0 Numara geçersiz															



VIII. RENK KODLAMALARI

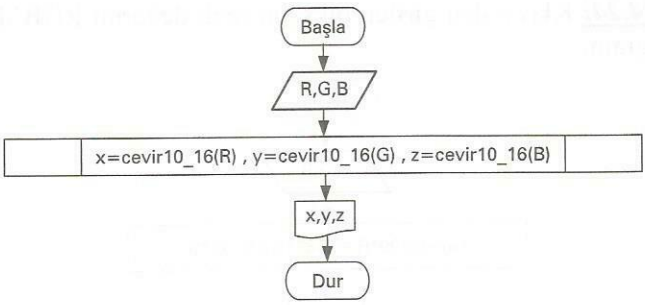
Renkler farklı şekillerde kodlanabilmektedirler: RGB (Red/Kırmızı-Green/Yeşil-Blue/Mavi), HSV (Hue/Renk-Saturation/Doygunluk-Value/Değer), CMYK (Cyan/Camgöbeği-Magenta/Eflatun-Yellow/Sarı-Key/Siyah), HSL (Hue/Renk-Saturation/Doygunluk-Lightness/Açıklık) vb. Bunlardan bazıları, aşağıdaki alt başlıklarda verilmektedir.

A. RGB → Hex

RGB olarak kodlanmış bir rengin HEX kodlamasında; her renk bileşeni, 16 tabanına dönüştürülmektedir.

Giriş	$0 \leq R, G, B \leq 255$
Çıkış	$\left.\begin{matrix} R = (x)_{16} \\ G = (y)_{16} \\ B = (z)_{16} \end{matrix}\right\} \rightarrow HEX = (xyz)$

Örnek-14.22: Klavyeden RGB olarak girilen rengin, HEX karşılığını bulan program.

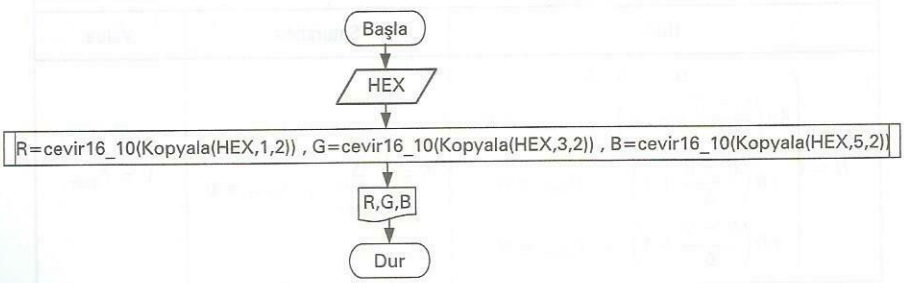


B. Hex → RGB

Altı basamaklı olarak HEX kodlanmış bir renkte; ilk iki basamak R, sonraki iki basamak G ve son iki basamak da B renk bileşenini ifade etmektedir. Bu basamaklar 16 tabanından 10 tabanına dönüştürüldüğünde, ilgili rengin RGB kodu elde edilmiş olmaktadır.

Giriş	HEX
Çıkış	$\left.\begin{matrix} R = (Kopyala(HEX, 1,2))_{16} \\ G = (Kopyala(HEX, 3,2))_{16} \\ B = (Kopyala(HEX, 5,2))_{16} \end{matrix}\right\}$

Örnek-14.23: Klavyeden HEX olarak girilen rengin, RGB karşılığını bulan program.

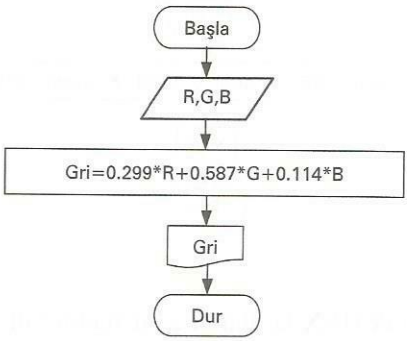


C. RGB → Gri

RGB olarak kodlanmış bir renk, gri seviyeye aşağıdaki ifadelerle dönüştürülebilmektedir.

Giriş	$0 \leq R, G, B \leq 255$
Alternatif çıkışlar	$Gray = \frac{R+G+B}{3}$
	$Gray = 0{,}299R + 0{,}587G + 0{,}114B$
	$Gray = 0{,}2126R + 0{,}7152G + 0{,}0722B$

Örnek-14.24: Klavyeden girilen pikselin renk değerini RGB'den griye dönüştüren program.



D. RGB → HSV

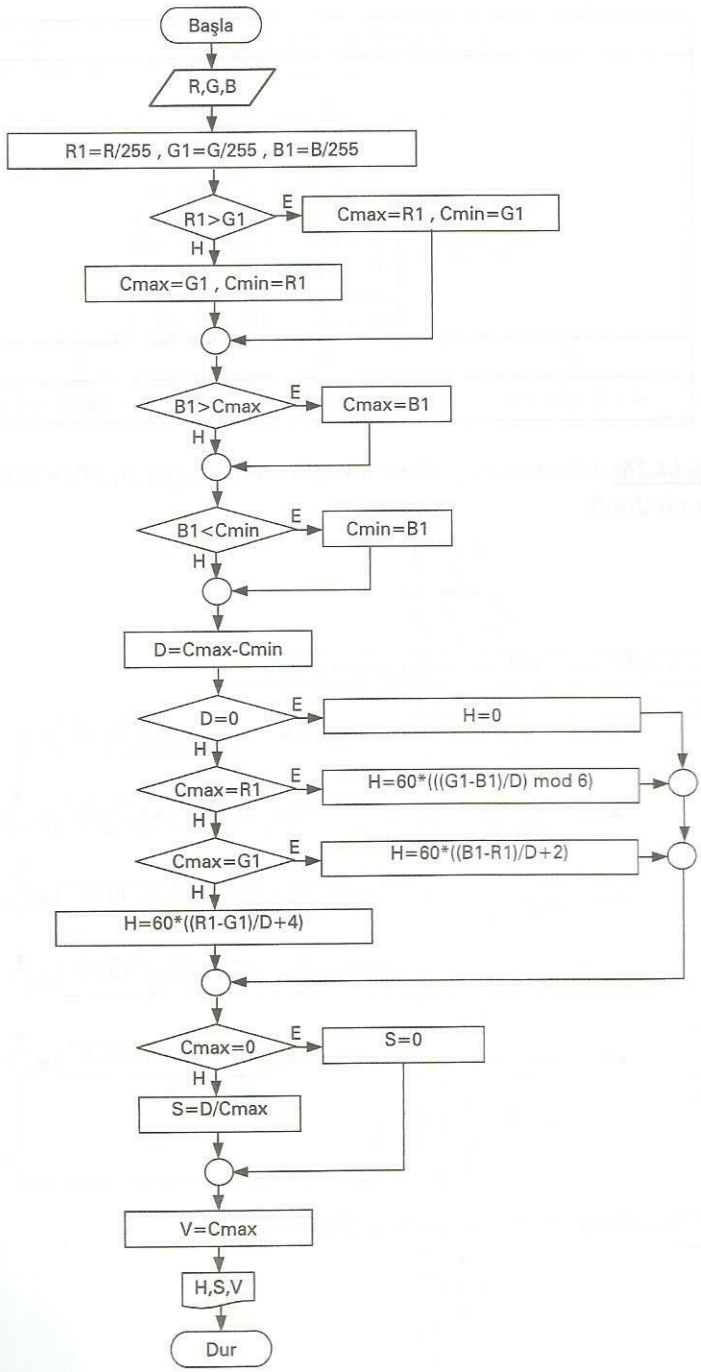
RGB'den HSV'ye dönüşüm aşağıdaki gibi yapılmaktadır.

$$0 \leq R, G, B \leq 255$$

$$\left.\begin{aligned} R' &= \frac{R}{255} \\ G' &= \frac{G}{255} \\ B' &= \frac{B}{255} \end{aligned}\right\} \rightarrow \begin{cases} C_{max} = max(R', G', B') \\ C_{min} = min(R', G', B') \\ \Delta = C_{max} - C_{min} \end{cases}$$

Hue	Saturation	Value
$H = \begin{cases} 0 \ , & \Delta = 0 \\ 60\left(\frac{G'-B'}{\Delta} \bmod 6\right) \ , & C_{max} = R' \\ 60\left(\frac{B'-R'}{\Delta} + 2\right) \ , & C_{max} = G' \\ 60\left(\frac{R'-G'}{\Delta} + 4\right) \ , & C_{max} = B' \end{cases}$	$S = \begin{cases} 0 \ , & C_{max} = 0 \\ \frac{\Delta}{C_{max}} \ , & C_{max} \neq 0 \end{cases}$	$V = C_{max}$

Örnek-14.25: Klavyeden girilen pikselin renk değerini RGB'den HSV'ye dönüştüren program.



E. HSV → RGB

HSV'den RGB'ye dönüşüm, aşağıdaki gibi yapılmaktadır.

$$0 \le H < 360 \,,\, 0 \le S \le 1 \,,\, 0 \le V \le 1$$

$$X = S.V$$

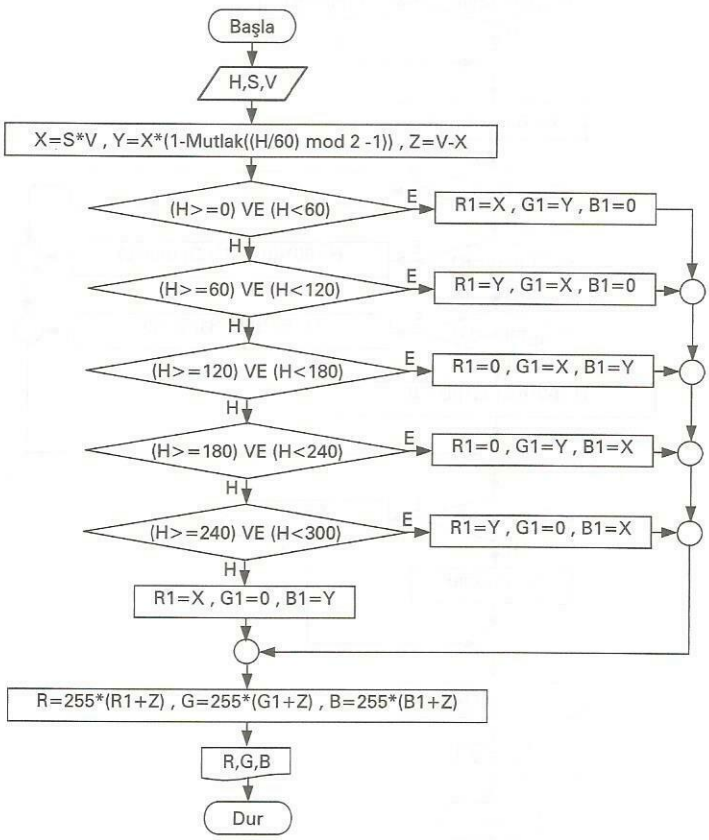
$$Y = X\left(1 - \left|\frac{H}{60} mod\ 2 - 1\right|\right)$$

$$Z = V - X$$

$$(R', G', B') = \begin{cases} (X, Y, 0) & , \ 0 \le H < 60 \\ (Y, X, 0) & , \ 60 \le H < 120 \\ (0, X, Y) & , \ 120 \le H < 180 \\ (0, Y, X) & , \ 180 \le H < 240 \\ (Y, 0, X) & , \ 240 \le H < 300 \\ (X, 0, Y) & , \ 300 \le H < 360 \end{cases}$$

R	G	B
$R = 255(R' + Z)$	$G = 255(G' + Z)$	$B = 255(B' + Z)$

Örnek-14.26: Klavyeden girilen pikselin renk değerini HSV'den RGB'ye dönüştüren program.

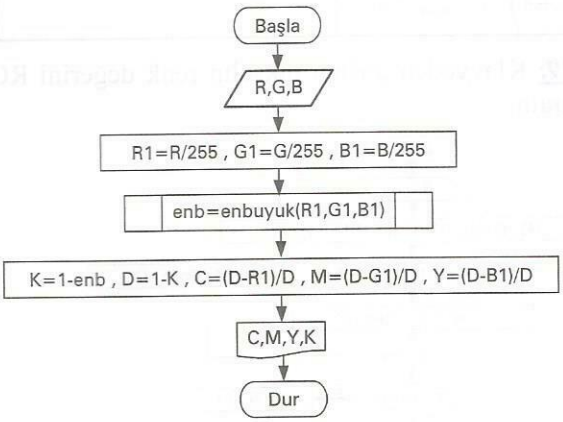


F. RGB → CMYK

RGB olarak kodlanmış bir renk, CMYK'ya aşağıdaki gibi dönüştürülmektedir.

$0 \le R, G, B \le 255$			
$R' = \frac{R}{255}, G' = \frac{G}{255}, B' = \frac{B}{255}$			
Cyan	**Magenta**	**Yellow**	**Black**
$C = \frac{1 - R' - K}{1 - K}$	$M = \frac{1 - G' - K}{1 - K}$	$Y = \frac{1 - B' - K}{1 - K}$	$K = 1 - max(R', G', B')$

Örnek-14.27: Klavyeden girilen pikselin renk değerini HSV'den RGB'ye dönüştüren program.

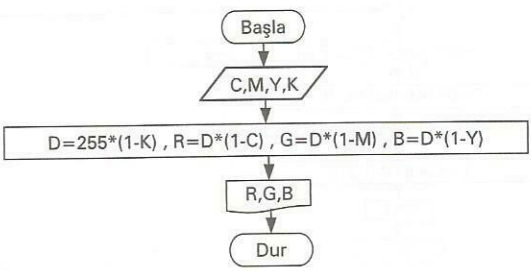


G. CMYK → RGB

CMYK'dan RGB'ye dönüşüm, aşağıdaki gibi yapılmaktadır.

$0 \le C, M, Y, K \le 1$		
R	G	B
$R = 255(1 - C)(1 - K)$	$G = 255(1 - M)(1 - K)$	$B = 255(1 - Y)(1 - K)$

Örnek-14.28: Klavyeden girilen pikselin renk değerini HSV'den RGB'ye dönüştüren program.



H. RGB → HSL

RGB'den HSL'ye dönüşüm aşağıdaki gibi yapılmaktadır.

$$0 \le R, G, B \le 255$$

$$R' = \frac{R}{255}, G' = \frac{G}{255}, B' = \frac{B}{255} \Rightarrow \begin{cases} C_{max} = max(R', G', B') \\ C_{min} = min(R', G', B') \\ \Delta = C_{max} - C_{min} \end{cases}$$

Hue

$$H = \begin{cases} 0 & , \Delta = 0 \\ 60\left(\frac{G' - B'}{\Delta} mod\ 6\right) & , C_{max} = R' \\ 60\left(\frac{B' - R'}{\Delta} + 2\right) & , C_{max} = G' \\ 60\left(\frac{R' - G'}{\Delta} + 4\right) & , C_{max} = B' \end{cases}$$

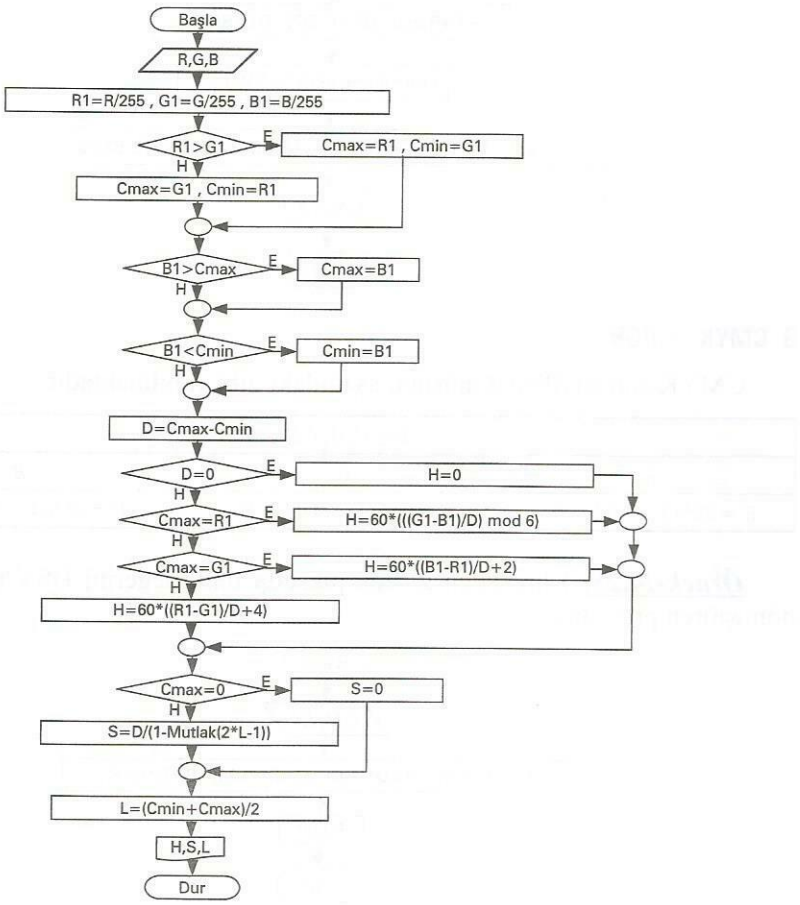
Saturation

$$S = \begin{cases} 0 & , C_{max} = 0 \\ \frac{\Delta}{1 - |2L - 1|} & , C_{max} \neq 0 \end{cases}$$

Lightness

$$L = \frac{C_{min} + C_{max}}{2}$$

Örnek-14.29: Klavyeden girilen pikselin renk değerini RGB'den HSV'ye dönüştüren program.



I. HSL → RGB

HSL'den RGB'ye dönüşüm aşağıdaki gibi yapılmaktadır.

$0 \le H < 360 \, , 0 \le S \le 1 \, , 0 \le V \le 1$

$$X = S(1 - |2L - 1|)$$

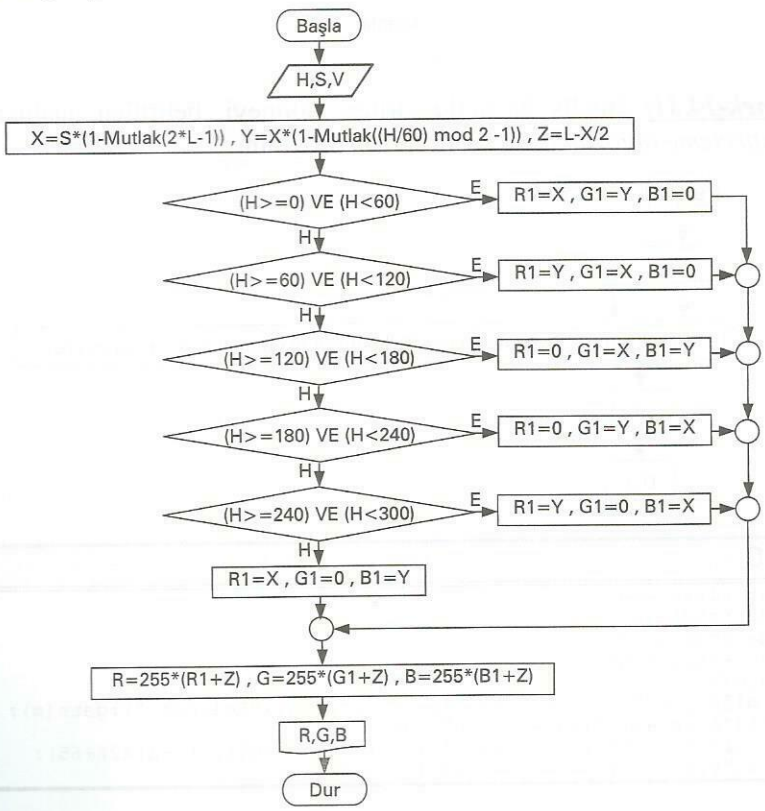
$$Y = X\left(1 - \left|\frac{H}{60} mod\ 2 - 1\right|\right)$$

$$Z = L - \frac{X}{2}$$

$$(R', G', B') = \begin{cases} (X, Y, 0) & , \ 0 \le H < 60 \\ (Y, X, 0) & , \ 60 \le H < 120 \\ (0, X, Y) & , \ 120 \le H < 180 \\ (0, Y, X) & , \ 180 \le H < 240 \\ (Y, 0, X) & , \ 240 \le H < 300 \\ (X, 0, Y) & , \ 300 \le H < 360 \end{cases}$$

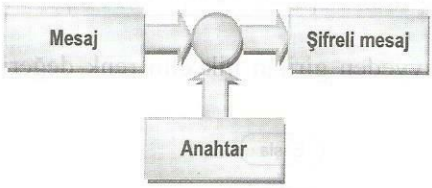
R	G	B
$R = 255(R' + Z)$	$G = 255(G' + Z)$	$B = 255(B' + Z)$

Örnek-14.30: Klavyeden girilen pikselin renk değerini HSV'den RGB'ye dönüştüren program.

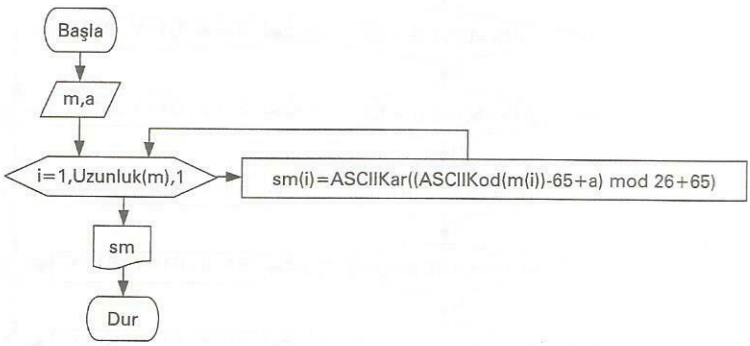


IX. SEZAR ŞİFRELEME

İlk şifreleme algoritmalarından kabul edilen Sezar şifreleme algoritması (Caesar cipher), eski Roma İmparatoru Julius Caesar tarafından savaş zamanlarındaki bilgi gönderiminde kullanıldığı ifade edilmektedir. Bu algoritmada; mesajdaki her karakter, 'anahtar' değeri kadar ötelenmiş başka karakterle yer değiştirerek şifreli mesaj elde edilmektedir. Eğer anahtar değeri 2 ise orijinal mesajdaki her harf, alfabedeki kendisinden iki sonraki harfle yer değiştirir. Yani orijinal mesajdaki "A"→"C", "B"→"D" olur. O zaman şifreleme işleminde, orijinal mesajın her bir harfi çağırılarak ASCII kod değerine anahtar değeri eklenerek şifreleme gerçekleştirilmiş olur. Burada dikkat edilmesi gereken nokta, örneğin sıradaki harf "Z" ise bunun ASCII koduna 2 eklendiğinde "B" harfinin ASCII kodunun elde edilemeyeceğidir. Bu sorun (26 harfli alfabede başa dönme) da 26'ya göre mod alma işlemiyle aşılmaktadır. Dolayısıyla Sezar şifreleme fonksiyonunun matematiksel ifadesi $e(m) = (m + a)\ mod\ 26$ şeklindedir.



Örnek-14.31: Büyük harflerle girilen kelimeyi, belirtilen anahtara göre "*Sezar şifreleme algoritması*"yla şifreleyen program.



C

```
/* Sezar şifreleme */
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{ char m[50],sm[50]; int i,a; clrscr();printf("Kelime: ");gets(m);
  printf("Anahtar: ");scanf("%u",&a);
  for(i=0;i<strlen(m);i++) sm[i]=(char)(((int)m[i]-65+a)%26+65);
  printf("Şifreli kelime: %s",sm);   getch(); }
```

C++

```
// Sezar sifreleme
#include <iostream>
using namespace std;
int main()
{   string m,sm="";int a;
    system("cls");cout<<"Kelime: ";cin>>m;cout<<"Anahtar: ";cin>>a;
    for(int i=0;i<m.size();i++) sm+=(char)(((int)m[i]-65+a)%26+65);
    cout << "Sifreli kelime: " << sm; system("pause");return 0; }
```

C#

```
// Sezar şifreleme
using System;
namespace ornek1431
{ class Program
   { static void Main(string[] args)
      { String m, sm = ""; int a;
        Console.Write("Kelime: ");m=Console.ReadLine();
        Console.Write("Anahtar: ");
        a=Convert.ToInt32(Console.ReadLine());
        for (int i=0;i<m.Length;i++)
          sm+=(char)(((int)m[i]-65+a)%26+65);
     Console.WriteLine("Şifreli kelime: {0}",sm);Console.ReadLine();}}}
```

JAVA

```
// Sezar şifreleme
import java.util.Scanner;
public class ornek1431 {
public static void main(String[] args)
{   String m,sm=""; int a; Scanner klavye=new Scanner(System.in);
   System.out.print("Kelime: ");m=klavye.nextLine();
   System.out.print("Anahtar: ");a=klavye.nextInt();
   for (int i=0;i<m.length();i++)
      sm+=(char)(((int)m.charAt(i)-65+a)%26+65);
   System.out.printf("Şifreli kelime: %s",sm);      }}
```

```
Kelime: BURSA
Anahtar: 2
Şifreli kelime: DWTUC
```

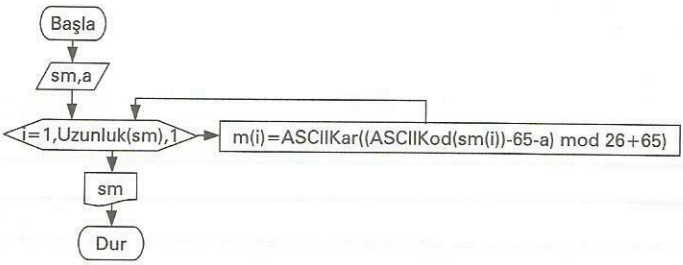
MATLAB

```
% Sezar şifreleme
clear all; clc; m=input('Kelime: ','s'); sm=''; a=input('Anahtar: ');
for i=1:length(m)
    sm(i)=char(mod((m(i)-65+a),26)+65);
end; fprintf('\nŞifreli kelime: %s\n',sm);
```

PYTHON

```
# Sezar şifreleme
m=input("Kelime: "); sm=""; a=int(input("Anahtar: "))
for i in range(len(m)):
    sm+=chr((ord(m[i])-65+a)%26+65)
print("\nŞifreli kelime: %s"%sm)
```

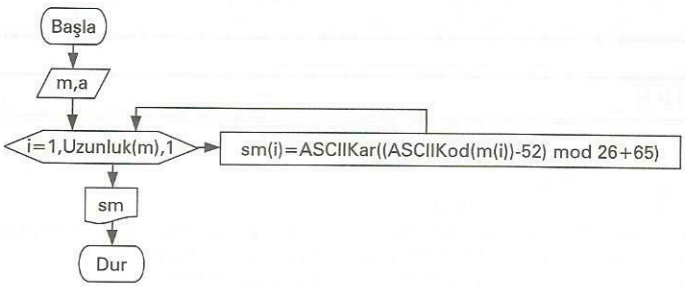
Örnek-14.32: *"Sezar şifreleme algoritması"*yla şifrelenmiş kelimeyi, verilen anahtara göre çözen program.



X. ROT13 ŞİFRELEME

Alfabedeki karakterlerin 13 birim kaydırıldığı, yani Sezar şifreleme algoritmasında anahtar değerinin 13 olduğu durumdur.

Örnek-14.33: Büyük harflerle girilen kelimeyi, *"ROT13 şifreleme algoritması"*yla şifreleyen program.



XI. AFFINE ŞİFRELEME

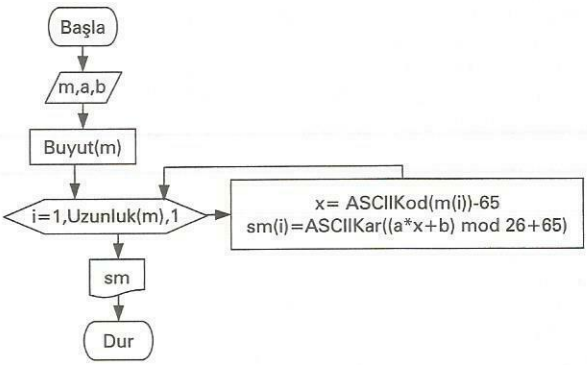
Affine şifreleme yönteminde harfleri ötelemek için $y(x) = ax + b$ doğru denklemi kullanılmaktadır. Yani şifrelenecek olan h harfi, m harfli bir alfabede a ile b anahtarları $(1 \leq a, b \leq m)$ kullanılarak $sh = \{a.h + b\}\ mod\ m$ şeklinde değiştirilmektedir. Benzer şekilde şifrenin çözülmesi de $x = \{a^{-1}(y - b)\}\ mod\ m$ eşitliğiyle yapılmaktadır. Bu eşitlikteki a^{-1} , $mod\ m$ tamsayı grubundaki a nın çarpımsal tersidir. Yani $\{a.a^{-1}\}\ mod\ m = 1$ dir.

Örneğin İngiliz alfabesini (26 harf) kullanarak "BURSA" isminin $a = 5$ ve $b = 1$ anahtarlarıyla şifrelenmesi/şifrenin çözülmesi aşağıda verilmektedir.

	Harf (h)	B	U	R	S	A
	ASCII kodu	66	85	82	83	65
Şifreleme	$x = h - 65$	1	20	17	18	0
	$y = ax + b = 5x + 1$	6	101	86	91	1
	$y\ mod\ 26$	6	23	8	13	1
	Öteleme (+65)	71	88	73	78	66
	Şifrelenmiş harf (sh)	G	X	I	N	B

Şifre çözme	Şifrelenmiş harf (sh)	G	X	I	N	B
	ASCII kodu	71	88	73	78	66
	$y = sh - 65$	6	23	8	13	1
	$a^{-1}(y - b) mod\ 26$	1	20	17	18	0
	ASCII kodu (+65)	66	85	82	83	65
	Harf (h)	B	U	R	S	A

Örnek-14.34: Klavyeden girilen girilen kelimeyi, belirtilen anahtarlara göre "*Affine şifreleme algoritması*"yla şifreleyen program.



