Bilgisayar Programlama ve Algoritma (BÖLÜM I)

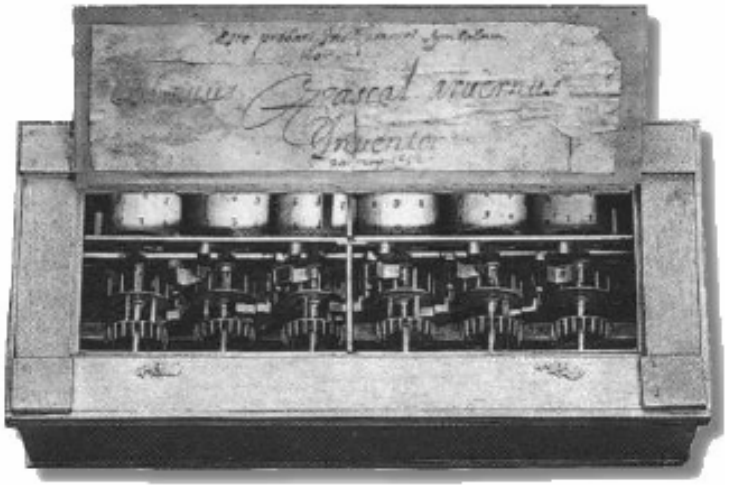
Dr. Gürcan SAMTAŞ

Bilgisayar

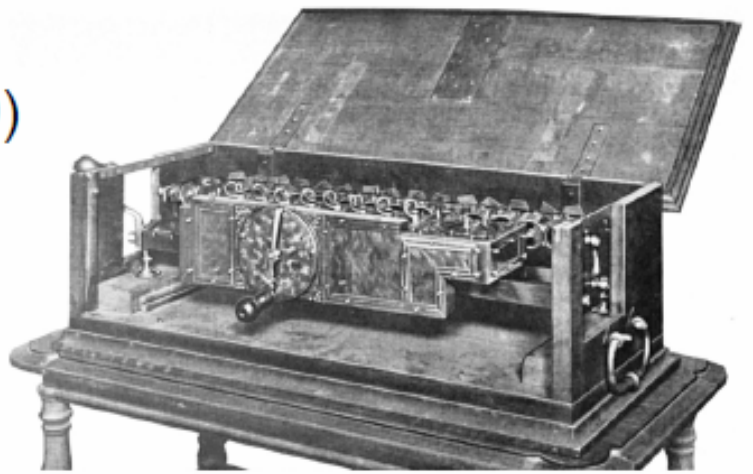
Verilen bilgileri saklayan, gerektiğinde bu bilgileri hızlı bir şekilde istenilen amaca uygun kullanmayı sağlayan/işleyen, mantıksal ve aritmetiksel işlemleri çok hızlı biçimde yapan bir makinedir. Bilgisayar terimi İngilizce “computer” kelimesinin dilimize çevrilmiş halidir. Bu terim de Latince “compurate” kelimesinden gelmektedir.

Bilgisayarların Gelişimi

Mekanik Çağ



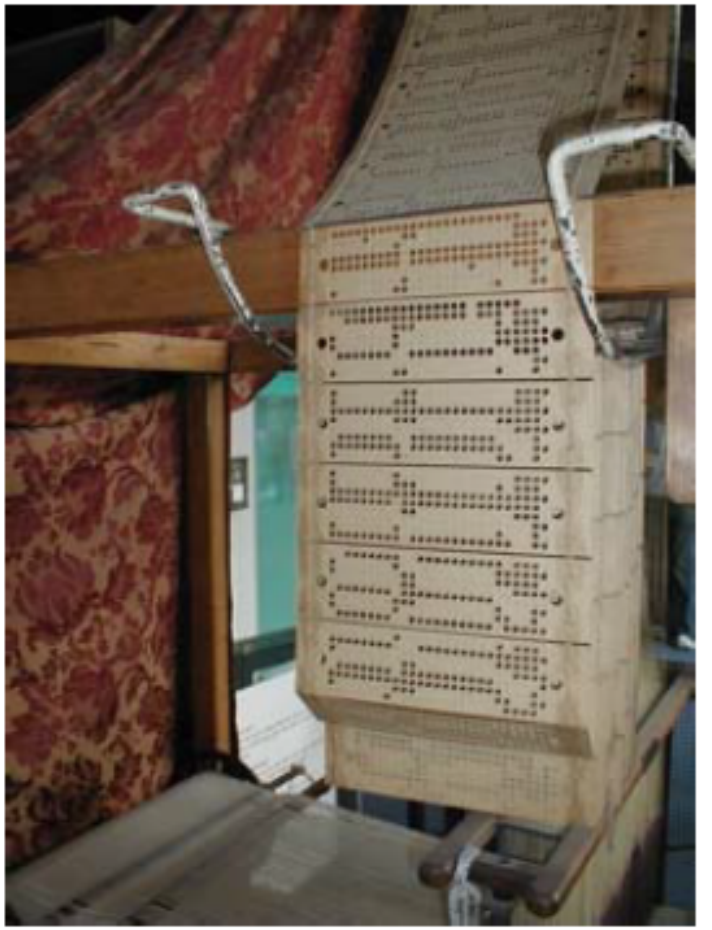
Blaise Pascal (1642)
Vites tabanlı toplama makinası



Gottfried Wilhelm von Leibniz (1670)
Toplama, çıkarma, çarpma, bölme
Mekanik olarak sık sık arızalanırdı.

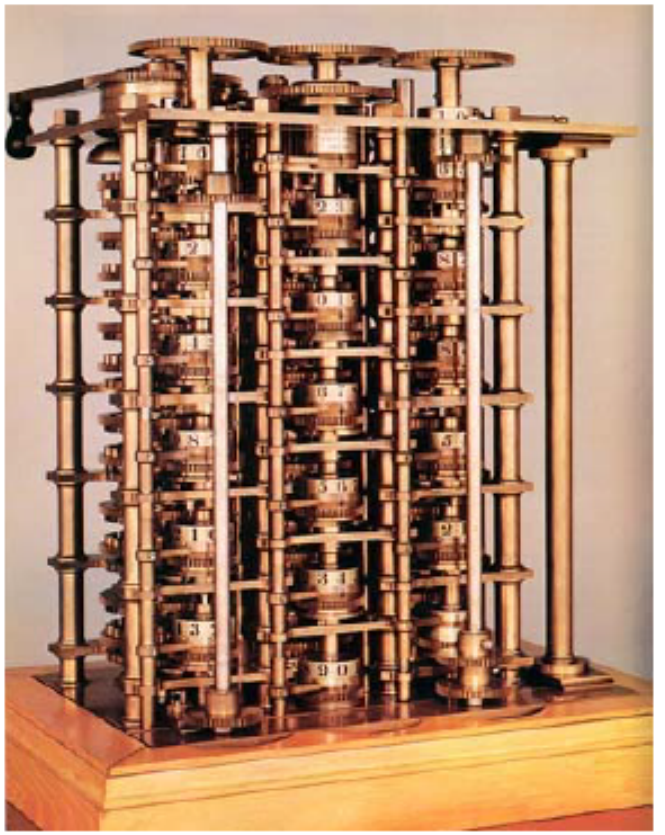
Bilgisayarların Gelişimi

Joseph Jacquard (1810)
Bilgisayar tabanlı halı dokuma makinesi



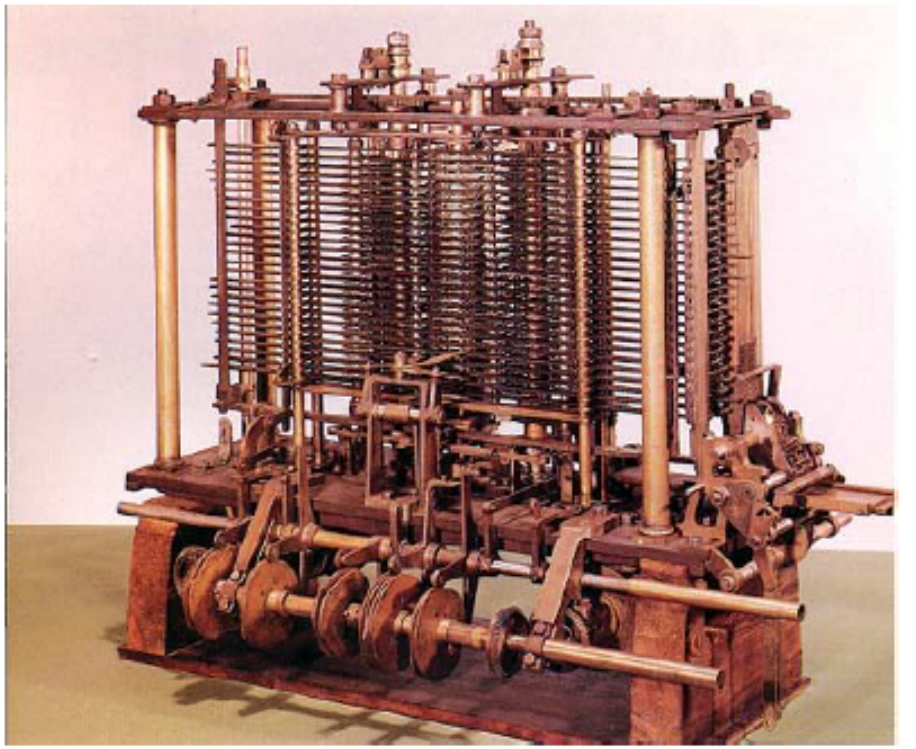
Delikli Kart (Punch Card)

Bilgisayarların Gelişimi



The Difference Engine (1822)

- Charles Babbage

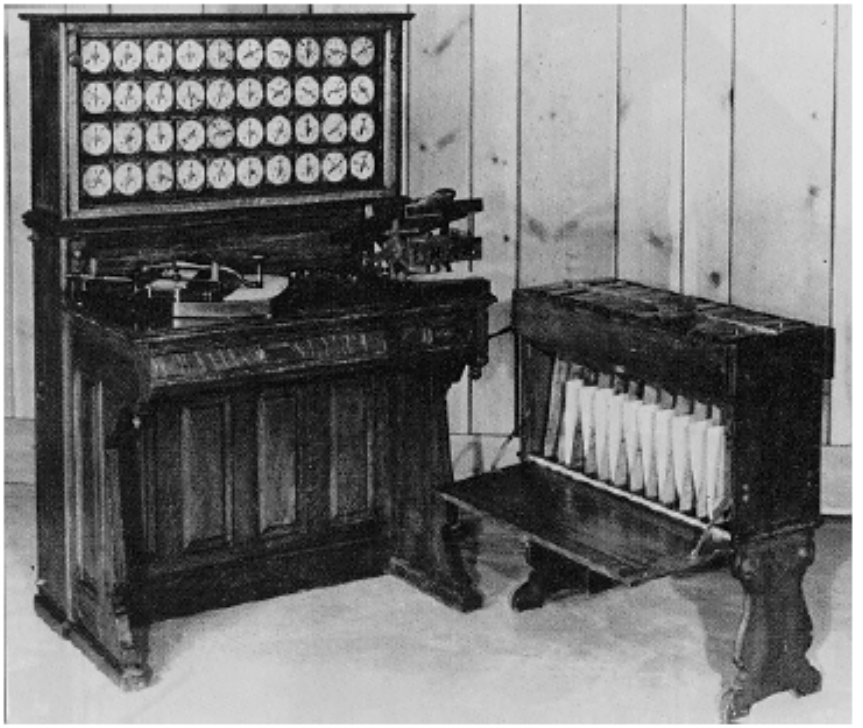


The Analytical Engine

- Punch card'lar üzerinde yazılan programları işleyebiliyordu
- Bilgiyi belleğinde saklayabiliyordu

Bilgisayarların Gelişimi

Elektro-mekanik Çağ (1840 – 1940)



- Hermann Hollerith (19'uncu yüzyılın sonları)
- Amerikan oy sayımlarına kullanıldı.
- Elektrik ile çalışıyor.
- Bilgi punch card ile veriliyor.

- Nüfus: 63 milyon; 6 hafta
- International Business Machines (IBM)'in ilk ürünü

The Harvard Mark I

Mark I

- Howard Aiken + IBM + Harvard (1930)
- Veri depolama: Mekanik röle telefon anahtarları (switch)
- Girdi: Punch Card

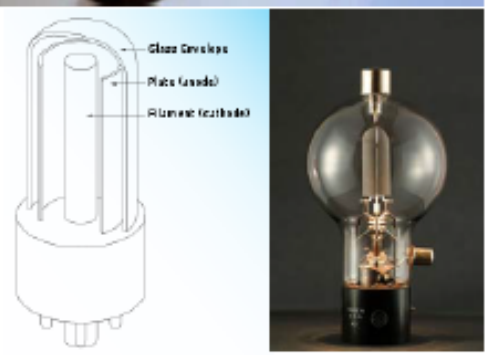
Bir bilgisayar ile bir hesap makinesi arasında ne fark var?

Bilgisayarların Gelişim

Elektronik Çağ (1840 – Bugün)

❖Elektronik ile ilgili ilk deneylerin vakum tüplerinde yapılan çalışmalar olduğu kabul edilir. Heinrich Geissler (1814-1879), cam tüpün içinden havanın çoğunu çıkartmış ve bu tüpün içinden elektrik akımı geçirildiğinde tüpün parıldadığını görmüştür.

❖Sir William Crookes (1832-1919) havası alınmış cam tüp'ün (vakum tüp) içinden akım geçirdiğinde, geçen akımın parçacıklardan oluştuğunu görmüştür.
❖Sir Joseph Thompson (1856-1940) bu parçacıkları ölçmeyi başarmıştır ve bu parçacıklara daha sonra ***elektron*** denilmiştir.



❖John Ambrose Fleming, 1904 yılında, vakum tüpünü kullanarak akımın tek yönlü olarak akmasına izin veren vakum tüp diode'u geliştirmiştir. Bu cihaza "Fleming valve" veya radio tube'de denir.

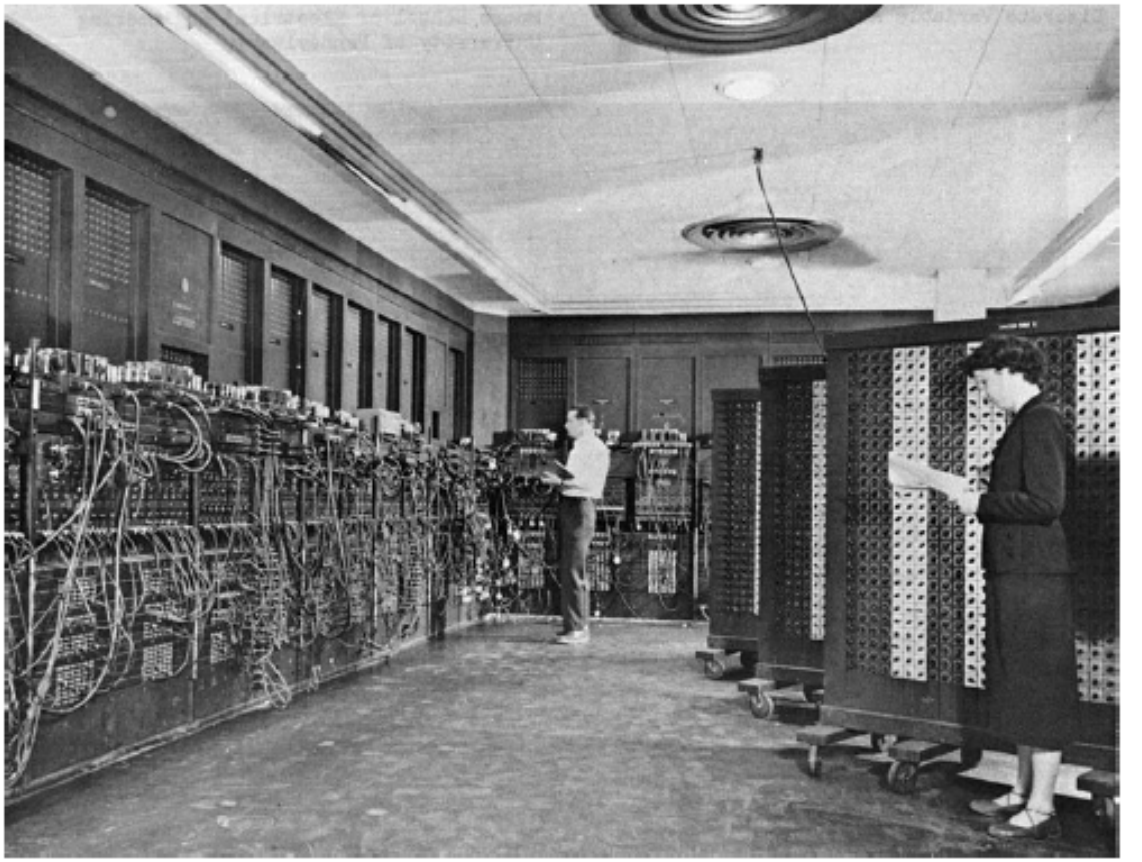
Bilgisayarların Gelişimi

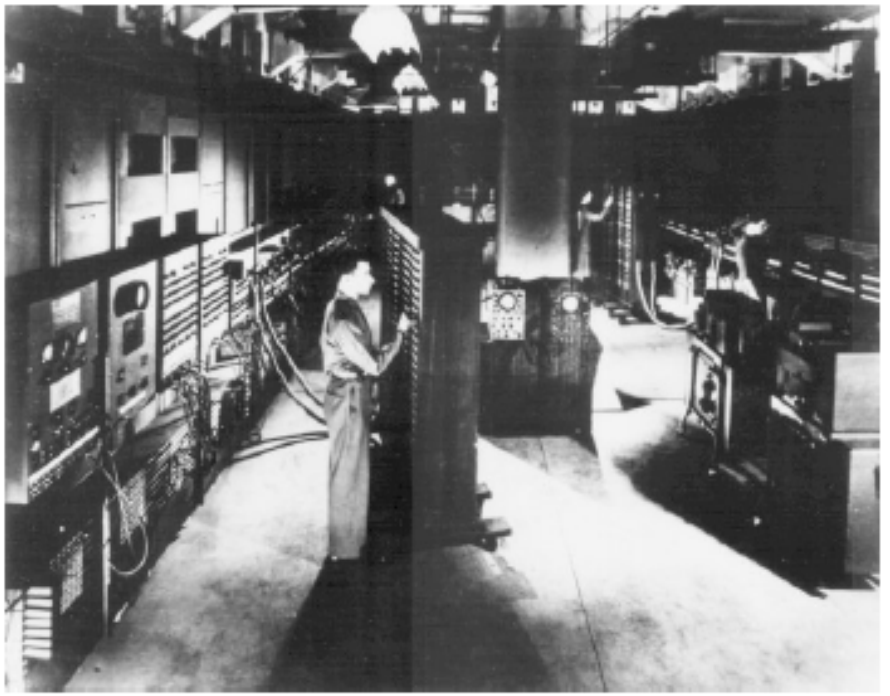
- 1930 yıllarında, elektronik dünyasında bir çok gelişme olmuştur. Bu yıllarda ilk elektronik hesap makineleri geliştirilmeye başlanmıştır.