C

```
/* Affine şifreleme */
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
void main()
{ char m[100],sm[100]=""; int i,a,b;
  clrscr();printf("Kelime: ");gets(m);
  printf("Anahtar (a): ");scanf("%d",&a);
  printf("Anahtar (b): ");scanf("%d",&b);
  for(i=0;i<strlen(m);i++)
    sm[i]=(char)((a*((int)toupper(m[i])-65)+b)%26+65);
  printf("\nŞifreli kelime: %s",sm);   getch(); }
```

C++

```
/ Affine sifreleme
#include <iostream>
#include <ctype.h>
using namespace std;
int main()
{   string m,sm="";int i,a,b;
    system("cls");cout<<"Kelime: ";cin>>m;
    cout<<"Anahtar (a): ";cin>>a; cout<<"Anahtar (b): ";cin>>b;
    for(int i=0;i<m.size();i++)
      sm+=(char)((a*((int)toupper(m[i])-65)+b)%26+65);
    cout << "\nSifreli kelime: " << sm; system("pause");return 0; }
```

C#

```
// Affine şifreleme
using System;
namespace ornek1434
{ class Program
   { static void Main(string[] args)
      { String m, sm = ""; int i,a,b;
        Console.Write("Kelime: "); m=Console.ReadLine().ToUpper();
        Console.Write("Anahtar (a): ");
        a=Convert.ToInt32(Console.ReadLine());
        Console.Write("Anahtar (b): ");
        b=Convert.ToInt32(Console.ReadLine());
        for (i=0;i<m.Length;i++)
          sm+=(char)((a*((int)m[i]-65)+b)%26+65);
        Console.WriteLine("\nŞifreli kelime: {0}", sm);
        Console.ReadLine(); }}}
```

JAVA

```
// Affine şifreleme
import java.util.Scanner;
public class ornek1434 {
public static void main(String[] args)
{ String m,sm=""; int i,a,b;
  Scanner klavye=new Scanner(System.in);
  System.out.print("Kelime: ");m=klavye.nextLine().toUpperCase();
  System.out.print("Anahtar (a): ");a=klavye.nextInt();
  System.out.print("Anahtar (b): ");b=klavye.nextInt();
  for (i=0;i<m.length();i++)
    sm+=(char)((a*((int)m.charAt(i)-65)+b)%26+65);
  System.out.printf("Şifreli kelime: %s",sm);      }}
```

MATLAB

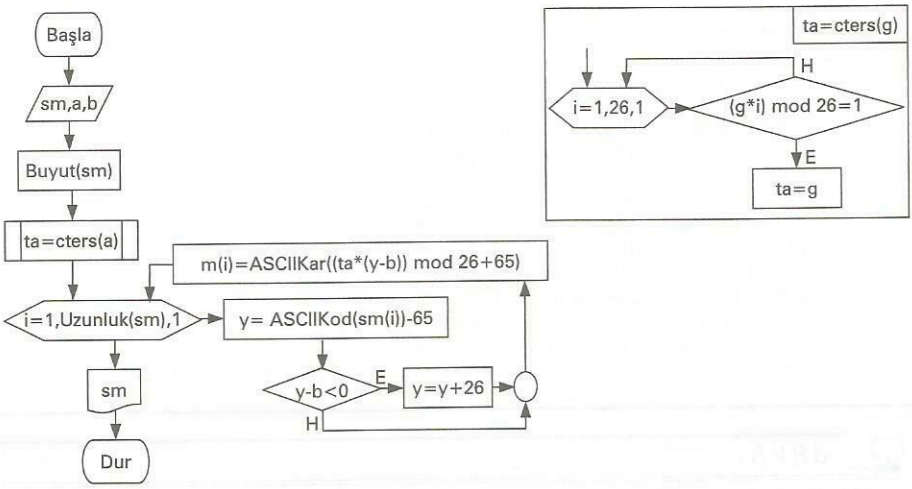
```
% Affine şifreleme
clear all; clc; m=upper(input('Kelime: ','s')); sm='';
a=input('Anahtar (a): '); b=input('Anahtar (b): ');
for i=1:length(m)
    sm(i)=char(mod(a*(m(i)-65)+b,26)+65);
end; fprintf('\nŞifreli kelime: %s\n',sm);
```

```
Kelime: Bursa
Anahtar (a): 5
Anahtar (b): 1

Şifreli kelime: GXINB
```

PYTHON

```
# Affine şifreleme
m=input("Kelime: ").upper(); sm=""
a=int(input("Anahtar (a): ")); b=int(input("Anahtar (b): "))
for i in range(len(m)):
    sm+=chr((a*(ord(m[i])-65)+b)%26+65)
print("\nŞifreli kelime: %s"%sm)
```

Örnek-14.35: Klavyeden girilen girilen ve "*Affine şifreleme algoritması*"yla şifrelenmiş kelimeyi, belirtilen anahtarlara göre çözen program.



C

```
/* Affine şifre çözme */
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>
void main()
{ char sm[100],m[100]=""; int i,a,b,y,ta;
  clrscr();printf("Şifreli kelime: ");gets(sm);
  printf("Anahtar (a): ");scanf("%d",&a);
  printf("Anahtar (b): ");scanf("%d",&b);
  for(i=1;i<27;i++) if ((a*i)%26==1) {ta=i;break;}
  for(i=0;i<strlen(sm);i++)
    { y=(int)toupper(sm[i])-65; if (y-b<0) y+=26;
      m[i]=(char)((ta*(y-b))%26+65); }
  printf("\nŞifresi çözülmüş kelime: %s",m);   getch(); }
```

C++

```
// Affine sifre çözme
#include <iostream>
#include <ctype.h>
using namespace std;
int main()
{ string sm,m="";int i,a,b,y,ta;
  system("cls");cout<<"Şifreli kelime: ";cin>>sm;
  cout<<"Anahtar (a): ";cin>>a; cout<<"Anahtar (b): ";cin>>b;
  for(i=1;i<27;i++) if ((a*i)%26==1) { ta=i;break; }
  for(int i=0;i<sm.size();i++)
    { y=(int)toupper(sm[i])-65; if (y-b<0) y+=26;
      m+=(char)((ta*(y-b))%26+65); }
  cout << "\nSifresi çözülmüş kelime: " << m;
  system("pause");return 0; }
```

C#

```
// Affine şifre çözme
using System;
namespace ornek1435
{ class Program
   { static void Main(string[] args)
      { String sm, m = ""; int i,a,b,y,ta=0;
        Console.Write("Şifreli kelime: ");
        sm=Console.ReadLine().ToUpper();
        Console.Write("Anahtar (a): ");
        a=Convert.ToInt32(Console.ReadLine());
        Console.Write("Anahtar (b): ");
        b=Convert.ToInt32(Console.ReadLine());
        for (i=1;i<27;i++) if ((a*i)%26==1) { ta=i; break; }
        for (i=0;i<sm.Length;i++)
         { y=(int)sm[i]-65; if (y-b<0) y+=26;
           m+=(char)((ta*(y-b))%26+65); }
        Console.WriteLine("\nŞifresi çözülmüş kelime: {0}",m);
        Console.ReadLine(); }}}
```

JAVA

```
// Affine şifre çözme
import java.util.Scanner;
public class ornek1435 {
public static void main(String[] args)
{ String sm,m=""; int i,a,b,y,ta=0;
  Scanner klavye=new Scanner(System.in);
  System.out.print("Şifreli kelime: ");
  sm=klavye.nextLine().toUpperCase();
  System.out.print("Anahtar (a): ");a=klavye.nextInt();
  System.out.print("Anahtar (b): ");b=klavye.nextInt();
  for(i=1;i<27;i++) if ((a*i)%26==1) { ta=i; break;}
  for (i=0;i<sm.length();i++)
    { y=(int)sm.charAt(i)-65; if (y-b<0) y+=26;
      m+=(char)((ta*(y-b))%26+65); }
  System.out.printf("Şifresi çözülmüş kelime: %s",m); }}
```

MATLAB

```
% Affine şifre çözme
clear all; clc; sm=upper(input('Şifreli kelime: ','s')); m='';
a=input('Anahtar (a): '); b=input('Anahtar (b): ');
for i=1:26
    if (mod(a*i,26)==1)
        ta=i;break; end; end;
for i=1:length(sm)
    y=sm(i)-65; if (y-b<0) y=y+26; end
    m(i)=char(mod(ta*(y-b),26)+65);
end; fprintf('\nŞifresi çözülmüş kelime: %s\n',m);
```

PYTHON

```
# Affine şifre çözme
def cters(g):
    for i in range(1,27):
        if ((g*i)%26==1):
            return i
sm=input("Şifreli kelime: ").upper(); m=""
```

```
a=int(input("Anahtar (a): ")); b=int(input("Anahtar (b): "))
ta=cters(a)
for i in range(len(sm)):
    y=ord(sm[i])-65
    if (y-b<0):
        y+=26
    m+=chr((ta*(y-b))%26+65)
print("\nŞifresi çözülmüş kelime: %s"%m)
```

```
Şifreli kelime: GXINB
Anahtar (a): 5
Anahtar (b): 1

Şifresi çözülmüş kelime: BUR-
SA
```

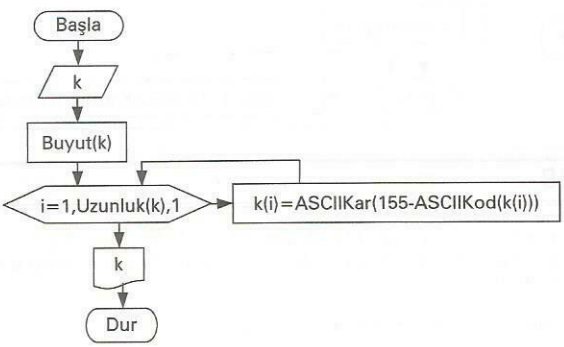
XII. ATBASH ŞİFRELEME

Atbash şifreleme yönteminde harfler, alfabedeki ters sırada yer alan konumdaki harflerle yer değiştirmektedir. Yani "A↔Z", "B↔Y", "C↔X"... (alfabenin ortasına göre simetriği) şeklinde değişimler yapılmaktadır.

Örnek-14.36: Klavyeden rastgele büyüklükte girilen kelimeyi "*Atbash şifrelem algoritması*"yla şifreleyen program.

Program harfleri "A↔Z", "B↔Y", "C↔X"... şeklinde değiştireceğinden bunların ASCII kodlarına dikkat edilirse toplamları 155 olmaktadır.

Harf		Yerini alacak harf		ASCII kodları toplamı
Harf	*ASCII kodu*	*Harf*	*ASCII kodu*	
A	65	Z	90	155
B	66	Y	89	155
...	...	...	...	155



XIII. HILL ŞİFRELEME

Lester S. Hill tarafından geliştirilen bu şifreleme algoritmasında, matrisler kullanılmaktadır. Öncelikle alfabede "A→0", "B→1", "C→2",..., "Z→25" şeklinde harflerin yerine sayılar atanmaktadır (sayısallaştırılmaktadır). Şifreleme anahtarı olarak *nxn* tipinde kare matris kullanıldığında, şifrelenecek mesaj *n*

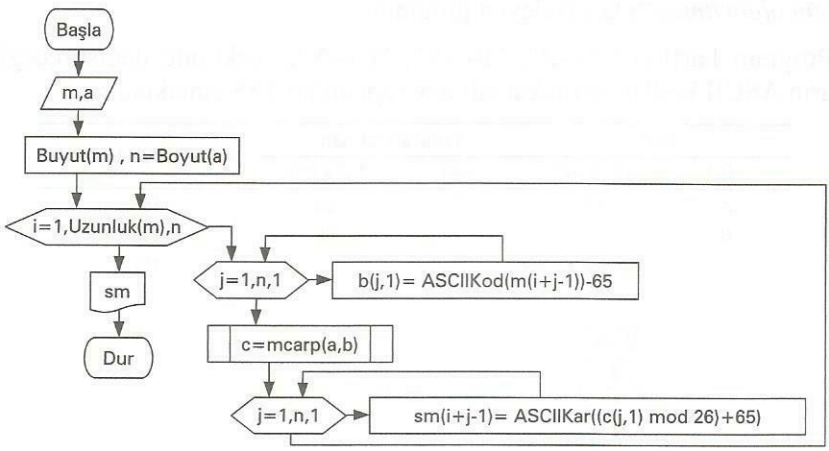
karakterlik gruplara ayrılmakta ve harflerin sayısal değerleri, şifreleme matrisiyle çarpılmakta ve sonucun 26'ya göre modu alınmaktadır.

Örneğin "ANKARA" isminin $\begin{bmatrix}3 & 9\\5 & 11\end{bmatrix}$ şifreleme anahtarı/matrisi kullanılarak Hill algoritmasıyla şifrelenmesi aşağıda verilmektedir.

A	N	K	A	R	A
0	13	10	0	17	0

Grup	İşlem	Sonuç
AN	$\begin{bmatrix}3 & 9\\5 & 11\end{bmatrix}\begin{bmatrix}0\\13\end{bmatrix} mod\ 26 = \begin{bmatrix}13\\13\end{bmatrix}$	NN
KA	$\begin{bmatrix}3 & 9\\5 & 11\end{bmatrix}\begin{bmatrix}10\\0\end{bmatrix} mod\ 26 = \begin{bmatrix}4\\24\end{bmatrix}$	EY
RA	$\begin{bmatrix}3 & 9\\5 & 11\end{bmatrix}\begin{bmatrix}17\\0\end{bmatrix} mod\ 26 = \begin{bmatrix}25\\7\end{bmatrix}$	ZH
ANKARA → NNEYZH		

Örnek-14.37: Klavyeden girilen girilen kelimeyi, belirtilen anahtar matrise göre "*Hill şifreleme algoritması*"yla şifreleyen program.



MATLAB

```
% Hill şifreleme
clear all; clc; m=upper(input('Kelime: ','s')); sm='';
a=input('Matrisi [a b;c d] şeklinde giriniz: '); n=length(a);
for i=1:n:length(m)
    for j=1:n
        b(j,1)=m(i+j-1)-65; end; c=a*b;
    for j=1:n
        sm(i+j-1)=char(mod(c(j,1),26)+65); end;end
fprintf('\nŞifrelenmiş kelime: %s\n',sm);
```

```
Kelime: Ankara
Matrisi [a b;c d] şeklinde giriniz:
[3 9;5 11]

Şifrelenmiş kelime: NNEYZH
```

XIV. POLYBIUS KARE ŞİFRELEME

Bu şifreleme yönteminde; metindeki her karakter, iki karakter ile yer değiştirmektedir. Önce 25 karakterlik anahtar, 5 şifreli karakter için 5x5 tipinde matrise yerleştirilmektedir. Sonra metindeki her karakter, oluşturulmuş olan matristeki konumuna uygun şifreli karakter ile yer değiştirmektedir.

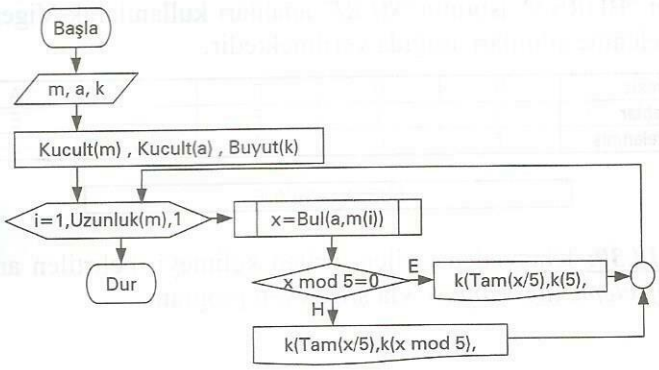
Örneğin "Bursa" isminin '*ywklcrfpebumvgaotzqnsxidh*' anahtarı ve '*ABCDE*' karakterleri kullanılarak Polybius kare şifreleme algoritmasıyla şifrelenmesi aşağıda verilmektedir.

	A	B	C	D	E
A	y	w	k	l	c
B	r	f	p	e	b
C	u	m	v	g	a
D	o	t	z	q	n
E	s	x	i	d	h

Şifresiz	b	u	r	s	a
Şifrelenmiş	BE	CA	BA	EA	CE

bursa → BECABAEACE

Örnek-14.38: Klavyeden girilen girilen kelimeyi, belirtilen anahtara ve karakterlere göre "*Polybius kare şifreleme algoritması*"yla şifreleyen program.



MATLAB

```
% Polybius kare şifreleme
clear all; clc; m=lower(input('Kelime: ','s'));
a=lower(input('Anahtar: ','s')); k=upper(input('Karakterler: ','s'));
fprintf('\nŞifrelenmiş kelime: ');
for i=1:length(m)
    x=strfind(a,m(i));
    if (mod(x,5)==0) fprintf('%c%c',k(ceil(x/5)),k(5));
    else fprintf('%c%c',k(ceil(x/5)),k(mod(x,5))); end; end
```

```
Kelime: bursa
Anahtar: ywklcrfpebumvgaotzqnsxidh
Karakterler: abcde

Şifrelenmiş kelime: BECABAEACE
```

XV. VIGENÈRE ŞİFRELEME

Vigenère şifreleme algoritması aşağıdaki tabloyu kullanmaktadır. Şifrelenecek mesajdaki her karakter (tabloda başlık sütunu), anahtar olarak kullanılan kelimedeki karakterle (tabloda başlık satırı) kesiştiği bu tabloda yer alan yerdeki yeni değeriyle değiştirilir.

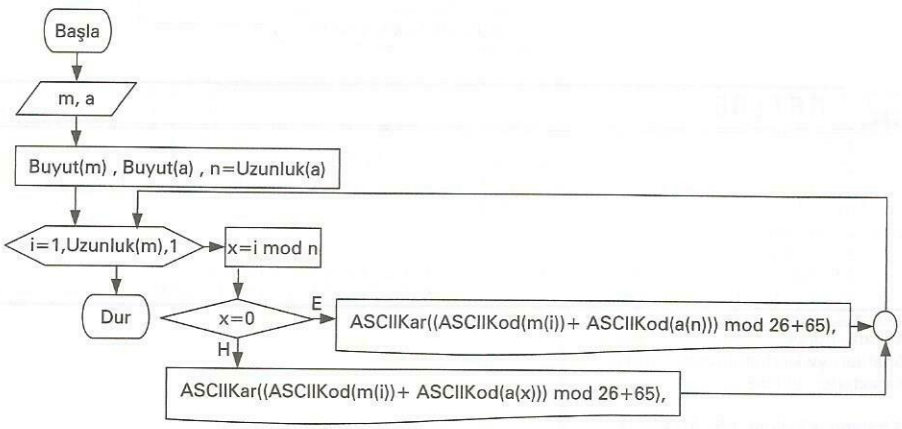
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Örneğin "BURSA" isminin '*SIFRE*' anahtarı kullanılarak Vigenère algoritmasıyla şifrelenme adımları aşağıda verilmektedir.

Şifresiz	B	U	R	S	A
Anahtar	S	I	F	R	E
Şifrelenmiş	T	C	W	J	E

BURSA → TCWJE

Örnek-14.39: Klavyeden girilen girilen kelimeyi, belirtilen anahtara göre "*Vigenère şifreleme algoritması*"yla şifreleyen program.



MATLAB

```
% Vigenère şifreleme
clear all; clc; m=upper(input('Kelime: ','s'));
a=upper(input('Anahtar: ','s')); n=length(a);
fprintf('\nŞifrelenmiş kelime: ');
for i=1:length(m)
    if (mod(i,n)==0)
        fprintf('%c',char(mod(m(i)+a(n),26)+65));
    else
        fprintf('%c',char(mod(m(i)+a(mod(i,n)),26)+65));
    end; end
```

```
Kelime: Bursa
Anahtar: sifre

Şifrelenmiş kelime: TCWJE
```

SORULAR

1. Klavyeden girilen pozitif tamsayıyı, 8 tabanına dönüştüren programın akış diyagramını çizip kodlayınız.

2. Klavyeden girilen 8 tabanındaki pozitif tamsayıyı, 10 tabanına dönüştüren programın akış diyagramını çizip kodlayınız.

3. Klavyeden girilen pozitif tamsayıyı, 16 tabanına dönüştüren programın akış diyagramını çizip kodlayınız.

4. Klavyeden girilen 16 tabanındaki pozitif tamsayıyı, 10 tabanına dönüştüren programın akış diyagramını çizip kodlayınız.

5. Klavyeden girilen pozitif tamsayıyı, Romen rakamlarıyla yazdıran programın akış diyagramını çizip kodlayınız.

6. Klavyeden HSV biçiminde girilen renk kodunu, CMYK biçimine dönüştüren programın akış diyagramını çizip kodlayınız.

7. Klavyeden CMYK biçiminde girilen renk kodunu, HSV biçimine dönüştüren programın akış diyagramını çizip kodlayınız.

8. Klavyeden HSV biçiminde girilen renk kodunu, HSL biçimine dönüştüren programın akış diyagramını çizip kodlayınız.

9. Klavyeden HSL biçiminde girilen renk kodunu, HSV biçimine dönüştüren programın akış diyagramını çizip kodlayınız.

10. Klavyeden HSL biçiminde girilen renk kodunu, CMYK biçimine dönüştüren programın akış diyagramını çizip kodlayınız.

11. Klavyeden CMYK biçiminde girilen renk kodunu, HSL biçimine dönüştüren programın akış diyagramını çizip kodlayınız.

12. ROT13 ile şifrelenmiş kelimeyi çözen programın akış diyagramını çizip kodlayınız.

13. Hill algoritmasıyla şifrelenmiş kelimeyi, belirtilen anahtara/matrise göre çözen programın akış diyagramını çizip kodlayınız.

14. Polybius kare şifreleme algoritmasıyla şifrelenmiş kelimeyi, belirtilen anahtara ve karakterlere göre çözen programın akış diyagramını çizip kodlayınız.

15. Vigenère şifreleme algoritmasıyla şifrelenmiş kelimeyi, belirtilen anahtara göre çözen programın akış diyagramını çizip kodlayınız.

16. Klavyeden girilen kelimeyi ADFGVX şifreleme algoritmasıyla şifreleyip yazdıran programın akış diyagramını çiziniz ve kodlayınız.

17. ADFGVX şifreleme algoritmasıyla şifrelenmiş kelimeyi çözen programın akış diyagramını çizip kodlayınız.

18. Klavyeden girilen kelimeyi Baconian şifreleme algoritmasıyla şifreleyip yazdıran programın akış diyagramını çiziniz ve kodlayınız.

19. Baconian şifreleme algoritmasıyla şifrelenmiş kelimeyi çözen programın akış diyagramını çizip kodlayınız.

20. Klavyeden girilen kelimeyi Beaufort şifreleme algoritmasıyla şifreleyip yazdıran programın akış diyagramını çiziniz ve kodlayınız.

21. Beaufort şifreleme algoritmasıyla şifrelenmiş kelimeyi çözen programın akış diyagramını çizip kodlayınız.

22. Klavyeden girilen kelimeyi Bifid şifreleme algoritmasıyla şifreleyip yazdıran programın akış diyagramını çiziniz ve kodlayınız.

23. Bifid şifreleme algoritmasıyla şifrelenmiş kelimeyi çözen programın akış diyagramını çizip kodlayınız.

24. Klavyeden girilen kelimeyi Porta şifreleme algoritmasıyla şifreleyip yazdıran programın akış diyagramını çiziniz ve kodlayınız.

25. Porta şifreleme algoritmasıyla şifrelenmiş kelimeyi çözen programın akış diyagramını çizip kodlayınız.

26. Girilen bir cümleyi; cümlenin karakterlerinin ASCII kodlarına sırayla 951753 tamsayısının hanelerini ekleyerek şifreleyen programın akış diyagramını çizip herhangi bir dilde yazılımını yapınız.

27. Bir önceki soruda şifrelenmiş cümleyi çözen programın akış diyagramını çizip yazılımını gerçekleyiniz.

28. Klavyeden girilen kelimede sadece sesli harfleri alfabede üç karakter öteleyerek ve yanlarına "*" koyarak şifreleyen programın akış diyagramını çizip kodlayınız.

29. Bir önceki soruda şifrelenmiş olan kelimeyi çözen programın akış diyagramını çizip kodlayınız.

30. Klavyeden girilen tamsayının tek sayı olan hanelerinden 1 çıkararak ve çift sayı olan hanelerine de 1 ekleyerek (örneğin 2017 → 3106) şifreleyen ve bu şifreyi çözen programların akış diyagramlarını çizip kodlayınız.

BÖLÜM 15

SAYISAL ANALİZ UYGULAMALARI

Bu bölümde

- Sayısal analiz yöntemlerini kavrayacak,
- Algoritmalarını hazırlayarak, kodlamalarını öğreneceksiniz.

Bölüm sonunda

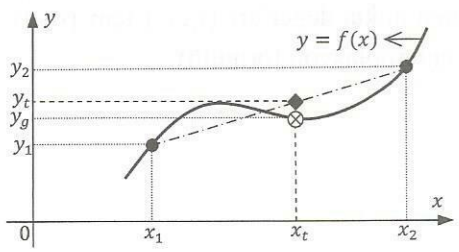
- Sayısal analiz yöntemlerini uygulamalarınızda kullanabilecek,
- Algoritmalarını hazırlayıp kodlayabileceksiniz.

I. ENTERPOLASYON

En genel anlamda enterpolasyon; belirli aralıkta, bilinen değerleri kullanarak, bilinmeyenleri hesaplamaktır (tahmin etmektir). Ölçümler, deneyler veya benzetimler sonucu elde edilen bu değerlerden faydalanılarak; ölçülemeyen, deneyle elde edilemeyen veya benzetimi yapılamayan ara değerler, enterpolasyon yöntemleriyle hesaplanabilmektedir. Bu ara değerleri hesaplamak için, uygun matematiksel fonksiyonlar oluşturulmaktadır.

A. Doğrusal (lineer) enterpolasyon

Bilinen/ölçülen (x_1, y_1) ve (x_2, y_2) değerleri kullanılarak, bu aralıktaki bilinmeyen $x_t \in [x_1, x_2]$ için y_t değeri; iki noktadan geçen doğru denklemi kullanılarak yaklaşık olarak hesaplanabilir. Burada, iki noktadan geçen doğrunun denklemi kullanıldığı için bu yöntem "*doğrusal (lineer) enterpolasyon*" olarak adlandırılır. Bilinen bu noktalardan geçen doğrunun denklemi $y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}.(x - x_1)$ olup x yerine x_t yazılırsa; y_g gerçek değeri, y_t olarak yaklaşık hesaplanır (Şekil 15.1).



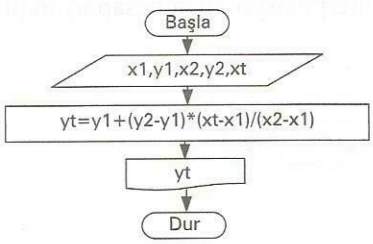
Şekil 15.1 Doğrusal enterpolasyon

Örneğin $(0,0)$ ve $(\pi/6, 0.5)$ değerlerini kullanarak (bu değerler *Sin* fonksiyonundan elde edilmiştir) $\pi/9$ 'daki değerini yaklaşık olarak

$$y_t = 0 + \frac{0.5 - 0}{\pi/6 - 0}.(\pi/9 - 0) = \frac{1}{3} \cong 0.333$$

olup gerçek sonuç $Sin(\pi/9) \cong 0.342$ dir.

Örnek-15.1: Klavyeden girilen iki bilinen değeri kullanarak aradaki istenen değeri, doğrusal enterpolasyonla hesaplayan program.



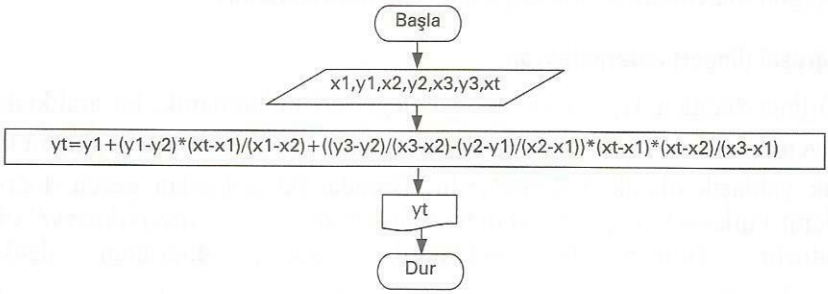
B. Parabolik (kuadratik) enterpolasyon

Bilinen/ölçülen (x_1, y_1) , (x_2, y_2) ve (x_3, y_3) değerleri kullanılarak, bu aralıktaki bu aralıktaki bilinmeyen $x_t \in [x_1, x_3]$ için y_t değeri; üç noktadan geçen parabol denklemi kullanılarak yaklaşık olarak hesaplanabilir. Dolayısıyla parabolik enterpolasyonda hesaplama ifadesi

$$y_t = y_1 + \frac{y_2 - y_1}{x_2 - x_1}.(x_t - x_1) + \left(\frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1} \right).\frac{(x_t - x_1).(x_t - x_2)}{(x_3 - x_1)}$$

şeklindedir.

Örnek-15.2: Klavyeden girilen üç bilinen değeri kullanarak aradaki istenen değeri, parabolik enterpolasyonla hesaplayan program.



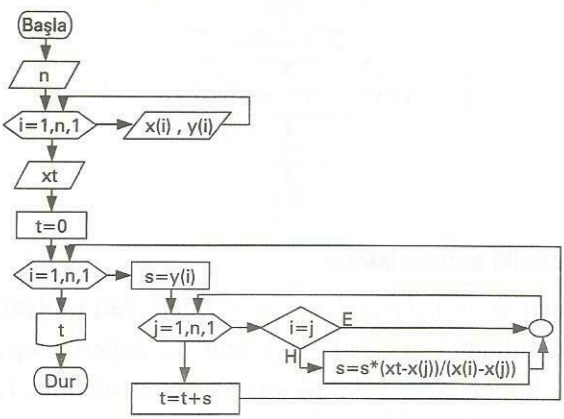
C. Lagrange enterpolasyonu

En popüler enterpolasyon yöntemlerinin başında "*Lagrange enterpolasyonu*" gelmektedir. n adet bilinen nokta değerleri (x_i, y_i) için $(n-1)$. dereceden Lagrange polinomu (Lagrange enterpolasyon formülü)

$$y(x) = \sum_{i=0}^{n} \left\{ y_i . \left[\prod_{\substack{j=0 \\ j \neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j} \right) \right] \right\}$$

şeklindedir. Zaten bu ifade; $n = 1$ için doğrusal enterpolasyon formülünü, $n = 2$ için parabolik enterpolasyon formülünü vermektedir.

Örnek-15.3: Klavyeden girilen n adet bilinen değeri kullanarak aradaki istenen değeri, Lagrange enterpolasyonuyla hesaplayan program.



C

```
/* Lagrange enterpolasyonu */
#include <stdio.h>
#include <conio.h>
void main()
{ int i,j,n; float x[30],y[30],xt,s,t=0;
   clrscr(); printf("Nokta sayısını giriniz: ");scanf("%u",&n);
   printf("\nx      y\n");printf("========== ==========\n ");
   for(i=0;i<n;i++)
    {gotoxy(2,5+i);scanf("%f",&x[i]);gotoxy(21,5+i);scanf("%f",&y[i]);}
   printf("\nAradaki tahmin edilecek değeri giriniz: ");
   scanf("%f",&xt);
   for(i=0;i<n;i++) {s=y[i];for(j=0;j<n;j++)
              {if (i!=j) s=s*(xt-x[j])/(x[i]-x[j]);}  t+=s; }
   printf("\nHesaplanan tahmini değer = %0.10f",t); getch(); }
```

```
Nokta sayısını giriniz: 3

x         y
===     ========
1         1
2         2
3         3

Aradaki tahmin edilecek değeri giriniz: 1.5

Hesaplanan tahmini değer: 1.5
```

C++

```
/* Lagrange enterpolasyonu*/
#include <iostream>
using namespace std;
int main()
{ int i,j,n; float x[30],y[30],xt,s,t=0;
  system("cls"); cout<<"Nokta sayısını giriniz: "; cin>>n;
  for(i=0;i<n;i++)
   {cout<<"x("<<i<<"): ";cin>>x[i];cout<<"y("<<i<<"): ";cin>>y[i];}
  cout<<"\nAradaki tahmin edilecek değeri giriniz: "; cin>>xt;
  for(i=0;i<n;i++) {s=y[i];for(j=0;j<n;j++)
       {if (i!=j) s=s*(xt-x[j])/(x[i]-x[j]);}  t+=s; }
  cout<<"\nHesaplanan tahmini değer: "<<t; system("pause"); return 0; }
```

C#

```
//Lagrange enterpolasyonu
using System;
namespace ornek153
{ class Program
   { static void Main(string[] args)
      { int i,j,n; double xt,s,t=0;
        Console.Write("Nokta sayısını giriniz: ");
        n=Convert.ToInt32(Console.ReadLine());
        double[] x = new double[n]; double[] y = new double[n];
        for (i = 0; i < n; i++)
         { Console.Write("x({0}): ", i);
           x[i] = Convert.ToDouble(Console.ReadLine());
           Console.Write("y({0}): ", i);
           y[i] = Convert.ToDouble(Console.ReadLine()); }
        Console.Write("Aradaki tahmin edilecek değeri giriniz:");
        xt = Convert.ToDouble(Console.ReadLine());
        for (i = 0; i < n; i++)
         { s = y[i]; for (j = 0; j < n; j++)
           { if (i!=j) s=s*(xt-x[j])/(x[i]-x[j]); } t+=s;  }
      Console.WriteLine("Hesaplanan tahmini değer:{0}",t);
      Console.ReadLine();   }    } }
```

JAVA

```
// Lagrange enterpolasyonu
import java.util.Scanner;
public class ornek153 {
  public static void main(String[] args) {
    int i,j,n; float xt,s,t=0;
    Scanner klavye=new Scanner(System.in);
    System.out.print("Nokta sayısını giriniz: ");n=klavye.nextInt();
    float[] x = new float[n]; float[] y = new float[n];
    for(i=0;i<n;i++)
    { System.out.printf("x(%d): ",i); x[i]=klavye.nextFloat();
      System.out.printf("y(%d): ",i); y[i]=klavye.nextFloat(); }
    System.out.print("Aradaki tahmin edilecek değeri giriniz: ");
    xt=klavye.nextFloat();
    for(i=0;i<n;i++)
      { s=y[i]; for(j=0;j<n;j++)
          { if (i!=j) s=s*(xt-x[j])/(x[i]-x[j]); } t+=s;  }
      System.out.printf("Hesaplanan tahmini değer:%.10f\n",t);}}
```

MATLAB

```
% Lagrance enterpolasyonu
clear all; clc; t=0; n=input('Nokta sayısını giriniz: ');
for i=1:n
  fprintf('x(%d): ',i); x(i)=input('');
  fprintf('y(%d): ',i); y(i)=input('');
end; xt=input('Aradaki tahmin edilecek değeri giriniz: ');
for i=1:n
    s=y(i); for j=1:n
       if (i~=j) s=s*(xt-x(j))/(x(i)-x(j)); end; end; t=t+s; end;
fprintf('\nHesaplanan tahmini değer: %0.10f\n',t)
```

PYTHON

```
# Lagrange enterpolasyonu
t=0; x=[]; y=[]; n=int(input("Nokta sayısını giriniz: "))
for i in range(n):
    print("x(%d): "%(i+1),end='')
    x.append(eval(input()))
    print("y(%d): "%(i+1),end='')
    y.append(eval(input()))
xt=eval(input("Aradaki tahmin edilecek değeri giriniz: "))
for i in range(n):
    s=y[i]
    for j in range(n):
        if (i!=j):
            s*=(xt-x[j])/(x[i]-x[j])
    t+=s
print("\nHesaplanan tahmini değer: %0.10f"%t)
-------------------- veya --------------------------------
# Lagrange enterpolasyonu
from scipy.interpolate import lagrange
print("Aralarına boşluk bırakarak x değerlerini giriniz!")
x=[eval(i) for i in input().split()]
print("Aralarına boşluk bırakarak y değerlerini giriniz!")
y=[eval(i) for i in input().split()]
xt=eval(input("Aradaki tahmin edilecek değeri giriniz: "))
lp=lagrange(x,y)
print("\nHesaplanan tahmini değer: %0.10f"%lp(xt))
```

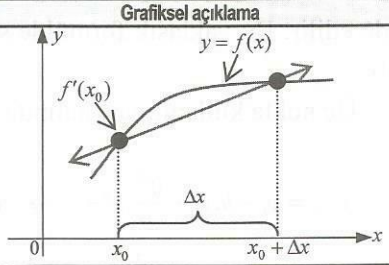
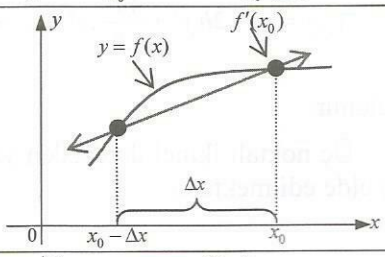
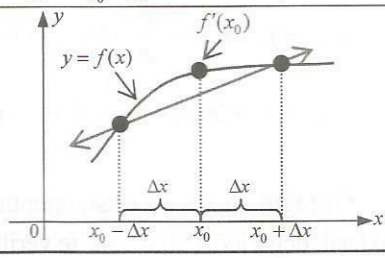
II. SAYISAL TÜREV

Büyüklükteki değişim miktarını gösteren ve geometrik anlamı ilgili noktadan geçen teğetin eğimi olan "*türev*", sayısal olarak "*sonlu farklar*" kullanılarak hesaplanabilmektedir. Genel olarak; herhangi bir $y = f(x)$ fonksiyonunda, h pozitif veya negatif adım miktarı olmak üzere

$$\left.\begin{aligned} &x_i \rightarrow y_i \\ &x_{i+h} = x_i + h \rightarrow y_{i+1} \\ &x_{i-h} = x_i - h \rightarrow y_{i-1} \\ &x_{i+h/2} = x_i + h/2 \rightarrow y_{i+1/2} \\ &x_{i-h/2} = x_i - h/2 \rightarrow y_{i-1/2} \end{aligned}\right\} \Rightarrow için \quad \left\{\begin{aligned} &ileri \quad fark \rightarrow \Delta y_i = y_{i+1} - y_i \\ &geri \quad fark \rightarrow \nabla y_i = y_i - y_{i-1} \\ &merkezi \quad fark \rightarrow \delta y_i = y_{i+1/2} - y_{i-1/2} \end{aligned}\right.$$

dir. Sonlu farklara dayanarak yaklaşık türev hesaplamaları; iki noktalı formülleri ve geometrik açıklamalarıyla özetle Tablo 15.1'de verilmektedir. Hesaplamalarda kullanılan nokta sayılarının artmasıyla, türev hesaplamalarındaki doğruluğun da artacağını unutmayınız. Nokta sayılarına göre türev eşitlikleri, ilgili terimleri yok ederek Tablo 15.2'te yer alan Taylor seri açılımlarından elde edilebilmektedir.

Tablo 15.1 Sonlu farklara dayalı yaklaşık türev hesaplamaları

	Yaklaşık türev	Grafiksel açıklama
İleri farklar	$f'(x_0) = \lim_{\Delta x \to 0} \dfrac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} = \dfrac{\Delta y_i}{h}$	
Geri farklar	$f'(x) = \lim_{\Delta x \to 0} \dfrac{f(x_0) - f(x_0 - \Delta x)}{\Delta x} = \dfrac{\nabla y_i}{h}$	
Merkezi farklar	$f'(x) = \lim_{\Delta x \to 0} \dfrac{f(x_0 + \Delta x) - f(x_0 - \Delta x)}{2\Delta x} = \dfrac{\delta y_i}{2h}$	

Tablo 15.2 İleri, geri ve merkezi sonlu farklar için Taylor seri açılımları

Sonlu fark	Taylor seri açılımı
İleri	$y_{i+1} = y_i + h.y_i' + \frac{h^2}{2!}.y_i'' + \frac{h^3}{3!}.y_i''' + \frac{h^4}{4!}.y_i^{(iv)} + \frac{h^5}{5!}.y_i^{(v)} + \ldots$
Geri	$y_{i-1} = y_i - h.y_i' + \frac{h^2}{2!}.y_i'' - \frac{h^3}{3!}.y_i''' + \frac{h^4}{4!}.y_i^{(iv)} - \frac{h^5}{5!}.y_i^{(v)} + \ldots$
Merkezi	$y_{i+1} = y_i + h.y_i' + \frac{h^2}{2!}.y_i'' + \frac{h^3}{3!}.y_i''' + \frac{h^4}{4!}.y_i^{(iv)} + \frac{h^5}{5!}.y_i^{(v)} + \ldots$ $y_{i-1} = y_i - h.y_i' + \frac{h^2}{2!}.y_i'' - \frac{h^3}{3!}.y_i''' + \frac{h^4}{4!}.y_i^{(iv)} - \frac{h^5}{5!}.y_i^{(v)} + \ldots$

Örneğin ileri farklarla iki noktalı birinci türev eşitliği

$$f(x_i + h) = y_{i+1} = y_i + h.y_i' + \frac{h^2}{2!}.y_i'' + \frac{h^3}{3!}.y_i''' + \frac{h^4}{4!}.y_i^{(iv)} + \ldots$$

Taylor seri açılımında $h.y_i'$ teriminden sonrası atılırsa

$$y_{i+1} = y_i + h.y_i' + \ldots \Rightarrow y_i' \cong \frac{y_{i+1} - y_i}{h} = \frac{\Delta y_i}{h}$$

elde edilir. Bu yaklaşık formülde sadece iki bilinen nokta değeri kullanılmaktadır.

Üç nokta kullanılması halinde

$$\left.\begin{aligned} y_{i+1} &= y_i + h.y_i' + \frac{h^2}{2!}.y_i'' + \ldots \rightarrow (A) \\ y_{i+2} &= y_i + 2h.y_i' + \frac{4h^2}{2!}.y_i'' + \ldots \rightarrow (B) \end{aligned}\right\} \Rightarrow \left\{\begin{aligned} &4A - B \\ &4y_{i+1} - y_{i+2} = 3y_i + 2h.y_i' + \ldots \\ &y_i' \cong \frac{1}{2h}\{-y_{i+2} + 4y_{i+1} - 3y_i\} \end{aligned}\right.$$

bulunur.

Üç noktalı ikinci dereceden yaklaşık sayısal türev formülü aşağıdaki şekilde elde edilmektedir.

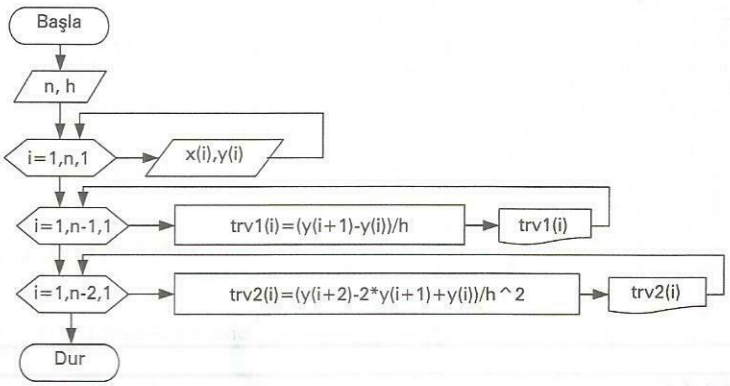
$$\left.\begin{aligned} y_{i+1} &= y_i + h.y_i' + \frac{h^2}{2!}.y_i'' + \ldots \rightarrow (A) \\ y_{i+2} &= y_i + 2h.y_i' + \frac{4h^2}{2!}.y_i'' + \ldots \rightarrow (B) \end{aligned}\right\} \Rightarrow \left\{\begin{aligned} &B - 2A \\ &y_{i+2} - 2y_{i+1} = -y_i + h^2.y_i'' + \ldots \\ &y_i'' \cong \frac{1}{h^2}\{y_{i+2} - 2y_{i+1} + y_i\} \end{aligned}\right.$$

Örneğin $Sin(\pi x)$ fonksiyonunun $x \in [0,0.1]$ için birinci ve ikinci türev gerçek ve yaklaşık değerleri Tablo 15.3'te verilmektedir.

Tablo 15.3 Sayısal türev hesaplama örneği

x	$Sin(\pi x)$	Gerçek 1. türev $\pi Cos(\pi x)$	Yaklaşık 1. türev (2 noktalı)	Gerçek 2. türev $-\pi^2 Sin(\pi x)$	Yaklaşık 2. türev (3 noktalı)
0	0	3.141592654	3.141075908	0	
0.01	0.031410759	3.140042467	3.137976045	-0.310011766	-0.309986269
0.02	0.06279052	3.135393438	3.131779379	-0.619717588	-0.61966662
0.03	0.094108313	3.127650154	3.122492025	-0.928811823	-0.928735434
0.04	0.125333234	3.116820257	3.110123148	-1.236989434	-1.236887699
0.05	0.156434465	3.102914435	3.094684955	-1.543946285	-1.543819304
0.06	0.187381315	3.085946411	3.076192681	-1.849379447	-1.849227347
0.07	0.218143241	3.06593293	3.054664577	-2.152987495	-2.152810425
0.08	0.248689887	3.042893743	3.030121887	-2.454470805	-2.454268939
0.09	0.278991106	3.016851588	3.002588834		
0.1	0.309016994				

Örnek-15.4: Klavyeden girilen n adet nokta çiftini kullanarak iki noktalı birinci ve üç noktalı ikinci türev ileri fark eşitlikleriyle türevleri hesaplayan program.



C

```
/* Sayısal türev */
#include <stdio.h>
#include <conio.h>
void main()
{ int i,n; float x[100],y[100],trv1[99],trv2[98],h;
  clrscr();printf("Nokta sayısı: "); scanf("%u",&n);
  printf("Adım değeri: "); scanf("%f",&h);
  printf("x(0): "); scanf("%f",&x[0]);
  for(i=0;i<n;i++)
   { if (i!=0) x[i]=x[i-1]+h;
     printf("y(%d): ",i); scanf("%f",&y[i]); }
  printf("\n1. türevler:\n");
  for(i=0;i<n-1;i++)
  { trv1[i]=(y[i+1]-y[i])/h;printf("%f\n",trv1[i]); }
  printf("\n2. türevler:\n");
  for(i=0;i<n-2;i++)
  { trv2[i]=(y[i+2]-2*y[i+1]+y[i])/(h*h);printf("%f\n",trv2[i]); }
  getch(); }
```

C++

```
/* Sayısal türev*/
#include <iostream>
using namespace std;
int main()
{ int i,n; float h; system("cls"); cout<<"Nokta sayısı: "; cin>>n;
  float x[n],y[n],trv1[n-1],trv2[n-2];
  cout<<"Adım değeri: "; cin>>h; cout<<"x(0): "; cin>>x[0];
  for(i=0;i<n;i++)
       { if (i!=0) x[i]=x[i-1]+h;
     cout<<"y("<<i<<"): "; cin>>y[i]; } cout<<"\n1. türevler:\n";
  for(i=0;i<n-1;i++)
       { trv1[i]=(y[i+1]-y[i])/h; cout<<trv1[i]<<endl; }
  cout<<"\n1. türevler:\n";
  for(i=0;i<n-2;i++)
       { trv2[i]=(y[i+2]-2*y[i+1]+y[i])/(h*h); cout<<trv2[i]<<endl; }
  system("pause"); return 0; }
```

C#

```
// Sayısal türev
using System;
namespace ornek154
{class Program
  {static void Main(string[] args)
    {int i,n; double h;
     Console.Write("Nokta sayısı: ");
     n=Convert.ToInt32(Console.ReadLine());
     double[] x=new double[n]; double[] y=new double[n];
     double[] trv1=new double[n-1]; double[] trv2=new double[n-2];
     Console.Write("Adım değeri: ");
     h= Convert.ToDouble(Console.ReadLine()); Console.Write("x(0): ");
     x[0]=Convert.ToDouble(Console.ReadLine());
     for (i = 0; i < n; i++)
      { if (i != 0) x[i]=x[i-1]+h; Console.Write("y({0}): ", i);
        y[i]=Convert.ToDouble(Console.ReadLine()); }
      Console.WriteLine("\n1. türevler");
      for (i = 0; i < n-1; i++)
      { trv1[i]=(y[i+1]-y[i])/h; Console.WriteLine("{0}",trv1[i]); }
      Console.WriteLine("\n2. türevler");
      for (i = 0; i < n - 2; i++)
      { trv2[i]=(y[i+2]-2*y[i+1]+y[i])/(h*h);
        Console.WriteLine("{0}", trv2[i]);} Console.ReadLine(); }}}
```

JAVA

```
// Sayısal türev
import java.util.Scanner;
public class ornek154 {
  public static void main(String[] args) {
    int i,n; float h; Scanner klavye=new Scanner(System.in);
    System.out.print("Nokta sayısı: ");n=klavye.nextInt();
    float[] x = new float[n]; float[] y = new float[n];
    float[] trv1 = new float[n-1]; float[] trv2 = new float[n-2];
    System.out.print("Adım değeri: ");h=klavye.nextFloat();
    System.out.print("x(0): ");x[0]=klavye.nextFloat();
    for(i=0;i<n;i++)
    { if (i!=0) x[i]=x[i-1]+h;
      System.out.printf("y(%d): ",i); y[i]=klavye.nextFloat(); }
    System.out.printf("\n1. türevler:\n");
    for (i=0; i<n-1;i++)
      { trv1[i]=(y[i+1]-y[i])/h; System.out.println(trv1[i]); }
    System.out.printf("\n2. türevler:\n");
    for (i=0; i<n-2;i++)
      {trv2[i]=(y[i+2]-2*y[i+1]+y[i])/(h*h);
       System.out.println(trv2[i]); } } }
```

MATLAB

```
% Sayısal türev
clear all; clc; n=input('Nokta sayısı: ');
h=input('Adım değeri: '); x(1)=input('x(1): ');
for i=1:n
    if (i~=1) x(i)=x(i-1)+h; end;
    fprintf('y(%d):',i); y(i)=input('');
end; fprintf('\n1. türevler:\n');
for i=1:n-1
    trv1(i)=(y(i+1)-y(i))/h; fprintf('%f\n',trv1(i));
end; fprintf('\n2. türevler:\n');
for i=1:n-2
    trv2(i)=(y(i+2)-2*y(i+1)+y(i))/(h*h); fprintf('%f\n',trv2(i));
end;
--------------------------- veya -------------------------------
% Sayısal türev
clear all; clc;
x=input('x değerlerini [a b c d] şeklinde giriniz !\n');
y=input('y değerlerini [a b c d] şeklinde giriniz !\n');
h=x(2)-x(1); fprintf('\n1. türevler:\n');
trv1=diff(y)./h; fprintf('%f\n',trv1');
fprintf('\n2. türevler:\n');
trv2=diff(y,2)./h^2; fprintf('%f\n',trv2');
```

```
Nokta sayısı: 5
Adım değeri: 1
x(1): 0
y(1):0
y(2):1
y(3):4
y(4):9
y(5):16

1. türevler:
1.000000
3.000000
5.000000
7.000000

2. türevler:
2.000000
2.000000
2.000000
```

PYTHON

```
# Sayısal türev
x=[]; y=[]; trv1=[]; trv2=[]
n=int(input("Nokta sayısı: ")); h=eval(input("Adım değeri: "))
print("x(1): ",end=''); x.append(eval(input()))
for i in range(n):
    if (i!=0):
        x.append(x[i-1]+h)
    print("y(%d): "%(i+1),end='')
    y.append(eval(input()))
print("\n1. türevler:")
for i in range(n-1):
    trv1.append((y[i+1]-y[i])/h); print(trv1[i])
print("\n2. türevler:")
for i in range(n-2):
    trv2.append((y[i+2]-2*y[i+1]+y[i])/h**2); print(trv2[i])
```

```
---------- veya ----------
# Sayısal türev
import numpy as np
trv1=[]; trv2=[]
print("Aralarına boşluk bırakarak x değerlerini giriniz!")
x=[eval(i) for i in input().split()]
print("Aralarına boşluk bırakarak y değerlerini giriniz!")
y=[eval(i) for i in input().split()]
h=x[1]-x[0]
print("\n1. türevler:")
trv1.append(np.diff(y)/h)
print(*trv1,sep=' ')
print("\n2. türevler:")
trv2.append(np.diff(y,2)/h**2)
print(" ".join(map(str,trv2)))
```

III. SAYISAL İNTEGRAL

Belirli integrallerin hesaplanması için çeşitli sayısal yöntemler mevcuttur. Geometriksel olarak belirli integral, fonksiyon grafiği ile eksen arasında kalan alandır. Bu alanı hesaplamak için, alanı kolaylıkla hesaplanabilen geometrik şekillerden faydalanılabilmektedir.

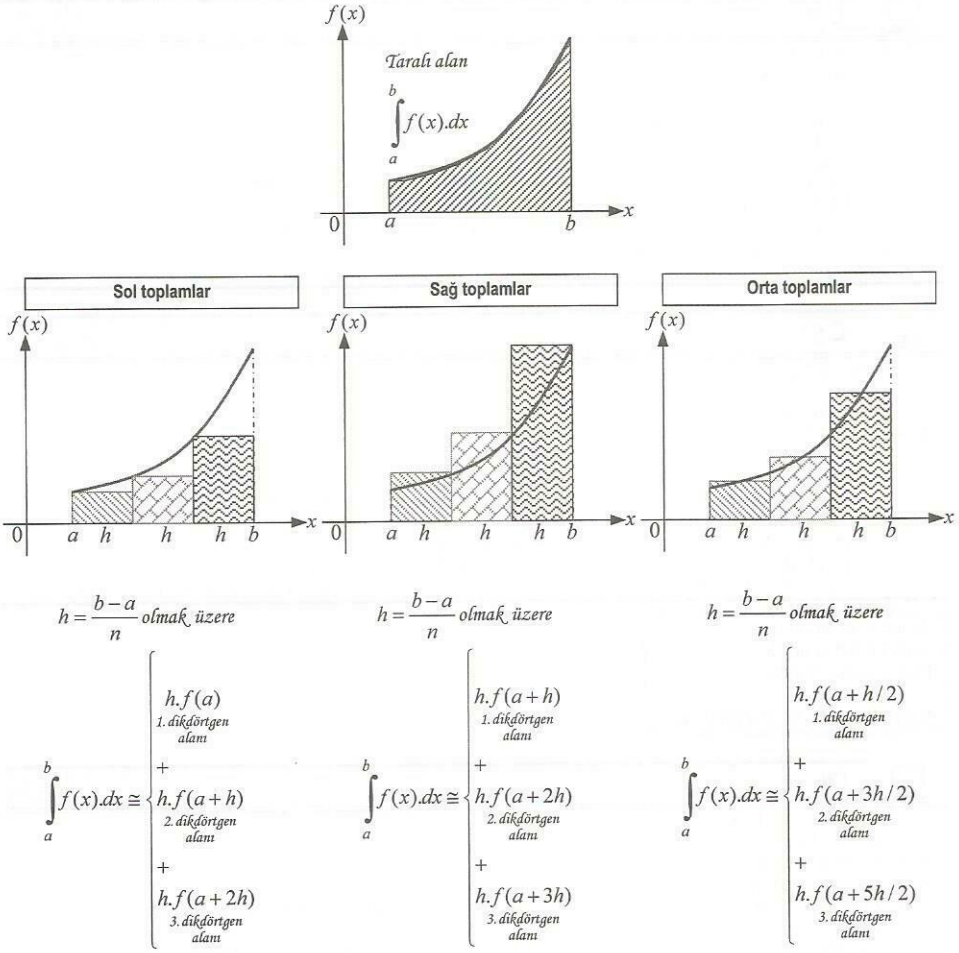
A. Dikdörtgenler yöntemi

Dikdörtgenler yönteminde; fonksiyon ile eksen arasında kalan bölge, küçük dikdörtgenlere bölünmekte ve oluşan dikdörtgenlerin alanları toplanmaktadır. Oluşturulacak dikdörtgenlerin sayısı arttıkça (daha çok sayıda parçaya bölündükçe), eğri ile dikdörtgenler arasındaki boşluklar (hata payları) azalacağından, sonuç daha yüksek doğrulukla hesaplanmış olmaktadır.

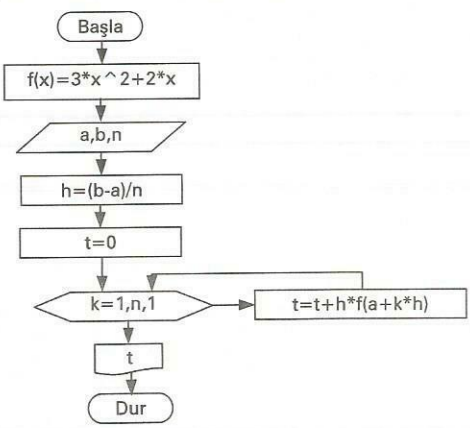
$\int_a^b f(x).dx$ belirli integrali, dikdörtgenler yöntemiyle hesaplanırken $[a,b]$ aralığı n tane parçaya bölünür ve bir kenarı $h=\frac{b-a}{n}$ olan dikdörtgenlere tamamlanmaktadır. Dikdörtgenin diğer kenarı olarak da sol tarafındaki nokta, sağ tarafındaki nokta veya ortasındaki nokta alınabilmektedir. Dikdörtgenin alanı, iki kenarının çarpımından ibaret olduğu için oluşan dikdörtgenlerin alanları toplanarak belirli integral yaklaşık olarak hesaplanmış olmaktadır(Şekil 15.2).

Örneğin sağ kenarların alınması durumunda toplam alan, yani belirli integral için $\int_a^b f(x).dx \cong h\sum_{k=1}^{n} f(a+kh)$ eşitliği elde edilmektedir.

Örnek-15.5: Klavyeden girilen $[a,b]$ aralığı ve parça (oluşturulacak dikdörtgen) sayısına göre $f(x)=3x^2+2x$ fonksiyonunun belirli integralini, "sağ toplamlar" ile hesaplayan program.



Şekil 15.2 Dikdörtgenler yöntemiyle belirli integral



C

```
/* Dikdörtgenler yöntemi */
#include <stdio.h>
#include <conio.h>
#define f(x) 3*x*x+2*x
void main()
{ float a,b,h,t=0.0,g; int k,n;
  clrscr();printf("Aralığın alt sınırı: ");scanf("%f",&a);
  printf("Aralığın üst sınırı: ");scanf("%f",&b);
  printf("Parça sayısı: ");scanf("%d",&n); h=(b-a)/n;
  for(k=1;k<=n;k++) { g=a+k*h;t+=h*(f(g)); }
  printf("\nBelirli integral: %0.10f\n",t); getch(); }
```

C++

```
// Dikdörtgenler yöntemi
#include <iostream>
using namespace std;
#define f(x) 3*x*x+2*x
int main()
{   float a,b,h,t=0.0,g; int n;
    system("cls"); cout<<"Aralığın alt sınırı: "; cin>>a;
    cout << "Aralığın üst sınırı: "; cin >> b;
    cout << "Parça sayısı: "; cin >> n; h=(b-a)/n;
    for (int k=1;k<=n;k++) { g=a+k*h; t+=h*(f(g)); }
    cout <<"\n"<<"Belirli integral: "<<t; system("pause");return 0; }
```

```
Aralığın alt sınırı: 1
Aralığın üst sınırı: 3
Parça sayısı: 10000

Belirli integral: 34.0027809143
```

C#

```
// Dikdörtgenler yöntemi
using System;
namespace ornek155
{ class Program
  { public static double f(double x) { return 3*x*x+2*x; }
    static void Main(string[] args)
     { double a,b,h,t=0; Console.Write("Aralığın alt sınırı: ");
       a = Convert.ToDouble(Console.ReadLine());
       Console.Write("Aralığın üst sınırı: ");
       b = Convert.ToDouble(Console.ReadLine());
       Console.Write("Parça sayısı: ");
       int n = Convert.ToInt32(Console.ReadLine()); h = (b - a) / n;
       for (int k = 1; k <= n; k++) t += h * f(a + k * h);
       Console.WriteLine("\nBelirli integral: {0:F10}", t);
       Console.ReadLine();   }   }   }
```

JAVA

```
// Dikdörtgenler yöntemi
import java.util.Scanner;
public class ornek155 {
public static double f(double x){ return 3*x*x+2*x; }
public static void main(String[] args)
{ double a,b,h,t=0; int n; Scanner klavye=new Scanner(System.in);
  System.out.print("Aralığın alt sınırı: ");a=klavye.nextDouble();
  System.out.print("Aralığın üst sınırı: ");b=klavye.nextDouble();
  System.out.print("Parça sayısı: "); n=klavye.nextInt();  h=(b-a)/n;
  for(int k=1;k<=n;k++) t+=h*f(a+k*h);
  System.out.printf("\nBelirli integral: %.10f",t); }}
```

MATLAB

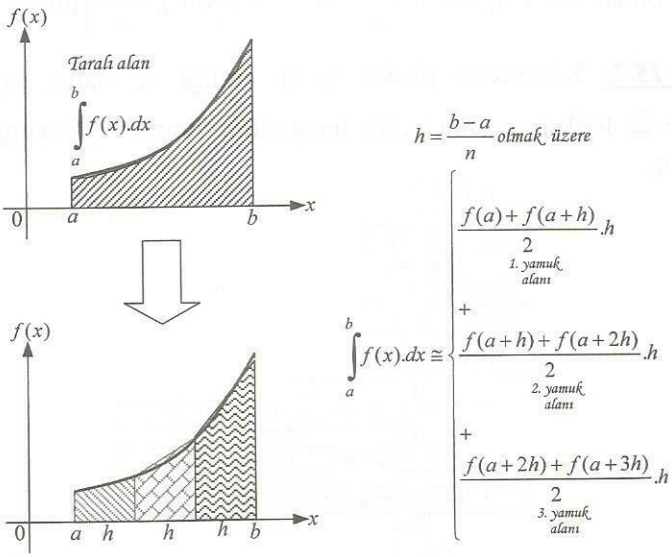
```
% Dikdörtgenler yöntemi
clear all; clc; t=0; f=inline('3*x^2+2*x');
a=input('Aralığın alt sınırı: '); b=input('Aralığın üst sınırı: ');
n=input('Parça sayısı: '); h=(b-a)/n;
for k=1:n
    t=t+h*f(a+k*h);
end ; fprintf('\nBelirli integral: %0.10f\n',t);
```

PYTHON

```
# Dikdörtgenler yöntemi
def y(x):
    return 3*x*x+2*x
t=0
a=eval(input("Aralığın alt sınırı: "))
b=eval(input("Aralığın üst sınırı: "))
n=int(input("Parça sayısı: "))
h=(b-a)/n
for k in range(1,n+1):
    t+=h*y(a+k*h)
print("\nBelirli integral: %0.10f"%t)
```

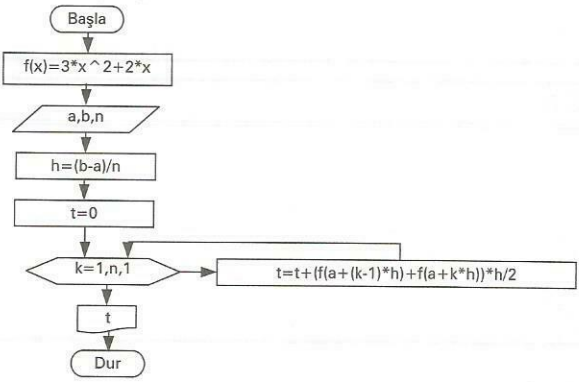
B. Yamuklar yöntemi

Dikdörtgenler yönteminde; fonksiyon ile eksen arasında kalan alan, dikdörtgenlere bölünürken yamuklar yönteminde ise yamuklara bölünmekte ve sonuç bu yamuk alanlarının toplamı olmaktadır. $f(x)$ fonksiyonunun $[a,b]$ aralığındaki belirli integrali hesaplanırken; aralık n eşit parçaya bölünmekte, yüksekliği $h=\dfrac{b-a}{n}$ olan yamuklara tamamlanmakta ve yamuk alanları $\int_a^b f(x).dx \cong h\sum_{k=1}^{n}\dfrac{f\{a+(k-1)h\}+f\{a+kh\}}{2}$ toplanmaktadır (Şekil 15.3).