- John Atanasoff ve lisansüstü öğrencisi Clifford Berry, 1939 yılında, ABC (Atanasoff-Berry Computer) olarak adlandırılan ilk ikili sayı sisteminde çalışan makineyi icat geliştirmişlerdir. Bu makinada lojik işlemler için vakum tüpleri ve hafıza için kondansatörler kullanılmıştır.

Savaş sırasında bilgisayar konusundaki çalışmalar çok daha hızlı bir şekilde geliştirilmiştir.

John von Neumann, 1946 yılında, ilk bilgisayar olarak kabul edilen Eniac'ı geliştirmiştir.



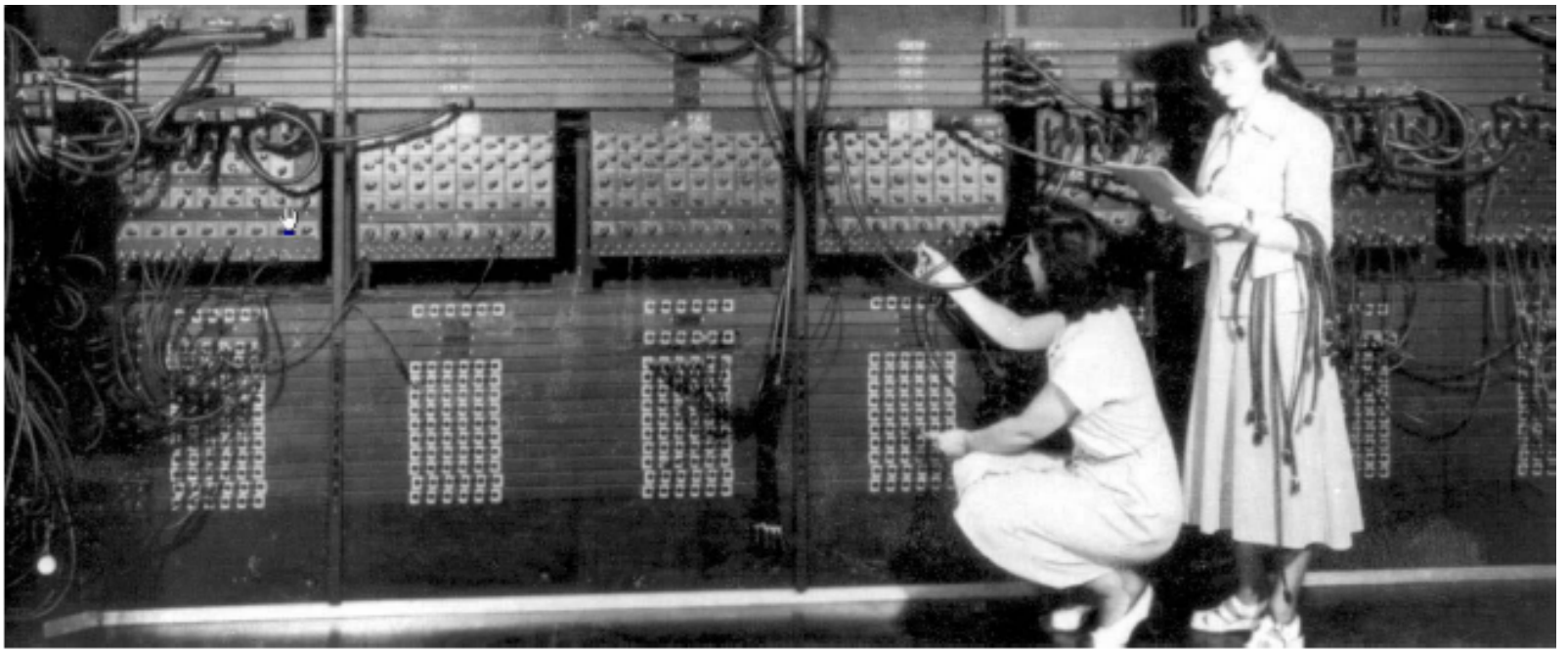


ENIAC

- 1000 metre kare alan
- 30 tons
- vacuum tüpleri kullanıyordu
  - >17,000
- Karar verebiliyordu: ilk gerçek bilgisayar
- Programlama kablo temasları ve switch ayarları ile yapılıyordu.

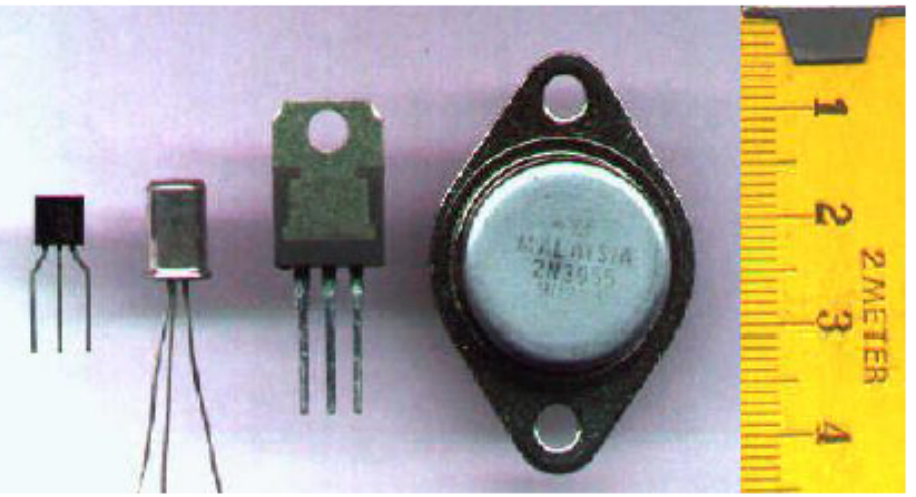
1945 yılında Bell laboratuarlarında bir araştırma grubu kurulmuştur. Grubun amacı: iletkenler, yarıiletkenler, yalıtkanlar, piezoelektrik malzemeler ve manyetik malzemeler üzerinde temel araştırmalar yapmak, olarak tanımlanmıştır. Burada yapılan yarıiletkenler konusundaki çalışmalar sonucunda, Walter Brattain, John Bardeen ve William Shockley tarafından tranzistör icat edilmiştir. 1950 yılında bu yeni devre elemanının patenti alınmış ve 1951 yılında da Allentown Pennsylvania'da ticari olarak üretilmeye başlanmıştır.

Tranzistörün icadı elektronikte devrim niteliğindedir.

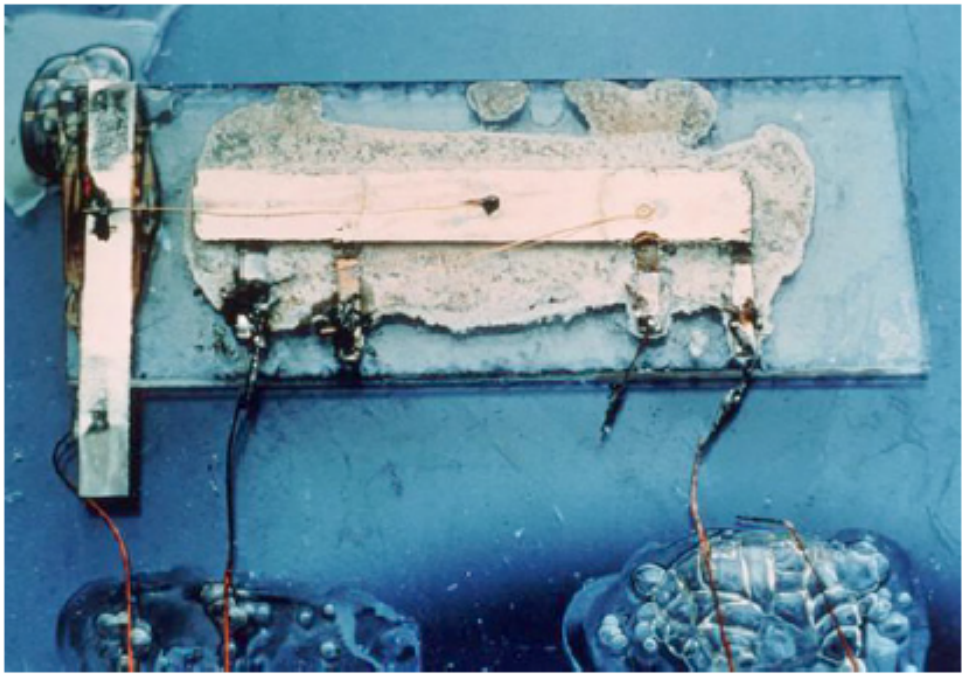


Electronic Numerical Integration and Calculator (ENIAC)

- John Mauchly and J. Presper Eckert (1946'da tamamlandı)
- İlk olarak 2'inci dünya savaşında gizli bir proje olarak başladı.
- University of Pennsylvania


MALAYSIA
2N3055
2 METER

1950'li yıllarda yapılan araştırmalar sonucunda çok sayıda tranzistör, diyot ve kapasiteden oluşan devrelerin bir bütün olarak gerçekleştirilmesi yolu bulunmuştur. Böylece ortaya tümdevreler veya entegre devreler (integrated circuit) çıkmıştır.

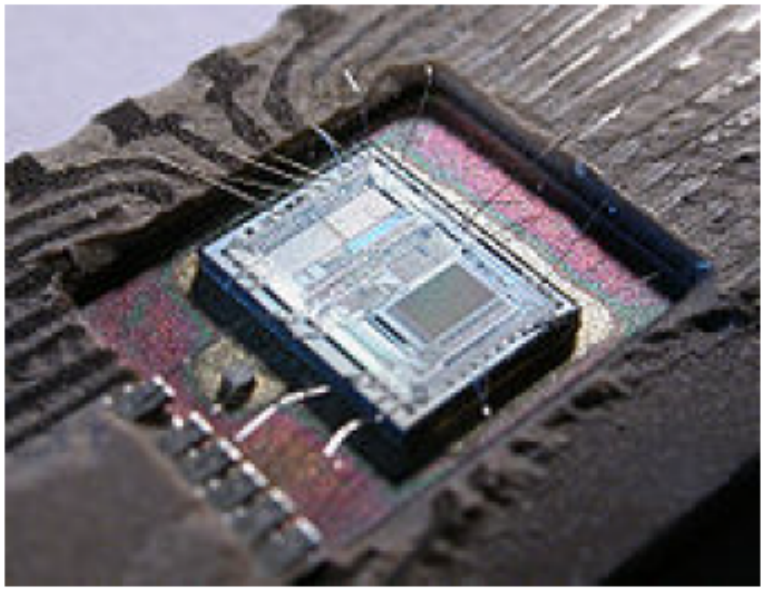


- Jack Kilby, 1958 yılında, Texas Instruments firmasında ilk tümdevreyi gerçekleştirmiştir.

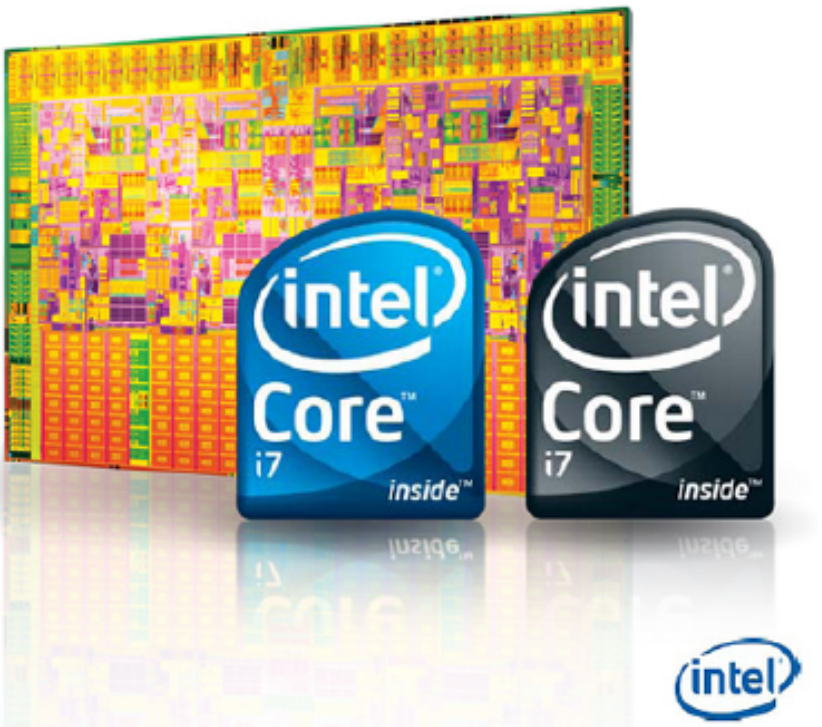
1960 ve 1962 yılında yapılan çalışmalarda tümdevre teknolojisine BJT'lere göre daha uygun olan Metal-oksit-yarıiletken alan etkili tranzistör (metal-oxide-semiconductor field effect transistor-MOSFET) geliştirilmiştir (Kahng ve Atalla, 1960), (Hofstein ve Heinman, 1963).

MOSFET transistorlerin gelişmesi ile birlikte tümdevre içine çok daha fazla sayıda transistor yerleştirilebilmiştir. Bir tümleşik devredeki eleman sayısı 1964'te 40'a ve 1972'de 1200'e yükselmiştir. 1982'li yıllarda VLSI (Very Large-Scale Integration) olarak isimlendirilen sistemlerde 100,000'ler mertebesinde eleman içeren tümleşik devreler gerçekleştirilmiştir.

Günümüzde bu elaman sayıları çok daha büyük değerlere ulaşmıştır.



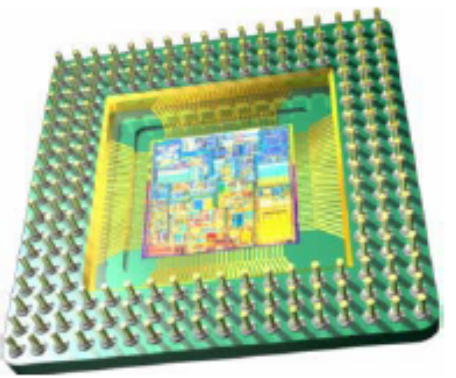
Intel 8742 8-bit mikrokontroller. İşlemci hızı 12 MHz, 128 bytes Ram, 2048 bytes EPROM, giriş çıkış uçları. Hepsi bir tümdevrede



781 Milyon tranzistör bir tümdevrenin içinde

- Microprocessor: CPU içeren tek bir chip
  - İlk olarak 1970 yılında Marcian Hoff (Intel Corporation) tarafından tasarlandı

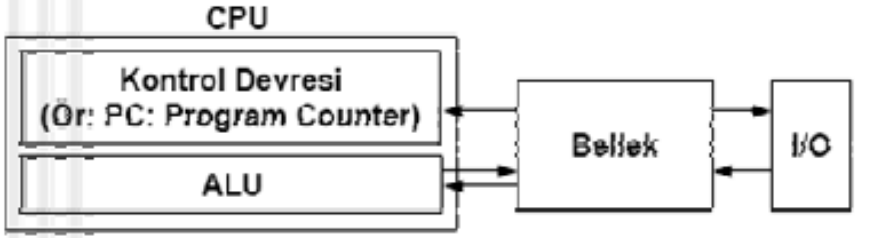
- Microcomputer: masaüstü boyutlarında bilgisayar
  - ALTAIR (1975)
  - Apple (Stephen Wozniak ve Steven Jobs; 1977)