Şekil 15.3 Yamuklar yöntemiyle belirli integral

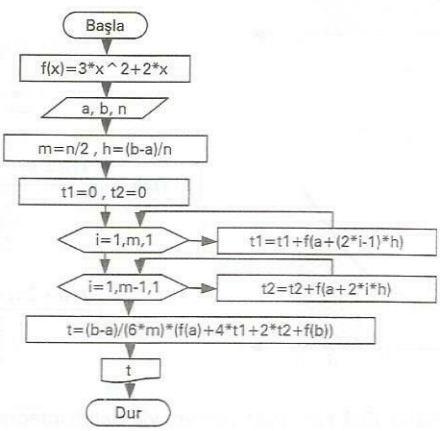
Örnek-15.6: Klavyeden girilen $[a,b]$ aralığı ve parça (oluşturulacak yamuk) sayısına göre $f(x)=3x^2+2x$ fonksiyonunun belirli integralini, yamuklar yöntemiyle hesaplayan program.



C. Simpson yöntemi

Sayısal analizde belirli integral hesaplamak için kullanılan yöntemlerin başında "*Simpson yöntemi*" ("*paraboller yöntemi*") gelmektedir. $[a,b]$ aralığında sürekli olan $y=f(x)$ fonksiyonu için aralık n eşit parçaya bölünür ve $m=\frac{n}{2}$ olmak üzere $\int_a^b f(x).dx \cong \frac{b-a}{6m}.\left\{y_0+4.\sum_{i=1}^{m} y_{2i-1}+2.\sum_{i=1}^{m-1} y_{2i}+y_{2m}\right\}$ dir. Burada $i,j=0,...,n$ olmak üzere $x_i=a+i.\frac{b-a}{n}$ ve $y_j=f\left(a+j.\frac{b-a}{n}\right)$ dir.

Örnek-15.7: Klavyeden girilen $[a,b]$ aralığı ve parça sayısına göre $f(x)=3x^2+2x$ fonksiyonunun belirli integralini, Simpson yöntemiyle hesaplayan program.



IV. KÖK HESAPLARI

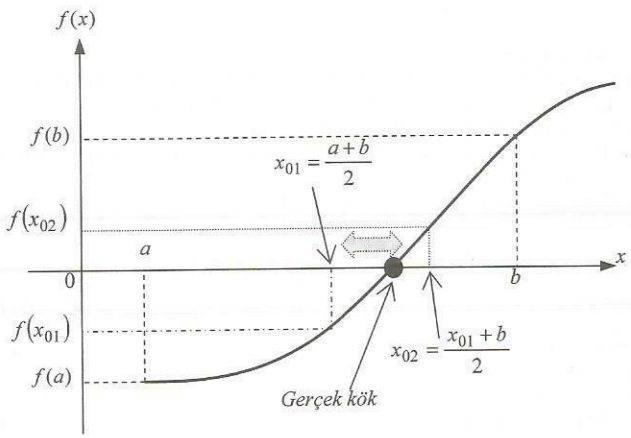
$f(x_k)=0$ denklemini sağlayan x_k değerine/değerlerine, denklemin kökü/kökleri denir.

A. Yarılama yöntemi

Sayısal analizde herhangi bir denklemin kökünü hesaplamak için kullanılan yöntemlerden birisi *"yarılama yöntemi"*dir. $f(x)=0$ denkleminin $[a,b]$ aralığında yarılama yöntemiyle kökü hesaplanırken, ilk iterasyonda aralığın orta noktası $x_0=\frac{a+b}{2}$ bulunur ve denklemde yerine yazılarak istenilen hassasiyette (hata miktarında) köke yakınsanıp yakınsanmadığı kontrol edilir (Şekil 15.4). Bu kontrol, iterasyon değerinin mutlak değerinin veya ardışık iki iterasyon arasındaki farkın hassasiyetten küçük olup olmadığı gibi sorgularla yapılabilir. Eğer kök istenilen hassasiyette hesaplanmadıysa, yeni aralık tespit edilir. Bu aralığın tespitinde de kökün temel var olma testi kullanılır:

$$\begin{cases} f(a).f(x_0)<0 \Rightarrow [a,x_0] \\ f(a).f(x_0)>0 \Rightarrow [x_0,b] \end{cases}$$

Yeni aralık belirlendikten sonra, yine bu aralığın orta noktası hesaplanarak iterasyonlara devam edilir. Böylece kökün bulunduğu aralık daraltılarak denklemin kökü hesaplanmaktadır.

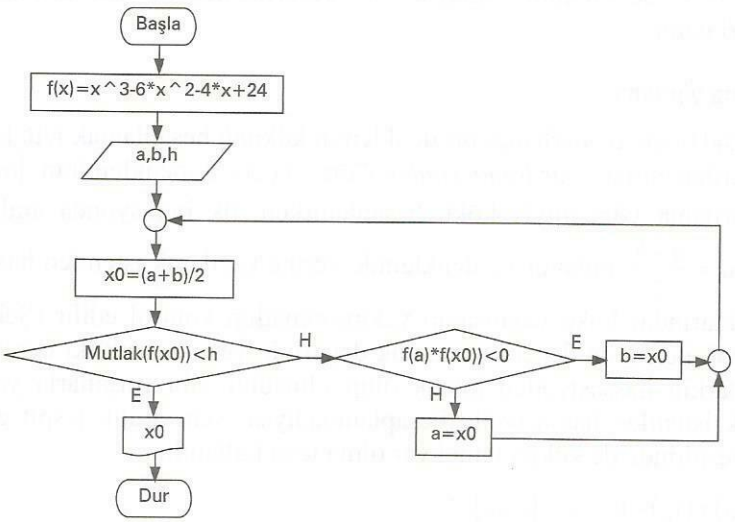


Şekil 15.4 Yarılama yöntemiyle kök hesabı

Örnek-15.8: $x^3-6x^2-4x+24=0$ denkleminin kökünü, klavyeden girilen aralıkta ve belirtilen hassasiyette "yarılama yöntemi"yle hesaplayan program.

Kök hesaplamalarıyla ilgili akış diyagramı çizilirken ve kodlamaları yapılırken; girilen aralıkta kökün var olduğu varsayılmıştır. Genel durumlar için akış

diyagramlarında/kodlarda; aralık girildikten sonra kökün olup olmadığı denetlenmelidir.



C

```
/* Yarılama yöntemi */
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define f(x) x*x*x-6*x*x-4*x+24
void main()
{ float a,b,h,x0,g1,g2;
  clrscr();printf("Aralığın alt sınırı: ");scanf("%f",&a);
  printf("Aralığın üst sınırı: ");scanf("%f",&b);
  printf("Hassasiyet değeri: ");scanf("%f",&h);
  do { x0=(a+b)/2;g1=f(a);g2=f(x0); if (g1*g2<0) b=x0; else a=x0;
     } while (fabs(f(x0))>h);
  printf("\nKök: %0.10f\n",x0); getch(); }
```

C++

```
// Yarılama yöntemi
#include <iostream>
#include <math.h>
using namespace std;
#define f(x) x*x*x-6*x*x-4*x+24
int main()
{   float a,b,h,x0,g1,g2;
    system("cls"); cout<<"Aralığın alt sınırı: "; cin>>a;
    cout << "Aralığın üst sınırı: "; cin >> b;
    cout << "Hassasiyet değeri: "; cin >> h;
    do { x0=(a+b)/2;g1=f(a);g2=f(x0); if (g1*g2<0) b=x0; else a=x0;
       } while (fabs(f(x0))>h);
    cout << "\n "<< "Kök: " << x0;  system("pause"); return 0; }
```

C#

```
// Yarılama yöntemi
using System;
namespace ornek158
{ class Program
   { public static double f(double x) { return x*x*x-6*x*x-4*x+24; }
     static void Main(string[] args) {
     double a, b, h, x0;
     Console.Write("Aralığın alt sınırı: ");
     a = Convert.ToDouble(Console.ReadLine());
     Console.Write("Aralığın üst sınırı: ");
     b = Convert.ToDouble(Console.ReadLine());
     Console.Write("Hassasiyet değeri: ");
     h = Convert.ToDouble(Console.ReadLine());
     do { x0=(a+b)/2; if (f(a)*f(x0)<0) b=x0; else a=x0;
        } while (Math.Abs(f(x0))>h);
     Console.WriteLine("\nKök: {0:F10}",x0); Console.ReadLine(); }}}
```

```
Aralığın alt sınırı: 0
Aralığın üst sınırı: 5
Hassasiyet değeri: 0.000001

Kök: 2.0000000000
```

JAVA

```
// Yarılama yöntemi
import java.util.Scanner;
public class ornek158 {
public static double f(double x){ return x*x*x-6*x*x-4*x+24; }
public static void main(String[] args)
{ double a,b,h,x0;
  Scanner klavye=new Scanner(System.in);
  System.out.print("Aralığın alt sınırı: ");a=klavye.nextDouble();
  System.out.print("Aralığın üst sınırı: ");b=klavye.nextDouble();
  System.out.print("Hassasiyet değeri: ");h=klavye.nextDouble();
  do { x0=(a+b)/2; if (f(a)*f(x0)<0) b=x0; else a=x0;
       } while (Math.abs(f(x0))>h);
  System.out.printf("\nKök: %.10f",x0); }}
```

MATLAB

```
% Yarılama yöntemi
clear all; clc; f=inline('x^3-6*x^2-4*x+24');
a=input('Aralığın alt sınırı: '); b=input('Aralığın üst sınırı: ');
h=input('Hassasiyet değeri: '); x0=(a+b)/2;
while (abs(f(x0))>h)
    if (f(a)*f(x0)<0) b=x0;
    else a=x0; end
    x0=(a+b)/2;
end; fprintf('\nKök: %0.10f\n',x0);
```

PYTHON

```
# Yarılama yöntemi
def f(x):
    return x**3-6*x*x-4*x+24
a=eval(input("Aralığın alt sınırı: "))
b=eval(input("Aralığın üst sınırı: "))
h=eval(input("Hassasiyet değeri: "))
x0=(a+b)/2
while (abs(f(x0))>h):
    if (f(a)*f(x0)<0):
        b=x0
    else:
        a=x0
    x0=(a+b)/2
print("\nKök: %0.10f"%x0)
```

B. Kirişler (Regula-False) yöntemi

"*Regula-False*" olarak da adlandırılan bu yöntemdeki iterasyon eşitliği; $[a, f(a)]$ ve $[b, f(b)]$ noktalarından geçen doğrunun denkleminden elde edilmektedir.

$$y = \frac{f(a)-f(b)}{a-b}.(x-a)+f(a)$$

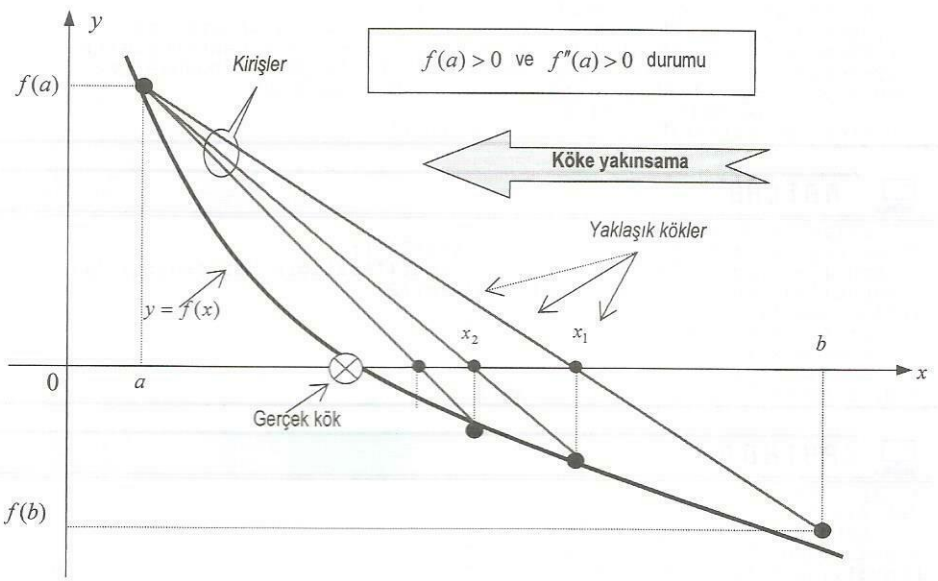
$y = 0$ şartı x-ekseninin kesildiği noktayı vereceğinden

$$0 = \frac{f(a)-f(b)}{a-b}.(x-a)+f(a) \Rightarrow x = \frac{a.f(b)-b.f(a)}{f(b)-f(a)}$$

eşitliği bulunur. Kesim noktası bulunduktan sonra köke istenilen hassasiyette yaklaşılmadıysa

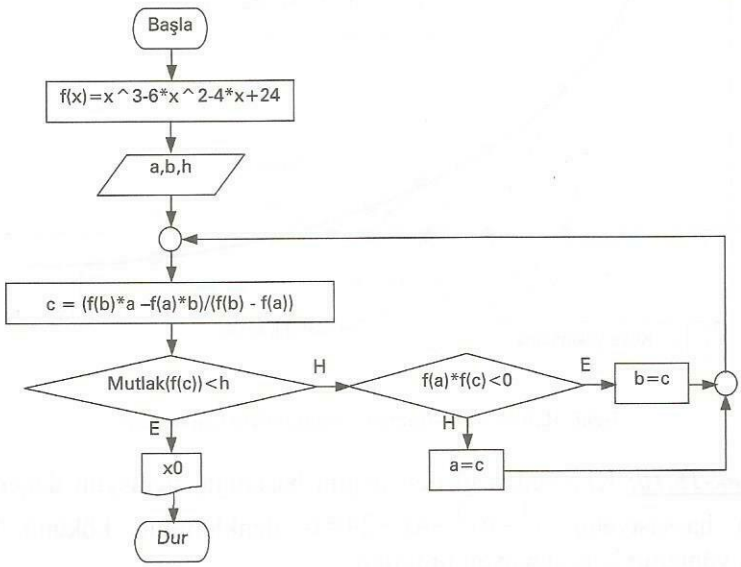
$$\begin{cases} f(a).f(x) < 0 \Rightarrow b = x \\ f(a).f(x) > 0 \Rightarrow a = x \end{cases}$$

yeni aralık tespitiyle devam edilir (Şekil 15.5).



Şekil 15.5 Kirişler yöntemiyle kök hesabı

Örnek-15.9: $x^3-6x^2-4x+24=0$ denkleminin kökünü, klavyeden girilen aralıkta ve belirtilen hassasiyette "kirişler yöntemi"yle hesaplayan program.



C. Teğetler (Newton-Raphson) yöntemi

Sayısal analizde herhangi bir fonksiyonun kökünü hesaplamak için kullanılan yöntemlerin başında Newton yöntemleri gelmektedir. Regula-False yönteminde köke, kirişlerle yaklaşılırken; birçok türü mevcut olan Newton yöntemlerinde köke, teğetler ile yaklaşılmaktadır. Yani $y=f(x)$ eğrisinin x-eksenini kestiği noktayı (gerçek kökü) bulmak için eğri üzerindeki noktalardan çizilen teğetlerin x-eksenini kestiği noktalar hesaplandığından; bu yöntemlere, *"teğetler yöntemi"* denir. $[x_0, f(x_0)]$ noktasına teğet olan doğrunun denklemi

$$y-f(x_0)=f'(x_0).(x-x_0)$$

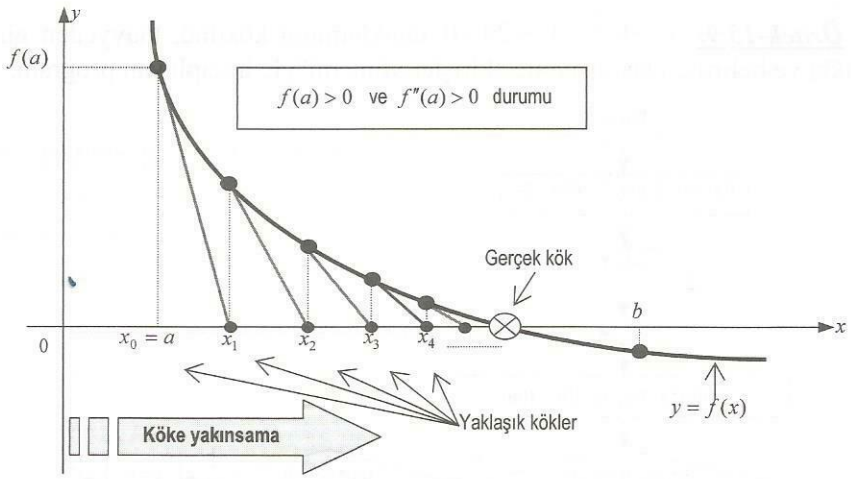
dir. x_1, bu teğetin x-eksenini kestiği nokta olduğundan $y=0$ için

$$0-f(x_0)=f'(x_0).(x_1-x_0)\Rightarrow x_1=x_0-\frac{f(x_0)}{f'(x_0)}$$

elde edilir. Özetle Newton-Raphson yönteminde $f(x)=0$ denkleminin $[a,b]$ aralığındaki kökü

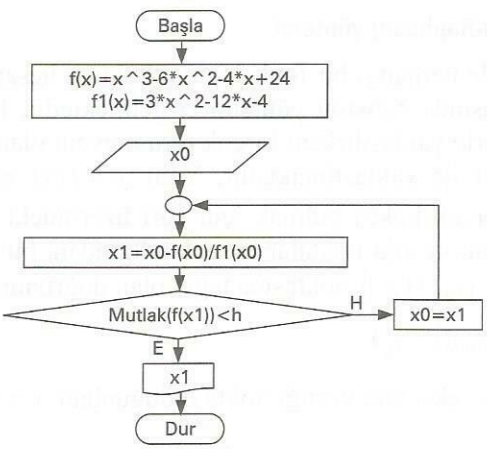
$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}$$

iterasyon eşitliğiyle hesaplanmaktadır. İstenen aralıkta, iterasyonun başlangıç noktasını seçmek için matematiksel yöntemler vardır (Şekil 15.6).



Şekil 15.6 Newton-Raphson yöntemiyle kök hesabı

Örnek-15.10: Klavyeden girilen uygun başlangıç iterasyon değerine göre belirtilen hassasiyette $x^3-6x^2-4x+24=0$ denkleminin kökünü Newton-Raphson yöntemiyle hesaplayan program.




```
/* Newton-Raphson yöntemi */
#include <stdio.h>
#include <conio.h>
#include <math.h>
#define f(x) x*x*x-6*x*x-4*x+24
#define f1(x) 3*x*x-12*x-4
void main()
{ float x0,x1,h,g0,g1;
  clrscr();printf("İterasyon başlangıç değeri: ");scanf("%f",&x0);
  printf("Hassasiyet değeri: ");scanf("%f",&h);
  do { g0=f(x0);g1=f1(x0); x1=x0-g0/g1; x0=x1;
     } while (fabs(f(x1))>h); printf("\nKök: %0.10f\n",x1); getch(); }
```

C++

```
// Newton-Raphson yöntemi
#include <iostream>
#include <math.h>
using namespace std;
#define f(x) x*x*x-6*x*x-4*x+24
#define f1(x) 3*x*x-12*x-4
int main()
{   float x0,x1,h,g0,g1;
    system("cls"); cout << "İterasyon başlangıç değeri: "; cin >> x0;
    cout << "Hassasiyet değeri: "; cin >> h;
    do { g0=f(x0);g1=f1(x0); x1=x0-g0/g1; x0=x1;
       } while (fabs(f(x1))>h);
    cout << "\n " << "Kök: " << x1;  system("pause");return 0; }
```

C#

```
// Newton-Raphson yöntemi
using System;
namespace ornek1510
{ class Program
  { public static double f(double x) { return x*x*x-6*x*x-4*x+24; }
    public static double f1(double x) { return 3*x*x-12*x-4; }
    static void Main(string[] args)
     { double x0, x1, h;
       Console.Write("İterasyon başlangıç değeri: ");
       x0 = Convert.ToDouble(Console.ReadLine());
       Console.Write("Hassasiyet değeri: ");
       h = Convert.ToDouble(Console.ReadLine());
       do { x1 = x0 - f(x0) / f1(x0); x0 = x1;
         } while (Math.Abs(f(x1)) > h);
       Console.WriteLine("\nKök: {0:F10}", x1); Console.ReadLine();}}}
```

JAVA

```
// Newton-Raphson yöntemi
import java.util.Scanner;
public class ornek1510 {
public static double f(double x){ return x*x*x-6*x*x-4*x+24; }
public static double f1(double x){ return 3*x*x-12*x-4; }
public static void main(String[] args)
{ double x0,x1,h;
  Scanner klavye=new Scanner(System.in);
  System.out.print("İterasyon başlangıç değeri: ");
  x0=klavye.nextDouble();
  System.out.print("Hassasiyet değeri: ");
  h=klavye.nextDouble();
  do
   {    x1=x0-f(x0)/f1(x0); x0=x1;
   } while (Math.abs(f(x1))>h);
  System.out.printf("\nKök: %.10f",x1);}}
```

```
İterasyon başlangıç değeri: -10
Hassasiyet değeri: 0.0000001

Kök: -2.0000000001
```

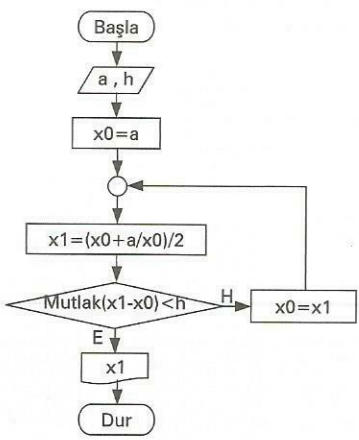
MATLAB

```
% Newton-Raphson yöntemi
clear all;clc;
f=inline('x^3-6*x^2-4*x+24');
f1=inline('3*x^2-12*x-4');
x0=input('İterasyon başlangıç değeri: ');
h=input('Hassasiyet değeri: ');
x1=x0-f(x0)/f1(x0);
while (abs(f(x1))>h)
    x0=x1;x1=x0-f(x0)/f1(x0);
end;
fprintf('\nKök: %0.10f\n',x1);
```

PYTHON

```
# Newton-Raphson yöntemi
def f(x):
    return x**3-6*x*x-4*x+24
def f1(x):
    return 3*x*x-12*x-4
x0=eval(input("İterasyon başlangıç değeri: "))
h=eval(input("Hassasiyet değeri: "))
x1=x0-f(x0)/f1(x0)
while (abs(f(x1))>h):
    x0=x1
    x1=x0-f(x0)/f1(x0)
print("\nKök: %0.10f"%x1)
------------ veya --------------------------
# Newton-Raphson yöntemi
import scipy.optimize as spop
def f(x):
    return x**3-6*x*x-4*x+24
x0=eval(input("İterasyon başlangıç değeri: "))
print("\nKök: %0.10f"%spop.newton(f,x0))
```

Örnek-15.11: Klavyeden girilen pozitif tamsayının karekökünü, belirtilen hassasiyetle hesaplayan program.



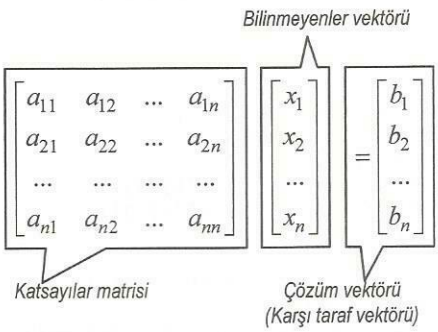
Newton-Raphson yöntemi $x^2-a=0$ denkleminin çözümü için düzenlenirse $x_{k+1}=\frac{1}{2}\left\{x_k+\frac{a}{x_k}\right\}$ iterasyon eşitliği elde edilir.

V. DOĞRUSAL DENKLEM SİSTEMLERİNİN ÇÖZÜMLERİ

n bilinmeyenli n denklemden oluşan sistem

$$\left.\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots\ldots + a_{1n}x_n = b_1 \\ &a_{21}x_1 + a_{22}x_2 + \ldots\ldots + a_{2n}x_n = b_2 \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_{n1}x_1 + a_{n2}x_2 + \ldots\ldots + a_{nn}x_n = b_n \end{aligned}\right\}$$

$$[A][x] = [b]$$

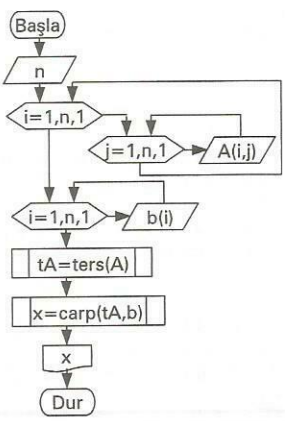


şeklinde ifade edilebilir. Denklem sisteminin çözümünün mevcut ve tek olması için A katsayılar matrisinin determinantının sıfırdan ($|A| \neq 0$) farklı olması gerekir.

A. Ters Matris Yöntemi

Doğrusal denklem sistemlerini çözümü için en basit yollardan birisi, ters matris yöntemidir. Katsayılar matrisinin tersinin kolaylıkla hesaplanabildiği durumlarda $A.x = b \Rightarrow x = A^{-1}.b$ ile çözüme ulaşılabilmektedir.

Örnek-15.12: Klavyeden girilen ve çözümü olan denklem sistemini, ters matris yöntemiyle çözen program.