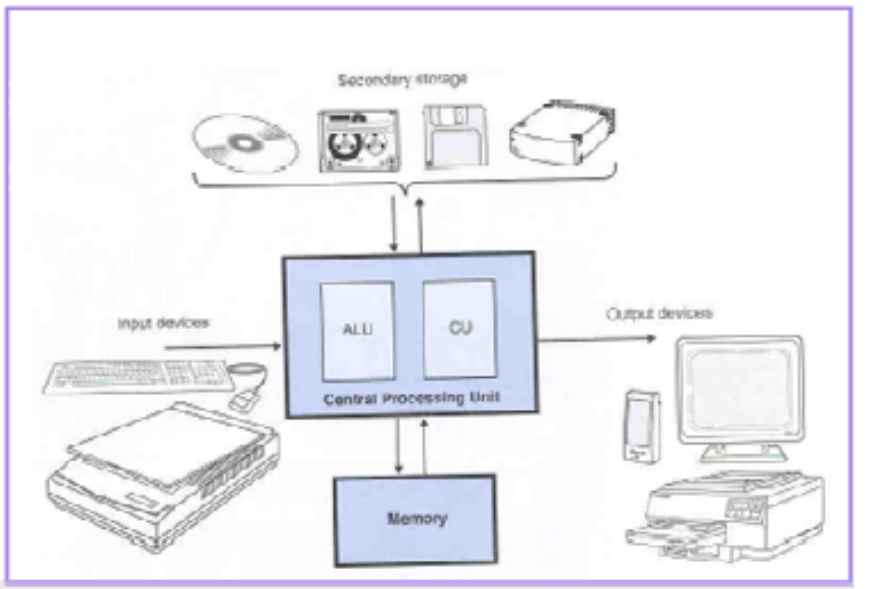


Temel Bilgisayar Mimarileri

- Von Neumannmimarisi
- Harvard Mimarisi

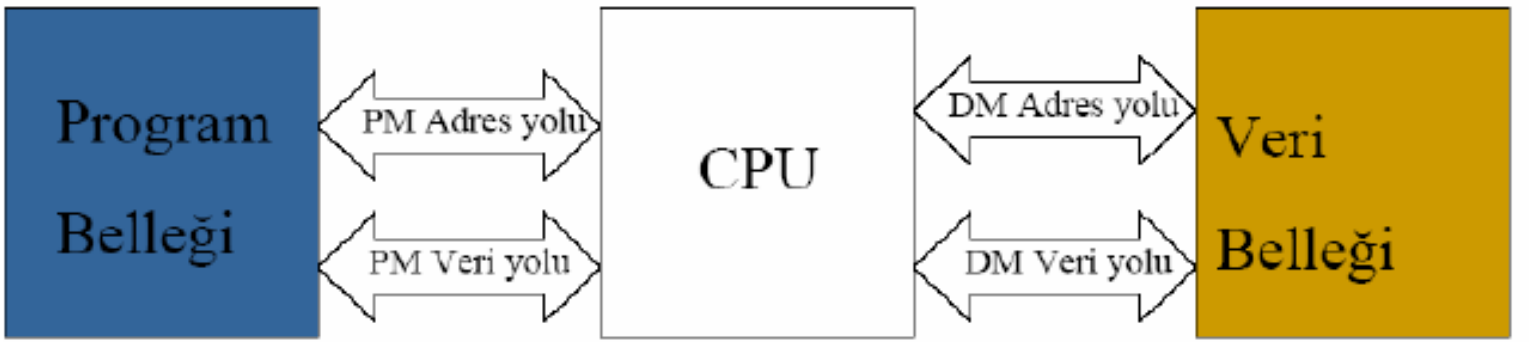
Von-Neumann Mimarisi

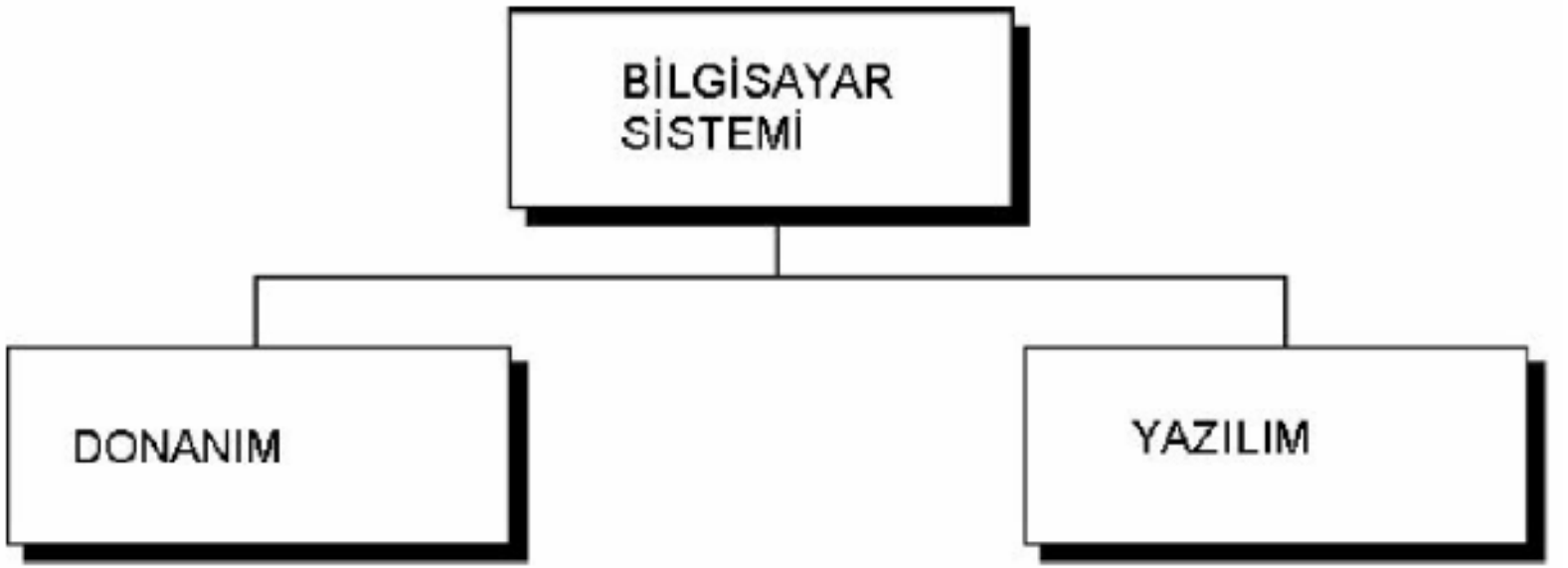




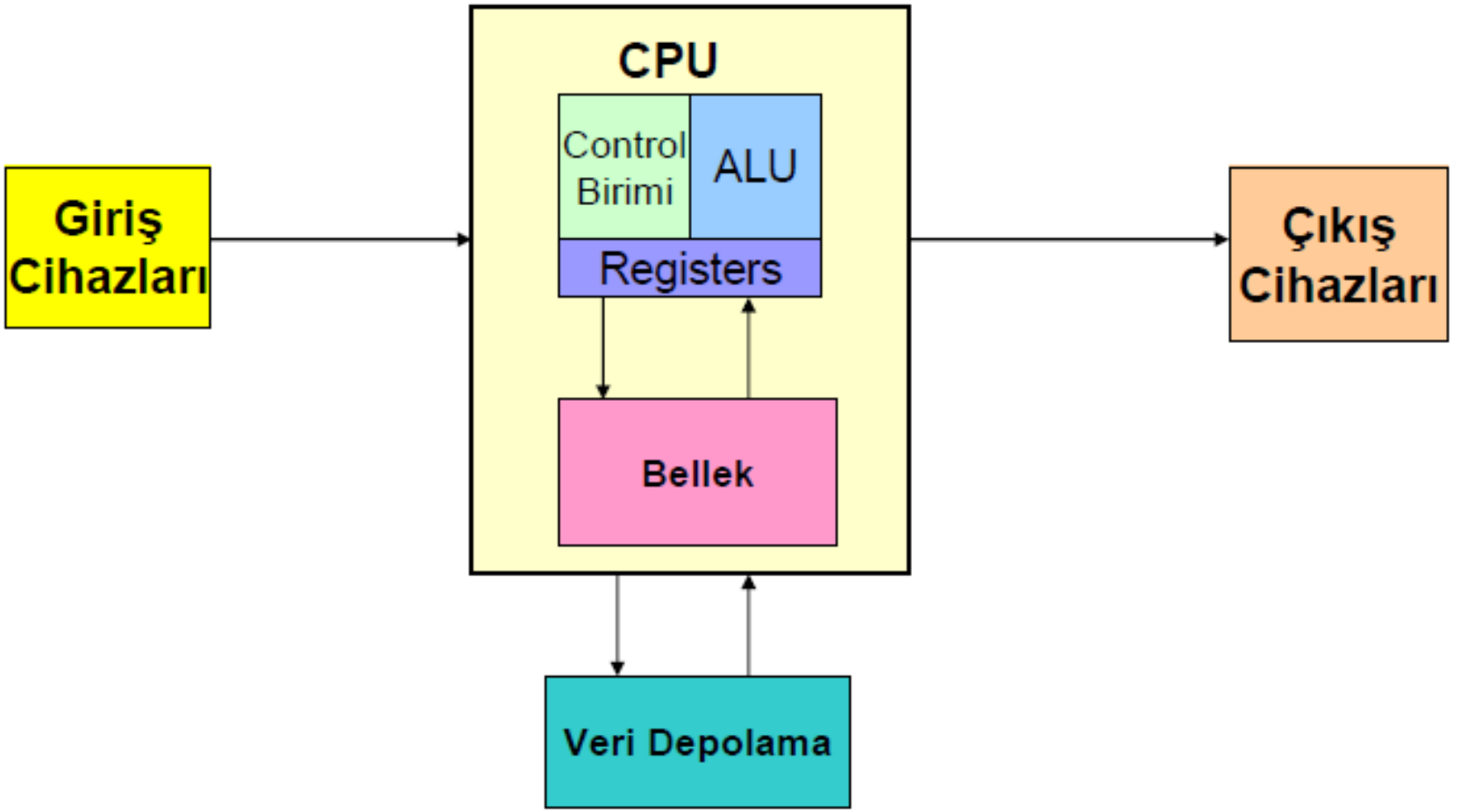
Harvard Mimarisi

Günümüz tipik bilgisayarları Von- Neumann Mimarisine sahip Mikroişlemciler kullanırken (Intel x86, Pentium, AMD Athlon..) , Özellikle Görüntü, ses işleme, yüksek hız gerektiren uygulamalarda Harward mimarisine sahip mikroişlemlerciler (DSP'ler, ARM Cortex..)