C

```
/* Ters matris yöntemi */
#include <stdio.h>
#include <conio.h>
float A[10][10],tA[10][10],x[10][1],b[10][1]; int i,j,k,n;
void main()
{ clrscr();printf("Denklem sayısı: ");scanf("%u",&n);
  printf("\nKatsayılar matrisi\n------------------\n");
  for(i=0;i<n;i++)
    { for(j=0;j<n;j++)
        { printf("A(%u,%u)= ",i+1,j+1);scanf("%f",&A[i][j]); }}
  printf("\nÇözüm vektörü\n-------------\n");
  for(i=0;i<n;i++) { printf("b(%u)= ",i+1);scanf("%f",&b[i][0]); }
  for(k=0;k<n;k++)
    { for(i=0;i<n;i++)
        { for(j=0;j<n;j++)
            if(!((i==k)||(j==k)))
              A[i][j]=A[i][j]-A[i][k]*A[k][j]/A[k][k];}
              A[k][k]=-1/A[k][k]; for(i=0;i<n;i++)
      if(!(i==k)) A[i][k]=A[i][k]*A[k][k];
      for(j=0;j<n;j++) if(!(j==k)) A[k][j]=A[k][j]*A[k][k];}
      for(i=0;i<n;i++) { for(j=0;j<n;j++) tA[i][j]=-A[i][j];}
  for(i=0;i<n;i++)
   {for(j=0;j<1;j++)
     { x[i][0]=0;for(k=0;k<n;k++) x[i][0]+=tA[i][k]*b[k][j];}}
  printf("\nBilinmeyenler\n-------------\n");
  for(i=0;i<n;i++) printf("x(%u)= %0.5f\n",i+1,x[i][0]);  getch(); }
```

C++

```
// Ters matris yöntemi
#include <iostream>
using namespace std;
int i,j,k,n;
int main()
{ system("cls");cout<<"Denklem sayısı: "; cin>>n;
  float A[n][n],tA[n][n],x[n][1],b[n][1];
  cout<<"\nKatsayılar matrisi\n------------------\n";
  for(i=0;i<n;i++)
    { for(j=0;j<n;j++)
        { cout<<"A("<<i+1<<","<<j+1<<")=";cin>>A[i][j]; }}
  printf("\nÇözüm vektörü\n-------------\n");
  for(i=0;i<n;i++) { cout<<"b("<<i+1<<")= ";cin>>b[i][0]; }
  for(k=0;k<n;k++)
    { for(i=0;i<n;i++)
        { for(j=0;j<n;j++)
            if(!((i==k)||(j==k)))
              A[i][j]=A[i][j]-A[i][k]*A[k][j]/A[k][k];}
              A[k][k]=-1/A[k][k]; for(i=0;i<n;i++)
      if(!(i==k)) A[i][k]=A[i][k]*A[k][k];
      for(j=0;j<n;j++) if(!(j==k)) A[k][j]=A[k][j]*A[k][k];}
      for(i=0;i<n;i++) { for(j=0;j<n;j++) tA[i][j]=-A[i][j];}
  for(i=0;i<n;i++)
   {for(j=0;j<1;j++)
     { x[i][0]=0;for(k=0;k<n;k++) x[i][0]+=tA[i][k]*b[k][j];}}
  cout<<"\nBilinmeyenler\n-------------\n";
  for(i=0;i<n;i++) cout<<"x("<<i+1<<")= "<<x[i][0]<<"\n";
  system("pause");return 0; }
```

C#

```
// Ters matris yöntemi
using System;
namespace ornek1512
{class Program
 {static void Main(string[] args)
  { int i, j, k, n;
    Console.Write("Denklem sayısı: ");
    n = Convert.ToInt32(Console.ReadLine());
    double[,] A = new double[n,n]; double[,] tA = new double[n,n];
    double[,] x = new double[n,1]; double[,] b = new double[n,1];
    Console.WriteLine("\nKatsayılar matrisi\n------------------");
    for (i=0;i<n;i++)
     { for (j=0;j<n;j++)
        { Console.Write("A({0},{1})= ",i+1,j+1);
          A[i,j]=Convert.ToDouble(Console.ReadLine());}}
    Console.WriteLine("\nÇözüm vektörü\n-------------");
    for (i=0;i<n;i++)
     { Console.Write("b({0})= ",i+1);
       b[i,0]=Convert.ToDouble(Console.ReadLine());}
    for (k=0;k<n;k++)
     { for (i=0;i<n;i++)
        { for (j=0;j<n;j++)
           if (!((i==k) || (j==k)))
            A[i,j]=A[i,j]-A[i,k]*A[k,j]/A[k,k]; }
       A[k,k]=-1/A[k,k];
       for (i=0;i<n;i++) if (!(i==k)) A[i,k]=A[i,k]*A[k,k];
       for (j=0;j<n;j++) if (!(j==k)) A[k,j]=A[k,j]* A[k,k];}
     for (i=0;i<n;i++) { for (j=0;j<n;j++) tA[i,j]=-A[i,j]; }
     for (i=0;i<n;i++)
      { for (j=0;j<1;j++)
         { x[i,0]=0; for (k=0;k<n;k++) x[i,0]+=tA[i,k]*b[k,j];}}
     Console.WriteLine("\nBilinmeyenler\n------------");
     for (i=0;i<n;i++) Console.WriteLine("x({0})={1:F5}",i+1,x[i,0]);
     Console.ReadLine();      }    }    }
```

JAVA

```
// Ters matris yöntemi
import java.util.Scanner;
public class ornek1512 {
public static void main(String[] args)
{ int i,j,k,n; Scanner klavye=new Scanner (System.in);
  System.out.print("Denklem sayısı: ");n=klavye.nextInt();
  double A[][]=new double[n][n]; double tA[][]=new double[n][n];
  double x[][]=new double[n][1]; double b[][]=new double[n][1];
  System.out.println("\nKatsayılar matrisi\n------------------\n");
  for(i=0;i<n;i++){
   for(j=0;j<n;j++)
    { System.out.print("A("+(i+1)+","+(j+1)+")= ");
      A[i][j]=klavye.nextDouble(); }}
  System.out.println("\nÇözüm vektörü\n-------------\n");
  for(i=0;i<n;i++){  System.out.print("b("+(i+1)+")= ");
                     b[i][0]=klavye.nextDouble();}
  for(k=0;k<n;k++)
   { for(i=0;i<n;i++)
      { for(j=0;j<n;j++)
          if(!((i==k)||(j==k)))
             A[i][j]=A[i][j]-A[i][k]*A[k][j]/A[k][k];}
             A[k][k]=-1/A[k][k]; for(i=0;i<n;i++)
     if(!(i==k)) A[i][k]=A[i][k]*A[k][k];
     for(j=0;j<n;j++) if(!(j==k)) A[k][j]=A[k][j]*A[k][k];}
     for(i=0;i<n;i++) { for(j=0;j<n;j++) tA[i][j]=-A[i][j];}
  for(i=0;i<n;i++)
   {for(j=0;j<1;j++)
     { x[i][0]=0;for(k=0;k<n;k++) x[i][0]+=tA[i][k]*b[k][j];}}
  System.out.println("\nBilinmeyenler\n------------\n");
  for(i=0;i<n;i++){ System.out.printf("x(%d)= %.5f\n",i+1,x[i][0]);}}}
```

MATLAB

```
% Ters matris yöntemi
clear all; clc;
A=input('Katsayılar matrisini [a b;c d] şeklinde giriniz !\n');
b=input('Çözüm vektörünü [a b c d] şeklinde giriniz !\n');
x=inv(A)*b';
fprintf('\nDenklem sisteminin çözümü:\n'); disp(x)
```

```
Katsayılar matrisini [a b;c d] şeklinde giriniz !
[5 4;2 3]
Çözüm vektörünü [a b c d] şeklinde giriniz !
[14 7]

Denklem sisteminin çözümü:
   2.0000
   1.0000
```

PYTHON

```
# Ters matris yöntemi
import numpy as np
g1=input("Katsayılar matrisini [a b;c d] şeklinde giriniz!\n")
A=np.matrix(g1)
g2=input("Çözüm vektörünü [a b] şeklinde giriniz\n")
b=np.matrix(g2)
x=(np.linalg.inv(A))*(b.transpose())
print("\nDenklem sisteminin çözümü")
print(x)
```

B. Cramer yöntemi

n bilinmeyenli n denklemden oluşan

$$\left.\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots\ldots + a_{1n}x_n = b_1 \\ &a_{21}x_1 + a_{22}x_2 + \ldots\ldots + a_{2n}x_n = b_2 \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_{n1}x_1 + a_{n2}x_2 + \ldots\ldots + a_{nn}x_n = b_n \end{aligned}\right\}$$

sistemin Cramer yöntemiyle çözümünde $(n+1)$ tane determinantın hesaplanması gerekmektedir. Determinant hesaplarının kolay olduğu durumlarda rahatlıkla kullanılabilen Kramer yönteminde

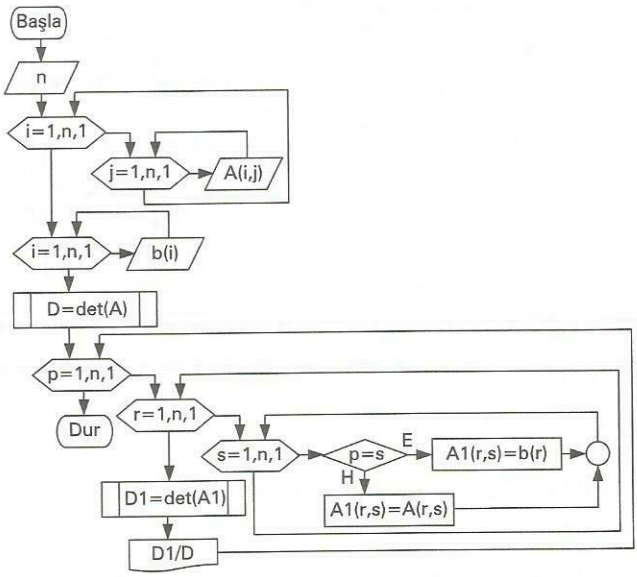
$$\Delta = \begin{vmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{vmatrix}, \qquad \Delta_1 = \begin{vmatrix} b_1 & a_{12} & \ldots & a_{1n} \\ b_2 & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ b_n & a_{n2} & \ldots & a_{nn} \end{vmatrix}$$

$$\Delta_2 = \begin{vmatrix} a_{11} & b_1 & \ldots & a_{1n} \\ a_{21} & b_2 & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & b_n & \ldots & a_{nn} \end{vmatrix} \qquad \ldots\ldots \qquad \Delta_n = \begin{vmatrix} a_{11} & a_{12} & \ldots & b_1 \\ a_{21} & a_{22} & \ldots & b_2 \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & b_n \end{vmatrix}$$

olmak üzere sistemin çözümü aşağıdaki gibi olmaktadır.

$$\begin{cases} \Delta \neq 0 & \Rightarrow \text{Çözüm var} : x_i = \dfrac{\Delta_i}{\Delta} \\ \Delta = 0 \quad ve \quad \forall \Delta_i = 0 & \Rightarrow \text{Sonsuz çözüm var} \\ \Delta = 0 \quad ve \quad \exists \Delta_i \neq 0 & \Rightarrow \text{Çözüm yok} \end{cases}$$

Örnek-15.13: Klavyeden girilen ve tek çözümü olan denklem sistemini, Cramer yöntemiyle çözen program.



C. Seidel Yöntemi

Seidel yönteminde

$$[A][x]=[b]$$

denklem sistemi $i=1,2,...,n$ için

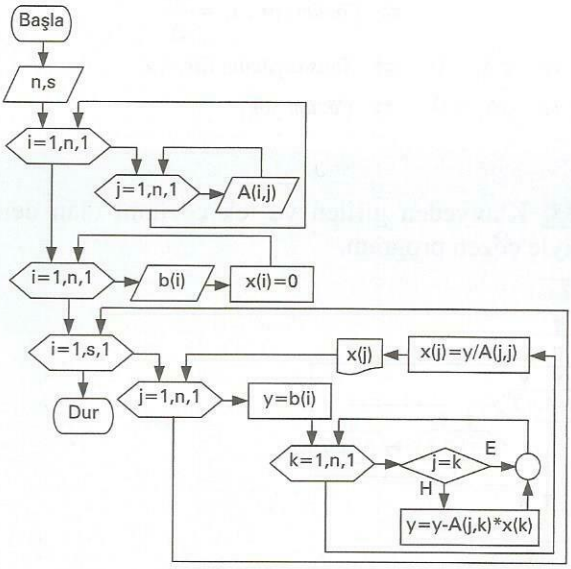
$$x_i = -\sum_{j=1, i\neq j}^{n} \frac{a_{ij}}{a_{ii}} x_j + \frac{b_i}{a_{ii}}$$

şekline dönüştürülmekte ve iterasyon işlemleri

$$x_i^{(k+1)} = -\sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(k+1)} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(k)} + \frac{b_i}{a_{ii}}$$

eşitliğine göre yapılmaktadır.

Örnek-15.14: Klavyeden girilen ve tek çözümü olan denklem sistemini, Seidel yöntemiyle çözen program.



C

```
/* Seidel yöntemi */
#include <stdio.h>
#include <conio.h>
float A[10][10],b[10],x[10],y; int i,j,k,n,s;
void main()
{ clrscr();printf("Denklem sayısı: ");scanf("%u",&n);
  printf("\nKatsayılar matrisi\n------------------\n");
  for(i=0;i<n;i++)
    { for(j=0;j<n;j++)
        { printf("A(%u,%u): ",i+1,j+1);scanf("%f",&A[i][j]); }}
  printf("\nÇözüm vektörü\n-------------\n");
  for(i=0;i<n;i++)
    { printf("b(%u): ",i+1);scanf("%f",&b[i]); x[i]=0; }
  printf("\nİterasyon sayısı: ");scanf("%u",&s);
  printf("\nİterasyonlar\n------------\n");
  for(i=1;i<=s;i++)
   { printf("\n%u. iterasyon\n",i);
     for (j=0;j<n;j++)
      { y=b[j]; for (k=0;k<n;k++) if (j!=k) y-=A[j][k]*x[k];
        x[j]=y/A[j][j];printf("x(%u)= %0.10f\n",j+1,x[j]); }} getch();}
```

C++

```
// Seidel yöntemi
#include <iostream>
using namespace std;
float A[10][10],b[10],x[10],y; int i,j,k,n,s;

int main()
{ system("cls");cout<<"Denklem sayısı: "; cin>>n;
  cout<<"\nKatsayılar matrisi\n------------------\n";
  for(i=0;i<n;i++)
    { for(j=0;j<n;j++)
        { cout<<"A("<<i+1<<","<<j+1<<"): ";cin>>A[i][j]; }}
```

```
  printf("\nÇözüm vektörü\n-------------\n");
  for(i=0;i<n;i++) { cout<<"b("<<i+1<<")= ";cin>>b[i]; x[i]=0;}
  cout<<"\nİterasyon sayısı: "; cin>>s;
  cout<<"\nİterasyonlar\n------------\n";
  for(i=1;i<=s;i++)
   { cout<<"\n"<<i<<". iterasyon\n";
     for (j=0;j<n;j++)
      { y=b[j]; for (k=0;k<n;k++) if (j!=k) y-=A[j][k]*x[k];
        x[j]=y/A[j][j];cout<<"x("<<j+1<<")= "<<x[j]<<"\n"; }}
  system("pause");return 0; }
```

C#

```
// Seidel yöntemi
using System;
namespace ornek1514
{class Program {
  static void Main(string[] args) {
   int i,j,k,n,s; double y;
   Console.Write("Denklem sayısı: ");
   n=Convert.ToInt32(Console.ReadLine());
   double[,] A = new double[n, n]; double[] b = new double[n];
   double[] x = new double[n];
   Console.WriteLine("\nKatsayılar matrisi\n-------------------");
   for (i=0;i<n;i++)
    { for (j=0;j<n;j++)
       { Console.Write("A({0},{1})= ", i+1, j+1);
         A[i,j]=Convert.ToDouble(Console.ReadLine()); }}
   Console.WriteLine("\nÇözüm vektörü\n-------------");
   for (i=0;i<n;i++)
    { Console.Write("b({0})= ", i + 1);
      b[i] = Convert.ToDouble(Console.ReadLine()); x[i] = 0; }
   Console.Write("\nİterasyon sayısı: ");
   s=Convert.ToInt32(Console.ReadLine());
   Console.WriteLine("\nİterasyonlar\n------------");
   for (i=1;i<=s;i++)
   { Console.WriteLine("\n{0}. iterasyon", i);
     for (j=0;j<n;j++)
      { y=b[j]; for (k=0;k<n;k++) if (j!=k) y-=A[j,k]*x[k];
        x[j]=y/A[j,j];Console.WriteLine("x({0})= {1:F10}",j+1,x[j]);}}
   Console.ReadLine();     }     }     }
```

JAVA

```
// Seidel yöntemi
import java.util.Scanner;
public class ornek1514 {
public static void main(String[] args)
{ int i,j,k,n,s; double y; Scanner klavye=new Scanner (System.in);
  System.out.print("Denklem sayısı: ");n=klavye.nextInt();
  double A[][]=new double[n][n]; double b[]=new double[n];
  double x[]=new double[n];
  System.out.println("\nKatsayılar matrisi\n-------------------\n");
  for(i=0;i<n;i++){
   for(j=0;j<n;j++)
    { System.out.print("A("+(i+1)+","+(j+1)+")= ");
      A[i][j]=klavye.nextDouble(); }}
  System.out.println("\nÇözüm vektörü\n-------------\n");
  for(i=0;i<n;i++){ System.out.print("b("+(i+1)+")= ");
                    b[i]=klavye.nextDouble();x[i]=0;}
  System.out.print("\nİterasyon sayısı: ");s=klavye.nextInt();
  System.out.println("\nİterasyonlar\n------------");
  for(i=1;i<=s;i++)
   { System.out.printf("\n%d. iterasyon\n",i);
     for (j=0;j<n;j++)
      {y=b[j]; for (k=0;k<n;k++) if (j!=k) y-=A[j][k]*x[k];
       x[j]=y/A[j][j];System.out.printf("x(%d)= %.10f\n",j+1,x[j]);}}}}
```

MATLAB

```
% Seidel yöntemi
clear all; clc;
A=input('Katsayılar matrisini [a b;c d] şeklinde giriniz !\n');
b=input('Çözüm vektörünü [a b c d] şeklinde giriniz !\n');
s=input('\nİterasyon sayısı: ');n=length(b); x(1:n)=0;
fprintf('\nİterasyonlar\n----------\n');
for i=1:s
    fprintf('\n%d. iterasyon\n',i);
    for j=1:n
        y=b(j); for k=1:n
                    if (j~=k) y=y-A(j,k)*x(k); end; end;
      x(j)=y/A(j,j);fprintf('x(%d)= %0.10f\n',j+1,x(j));end;end
```

PYTHON

```
# Seidel yöntemi
import numpy as np
g1=input("Katsayılar matrisini [a b;c d] şeklinde giriniz!\n")
A=np.matrix(g1)
print("\nÇözüm vektörünü [a b] şeklinde giriniz!")
b=[eval(i) for i in input().split()]
n=len(b); x=np.zeros(n)
s=int(input("\nİterasyon sayısı: "))
print("\nİterasyonlar\n----------");
for i in range(s):
    print("\n%d. iterasyon"%(i+1))
    for j in range(n):
        y=b[j]
        for k in range(n):
            if (j!=k):
                y-=A[j,k]*x[k]
        x[j]=y/A[j,j]
        print("x(%d)= %0.10f"%(j+1,x[j]))
```

```
Katsayılar matrisini [a b;c d] şeklinde giriniz!
5 -4;3 10

Çözüm vektörünü [a b] şeklinde giriniz!
7 29

İterasyon sayısı: 6

İterasyonlar
----------

1. iterasyon
x(1)= 1.4000000000
x(2)= 2.4800000000

2. iterasyon
x(1)= 3.3840000000
x(2)= 1.8848000000

3. iterasyon
x(1)= 2.9078400000
x(2)= 2.0276480000

4. iterasyon
x(1)= 3.0221184000
x(2)= 1.9933644800

5. iterasyon
x(1)= 2.9946915840
x(2)= 2.0015925248

6. iterasyon
x(1)= 3.0012740198
x(2)= 1.9996177940
```

VI. DİFERANSİYEL DENKLEM ÇÖZÜMLERİ

Bilinmeyen fonksiyonu ve türevlerini içeren eşitlikler "*diferansiyel denklem*" olarak adlandırılmaktadır. Birçok olay ve sistemler; değişim içerdiğinden, bunların matematiksel olarak modellenmesi de diferansiyel denklemlerle yapılmaktadır.

A. Euler Yöntemi

Adi diferansiyel denklemlerin sayısal çözümleri için en temel/basit yöntemlerden birisi "*Euler yöntemi*"dir.

$$\frac{dy}{dx} = f(x,y) \qquad \text{veya} \qquad y' = f(x,y)$$

diferansiyel denkleminin her iki tarafının $[x_0, x_0+h]$ aralığında belirli integrali alınırsa

$$\int_{x_0}^{x_0+h} y'.dx = \int_{x_0}^{x_0+h} f(x,y).dx \quad , \quad y\Big]_{x_0}^{x_0+h} = \int_{x_0}^{x_0+h} f(x,y).dx$$

$$y(x_0+h) - y(x_0) = \int_{x_0}^{x_0+h} f(x,y).dx \quad , \quad y(x_0+h) = y(x_0) + \int_{x_0}^{x_0+h} f(x,y).dx$$

eşitliği elde edilir. Eğer $f(x,y)$ fonksiyonunun $[x_0, x_0+h]$ aralığındaki değişimi çok küçük ise

$$f(x,y) \cong f(x_0,y_0) \Rightarrow \int_{x_0}^{x_0+h} f(x,y).dx \cong h.f(x_0,y_0)$$

alınabilir. Böylece

$$y(x_0+h) \cong y(x_0) + h.f(x_0,y_0)$$

elde edilmiş olur. Genelleştirme yapılırsa $y(x_0) = y_0$ başlangıç koşulu altında, $k = 0,1,2,...,N$ için $x_{k+1} = x_k + (k+1)h$ yani $h = x_1 - x_0 = x_2 - x_1 = x_3 - x_2 = ...$ sabit artım miktarı olmak üzere Euler yöntemi

$$y(x_{k+1}) = y(x_k) + h.f(x_k,y_k)$$

iterasyon eşitliğiyle verilir. Aynı sonuca Taylor seri açılımıyla da ulaşılabilir:

$$y(x_0+h) = y(x_0) + h.y'(x_0) + \frac{h^2}{2!}.y''(x_0) + \frac{h^3}{3!}.y'''(x_0) + ...$$

seri açılımının ilk iki terimi kullanılarak $x_1 = x_0 + h$ olmak üzere

$$y(x_1) \cong y(x_0) + h.y'(x_0)$$

yazılabilir. $y(x_0) = y_0$ başlangıç koşulu bilindiğinden ve fonksiyonun değişimin bu aralıkta çok küçük olduğu, yani $y'(x_0) \cong f(x_0, y_0)$ varsayımı ile

$$y(x_1) \cong y(x_0) + h.f(x_0, y_0) \quad , \quad y(x_{k+1}) = y(x_k) + h.f(x_k, y_k)$$

önceden elde edilmiş olan Euler yöntemine ait iterasyon eşitliği bulunur.

Örnek-15.15: $y' = -\dfrac{y}{2+x}$ diferansiyel denkleminin girilen başlangıç koşulu, adım miktarı ve değer sayısınca sayısal çözümünü Euler yöntemiyle gerçekleştiren program.

Yöntemin etkinliğini göstermek açısından $y(0) = 0.5$ başlangıç koşulu altındaki çözümünü $h = 0.2$ alarak [0,2] aralığında karşılaştırmalı olarak gerçekleştirelim.

$$\frac{dy}{y} = -\frac{dx}{2+x} \quad , \quad \int\frac{dy}{y} = -\int\frac{dx}{2+x} \quad , \quad Lny = -Ln|2+x| + LnC \quad , \quad y = \frac{C}{2+x}$$

olarak genel çözüm bulunur. $y(0) = 0.5$ başlangıç şartı için

$$x = 0 \rightarrow y = 0.5$$

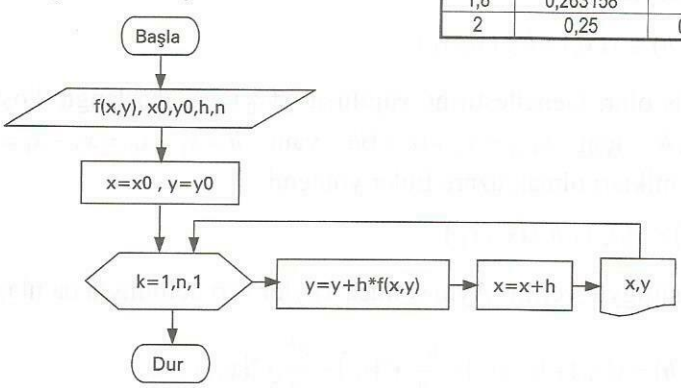
$$y = \frac{C}{2+x} \Rightarrow C = y(2+x) = 0.5(2+0) = 1$$

$$y = \frac{1}{2+x}$$

olarak bulunur (Tablo 15.4).

Tablo 15.4 Euler yöntemiyle sayısal çözüm

x	Gerçek	Euler
0	0,5	0,5
0,2	0,454545	0,45
0,4	0,416667	0,409091
0,6	0,384615	0,375
0,8	0,357143	0,346154
1	0,333333	0,321429
1,2	0,3125	0,3
1,4	0,294118	0,28125
1,6	0,277778	0,264706
1,8	0,263158	0,25
2	0,25	0,236842



C

```
/* Euler yöntemi */
#include <stdio.h>
#include <conio.h>
void main()
{ int i,n; float x,y,x0,y0,h;
  clrscr();printf("x(0): "); scanf("%f",&x0);
  printf("y(0): "); scanf("%f",&y0);
  printf("Adım değeri: "); scanf("%f",&h);
  printf("Değer sayısı: "); scanf("%u",&n);
  x=x0;y=y0;printf("\nSonuçlar (x,y)\n"); printf("%f\t%f\n",x,y);
  for(i=1;i<=n;i++)
   { y+=h*(-y/(2+x));x+=h; printf("%f\t%f\n",x,y); } getch(); }
```

C++

```
/* Euler yöntemi */
#include <iostream>
using namespace std;
int main()
{ int i,n; float x,y,x0,y0,h;
  system("cls"); cout<<"x(0): "; cin>>x0;
  cout<<"y(0): "; cin>>y0; cout<<"Adım değeri: "; cin>>h;
  cout<<"Değer sayısı: "; cin>>n;
  x=x0;y=y0;cout<<"\nSonuçlar (x,y):\n";  cout<<x<<"\t"<<y<<"\n";
  for(i=1;i<=n;i++) { y+=h*(-y/(2+x));x+=h; cout<<x<<"\t"<<y<<"\n"; }
  system("pause"); return 0; }
```

C#

```
// Euler yöntemi
using System;
namespace ornek1515
{class Program
  {static void Main(string[] args)
   {int i, n; double x, y, x0, y0, h;
    Console.Write("x(0): ");x0=Convert.ToDouble(Console.ReadLine());
    Console.Write("y(0): ");y0=Convert.ToDouble(Console.ReadLine());
 Console.Write("Adım değeri: ");h=Convert.ToDouble(Console.ReadLine());
 Console.Write("Değer sayısı: ");n=Convert.ToInt32(Console.ReadLine());
    x=x0; y=y0; Console.WriteLine("\nSonuçlar (x,y):");
    Console.WriteLine("{0}\t{1}",x,y);
    for (i=1;i<=n;i++) { y+=h*(-y/(2+x)); x+=h;
         Console.WriteLine("{0}\t{1}", x, y); } Console.ReadLine();}}}
```

JAVA

```
// Euler yöntemi
import java.util.Scanner;
public class ornek1515 {
  public static void main(String[] args) {
    int i,n; float x,y,x0,y0,h;
    Scanner klavye=new Scanner(System.in);
    System.out.print("x(0): ");x0=klavye.nextFloat();
    System.out.print("y(0): ");y0=klavye.nextFloat();
    System.out.print("Adım değeri: ");h=klavye.nextFloat();
    System.out.print("Değer sayısı: ");n=klavye.nextInt();
    x=x0; y=y0; System.out.printf("\nSonuçlar (x,y)\n");
    System.out.printf("%f\t%f\n",x,y);
    for(i=1;i<=n;i++)
    { y+=h*(-y/(2+x)); x+=h; System.out.printf("%f\t%f\n",x,y); }}}
```

MATLAB

```
% Euler yöntemi
clear all; clc; x0=input('x(0): ');y0=input('y(0): ');
h=input('Adım değeri: '); n=input('Değer sayısı: ');
x=x0; y=y0; fprintf('\nSonuçlar (x,y)\n'); fprintf('%f\t%f\n',x,y);
for i=1:n  y=y+h*(-y/(2+x)); x=x+h; fprintf('%f\t%f\n',x,y); end
```

```
x(0): 0
y(0): 0.5
Adım değeri: 0.2
Değer sayısı: 10

Sonuçlar (x,y)
0.000000  0.500000
0.200000  0.450000
0.400000  0.409091
0.600000  0.375000
0.800000  0.346154
1.000000  0.321429
1.200000  0.300000
1.400000  0.281250
1.600000  0.264706
1.800000  0.250000
2.000000  0.236842
```

PYTHON

```
# Euler yöntemi
x0=eval(input("x(0): ")); y0=eval(input("y(0): "))
h=eval(input("Adım değeri: ")); n=int(input("Değer sayısı: "))
x=x0; y=y0; print("\nSonuçlar (x,y)"); print("%f\t%f"%(x,y))
for i in range(n):
    y+=h*(-y/(2+x)); x+=h; print("%f\t%f"%(x,y))
```

B. Heun Yöntemi

Kestirme ve düzeltme prensibine göre çalışan Heun yönteminin bir diğer adı da, "*Düzeltilmiş Euler yöntemi*"dir. Euler yönteminde;

$$y(x_0+h)=y(x_0)+\int_{x_0}^{x_0+h} f(x,y).dx$$

hesaplanırken, fonksiyondaki değişimin küçük olduğu varsayımı ile aralığın solundaki $f(x_0,y_0)$ değeri yaklaşık olarak $f(x,y)$'ye eşit alınıyordu. Oysa Heun yönteminde; aralığın solundaki değer değil, solundaki ve sağındaki değerin aritmetik ortalaması alınmaktadır.

$$f(x,y)\cong\frac{f(x_0,y_0)+f(x_1,y_1^*)}{2}$$

Burada, $x_1=x_0+h$ olup $y_1^*=y(x_1)$ dir. y_1^* için başlangıç değeri, Euler yönteminden hesaplanmaktadır. Yani

$$y_1^*=y(x_0)+h.f(x_0,y_0)$$

hesaplanır ve sonra da

$$y(x_1) = y(x_0) + \frac{h}{2}\left\{f(x_0, y_0) + f\left(x_1, y_1^*\right)\right\}$$

Heun yöntemi eşitliğinde yerine yazılmaktadır. Görüldüğü gibi Heun yöntemindeki genel iterasyon eşitliği

$$y(x_{k+1}) = y(x_k) + \frac{h}{2}\left\{f(x_k, y_k) + f\left(x_{k+1}, y_{k+1}^*\right)\right\}$$

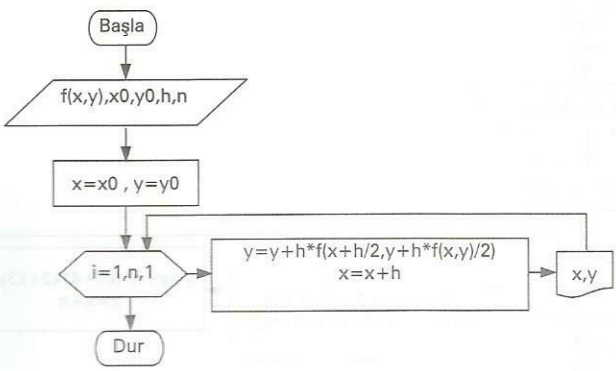
şeklindedir.

Örnek-15.16: $y' = -\dfrac{y}{2+x}$ diferansiyel denkleminin girilen başlangıç koşulu, adım miktarı ve değer sayısınca sayısal çözümünü Heun yöntemiyle gerçekleştiren program.

Örnekteki diferansiyel denklemin gerçek, Euler ve Heun yöntemleriyle ilgili karşılaştırmalı sayısal çözümleri Tablo 15.5'te verilmektedir.

Tablo 15.5 Heun yöntemiyle sayısal çözüm

x	Gerçek	Euler	Heun
0	0,5	0,5	0,5
0,2	0,454545	0,45	0,454762
0,4	0,416667	0,409091	0,417015
0,6	0,384615	0,375	0,385044
0,8	0,357143	0,346154	0,357619
1	0,333333	0,321429	0,333836
1,2	0,3125	0,3	0,313017
1,4	0,294118	0,28125	0,294639
1,6	0,277778	0,264706	0,278297
1,8	0,263158	0,25	0,263672
2	0,25	0,236842	0,250506



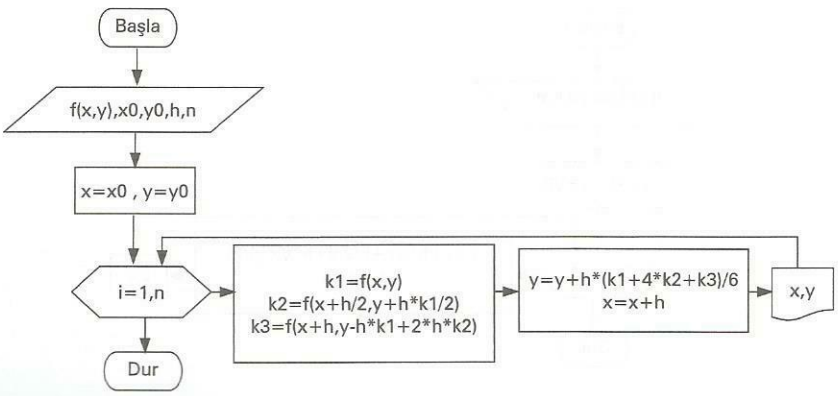
C. Runge-Kutta Yöntemleri

Alman matematikçilerden 1891 yılında Carl Runge(1856-1927) tarafından önerilen ve 1901 yılında da Wilhelm Kutta(1867-1944) tarafından geliştirilen "*Runge-Kutta*" yöntemleri, diferansiyel denklemlerin sayısal çözümlerinde kullanılan en etkin yöntemlerdendir. Tablo 15.6'da ikinci, üçüncü ve dördüncü derece Runge-Kutta iterasyon eşitlikleri verilmektedir.

Tablo 15.6 İkinci, üçüncü ve dördüncü derece Runge-Kutta iterasyon eşitlikleri

Derece	Katsayılar	İterasyon eşitliği
İkinci	$k_1 = f(x_i, y_i)$ $k_2 = f(x_i + h, y_i + hk_1)$	$y_{i+1} = y_i + \frac{h}{2}(k_1 + k_2)$
Üçüncü	$k_1 = f(x_i, y_i)$ $k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{hk_1}{2}\right)$ $k_3 = f(x_i + h, y_i - hk_1 + 2hk_2)$	$y_{i+1} = y_i + \frac{h}{6}(k_1 + 4k_2 + k_3)$
Dördüncü	$k_1 = f(x_i, y_i)$ $k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{hk_1}{2}\right)$ $k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{hk_2}{2}\right)$ $k_4 = f(x_i + h, y_i + hk_3)$	$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

Örnek-15.17: $y' = -\frac{y}{2+x}$ diferansiyel denkleminin girilen başlangıç koşulu, adım miktarı ve değer sayısınca sayısal çözümünü 3. derece Runge-Kutta yöntemiyle gerçekleştiren program.



C

```
/* 3. derece Runge-Kutta yöntemi */
#include <stdio.h>
#include <conio.h>
float f(float a, float b) { return -b/(2+a); }
void main()
{ int i,n; float x,y,x0,y0,h,k1,k2,k3;
  clrscr();printf("x(0): "); scanf("%f",&x0);
  printf("y(0): "); scanf("%f",&y0);
  printf("Adım değeri: "); scanf("%f",&h);
  printf("Değer sayısı: "); scanf("%u",&n);
  x=x0;y=y0;printf("\nSonuçlar (x,y)\n"); printf("%f\t%f\n",x,y);
  for(i=0;i<n;i++)
   { k1=f(x,y); k2=f(x+h/2,y+h*k1/2); k3=f(x+h,y-h*k1+2*h*k2);
     y+=h*(k1+4*k2+k3)/6; x+=h; printf("%f\t%f\n",x,y); } getch(); }
```

C++

```
/* 3. derece Runge-Kutta yöntemi */
#include <iostream>
using namespace std;
float f(float a, float b) { return -b/(2+a); }
int main()
{ int i,n; float x,y,x0,y0,h,k1,k2,k3;
  system("cls"); cout<<"x(0): "; cin>>x0; cout<<"y(0): "; cin>>y0;
  cout<<"Adım değeri: "; cin>>h; cout<<"Değer sayısı: "; cin>>n;
  x=x0;y=y0;cout<<"\nSonuçlar (x,y):\n"; cout<<x<<"\t"<<y<<"\n";
  for(i=0;i<n;i++)
    { k1=f(x,y); k2=f(x+h/2,y+h*k1/2); k3=f(x+h,y-h*k1+2*h*k2);
      y+=h*(k1+4*k2+k3)/6; x+=h; cout<<x<<"\t"<<y<<"\n"; }
  system("pause"); return 0; }
```

C#

```
// 3. derece Runge-Kutta yöntemi
using System;
namespace ornek1517
{ class Program
  { public static double f(double a,double b) { return -b/(2+a); }
    static void Main(string[] args)
    { int i, n; double x, y, x0, y0, h, k1, k2, k3;
      Console.Write("x(0): ");x0=Convert.ToDouble(Console.ReadLine());
      Console.Write("y(0): ");y0=Convert.ToDouble(Console.ReadLine());
      Console.Write("Adım değeri: ");
      h=Convert.ToDouble(Console.ReadLine());
      Console.Write("Değer sayısı: ");
      n=Convert.ToInt32(Console.ReadLine());
      x=x0; y=y0; Console.WriteLine("\nSonuçlar (x,y):");
      Console.WriteLine("{0}\t{1}",x,y);
       for (i=0;i<n;i++)
       { k1=f(x,y); k2=f(x+h/2,y+h*k1/2); k3=f(x+h,y-h*k1+2*h*k2);
         y+=h*(k1+4*k2+k3)/6; x+=h;Console.WriteLine("{0}\t{1}",x,y);}
      Console.ReadLine(); }}}
```

JAVA

```
// 3. derece Runge-Kutta yöntemi
import java.util.Scanner;
public class ornek1517 {
  public static float f(float a,float b){ return -b/(2+a); }
  public static void main(String[] args) {
    int i,n; float x,y,x0,y0,h,k1,k2,k3;
    Scanner klavye=new Scanner(System.in);
    System.out.print("x(0): ");x0=klavye.nextFloat();
    System.out.print("y(0): ");y0=klavye.nextFloat();
    System.out.print("Adım değeri: ");h=klavye.nextFloat();
    System.out.print("Değer sayısı: ");n=klavye.nextInt();
    x=x0; y=y0; System.out.printf("\nSonuçlar (x,y)\n");
    System.out.printf("%f\t%f\n",x,y);
    for(i=0;i<n;i++)
    { k1=f(x,y); k2=f(x+h/2,y+h*k1/2); k3=f(x+h,y-h*k1+2*h*k2);
      y+=h*(k1+4*k2+k3)/6; x+=h;System.out.printf("%f\t%f\n",x,y); }}}
```

MATLAB

```
% 3. derece Runge-Kutta yöntemi
clear all; clc;
f=inline('-y/(2+x)');
x0=input('x(0): ');y0=input('y(0): ');
h=input('Adım değeri: '); n=input('Değer sayısı: ');
x=x0; y=y0; fprintf('\nSonuçlar (x,y)\n'); fprintf('%f\t%f\n',x,y);
for i=1:n
    k1=f(x,y); k2=f(x+h/2,y+h*k1/2); k3=f(x+h,y-h*k1+2*h*k2);
    y=y+h*(k1+4*k2+k3)/6; x=x+h; fprintf('%f\t%f\n',x,y);
end
```

PYTHON

```
# 3. derece Runge-Kutta yöntemi
def f(a,b):
    return -b/(2+a)
x0=eval(input("x(0): ")); y0=eval(input("y(0): "))
h=eval(input("Adım değeri: ")); n=int(input("Değer sayısı: "))
x=x0; y=y0; print("\nSonuçlar (x,y)"); print("%f\t%f"%(x,y))
for i in range(n):
    k1=f(x,y); k2=f(x+h/2,y+h*k1/2); k3=f(x+h,y-h*k1+2*h*k2)
    y=y+h*(k1+4*k2+k3)/6; x=x+h; print("%f\t%f"%(x,y))
```

```
x(0): 0
y(0): 0.5
Adım değeri: 0.2
Değer sayısı: 10

Sonuçlar (x,y)
0.000000  0.500000
0.200000  0.454545
0.400000  0.416667
0.600000  0.384615
0.800000  0.357143
1.000000  0.333333
1.200000  0.312500
1.400000  0.294118
1.600000  0.277778
1.800000  0.263158
2.000000  0.250000
```

D. Diferansiyel Denklem Sistemleri Çözümleri

İki denklemden oluşan örnek sistem

$$\begin{cases} y' = f(x,y,z) & , \quad y(x_0) = y_0 \\ z' = f(x,y,z) & , \quad z(x_0) = z_0 \end{cases}$$

şeklindedir. $y' = f(x,y)$ diferansiyel denkleminin $y(x_0) = y_0$ başlangıç koşulu altındaki sayısal çözümü için verilen yöntemler, N denklemden oluşan $i = 1,2,3,.....,N$ olmak üzere $y_i' = f_i(x,y_i)$ şeklindeki diferansiyel denklem sistemlerine de kolaylıkla uygulanabilir.

1. Euler Yöntemiyle Diferansiyel Denklem Sistemi Çözümü

Diferansiyel denklemlerin çözümünde Euler iterasyon eşitliği

$$y_{k+1} = y_k + h.f(x_k, y_k)$$

şeklindeydi. Diferansiyel denklem sistemlerine gelince;

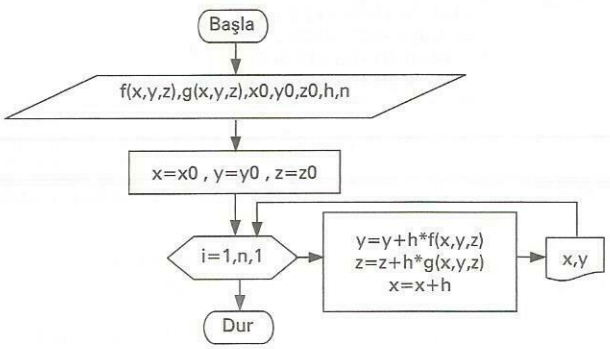
$$\begin{cases} y' = f(x,y,z) & , \quad y(x_0) = y_0 \\ z' = f(x,y,z) & , \quad z(x_0) = z_0 \end{cases}$$

Euler iterasyon eşitlikleri

$$\begin{cases} y_{k+1} = y_k + h.f(x_k, y_k, z_k) \\ \\ z_{k+1} = z_k + h.g(x_k, y_k, z_k) \end{cases}$$

şeklinde uygulanmaktadırlar.

Örnek-15.18: $\begin{cases} y' = 2x + y - 3z \\ z' = 3x - 2y + z \end{cases}$ diferansiyel denklem sisteminin girilen başlangıç koşulu, adım miktarı ve değer sayısınca sayısal çözümünü Euler yöntemiyle gerçekleştiren program.



2. Runge-Kutta Yöntemiyle Diferansiyel Denklem Sistemi Çözümü

Dördüncü mertebe Runge-Kutta yöntemi, diferansiyel denklem sistemine

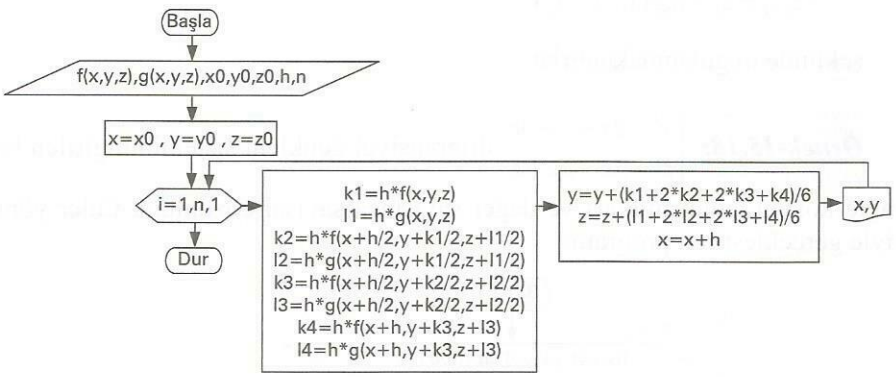
$$\left.\begin{array}{l} y' = f(x,y,z) \quad , \quad y(x_0) = y_0 \\ z' = f(x,y,z) \quad , \quad z(x_0) = z_0 \end{array}\right\} \Rightarrow \left.\begin{array}{l} y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ z_{i+1} = z_i + \frac{1}{6}(l_1 + 2l_2 + 2l_3 + l_4) \end{array}\right\}$$

şeklinde uygulanmaktadır. Buradaki katsayılar Tablo 15.7'de verilmektedir.

Tablo 15.7 Sistemler için dördüncü derece Runge-Kutta katsayıları

Katsayılar	
$k_1 = h.f(x_i, y_i, z_i)$	$l_1 = h.g(x_i, y_i, z_i)$
$k_2 = h.f\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}, z_i + \frac{l_1}{2}\right)$	$l_2 = h.g\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}, z_i + \frac{l_1}{2}\right)$
$k_3 = h.f\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}, z_i + \frac{l_2}{2}\right)$	$l_3 = h.g\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}, z_i + \frac{l_2}{2}\right)$
$k_4 = h.f(x_i + h, y_i + k_3, z_i + l_3)$	$l_4 = h.g(x_i + h, y_i + k_3, z_i + l_3)$

Örnek-15.19: $\begin{cases} y' = 2x + y - 3z \\ z' = 3x - 2y + z \end{cases}$ diferansiyel denklem sisteminin girilen başlangıç koşulu, adım miktarı ve değer sayısınca sayısal çözümünü 4. derece Runge-Kutta yöntemiyle gerçekleştiren program.



C

```
/* 4. derece Runge-Kutta yöntemiyle dif. denk. sistemi çözümü */
#include <stdio.h>
#include <conio.h>
float f(float v,float w,float t){ return 2*v+w-3*t; }
float g(float p,float r,float s){ return 3*p-2*r+s; }
void main()
{ int i,n; float x,y,z,x0,y0,z0,h,k1,k2,k3,k4,l1,l2,l3,l4;
  clrscr();printf("x(0): "); scanf("%f",&x0);
```

```
printf("y(0): "); scanf("%f",&y0);
printf("z(0): "); scanf("%f",&z0);
printf("Adım değeri: "); scanf("%f",&h);
printf("Değer sayısı: "); scanf("%u",&n); x=x0;y=y0;z=z0;
printf("\nSonuçlar (x,y,z)\n"); printf("%f\t%f\t%f\n",x,y,z);
for(i=0;i<n;i++)
 { k1=h*f(x,y,z); l1=h*g(x,y,z);
    k2=h*f(x+h/2,y+k1/2,z+l1/2); l2=h*g(x+h/2,y+k1/2,z+l1/2);
    k3=h*f(x+h/2,y+k2/2,z+l2/2); l3=h*g(x+h/2,y+k2/2,z+l2/2);
    k4=h*f(x+h,y+k3,z+l3); l4=h*g(x+h,y+k3,z+l3);
    y+=(k1+2*k2+2*k3+k4)/6; z+=(l1+2*l2+2*l3+l4)/6; x+=h;
    printf("%f\t%f\t%f\n",x,y,z); } getch(); }
```

C++

```
// 4. derece Runge-Kutta yöntemiyle dif. denk. sistemi çözümü
#include <iostream>
using namespace std;
float f(float v,float w,float t){ return 2*v+w-3*t; }
float g(float p,float r,float s){ return 3*p-2*r+s; }
int main()
{ int i,n; float x,y,z,x0,y0,z0,h,k1,k2,k3,k4,l1,l2,l3,l4;
  system("cls"); cout<<"x(0): "; cin>>x0; cout<<"y(0): "; cin>>y0;
  cout<<"z(0): "; cin>>z0;cout<<"Adım değeri: "; cin>>h;
  cout<<"Değer sayısı: "; cin>>n; x=x0;y=y0;z=z0;
  cout<<"\nSonuçlar (x,y,z):\n"; cout<<x<<"\t"<<y<<"\t"<<z<<"\n";
  for(i=0;i<n;i++)
    { k1=h*f(x,y,z); l1=h*g(x,y,z);
      k2=h*f(x+h/2,y+k1/2,z+l1/2); l2=h*g(x+h/2,y+k1/2,z+l1/2);
      k3=h*f(x+h/2,y+k2/2,z+l2/2); l3=h*g(x+h/2,y+k2/2,z+l2/2);
      k4=h*f(x+h,y+k3,z+l3); l4=h*g(x+h,y+k3,z+l3);
      y+=(k1+2*k2+2*k3+k4)/6; z+=(l1+2*l2+2*l3+l4)/6; x+=h;
      cout<<x<<"\t"<<y<<"\t"<<z<<"\n"; } system("pause"); return 0; }
```

C#

```
// 4. derece Runge-Kutta yöntemiyle dif. denk. sistemi çözümü
using System;
namespace ornek1519
{class Program {
  public static double f(double v,double w,double t){return 2*v+w-3*t;}
  public static double g(double p,double r,double s){return 3*p-2*r+s;}
  static void Main(string[] args)
   { int i,n; double x,y,z,x0,y0,z0,h,k1,k2,k3,k4,l1,l2,l3,l4;
     Console.Write("x(0): "); x0=Convert.ToDouble(Console.ReadLine());
     Console.Write("y(0): "); y0=Convert.ToDouble(Console.ReadLine());
     Console.Write("z(0): "); z0=Convert.ToDouble(Console.ReadLine());
 Console.Write("Adım değeri: ");h=Convert.ToDouble(Console.ReadLine());
 Console.Write("Değer sayısı: ");n=Convert.ToInt32(Console.ReadLine());
     x=x0; y=y0; z=z0; Console.WriteLine("\nSonuçlar (x,y,z):");
     Console.WriteLine("{0}\t{1}\t{2}",x,y,z);
     for (i=0;i<n;i++){
       k1=h*f(x,y,z); l1=h*g(x,y,z);
       k2=h*f(x+h/2,y+k1/2,z+l1/2); l2=h*g(x+h/2,y+k1/2,z+l1/2);
       k3=h*f(x+h/2,y+k2/2,z+l2/2); l3=h*g(x+h/2,y+k2/2,z+l2/2);
       k4=h*f(x+h,y+k3,z+l3); l4=h*g(x+h,y+k3,z+l3);
       y+= (k1+2*k2+2*k3+k4)/6; z+= (l1+2*l2+2*l3+l4)/6; x+= h;
       Console.WriteLine("{0}\t{1}\t{2}", x, y, z); }
     Console.ReadLine();   } } }
```

JAVA

```
// 4. derece Runge-Kutta yöntemiyle dif. denk. sistemi çözümü
import java.util.Scanner;
public class ornek1519 {
  public static float f(float v,float w,float t){ return 2*v+w-3*t; }
  public static float g(float p,float r,float s){ return 3*p-2*r+s; }
  public static void main(String[] args) {
    int i,n; float x,y,z,x0,y0,z0,h,k1,k2,k3,k4,l1,l2,l3,l4;
    Scanner klavye=new Scanner(System.in);
    System.out.print("x(0): ");x0=klavye.nextFloat();
    System.out.print("y(0): ");y0=klavye.nextFloat();
    System.out.print("z(0): ");z0=klavye.nextFloat();
    System.out.print("Adım değeri: ");h=klavye.nextFloat();
    System.out.print("Değer sayısı: ");n=klavye.nextInt();
    x=x0; y=y0; z=z0; System.out.printf("\nSonuçlar (x,y,z)\n");
    System.out.printf("%f\t%f\t%f\n",x,y,z);
    for(i=0;i<n;i++)
    { k1=h*f(x,y,z); l1=h*g(x,y,z);
      k2=h*f(x+h/2,y+k1/2,z+l1/2); l2=h*g(x+h/2,y+k1/2,z+l1/2);
      k3=h*f(x+h/2,y+k2/2,z+l2/2); l3=h*g(x+h/2,y+k2/2,z+l2/2);
      k4=h*f(x+h,y+k3,z+l3); l4=h*g(x+h,y+k3,z+l3);
      y+=(k1+2*k2+2*k3+k4)/6; z+=(l1+2*l2+2*l3+l4)/6; x+=h;
      System.out.printf("%f\t%f\t%f\n",x,y,z); }}}
```

MATLAB

```
% 4.mertebe Runge-Kutta yöntemiyle dif. denk. sistemi çözümü
clear all; clc; f=inline('2*x+y-3*z'); g=inline('3*x-2*y+z');
x0=input('x(0): '); y0=input('y(0): '); z0=input('z(0): ');
h=input('Adım değeri: '); n=input('Değer sayısı: '); x=x0;y=y0;z=z0;
fprintf('\nSonuçlar (x,y,z)\n');fprintf('%f\t%f\t%f\n',x,y,z);
for i=1:n
    k1=h*f(x,y,z);l1=h*g(x,y,z);
    k2=h*f(x+h/2,y+k1/2,z+l1/2);l2=h*g(x+h/2,y+k1/2,z+l1/2);
    k3=h*f(x+h/2,y+k2/2,z+l2/2); l3=h*g(x+h/2,y+k2/2,z+l2/2);
    k4=h*f(x+h,y+k3,z+l3); l4=h*g(x+h,y+k3,z+l3);
    y=y+(k1+2*k2+2*k3+k4)/6; z=z+(l1+2*l2+2*l3+l4)/6; x=x+h;
    fprintf('%f\t%f\t%f\n',x,y,z);
end
```

PYTHON

```
# 4. derece Runge-Kutta yöntemiyle dif. denk. sistemi çözümü
def f(v,w,t):
    return 2*v+w-3*t
def g(p,r,s):
    return 3*p-2*r+s
x0=eval(input("x(0): ")); y0=eval(input("y(0): "))
z0=eval(input("z(0): ")); h=eval(input("Adım değeri: "))
n=int(input("Değer sayısı: ")); x=x0; y=y0; z=z0
print("\nSonuçlar (x,y,z)"); print("%f\t%f\t%f"%(x,y,z))
for i in range(n):
    k1=h*f(x,y,z); l1=h*g(x,y,z)
    k2=h*f(x+h/2,y+k1/2,z+l1/2); l2=h*g(x+h/2,y+k1/2,z+l1/2)
    k3=h*f(x+h/2,y+k2/2,z+l2/2); l3=h*g(x+h/2,y+k2/2,z+l2/2)
    k4=h*f(x+h,y+k3,z+l3); l4=h*g(x+h,y+k3,z+l3)
    y+=(k1+2*k2+2*k3+k4)/6; z+=(l1+2*l2+2*l3+l4)/6; x+=h
    print("%f\t%f\t%f"%(x,y,z))
```

```
x(0): 0
y(0): 0
z(0): 0
Adım değeri: 0.05
Değer sayısı: 10

Sonuçlar (x,y,z)
0.000000  0.000000  0.000000
0.050000  0.002353  0.003733
0.100000  0.008813  0.014889
0.150000  0.018450  0.033477
0.200000  0.030275  0.059599
0.250000  0.043229  0.093473
0.300000  0.056152  0.135436
0.350000  0.067764  0.185972
0.400000  0.076631  0.245725
0.450000  0.081135  0.315530
0.500000  0.079429  0.396442
```

SORULAR

1. Enterpolasyon hesaplamaları için diğer yöntemleri araştırarak akış diyagramlarını çiziniz ve kodlayınız.

2. En küçük kareler yöntemini araştırarak, ilgili fonksyonlar için akış diyagramlarını çizerek kodlayınız.

3. Sayısal türev hesaplamaları için diğer yöntemleri araştırarak akış diyagramlarını çiziniz ve kodlayınız.

4. Sayısal integral hesaplamaları için diğer yöntemleri araştırarak akış diyagramlarını çiziniz ve kodlayınız.

5. İki katlı integrallerin sayısal hesaplanmalarını araştırarak akış diyagramlarını çiziniz ve kodlayınız.

6. Üç katlı integrallerin sayısal hesaplanmalarını araştırarak akış diyagramlarını çiziniz ve kodlayınız.

7. Kök bulma/hesaplama için kullanılan diğer yöntemleri araştırarak akış diyagramlarını çiziniz ve kodlayınız.

8. Doğrusal denklem sistemlerinin çözümü için diğer yöntemleri araştırarak akış diyagramlarını çiziniz ve kodlayınız.

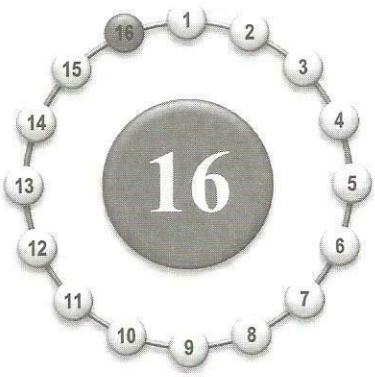
9. Diferansiyel denklemlerin çözümünde kullanılan diğer yöntemleri araştırarak akış diyagramlarını çiziniz ve kodlayınız.

10. Diferansiyel denklem sistemlerinin çözümünde kullanılan diğer yöntemleri araştırarak akış diyagramlarını çiziniz ve kodlayınız.

11. Programlama dillerinde sayısal analiz yöntemleri için hazır/mevcut fonksiyonları/komutları araştırarak karşılaştırınız.

BÖLÜM 16

DONANIM UYGULAMALARI



Bu bölümde

- Bilgisayar donanımının bazı kısımlarını tanıyacak,
- Dış dünya ile iletişimini kavrayacak,
- Veri alış-verişi için algoritmalarını hazırlayarak kodlamalarını öğreneceksiniz.

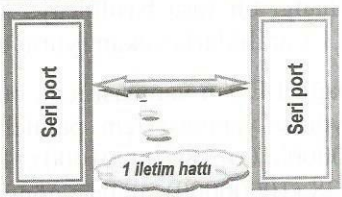
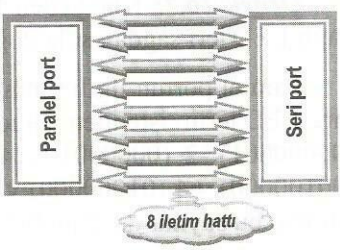
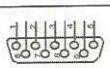
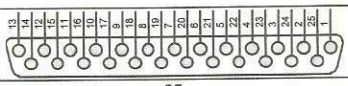
Bölüm sonunda

- Bilgisayar ile dış dünya arasında haberleşemeyi sağlayacak,
- Veri alış-verişi için algoritmalar hazırlayıp kodlayabileceksiniz.

I. BİLGİSAYAR PORTLARI

Kelime anlamı "liman" olan bilgisayarın portları, birer veri kanallarıdır. Bu portlar kullanılarak dışarıdan veri alınabilmekte veya dışarıya veri gönderilebilmektedir. Yani portlar, bilgisayar ile harici bağlı cihaz arasında veri alışverişini sağlamaktadırlar (bilgi alıp göndermektedirler). Bilgisayarlarda seri port, paralel port, USB portu, PS/2 portu, SCSI portu, RJ-45 portu, RJ-11 portu, IEEE1394 Firewire portu gibi birçok port bulunabilmektedir. İletişim şekillerine bakıldığında ise temelde veriler seri veya paralel olarak aktarılmaktadırlar (Tablo 16.1).

Tablo 16.1: Bilgisayarda seri/paralel iletişim

<table>
<tr><th></th><th>Seri</th><th>Paralel</th></tr>
<tr><td>Blok diyagram</td><td> 
</td><td> 
</td></tr>
<tr><td>Veri alma ve gönderme için iletim hattı sayısı</td><td>1</td><td>8</td></tr>
<tr><td>Tek seferde gönderilen veya alınan bit sayısı</td><td>1</td><td>8</td></tr>
<tr><td>Port</td><td>  </td><td>  </td></tr>
<tr><td>Konnektör pin sayısı</td><td>9 (yeni), 25 (eski)</td><td>25</td></tr>
<tr><td>Konnektör pin bağlantıları</td><td>
<table>
<tr><th>9 pinli</th><th>25 pinli</th><th>Açıklama</th></tr>
<tr><td>1</td><td>8</td><td>Port için "Veri algılandı" işareti</td></tr>
<tr><td>2</td><td>3</td><td>Veri alma</td></tr>
<tr><td>3</td><td>2</td><td>Veri gönderme</td></tr>
<tr><td>4</td><td>20</td><td>Portun "Veri almaya hazır" işareti</td></tr>
<tr><td>5</td><td>7</td><td>Toprak hattı</td></tr>
<tr><td>6</td><td>6</td><td>Karşı tarafın "Veri almaya hazır" işareti</td></tr>
<tr><td>7</td><td>4</td><td>Portun "Veri gönderme isteği" işareti</td></tr>
<tr><td>8</td><td>5</td><td>Portun 'veri gönderme isteğine' cevap işareti</td></tr>
<tr><td>9</td><td>22</td><td>"Ring indicator" işareti</td></tr>
</table>
</td><td>
<table>
<tr><th>Pin</th><th>Açıklama</th></tr>
<tr><td>1</td><td>Bilgisayarla yazıcı arasındaki "veri alış-veriş kontrol" işareti</td></tr>
<tr><td>2 - 9</td><td>1 byte'lık veri hatları (her birinden 1'er bit gider)</td></tr>
<tr><td>10</td><td>"Veri alındı" ve "meşgul değil" işareti</td></tr>
<tr><td>11</td><td>Yazıcının "meşgul" işareti</td></tr>
<tr><td>12</td><td>Yazıcının "kağıt bitti" işareti</td></tr>
<tr><td>13</td><td>Bilgisayarla yazıcı arasındaki "seçim" işareti</td></tr>
<tr><td>14</td><td>Yazıcının "satırbaşı yap" işareti</td></tr>
<tr><td>15</td><td>Yazıcıda "hata" işareti</td></tr>
<tr><td>16</td><td>Yazıcının "sıfırlama(kapanıp açılma)" işareti</td></tr>
<tr><td>17</td><td>Seçim</td></tr>
<tr><td>18 - 25</td><td>Toprak hatları</td></tr>
</table>
</td></tr>
</table>

Bilgisayardaki herbir portun, kendine ait adresi vardır. Bu adresler kullanılarak ilgili porta erişilebilmekte ve üzerinden veri aktarımı yapılabilmektedir. Portun ilgili pininde "1" bitinin olması, +5 V; "0" bitinin olması da 0 V gerilim değerindedir/anlamındadır.