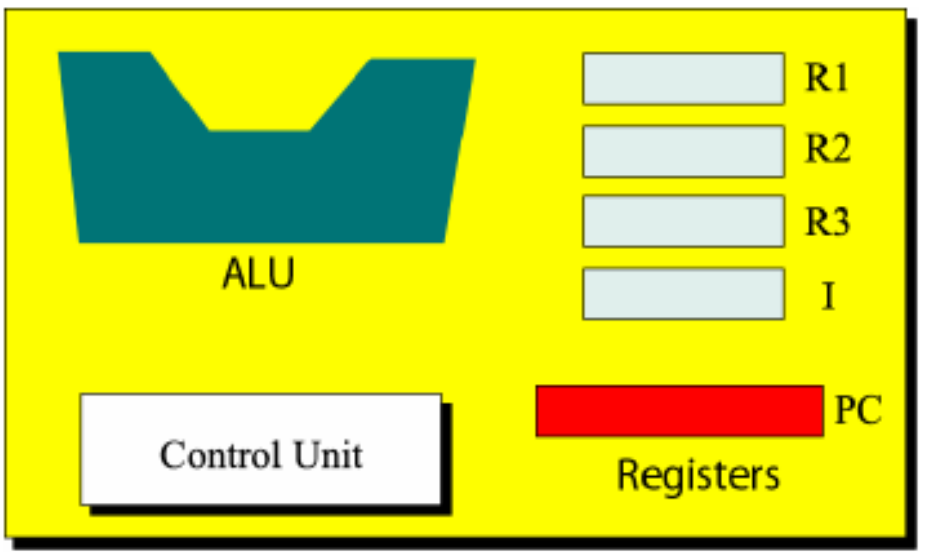




1. Donanım fiziksel aygıtlardır.
2. Yazılım ise yapılması gereken işleri yapabilmek için donanıma komutlar veren programlar topluluğudur.


CPU
Control Birimi
ALU
Registers
Giriş Cihazları
Çıkış Cihazları
Bellek
Veri Depolama

- Görevleri yapabilmek için komutları işleyen mikroişlemciye CPU denir.
- CPU nelerden oluşur:
  - Kontrol Ünitesi
  - Aritmetik mantık ünitesi (Arithmetic Logic Unit)
  - Register

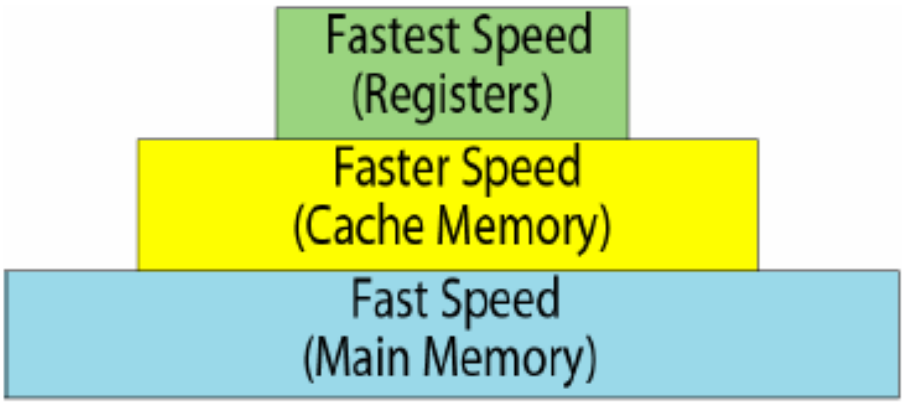


CPU'daki Komut Döngüsü

Bellek Türleri

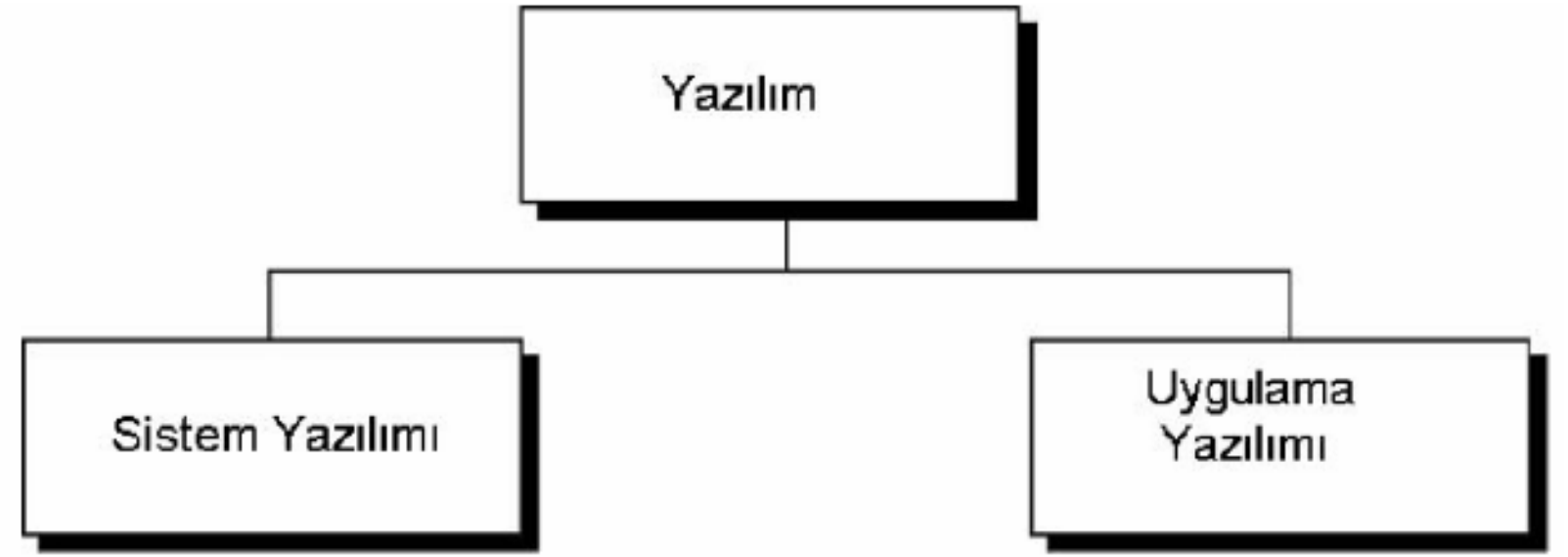
- **Registers** – CPU'nun bir parçası; çok hızlı; sınırlı büyüklük
- **Cache Memory** – CPU'nun bir parçası; RAM'den daha hızlı
- **Read-only Memory (ROM)**

  – Bilgisayarın sürekli ihtiyaç duyduğu sistem komutlarını barındıran chip
- **Random Access Memory (RAM)** – Ana karta eklenen bellek; program komutları ve veriler için birincil depo



Diğer Bilgisayar Bileşenleri

- Veri depolama sistemi
  - Hard disk, tape, floppy, DVD vs.
    - Geniş alan, ucuz, yavaş, manyetik ve optik
- Input Cihazları
  - Klavye, Fare, Dokunmatik ekran, Tarayıcı, Webcam,Joystick,Mikrofon
- Output Cihazları
  - Monitör, Yazıcı, Plotter, Hoparlör



Sistem Yazılımı:

1. Aygıt Yazılımı (Firmware) (BIOS).
2. İşletim Sistemi
3. Sistem destek yazılımı
4. Sistem Geliştirme Yazılımı

Uygulama yazılımı:

1. Genel Amaçlı
2. Uygulamaya Özel

Sistem Yazılımı

•Aygıt Yazılımı: Sistemi oluşturan donanımların çalışması için gerekli olan yazılımlardır.

•İşletim sistemi: Kullanıcı arayüzü, ağ bağlantı arayüzleri, Dosya erişimi ve organizasyonu, Çoklu çalışma gibi hizmetleri sağlayan yazılımlardır. Örneğin: DOS, Windows, Linux, PARDUS, Unixvs..

•Sistem destek yazılımları: Sistemle ilişkili faydalı yazılımlardır. Örneğin,Disk formatlayıcı, hesap makinesi, test ve iletişim yazılımları, Hyperterminal, Telnet vs..

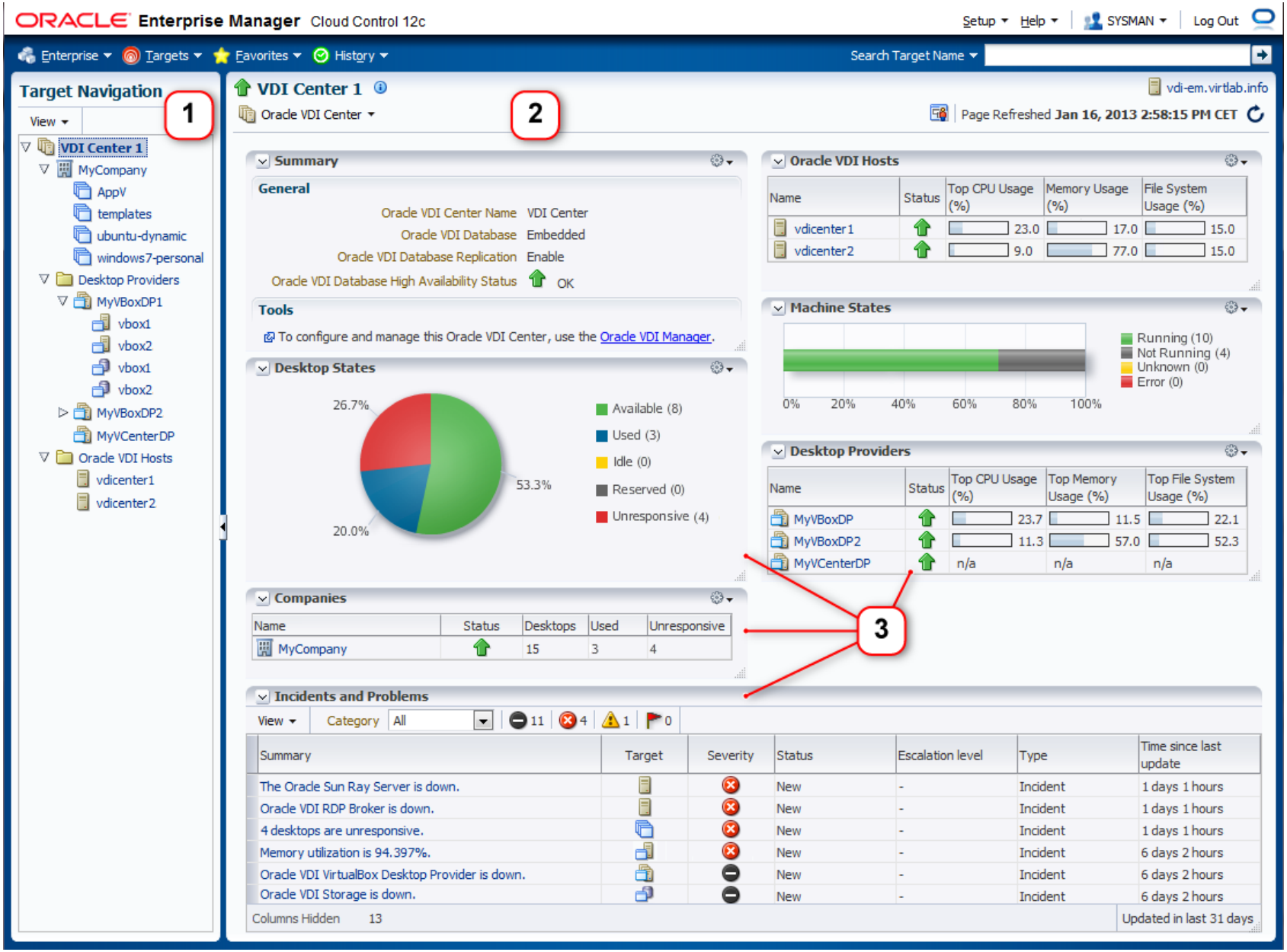
•Sistem Geliştirme Yazılımları: Bunlar, çeşitli kütüphaneler, Uygulama Programı arayüzü(API) (Winsock, setupapi, mmtools, SAPI, DDK..), Derleyiciler, Debugger'lar..

Uygulama Yazılımları

•Genel Amaçlı

- •Kelime işlem programları: MS-Word, Word-Pro, …
- •Veri tabanı yönetim programları: Oracle, Access, SQL, …
- •Hesap Tablosu programları: MS-Excel, Lotus, …
- •Grafik ve çizim programları: AutoCAD, 3D MAX, Photoshop, Corel Draw, …
- •Matematik tabanlı programlar: MATLAB, MatCAD, Mathematica, …
- •…

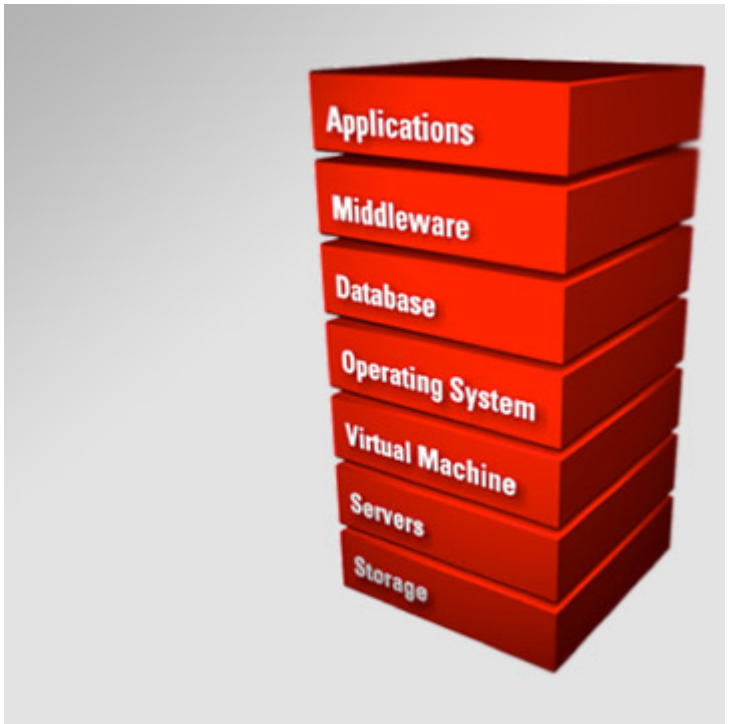
•Özel yazılımlar


ORACLE Enterprise Manager Cloud Control 12c
Setup
Help
SYSMAN
Log Out
Enterprise
Targets
Favorites
History
Search Target Name
Target Navigation
1
View
VDI Center 1
MyCompany
AppV
templates
ubuntu-dynamic
windows7-personal
Desktop Providers
MyVBoxDP1
vbox1
vbox2
vbox1
vbox2
MyVBoxDP2
MyVCenterDP
Oracle VDI Hosts
vdicenter1
vdicenter2
VDI Center 1
Oracle VDI Center
2
vdi-em.virtlab.info
Page Refreshed Jan 16, 2013 2:58:15 PM CET
Summary
General
Oracle VDI Center Name VDI Center
Oracle VDI Database Embedded
Oracle VDI Database Replication Enable
Oracle VDI Database High Availability Status OK
Tools
To configure and manage this Oracle VDI Center, use the Oracle VDI Manager.
Desktop States
26.7%
53.3%
20.0%
Available (8)
Used (3)
Idle (0)
Reserved (0)
Unresponsive (4)
Companies
Name Status Desktops Used Unresponsive
MyCompany 15 3 4
Oracle VDI Hosts
Name Status Top CPU Usage (%) Memory Usage (%) File System Usage (%)
vdicenter1 23.0 17.0 15.0
vdicenter2 9.0 77.0 15.0
Machine States
0% 20% 40% 60% 80% 100%
Running (10)
Not Running (4)
Unknown (0)
Error (0)
Desktop Providers
Name Status Top CPU Usage (%) Top Memory Usage (%) Top File System Usage (%)
MyVBoxDP 23.7 11.5 22.1
MyVBoxDP2 11.3 57.0 52.3
MyVCenterDP n/a n/a n/a
3
Incidents and Problems
View
Category All
11
4
1
0
Summary Target Severity Status Escalation level Type Time since last update
The Oracle Sun Ray Server is down. New - Incident 1 days 1 hours
Oracle VDI RDP Broker is down. New - Incident 1 days 1 hours
4 desktops are unresponsive. New - Incident 1 days 1 hours
Memory utilization is 94.397%. New - Incident 6 days 2 hours
Oracle VDI VirtualBox Desktop Provider is down. New - Incident 6 days 2 hours
Oracle VDI Storage is down. New - Incident 6 days 2 hours
Columns Hidden 13
Updated in last 31 days