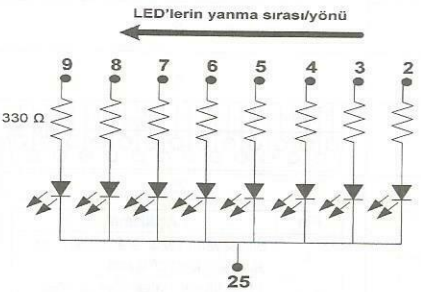
Örnek-16.1: Paralel porta 10 tabanında "11" değeri gönderildiğinde, pinlerdeki gerilim durumlarını tespit ediniz.

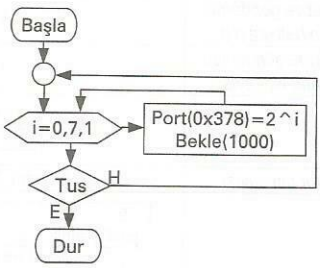
"`Port(adres)=değer`" ile 10 tabanında bir değer, paralel porta gönderildiğinde; bu değer bilgisayar tarafından 2 tabanına dönüştürülmekte ve en düşük değerlikli biti 2 numaralı pine; en yüksek değerlikli biti de 9 numaralı pine gelecek şekilde yerleştirilmektedir. $(11)_{10} = (1011)_{10}$ olduğundan pinlerdeki gerilim durumları aşağıdaki gibi olacaktır.

Pin	9	8	7	6	5	4	3	2
Verinin ikili değeri	(2^7)	(2^6)	(2^5)	(2^4)	(2^3)	(2^2)	(2^1)	(2^0)
	0	0	0	0	1	0	1	1
Gerilim (V)	0	0	0	0	5	0	5	5
Veri	$2^3+2^1+2^0=8+2+1=11$							

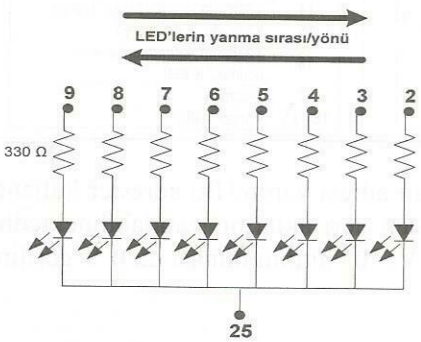
Örnek-16.2: Klavyeden herhangi bir tuşa basılıncaya kadar paralel porta bağlı LED'leri sağdan sola doğru 1 s aralıklarla yakan (yürüyen ışık) program.

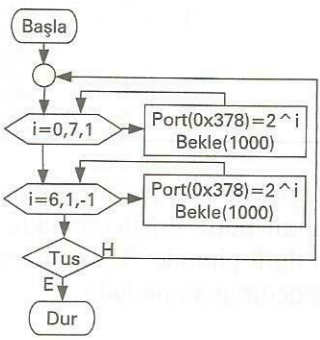
Paralel portun ilgili pinine bağlı LED'in yanması için, burada +5 V olmalıdır. LED'lerin istenilen sırada/yönde yanması için paralel porta sırasıyla 1 (00000001), 2 (00000010), 4 (00000100), 8 (00001000), 16 (00010000), 32 (00100000), 64 (01000000) ve 128 (10000000) sayıları gönderilmelidir. Dikkat edilirse bu sayılar da 2'nin 0-7 arası kuvvetleridir.





Örnek-16.3: Klavyeden herhangi bir tuşa basılıncaya kadar paralel porta bağlı LED'leri önce sağdan sola, sonra da soldan sağa doğru 1 s aralıklarla yakan program.

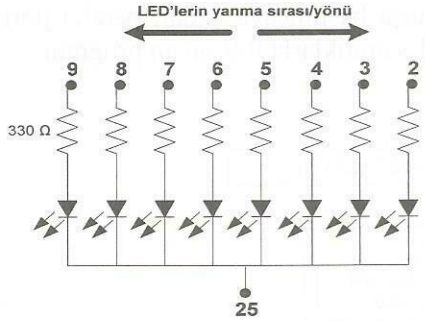


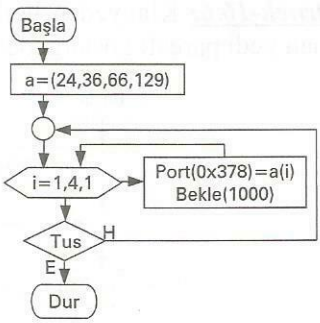


Örnek-16.4: Klavyeden herhangi bir tuşa basılıncaya kadar paralel porta bağlı LED'leri ortadan iki yana doğru 1 s aralıklarla yakan program.

LED'lerin istenilen sırada/yönde yanması için öncelikle paralel porta gönderilecek sayılar hesaplanmalıdır.

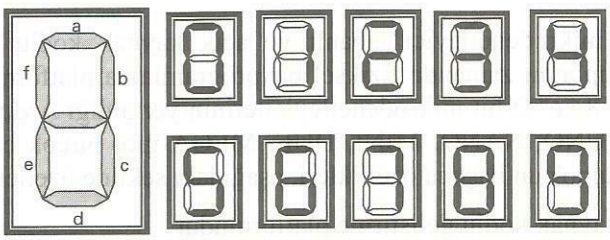
9	8	7	6	5	4	3	2	Sayı
0	0	0	1	1	0	0	0	*24*
0	0	1	0	0	1	0	0	*36*
0	1	0	0	0	0	1	0	*66*
1	0	0	0	0	0	0	1	*129*



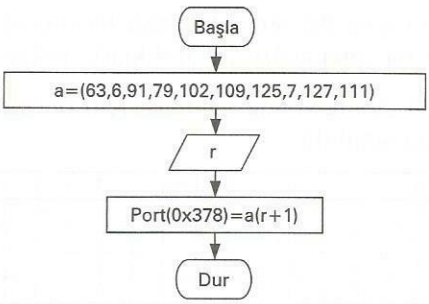


Örnek-16.5: Klavyeden girilen 0-9 arası rakamı, paralel porta bağlanan yedi parçalı göstergede (display) yakan program.

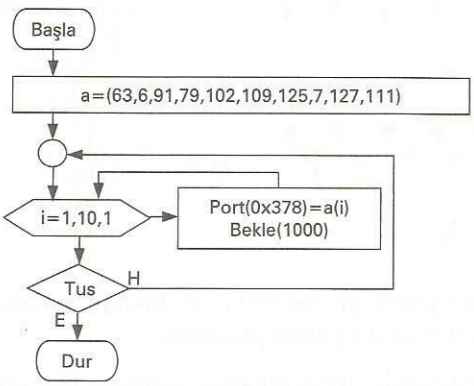
Yedi parçalı göstergede, ilgili rakamın yanması için öncelikle paralel porta gönderilecek sayılar hesaplanmalıdır.



Pin	9	8	7	6	5	4	3	2	Sayı
		g	*f*	*e*	*d*	*c*	*b*	*a*	
0	0	0	1	1	1	1	1	1	63
1	0	0	0	0	0	1	1	0	6
2	0	1	0	1	1	0	1	1	91
3	0	1	0	0	1	1	1	1	79
4	0	1	1	0	0	1	1	0	102
5	0	1	1	0	1	1	0	1	109
6	0	1	1	1	1	1	0	1	125
7	0	0	0	0	0	1	1	1	7
8	0	1	1	1	1	1	1	1	127
9	0	1	1	0	1	1	1	1	111



Örnek-16.6: Klavyeden herhangi bir tuşa basılıncaya kadar, paralel porta bağlanan yedi parçalı göstergede (display) 1 s aralıklarla 0-9 sayan program.



II. ARDUINO

Arduino; mikrodenetleyici tabanlı ve açık kaynak kodlu bir giriş/çıkış (G/Ç) kartı olup aynı zamanda fiziksel bir programlama platformudur. Üzerlerinde Atmel'in 8 ve 32 bit mikrodenetleyicilerinin yer aldığı Arduino'nun; piyasada Arduino UNO, NANO, MINI, DUE, MEGA gibi birçok çeşitleri bulunmaktadır. Arduino'nun başlıca özellikleri, aşağıdaki şekilde özetlenebilir:

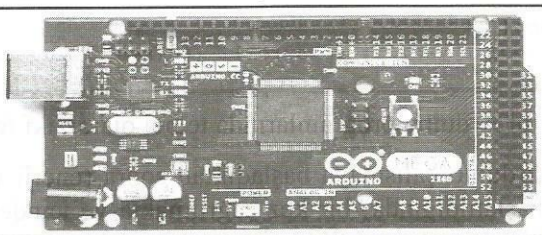
- Açık kaynak kodlu geliştirme platformudur.
- Bütün işletim sistemlerinde çalışabilmektedir.
- Çok geniş kütüphane/komut ve bileşen (ek donanım - shield) desteğine sahiptir.
- Kurulumu ve programlanması son derece basittir.
- Üzerinde bulunan çok sayıda analog ve sayısal giriş/çıkışlar sayesinde bilgisayar ile dış dünya arasındaki veri alış-verişi hızlı, kolay ve etkin bir şekilde yapılabilmektedir.

Örneğin kitaptaki uygulamalarda kullanılacak olan Arduino Mega 2560 R3 kartının teknik özellikleri, Tablo 16.2'de verilmektedir.

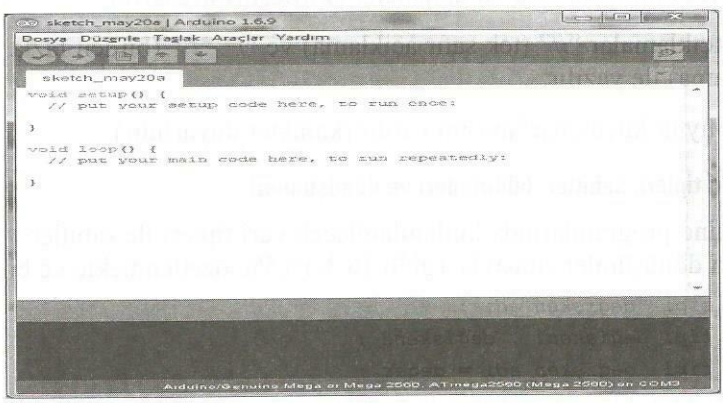
A. Arduino programlama

Arduino yazılım geliştirme ekranı, Şekil 16.1'de verilmektedir. Arduino programları burada kolaylıkla yazılabilir, derlenebilir ve çalıştırılabilmektedirler.

Tablo 16.2: Arduino Mega 2560 R3 teknik özellikleri



Mikrodenetleyici	ATmega2560 (8-bit)
Çalışma gerilimi	5 V
Giriş gerilimi (önerilen)	7-12 V
Giriş gerilimi (sınır)	6-20 V
Sayısal G/Ç pinleri	54 (15 tanesi PWM çıkış)
Analog giriş pinleri	16
Her G/Ç pini için DC akım	20 mA
3.3 V pini için DC akım	50 mA
Flash bellek	256 KB (8 KB bootloader tarafından kullanılmaktadır)
SRAM	8 KB
EEPROM	4 KB
Saat (Clock) hızı	16 MHz
Boyutlar /Ağırlık	101.52 mm x 53.3 mm / 37 gr



Şekil 16.1: Arduino yazılım geliştirme ortamı

1. Arduino program yapısı

Arduino programlarının temel yapısı aşağıdaki gibidir:

```
// Genel (global) tanımlamalar
void setup() {
// kurulum/ayar kodları yazılır ve bir defa çalışır
}

void loop() {
// ana kodlar yazılır ve sürekli (tekrarlamalı) çalışır
}
```

Görüldüğü gibi Arduino programlarında temel olarak iki fonksiyon vardır:

- *setup()*: Arduino çalışmaya başladıktan sonra (enerji verildiğinde veya yeniden başlatıldığında) bir kez çağrılmakta ve içindeki işlemler yürütülmektedir. Burada genellikle programlardaki başlangıç ayarları (değişkenler, pin modları, seri iletişim, kütüphaneler vb.) yapılmaktadır.
- *loop()*: Kelime anlamı "döngü" olan bu fonksiyon içindeki işlemler, Arduino çalışmaya devam ettiği - yeniden başlatılmadığı - enerjisi kesilmediği sürece yürütülür. Buraya ana/esas işlemler yazılmaktadır.

Arduino program yazımının temel özellikleri, aşağıdaki gibi özetlenebilir:

- Program yazımları belirli kalıpta, bloklar halinde olur.
- Bloklar, { } parantezleri ile oluşturulur.
- Komutlar aynı veya alt alta satırlara yazılabilirler.
- Tüm komutlar, noktalı virgül (;) ile biter. Yalnız blok başlatan ifadelerden sonra noktalı virgül kullanılmaz.
- Programda kullanılan tüm değişkenler ve bilgi tipleri bildirilir.
- Programın başında kütüphaneler aktifleştirilir/çağrılır.
- Açıklamalar "//" (tek satır açıklama) ve "/* */ " (birden fazla satır açıklama) ile yazılır.
- Büyük-küçük harf ayrımı vardır (karakter duyarlıdır)

2. Veri tipleri, sabitler, bildirimleri ve dönüşümleri

Arduino programlarında kullanılabilecek veri tipleri ile sabitler ve bu tipler arasındaki dönüşümler sırasıyla Tablo 16.3-16.5'te özetlenmekte ve bildirimleri

```
veri_tipi  değişken adı ;
veri_tipi değişken1 , değişken2 ;
veri_tipi  değişken_adı = değer ;
const veri_tipi değişken_adı = değer;
```

şeklinde yapılmaktadır.

Tablo 16.3: Arduino veri tipleri

Veri tipi	Açıklama	En küçük	En büyük	Bellek (Byte)
boolean	Mantıksal veri (true/false)			
char	Tek karakter			1
unsigned char	byte ile aynı	0	255	1
byte	0-255 arası işaretsiz tamsayı	0	255	1
int	Tamsayı	-2^{15}	$2^{15}-1$	2
unsigned int	İşaretsiz tamsayı	0	$2^{16}-1$	2
word	İşaretsiz tamsayı	0	$2^{16}-1$	2
long	Uzun tamsayı	-2^{31}	$2^{31}-1$	4
unsigned long	İşaretsiz uzun tamsayı	0	$2^{32}-1$	4
short	Kısa tamsayı	-2^{15}	$2^{15}-1$	2
float	Ondalıklı sayı	$-3,4028235.10^{38}$	$3,4028235.10^{38}$	4
double	Ondalıklı sayı			4 veya 8

Tablo 16.4: Arduino'daki sabitler

Sabit	Değerleri
Mantıksal	true , false
Pin seviyesi	HIGH , LOW
Pin veri yönü	INPUT , OUTPUT

Tablo 16.5: Veri tipi dönüşümleri

Sabit	Değerleri
char()	*Karaktere*
byte()	*Tamsayıya*
int()	*Tamsayıya*
word()	*Tamsayıya*
long()	*Tamsayıya*
float()	*Ondalıklı sayıya*

3. İşlemler

Arduino programlarında kullanılabilecek matematiksel, karşılaştırma, mantıksal(bitsel), birleşik ve işaretçi (pointer) operatörler sırasıyla Tablo 16.6-16.10'da verilmektedir.

Tablo 16.6: Matematiksel operatörler

Operatör	İşlem
+	*Toplama*
-	*Çıkarma*
*	*Çarpma*
/	*Bölme*
%	*Mod alma*

Tablo 16.7: Karşılaştırma operatörleri

Operatör	İşlem
==	*Eşit*
!=	*Eşit değil*
>	*Büyük*
<	*Küçük*
>=	*Büyük eşit*
<=	*Küçük eşit*

Tablo 16.9: Birleşik operatörler

Operatör	İşlem
++	*Bir arttır*
--	*Bir azalt*
+=	*Topla-aktar*
-=	*Çıkar-aktar*
*=	*Çarp-aktar*
/=	*Böl-aktar*
%=	*Mod-aktar*
&=	*VE-aktar*
|=	*VEYA-aktar*

Tablo 16.8: Mantıksal/bitsel operatörler

Operatör	İşlem
&	*VE*
|	*VEYA*
~	*DEĞİL*
^	*Özel VEYA*
>>	*Bir bit sağa kaydır*
<<	*Bir bit sola kaydır*

Tablo 16.10: İşaretçi operatörler

Operatör	İşlem
*	*Adresteki değer (de-referans)*
&	*Referans*

4. Karar ve döngü yapıları

Arduino'daki karar ve döngü yapıları diğer dillerdeki gibidir (C/C++/C#/Java) (Tablo 16.11).

Tablo 16.11: Karar ve döngü yapıları

Karar yapısı	Döngü yapısı	Döngü kontrol

`if`
`if-else`
`switch-case`
	`for`
	`while`
	`do-while`
		`break`
		`continue`

5. Bazı fonksiyonlar

Arduino program yazımlarında yararlanılabilecek bazı temel fonksiyonlar, Tablo 16.12'de özetlenmektedir.

Tablo 16.12: Bazı fonksiyonlar

Fonksiyon	Kullanım şekli	Açıklama
min	**`min`**`(a, b)`	İki sayıdan (a, b) küçük olanı verir
max	**`max`**`(a, b)`	İki sayıdan (a, b) büyük olanı verir
abs	**`abs`**`(a)`	Sayının (a) mutlak değerini verir
sqrt	**`sqrt`**`(a)`	Sayının (a) karekökünü verir
pow	**`pow`**`(a, b)`	Üs (a^b) hesaplar
sin	**`sin`**`(a)`	Radyan cinsinden açının sinüsünü hesaplar
cos	**`cos`** `(a)`	Radyan cinsinden açının kosinüsünü hesaplar
tan	**`tan`**`(a)`	Radyan cinsinden açının tanjantını hesaplar
random	**`random`**`(max)` **`random`**`(min,max)`	Rastgele sayı üretir
delay	**`delay`**`(süre)`	ms cinsinden program akışını bekletir.
delayMicroseconds	**`delayMicroseconds`**`(süre)`	Mikrosaniye cinsinden program akışını bekletir.

6. Giriş/Çıkış fonksiyonları

Arduino'da kullanılabilecek analog ve sayısal giriş/çıkış fonksiyonları Tablo 16.13-16.14'te verilmektedir.

Tablo 16.13: Analog giriş/çıkış fonksiyonları

Fonksiyon	Kullanım şekli	Açıklama
analogReference	**`analogReference`**`(tip)` tip: DEFAULT *(5 V)* EXTERNAL *(Aref girişine 0-5 V)*	Analog giriş için kullanılan referans gerilimini ayarlar.
analogWrite	**`analogWrite`**`(pin no,değer)` değer: duty cycle (0-255)	Pine değeri (PWM dalga) yazar/gönderir.
analogRead	`değer` **`=`** **`analogRead`**`(pin no)`	Pinden değeri okur/alır. *16 kanal, 10 bit ADC [0 V -> 0 , 5 V -> 1023]*

Tablo 16.14: Sayısal giriş/çıkış fonksiyonları

Fonksiyon	Kullanım şekli	Açıklama
pinMode	**pinMode**(*pin no,mod*) mod: INPUT, OUTPUT, INPUT_PULLUP	Pini giriş-çıkış için ayarlar.
digitalWrite	**digitalWrite**(*pin no,değer*) değer: HIGH, LOW	Pine değeri yazar/gönderir.
digitalRead	*değer* = **digitalRead**(*pin no*)	Pinden değeri (HIGH, LOW) okur/alır.

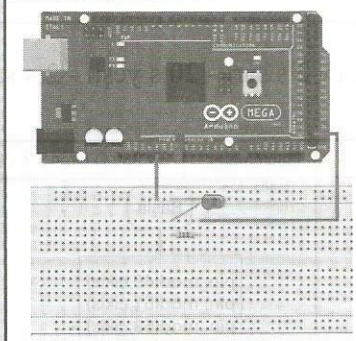
Örnek-16.7: Aşağıdaki bağlantıya göre LED'in 1 s aralıklarla yanıp sönmesini sağlayan program.

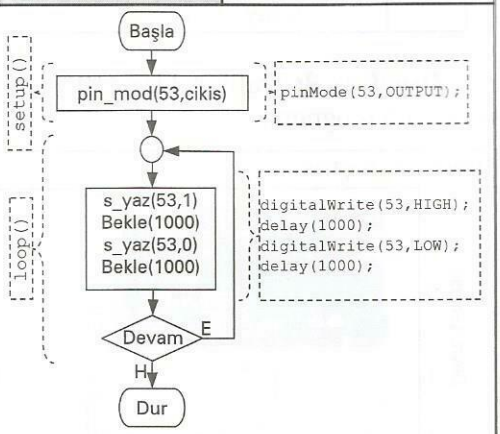
Çıkış için ayarlanan 53 numaralı pine bağlı LED'in yanması için bu pinin gerilim seviyesinin "YÜKSEK" (HIGH), yani +5 V; sönmesi için de "DÜŞÜK" (LOW), yani 0 V yapılması gerekmektedir.

Örnek LED yakıp söndürme

Malzemeler 1 x 330 Ω direnç, 1 x LED

Devre şeması





Kod

```
// LED yakıp söndürme
void setup() {
  pinMode(53,OUTPUT);
}
void loop() {
  digitalWrite(53,HIGH);
  delay(1000);
  digitalWrite(53,LOW);
  delay(1000);
}
```

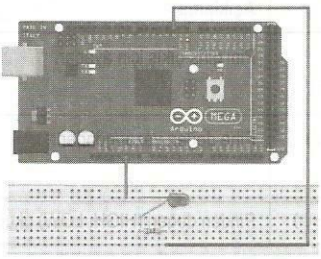
Örnek-16.8: Aşağıdaki bağlantıya göre LED'in parlaklığını (ışık şiddetini) değiştiren (arttırıp azaltan) program.

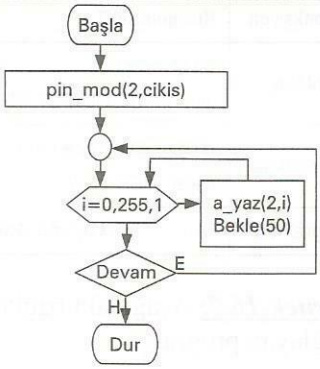
LED'in ışık şiddetini (parlaklığını) değiştirme, uygulanan gerilimin değişimiyle sağlanmaktadır. Arduino kartta "Darbe Genişlik Modülasyon" lu (PWM) çıkış sağlayan pinler mevcuttur. 0-255 arası gönderilen tamsayı değerler, bu pinlerde 0-5 V çıkış oluşturmaktadırlar.

Örnek PWM ile LED'in ışık şiddetini değiştirme

Malzemeler 1 x 330 Ω direnç, 1 x LED

Devre şeması





Kod

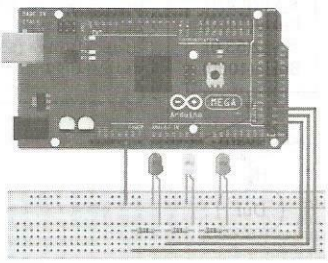
```
// PWM ile LED'in ışık şid-
detini değiştirme

void setup() {
  pinMode(2,OUTPUT); }

void loop() {
 for(int i=0;i<256;i++)
   { analogWrite(2,i);
     delay(50); } }
```

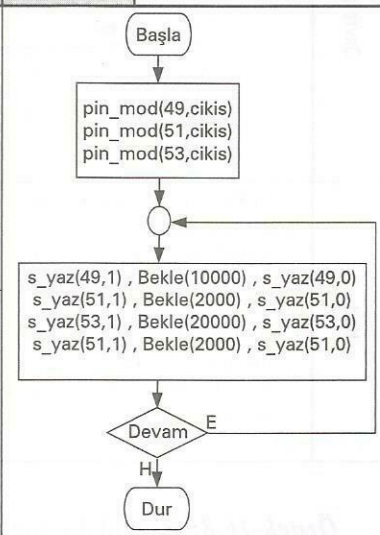
Örnek-16.9: Trafik ışıkları (10 s kırmızı - 2 s sarı - 20 s yeşil - 2 s sarı) simülasyon programı.

Örnek Trafik ışıkları

Malzemeler 3 x 330 Ω direnç, 3 x LED

Devre şeması

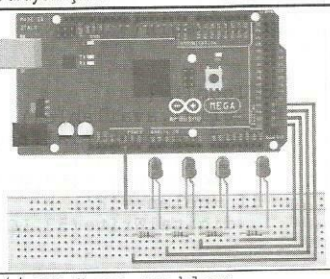
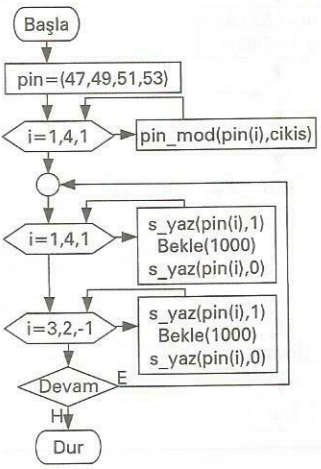




Kod

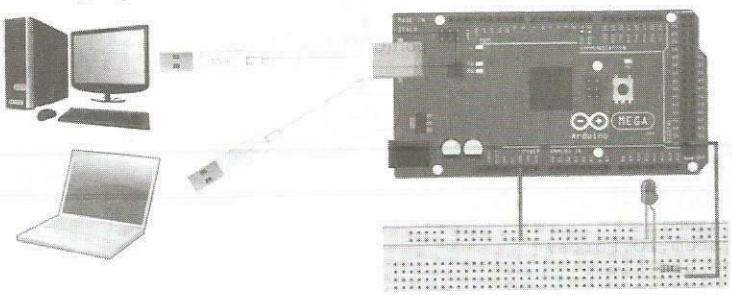
```
// Trafik ışıkları

void setup() {
 pinMode(49,OUTPUT);
 pinMode(51,OUTPUT);
 pinMode(53,OUTPUT); }

void loop() {
digitalWrite(49,HIGH);delay(10000);
digitalWrite(49,LOW);
digitalWrite(51,HIGH);delay(2000);
digitalWrite(51,LOW);
digitalWrite(53,HIGH);delay(20000);
digitalWrite(53,LOW);
digitalWrite(51,HIGH);delay(2000);
digitalWrite(51,LOW); }
```

Örnek-16.10: Yürüyen ışıklar programı.

Örnek	Yürüyen ışıklar	Malzemeler	4 x 330 Ω direnç, 4 x LED
Devre şeması			

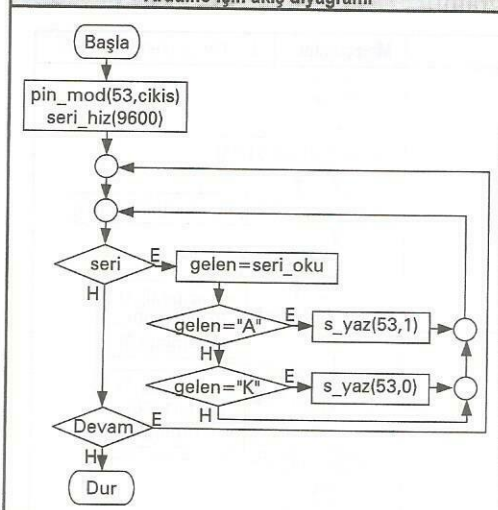
Kod	`// Yürüyen ışıklar`
	`byte pin[]={47,49,51,53};`
	`int i,t=1000;`
	`void setup() {`
	`for(i=0;i<4;i++)`
	`pinMode(pin[i],OUTPUT);`
	`}`
	
	`void loop() {`
	`for(i=0;i<4;i++)`
	`{ digitalWrite(pin[i],HIGH);`
	`delay(t);`
	`digitalWrite(pin[i],LOW); }`
	`for(i=2;i>0;i--)`
	`{ digitalWrite(pin[i],HIGH);`
	`delay(t);`
	`digitalWrite(pin[i],LOW); }`
	`}`		

Örnek-16.11: Bilgisayardan Arduino'ya veri gönderme uygulaması: LED yakıp söndüren program.



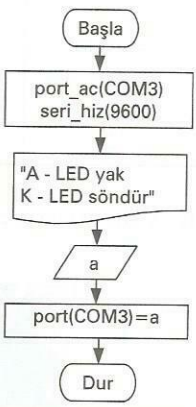
Bilgisayar ile Arduino'yu haberleştirebilmek için her iki tarafa da uygun programların yazılması gerekmektedir. Öncelikle Arduino tarafındaki programı hazırlayalım. Arduino kartı, bilgisayardan gelecek olan veriyi sürekli dinlemelidir/beklemelidir. Eğer bilgisayardan gelen veri, örneğin "A" harfi ise; 53 numaralı pininin gerilimini +5 V yapmalı, böylece LED'in yanmasını sağlamalıdır. Benzer şekilde bilgisayardan gelen veri, örneğin "K" harfi ise; 53 numaralı pininin gerilimini 0 V yaparak sönmesini sağlamalıdır. Buna karşın bilgisayar da Arduino kartın bağlı olduğu portu seri iletişim için açıp ayarlarını yaparak; LED'i yakmak için "A", söndürmek için de "K" harfini göndermelidir.

Arduino için akış diyagramı



Arduino kodu

```
// PC - Arduino LED yak-söndür
void setup() {
 pinMode(53,OUTPUT);
 Serial.begin(9600);
}

void loop() {
 if (Serial.available())
  { char gelen=Serial.read();
    if (gelen=='A')
      digitalWrite(53,HIGH);
    if (gelen=='K')
      digitalWrite(53,LOW); }
}
```



C#

```
// PC-Arduino LED yak-söndür
using System;
using System.IO.Ports;
namespace ornek1611
{ class Program
   { static void Main(string[] args)
      { SerialPort port;port=new SerialPort("COM3", 9600);port.Open();
        Console.WriteLine("A - LED yak\nK - LED söndür");
        Console.Write("İşleminiz: ");
        String x=Console.ReadLine().ToUpper(); port.Write(x);
        Console.ReadLine(); port.Close(); }}}
```

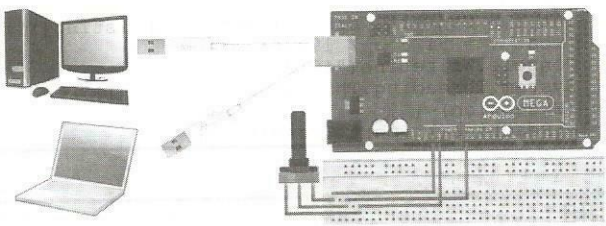
MATLAB

```
% PC-Arduino LED yak-söndür
clear all; clc; port=serial('COM3','BaudRate',9600); fopen(port);
fprintf('A - LED yak\nK - LED söndür\n');
x=upper(input('İşleminiz: ','s')); fprintf(port,x); fclose(port);
```

PYTHON

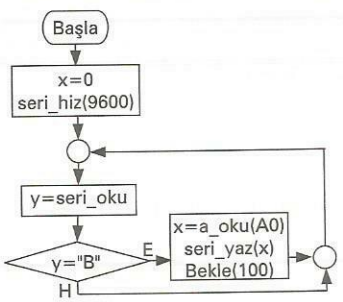
```
# PC - Arduino LED yak-söndür
import serial
port=serial.Serial('COM3',9600)
print("A - LED yak\nK - LED söndür")
x=input("İşleminiz: "); x=x.upper(); port.write(x.encode())
port.close()
```

Örnek-16.12: Arduino'dan bilgisayara veri okuma/alma uygulaması: 10 kΩ potansiyometre üzerindeki gerilimi okuyan program.



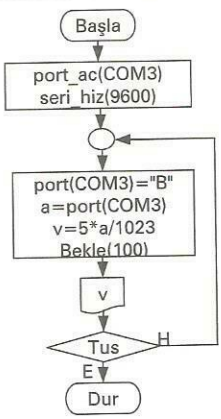
Arduino kartı, bilgisayardan gelecek olan başlama işaretiyle analog giriş pinine bağlı potansiyometredeki gerilimi okumalı ve bu değeri aynı seri iletim hattı üzerinden bilgisayara göndermelidir. Bilgisayar da Arduino karta başlama işaretini göndermeli ve gelecek veriyi okumalıdır.

Arduino için akış diyagramı



Arduino kodu

```
// PC - Arduino 10k potansiyometreden
gerilim okuma
int x=0;
void setup() {
  Serial.begin(9600);
}
void loop() {
        char y=Serial.read();
        if (y=='B')
           { x=analogRead(A0);
             Serial.println(x);
             delay(100); }
}
```



C#

```
// PC - Arduino 10k potansiyometreden gerilim okuma
using System;
using System.IO.Ports;
namespace ornek1612
{ class Program
   { static void Main(string[] args)
      { SerialPort port;
        port=new SerialPort("COM3", 9600); port.Open();
        System.Threading.Thread.Sleep(100);
        try
         { while (1==1)
            { port.Write("B");
              int a = Convert.ToInt32(port.ReadExisting());
              float v = 5f * a / 1023f;
              Console.WriteLine("{0}", v);
              System.Threading.Thread.Sleep(100); }}
        catch (Exception) { }
        Console.ReadLine(); port.Close(); }}}
```

MATLAB

```
% PC - Arduino 10k potansiyometreden gerilim okuma
clear all; clc; port=serial('COM3','BaudRate',9600); fopen(port);
while (1)
    fprintf(port,'B'); a=fscanf(port,'%d'); v=5*a/1023;
    fprintf('%f\n',v); pause(0.1);
end
fclose(port);
```

PYTHON

```
# PC - Arduino 10k potansiyometreden gerilim okuma
import serial
import time
port=serial.Serial('COM3',9600)
time.sleep(2); a=0
while a<=1023:
    port.write("B".encode()); a=int(port.readline().rstrip())
    v=5*a/1023; print(v); time.sleep(0.1)
port.close()
```

SORULAR

1. Klavyeden herhangi bir tuşa basılıncaya kadar paralel porta bağlı LED'leri önce ortadan iki yana, sonra da iki yandan ortaya doğru 1 s aralıklarla yakan programın akış diyagramını çizip kodlayınız.

2. Klavyeden herhangi bir tuşa basılıncaya kadar paralel porta bağlı LED'leri soldan sağa doğru doldurarak (önce bir, sonra iki, sonra üç ... tane yanacak) 1 s aralıklarla yakan programın akış diyagramını çizip kodlayınız.

3. Klavyeden girilen 0-15 arası tamsayının 16 tabanındaki karşılığını paralel porta bağlı yedi parçalı göstergede yakan programın akış diyagramını çizip kodlayınız.

4. Arduino kartın sayısal çıkış pinlerine bağlı 8 LED'i önce soldan sağa, sonra da sağdan sola 1 s aralıklarla yakan programın akış diyagramını çizip kodlayınız.

5. Arduino kartın sayısal çıkış pinlerine bağlı 7 parçalı göstergede 1 s aralıklaral 0-9 arası sayan programın akış diyagramını çizip kodlayınız.

6. Klavyeden girilen 0-9 arası rakamı, Arduino kartın sayısal çıkış pinlerine bağlı 7 parçalı göstergede yakan programın akış diyagramını çizip kodlayınız.

7. Klavyeden girilen 0-15 arası sayının onaltı tabanındaki karşılığını, Arduino kartın sayısal çıkış pinlerine bağlı 7 parçalı göstergede yakan programın akış diyagramını çizip kodlayınız.

8. Arduino kartın analog girişine bağlı LM35 sensöründen 10 s aralıklarla sıcaklığı okuyup ekranda gösteren programın akış diyagramını çizip kodlayınız.

9. Arduino karta bağlı adım motorunu istenilen sayıda ve yönde döndüren programın akış diyagramını çizip kodlayınız.

Ek - A
ASCII Kodları

Tablo A.1: ASCII tablosu

000 (nul)	016 ► (dle)	032 sp	048 0	064 @	080 P	096 `	112 p
001 ☺ (soh)	017 ◄ (dc1)	033 !	049 1	065 A	081 Q	097 a	113 q
002 ☻ (stx)	018 ↕ (dc2)	034 "	050 2	066 B	082 R	098 b	114 r
003 ♥ (etx)	019 ‼ (dc3)	035 #	051 3	067 C	083 S	099 c	115 s
004 ♦ (eot)	020 ¶ (dc4)	036 $	052 4	068 D	084 T	100 d	116 t
005 ♣ (enq)	021 § (nak)	037 %	053 5	069 E	085 U	101 e	117 u
006 ♠ (ack)	022 ▬ (syn)	038 &	054 6	070 F	086 V	102 f	118 v
007 • (bel)	023 ↨ (etb)	039 '	055 7	071 G	087 W	103 g	119 w
008 ◘ (bs)	024 ↑ (can)	040 (	056 8	072 H	088 X	104 h	120 x
009 (tab)	025 ↓ (em)	041)	057 9	073 I	089 Y	105 i	121 y
010 (lf)	026 (eof)	042 *	058 :	074 J	090 Z	106 j	122 z
011 ♂ (vt)	027 ← (esc)	043 +	059 ;	075 K	091 [	107 k	123 {
012 ♀ (np)	028 ∟ (fs)	044 ,	060 <	076 L	092 \	108 l	124 |
013 (cr)	029 ↔ (gs)	045 -	061 =	077 M	093]	109 m	125 }
014 ♫ (so)	030 ▲ (rs)	046 .	062 >	078 N	094 ^	110 n	126 ~
015 ☼ (si)	031 ▼ (us)	047 /	063 ?	079 O	095 _	111 o	127 ⌂

Tablo A.2: Genişletilmiş ASCII tablosu

128 Ç	143 Å	158 Ş	172 ¼	186 ‖	200 ╚	214 Í	228 õ	242 ▯
129 ü	144 É	159 ş	173 ¡	187 ╗	201 ╔	215 Î	229 Õ	243 ¾
130 é	145 æ	160 á	174 «	188 ╝	202 ╩	216 Ï	230 µ	244 ¶
131 â	146 Æ	161 í	175 »	189 ¢	203 ╦	217 ┘	231 ▯	245 §
132 ä	147 ô	162 ó	176 ░	190 ¥	204 ╠	218 ┌	232 ×	246 ÷
133 à	148 ö	163 ú	177 ▒	191 ┐	205 ═	219 ▉	233 Ú	247 ¸
134 å	149 ò	164 ñ	178 ▓	192 └	206 ╬	220 ▄	234 Û	248 °
135 ç	150 û	165 Ñ	179 |	193 ┴	207 ¤	221 ¦	235 Ù	249 ¨
136 ê	151 ù	166 Ğ	180 ┤	194 ┬	208 º	222 Ì	236 ì	250 ·
137 ë	152 İ	167 ğ	181 Á	195 ├	209 ª	223 ▀	237 ÿ	251 ¹
138 è	153 Ö	168 ¿	182 Â	196 —	210 Ê	224 Ó	238 ¯	252 ³
139 ï	154 Ü	169 ®	183 À	197 ┼	211 Ë	225 ß	239 ´	253 ²
140 î	155 ø	170 ¬	184 ©	198 ã	212 È	226 Ô	240 -	254 ■
141 ı	156 £	171 ½	185 ╣	199 Ã	213 ▯	227 Ò	241 ±	255
142 Ä	157 Ø							

Kaynaklar

http://java.sun.com/

http://mathworld.wolfram.com/

http://msdn.microsoft.com/

http://wikipedia.org/

http://www.cplusplus.com/

http://www.mathworks.com/

https://www.arduino.cc/

https://www.computerhope.com/

https://www.python.org/

https://www.tutorialspoint.com/index.htm

https://www.wikibooks.org/

Turbo C++ (v3.0) Documentation, Borland International, Inc., 1990-1992.

Vatansever, F., "Algoritma Geliştirme ve Programlamaya Giriş", Ders Notları, Sakarya, 1999.

Vatansever, F., "Borland/Turbo Pascal Programlama Dili", Seçkin Yayıncılık, Ankara, 2009.

Vatansever, F., "C Programlama Dili", Ders Notları, Sakarya, 2001.

Vatansever, F., "İleri Programlama Uygulamaları", Seçkin Yayıncılık, Ankara, 2006.

Vatansever, F., "MATLAB ile Temel Programlama", Ders Notları, Bursa, 2011.

Dizin

- · 32, 54, 120
-- · 120
! · 120
!= · 120
" · 115, 120
""" · 115
· 115
% · 115, 120
%= · 120
& · 120
&& · 120
* · 32, 54, 120
** · 120
**= · 120
*/ · 115
*= · 120
. · 32, 54
... · 115
.net framework · 102
/ · 32, 54, 120
/* · 115
// · 115, 120
//= · 120
/= · 120
? · 159
?: · 120
@ · 120
^ · 32, 54, 120
` · 54
{ · 115
| · 120
|| · 120
} · 115
~ · 120
~= · 120
+ · 32, 54, 120
++ · 120
+= · 120
+3 kodu · 527
< · 38, 54, 120
<< · 120
<= · 38, 54, 120
<> · 38, 54
= · 38, 54
-= · 120
== · 120
> · 38, 54, 120
>= · 38, 54, 120
>> · 120
π · 409
1,1,2, · 512
5'te 2 (74210) kodu · 527

A

abs · 164, 166, 622
acos · 165
acosh · 165
adfgvx şifreleme · 564
affine şifreleme · 554
ağırlıklı aritmetik ortalama · 305
ağırlıklı geometrik ortalama · 305
ağırlıklı harmonik ortalama · 305
aiken (2421) kodu · 528
akış diyagramı · 52, 71
aktarma · 58
alfasayısal veri · 55
algorithms · 99
algoritma · 52
algorizm · 52
alt program · 21
alt üçgen matris · 337
altın açı · 453
altın oran · 412, 431
altın üçgen · 413
amaç program · 30
ana program · 21
analogread · 622
analogreference · 622
analogwrite · 622
anders hejlsberg · 102
angle · 164
ans · 109
ansi · 29
applet · 107
ara asal sayı · 457
aralarında asal sayılar · 450
araya eklemeli sıralama · 477
arduino · 618
aritmetik ortalama · 254
aritmetik sayı · 427
aritmetik-geometrik ortalama · 256, 450
aritmetik-harmonik ortalama · 256
armstrong sayısı · 423
artan döngü · 73
artık yıl · 208
asal çarpan · 417
asal sayı · 414
asal toplayanlar · 450
asal üçüz · 456
ascii · 29, 533, 631
asin · 165
asinh · 165
atama · 58
atan · 165
atanh · 165
atbash şifreleme · 559
atlamalı/blok arama · 507
atof · 166
atoi · 166
awt · 107
azalan döngü · 73

B

backgroundcolor · 164
baconian şifreleme · 565
bağdaşık sayılar · 424
bailey-borwein-plouffle formülü · 450
bar (çubuk) kodu · 541
barkod · 541
base2dec · 166
bazal metabolizma hızı · 207
bcd (8421) kodu · 523
beaufort şifreleme · 565
beden kütle/kitle indeksi · 205
beden yüzey alanı · 205
bell sayıları · 430, 442
bell üçgeni · 442
bellard formülü · 450
bellek · 27
bidirectional bubble sort · 471
bifid şifreleme · 565
bileşik sayı · 454
bilgisayar · 20
bin · 166
bin2dec · 166
binary · 521
binary search · 508
binary_search · 511
binarysearch · 511
binom açılımı · 407
binom katsayıları · 240, 445
birim matris · 327
birleştirme/birleştirmeli sıralama · 489
bisect · 511
bit · 29

bjarne stroustrup · 97
blum tamsayısı · 457
bogo sort · 488
boolalpha · 125, 134
boole cebri · 40
boolean · 621
boolean cebri · 40
böl ve üstesinden gel · 21
böl ve yönet · 21
break · 155, 622
brick sort · 472
brothers formülü · 408
brute-force search · 215
bubble sort · 467
bucket sort · 496
byte · 29, 90, 91, 621

C

c · 92
c# · 102
c++ · 97
caesar cipher · 552
camel case/casing/notation · 56
carol sayıları · 454
case · 157
catalan sayıları · 454
ceil · 166
ceiling · 166
center · 145
char · 87, 89, 90, 91, 166, 621
charat · 127
chen asalı · 419
chr · 166
chudnovsky algoritması · 450
cin · 125
circle · 383
class · 104, 107, 112
classdef · 110
clear · 164, 384
cleardevice · 384
cleve moler · 108
close · 396
closegraph · 384
clrscr(· 164
cls · 164
cmyk · 544, 549
cocktail shaker sort · 471
cocktail sort · 471
color · 164
comb sort · 474
common language infrastructure · 102
complex · 99, 164
conio · 93
conj · 164
conjugate · 164
consolekeyinfo · 126
const · 95, 101
contains · 166
continue · 155, 622
copy · 166
cos · 165, 622
cosh · 165
cosinus bağıntısı · 176
counting sort · 495
cout · 133
cprintf · 133
cputs · 133
cramer yöntemi · 594
ctype · 93
cullen asal sayıları · 453
cullen sayısı · 422
cüce sıralama · 482

Ç

çalışma(çalıştırma) zamanı hatası · 31
çevrim · 72
çok koşullu form · 155

D

dec · 125, 134
dec2base · 166
dec2bin · 166
dec2hex · 166
decimal · 87, 90, 521
dedekind sayıları · 455
def · 112
default · 157
define · 94, 100
degrees · 164
degtorad · 164
değişken · 56
del · 111
delannoy sayısı · 437
delay · 167, 622
delaymicroseconds · 622
dennis ritchie · 92
derleme zamanı hatası · 31
derleyici · 30
detectgraph · 382
determinant · 349
diferansiyel denklem · 599
digitalread · 623
digitalwrite · 623
dikdörtgenler yöntemi · 578
disp · 139
div · 166
divide · 166
dizi · 245
do – while · 149
dobiński formülü · 444
doğrusal (lineer) enterpolasyon · 569
doğrusal/sıralı arama · 507
donanım · 26
dos · 93
dost sayılar · 424
dosya · 395
double · 88, 89, 90, 91, 166, 621
do-while · 622
döngü · 61, 72
drawellipse · 383
drawline · 382
drawoval · 383
drawrect · 383
drawrectangle · 383
düşük seviyeli dil · 29
düzeltilmiş euler yöntemi · 602

E

e · 164, 405
ebob · 233
ecvt · 166
eksik sayı · 425
el-harezmi · 52
en küçük kareler yöntemi · 611
end · 115, 142
endl · 134
ends · 134
enterpolasyon · 569
enterpolasyon arama · 511
enum · 100, 105
erase · 166
eşikleme · 369
etkin değer (rms) · 267
euler · 451
euler üçgeni · 429
euler yöntemi · 599
eulerian sayısı · 428
exhaustive search · 215
exp · 165

F

factorial · 166
faktorion · 429
faktoriyel · 190, 229

false · 39, 621
fclose · 396
fcvt · 166
fermat sayısı · 422
fibonacci · 431, 455
fibonacci asal sayıları · 435
fileaccess · 396
filemode · 396
filereader · 396
filestream · 396
filewriter · 396
fillrect · 384
final · 108
find · 166
findinline · 127
fixed · 134
float · 88, 89, 90, 91, 621
floor · 166
floyd üçgeni · 440
fopen · 396
for · 147, 151, 152, 622
foreach · 150
foregroundcolor · 164
format · 143
fprintf · 139
françois viète · 451
frekans · 502
frekans modülasyonu · 390
friedman sayısı · 453
function · 95, 110

G

gama iyileştirme/düzeltme · 372
gb · 29
genişletilmiş ascii · 29, 632
genlik modülasyonu · 389
geometrik ortalama · 254
geometrik-harmonik ortalama · 256
geri fark · 573
get · 126
getch · 124
getchar · 124
getche · 124
getline · 126
gets · 124
global · 109
gnome sort · 482
goto · 167
gotoxy · 164
grad · 173
graphics · 93
gray kodu · 530
gri · 546
guido van rossum · 111
güçlü sayılar · 454
gümüş oran · 414, 437

H

hane/basamak sıralama · 495
hanoi kuleleri · 241
harmonik bölücü sayısı · 426
harmonik ortalama · 254
harris-benedict denklemi · 208
harshad sayıları · 453
heap sort · 499
heinz ortalama · 305
heptagonal · 455
hermite sayısı · 430
heron formülü · 177
heronian ortalama · 305
heterometrik sayı · 427
heun yöntemi · 602
hex · 125, 134, 166, 545
hex2dec · 166
hexadecimal · 87
hexagonal · 455
hızlı sıralama · 492
high · 621
hilbert asalları · 457
hilbert sayısı · 423
hill şifreleme · 559
histogram · 370
horner yöntemi · 263
hsl · 544, 550, 551
hsv · 544, 546, 548

İ

iban · 542
ideal kilo · 205
identric ortalama · 305
idoneal sayılar · 454
if · 622
if-elif-else · 162
if-else · 622
if-else if-else · 155
if-elseif-else · 160
ikili arama · 508
ikili kod · 28
ikiz asallar · 419
ileri fark · 573
imag · 164
imei · 542
import · 107, 112, 115
impulse (birim dürtü) cevabı · 264
in · 150, 153
include · 93, 99, 115
indeks erişimli dosyalar · 395
indexof · 166
inf · 164
initgraph · 382
inline · 167
inp · 167
inport · 167
input · 128, 129, 621
inputdlg · 128
insertion sort · 477
int · 87, 89, 90, 91, 166, 621
int16 · 90
int2str · 166
int32 · 90
int64 · 90
integral · 578
internal · 134
interpolation search · 511
io · 107
iostream · 99
isaac newton · 451
isimlendirme kuralları · 56
işaretçi · 58
işlem öncelik sıraları · 32
itoa · 166

J

jacobsthal sayıları · 458
jacobsthal-lucas sayıları · 458
james gosling · 105
java · 105
java sanal makinesi · 106
john wallis · 409
jump/block search · 507

K

kabarcık sıralama · 467
kabuk sıralama · 481
kaprekar sabiti · 459
kaprekar sayısı · 457
kare matris · 309
kare üçgensel (üçgen karesel) sayı · 439
karnaugh haritaları · 42
kartezyen · 271
kaynak program · 30
kb · 29
kbhit · 124

kepler üçgeni · 453
keyavailable · 127
kinetik · 178
kirişler (regula-false) yöntemi · 586
kodlama · 20
kokteyl sıralama · 471
kombinasyon · 188
komut · 20
kontraharmonik ortalama · 254
kontrast iyileştirme · 371
konvolüsyon · 264
kova sıralama · 496
köşegen matris · 337
kuzen asallar · 457
kynea sayıları · 454

L

lagrange enterpolasyonu · 570
lah sayıları · 459
lang · 107
left · 134
leibnitz- gregory–madhava serisi · 450
len · 166
length · 166
leonardo sayıları · 458
leyland sayısı · 423
line · 382
linear/sequential search · 507
ljust · 145
log · 165
log10 · 165
logaritmik ortalama · 305
long · 88, 89, 90, 91, 621
loop · 620
low · 621
lower · 166
lucas · 452
luhn algoritması · 543

M

m_e · 164
m_pi · 164
maclaurin serisi · 225
madhava of sangamagrama · 409
main · 97, 101, 102, 105, 115
makine dili · 28
makine kodu · 28
mantık hatası · 31
markov (markoff) sayısı · 427
math · 93
matlab · 108
matplotlib · 112
matris · 309
max · 166, 622
mb · 29
medyan · 502
merge sort · 489
merkezi fark · 573
merkezi işlem birimi · 27
mersenne asalları · 456
mersenne sayıları · 453
mifflin-st jeor denklemi · 208
min · 166, 622
mod · 166, 502
mod-10 algoritması · 543
mol sayısı · 178
monkey sort · 488
monte carlo yöntemi · 409
morse alfabesi · 538
mükemmel sayı · 420

N

namespace · 103
nan · 164
negafibonacci · 436
net · 107
new · 99
next · 127, 165
nextboolean · 127
nextbyte · 127
nextdouble · 127
nextfloat · 127
nextint · 127
nextline · 127
nextlong · 127
nextshort · 127
nibble · 29
nilakantha somayaji · 451
noboolalpha · 125, 134
noshowbase · 134
noshowpoint · 134
noshowpos · 134
noskipws · 125, 134
nouppercase · 134
num2str · 166
numpy · 112

O

obeb · 231, 233
oct · 125, 134, 166
oct2dec · 166
octagonal · 455
odd–even sort · 472
odd–even transposition sort · 472
ohm kanunu · 179
okek · 231
open · 396
operatör · 54
ord · 166
ore sayısı · 426
orta seviyeli dil · 29
ortalama değer (da/dc) · 267
otherwise · 161
otomatik kontrast · 371
otomorf sayılar · 457
outp · 167
outport · 167
output · 621
owen denklemi · 208

Ö

öklid algoritması · 233
ön koşullu döngü · 146

P

palindrom · 295, 428
palindromik · 295
pancake sort · 484
pankek sıralama · 484
parabolik (kuadratik) enterpolasyon · 569
paraboller yöntemi · 582
parçala ve fethet · 21
parçalı sıralama · 497
parity (eşlik) kodu · 529
parsefloat · 166
parseint · 166
pascal case/casing/notation · 56
pascal matrisi · 446
pascal üçgeni · 445
pause · 167
pell sayısı · 436
pen · 384
pentagonal · 455
permutation sort · 488
permütasyon · 301
permütasyon sıralama · 488
perrin sayıları · 460
phase · 164
pi · 164
piksel · 363, 381

piksel kopyalama · 373
pinmode · 623
pisagor teoremi · 174
pisagor üçlemesi · 441
plot · 382
pointer · 58
poligonsal sayılar · 455
polinom · 261
polybius kare şifreleme · 561
port · 615
porta şifreleme · 565
potansiyel · 178
pow · 165, 622
pratik sayılar · 457
print · 137, 142
printf · 130, 137
println · 137
prod · 166
program · 20
programlama dili · 28
proth sayıları · 453
putchar · 133
putpixel · 384
puts · 133
python · 111

Q

qsort · 166
quick sort · 492

R

radians · 164
radix sort · 495
radtodeg · 164
radyan · 173
rand · 165
randint · 165
random · 165, 622
random sort · 488
randomize · 165
randrange · 165
range · 154
rastgele erişimli dosya · 395
rastgele sıralama · 488
read · 127
readkey · 126
readline · 126
real · 164
rectangle · 383
rekürsif · 229
remove · 166
rené descartes · 66
replace · 166
return · 96, 101
reverse · 166
rgb · 364, 544, 545, 546, 548, 549, 550, 551
right · 134
ripple sort · 471
rjust · 145
romen rakamları · 534
rot13 · 554
round · 166
runge-kutta yöntemleri · 604

S

sabit · 58
sarrus yöntemi · 349
sayaç · 60
sayıcı · 60
sayıcılı döngü · 146
sayısal veri · 55
sayma sıralama · 495
sbyte · 90
scanf · 122
scanner · 127
schofield denklemi · 208
scientific · 134
scipy · 112
seçimli form · 155
seçme sıralama · 463
seidel yöntemi · 595
selection sort · 463
sep · 142
setbackground · 384
setbase · 134
setbkcolor · 384
setcolor · 384
setconsolecursorposition · 164
setconsoletextattribute · 164
setcursorposition · 164
setfill · 134
setpixel · 384
setprecision · 134
setup · 620
setw · 134
sezar şifreleme · 552
shaker sort · 471
shell sort · 481
short · 88, 89, 90, 91, 621
shotgun sort · 488
showbase · 134
showpoint · 134
showpos · 134
shuffle sort · 471
shuttle sort · 471
sıralı erişimli dosya · 395
signed · 88, 90
signum · 201
sihirli kare · 460
sihirli sabit · 460
simpson yöntemi · 582
sin · 165, 622
single · 90
sinh · 165
sistem yazılımları · 26
skaler çarpım · 253
skipws · 125, 134
sleep · 167
smith sayısı · 456
solinas asalları · 456
son koşullu döngü · 146
sonlu fark · 573
sort · 166
sprintf · 141
sql · 107
sqrt · 165, 622
srand · 165
standart sapma · 259
stdio · 93
stdlib · 93
stirling sayıları · 444
stirling üçgeni · 444
stolarsky ortalama · 305
stooge sort · 497
str · 166
str2double · 166
str2num · 166
strfind · 166
string · 91, 93
strlen · 166
strrep · 166
strrev · 166
strstr · 166
strtol · 166
struct · 95, 100, 104, 109
stupid sort · 482
substring · 166
sum · 166
switch · 157, 161
switch-case · 622
syms · 109
system · 164

T

taban aritmetiği · 517
tablo (quine-mccluskey) yöntemi · 42
tam bölenler · 183
tam form · 155

tan · 165, 622
tanh · 165
tanımlayıcı · 56
tarak sıralama · 474
tau · 164
tau sayısı · 426
taylor serisi · 225
tb · 29
teğetler (newton-raphson) yöntemi · 587
tek-çift sıralama · 472
ters matris yöntemi · 591
tetranacci · 452
textbackground · 164
textcolor · 164
thabit sayısı · 454
toascii · 166
tochar · 166
todegrees · 164
todouble · 166
toint32 · 166
tolower · 166
toradians · 164
tostring · 166
totient · 448
toupper · 166
transformatör · 180
transpoze · 324
tribonacci · 436
trimorf sayılar · 457
true · 39, 621
trunc · 166
truncate · 166
türev · 573
typedef · 94, 100

U

uint · 90
uint16 · 90
uint32 · 90
uint64 · 90
ulong · 90
undef · 94, 100
underscore case/notation · 56
unsigned · 88, 90
unsigned char · 621
unsigned int · 621
unsigned long · 621
upper · 166
upper (upperscore) case/notation · 56
uppercase · 134
ushort · 90
using · 103, 115
util · 107
uygulama yazılımları · 26

Ü

üçgensel sayı · 438
üst üçgen matris · 335
üstel faktöriyel · 240

V

vampir sayı · 456
varyans · 259
vectors · 99
veri · 55
veritabanı · 395
veriyol · 27
viète formülleri · 197
vigenère şifreleme · 562
void · 97

W

wagstaff asalları · 456
wallis çarpımı · 450
while · 148, 152, 154, 622
width · 134
william brouncker · 451
windows · 99
woodall sayıları · 453
word · 621
worpitzky sayıları · 454
write · 136
writeline · 136
ws · 125

Y

yağsız beden ağırlığı · 205
yamuklar yöntemi · 581
yarıasal sayı · 455
yarılama yöntemi · 583
yarılanma süresi · 179
yarım form · 155
yazılım · 26
yazım hatası · 30
yığın sıralama · 499
yinelemeli · 229
yorumlayıcı · 30
yüksek seviyeli dil · 29

Z

zfill · 145
zoom · 373, 374

algoritma geliştirme

ve programlamaya giriş

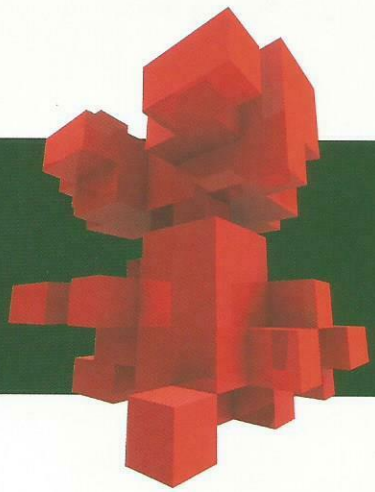
Doç. Dr. Fahri VATANSEVER

Yenilenmiş 13. baskısını yapan kitaba, okuyuculardan gelen yoğun istek üzerine Python ve Arduino da eklenmiştir. Kitap, daha önce programlama ile ilgili bilgisi bulunmayan okuyucuları da kapsayacak şekilde başlangıç seviyesinden ileri seviyelere kadar, adım adım "Algoritma Geliştirme" ve "Programlama Teknikleri"ni anlatmaktadır.

Kitapta farklı alanlara ait çok sayıda örnek problemin çözümüne ilişkin akış diyagramları çizilerek altı ayrı programlama dilinde (C, C++, C#, Java, MATLAB ve Python) kodlanmıştır. Okuyucuların "algoritma" ve "programlama" bilgilerini geliştirmeleri için konular çok sayıda çözümlü örnekler ve bölüm sonu soruları ile desteklenmiştir (375'i çözümlü, 680 örnek/problem)

Kitaptaki konuları, sırası ile okuyup uygulayan okuyucuya "algoritma geliştirme" ve "program yazma" tekniklerinin kazandırılması amaçlanmıştır.

- Programlamaya Giriş
- Algoritma
- Akış Diyagramı
- Programlama Dilleri
- Akış Diyagramından Kodlamaya Geçiş
- Genel Uygulamalar
- Bir Boyutlu Dizi Uygulamaları
- İki Boyutlu Dizi Uygulamaları
- Grafik Uygulamaları
- Dosyalama Uygulamaları
- Özel Sayı ve Şekiller
- Sıralama Algoritmaları
- Arama Algoritmaları
- Kodlama/Şifreleme Uygulamaları
- Sayısal Analiz Uygulamaları
- Donanım Uygulamaları



seçkin | Akademik ve Mesleki Yayınlar
Mustafa Kemal Mahallesi 2158. Sokak
No:13 Çankaya | ANKARA
Tel: (312) 435 30 30 Faks: (312) 435 24 72
www.seckin.com.tr seckin@seckin.com.tr

39,90 TL

ISBN 978 975 02 4430 8
9 789750 244308

Kategori: Bilgisayar
Konu: Programlama