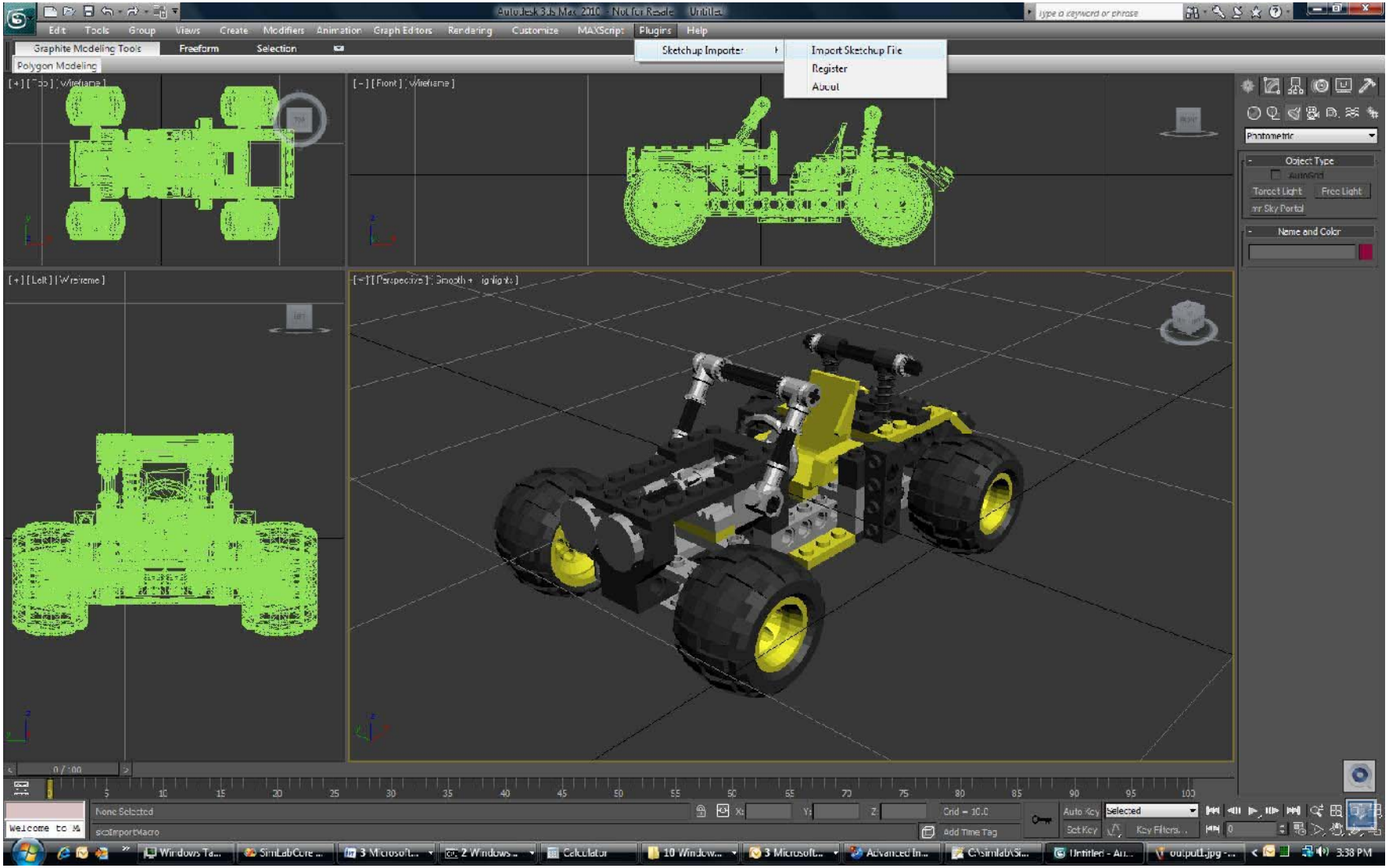
ORACLE CLOUD

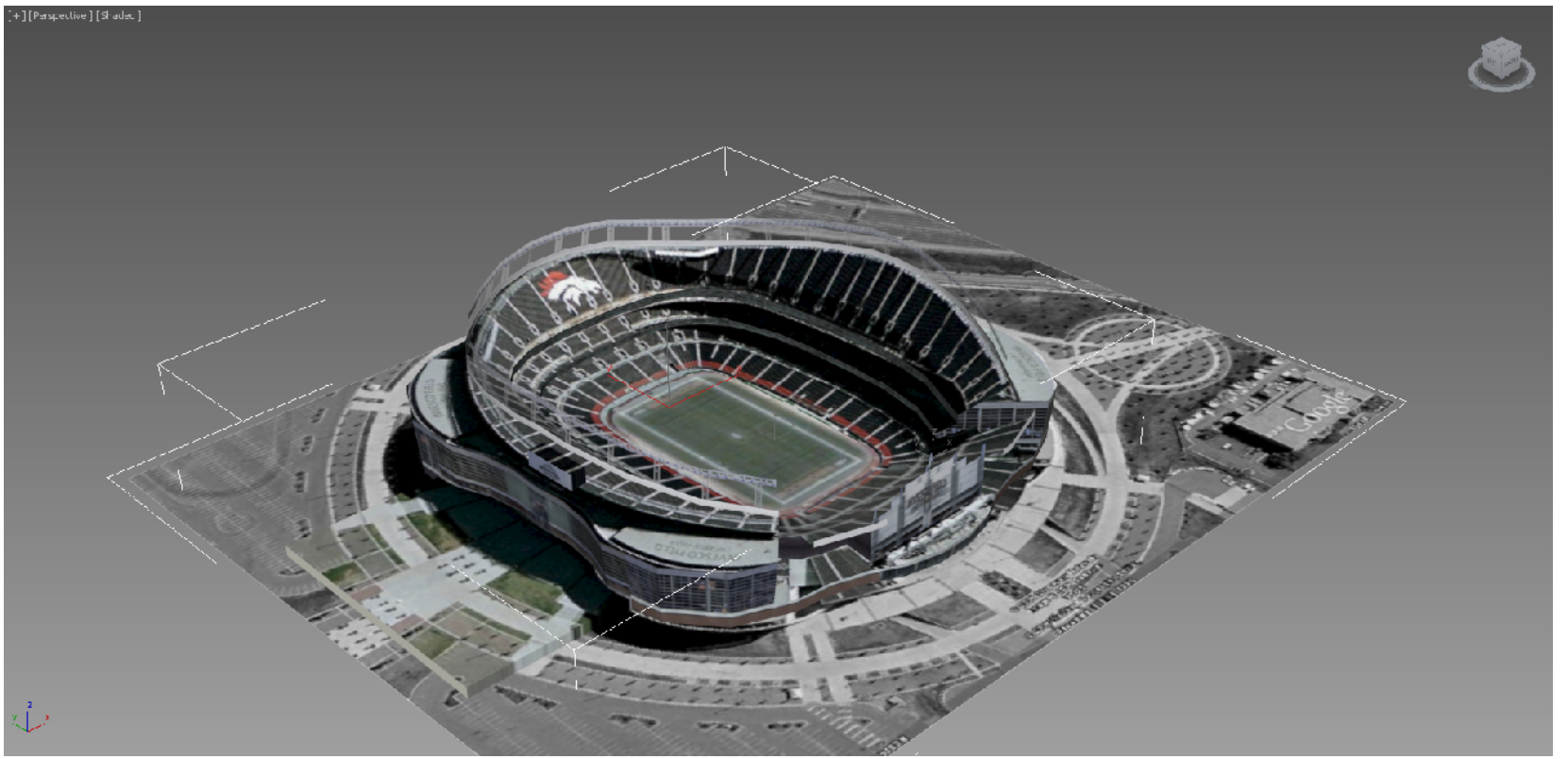
Social. Mobile. Complete.

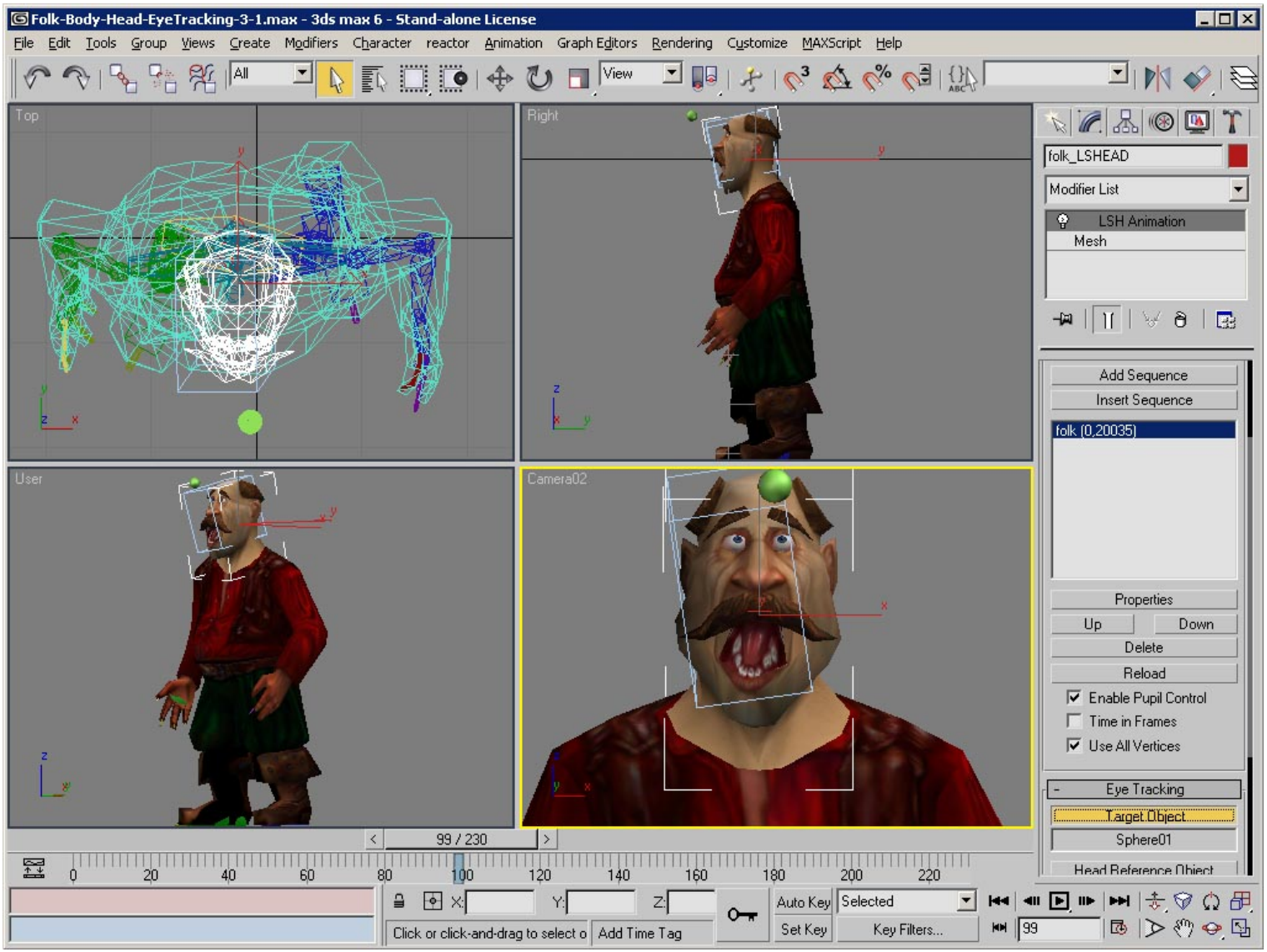
ORACLE


Applications
Middleware
Database
Operating System
Virtual Machine
Servers
Storage

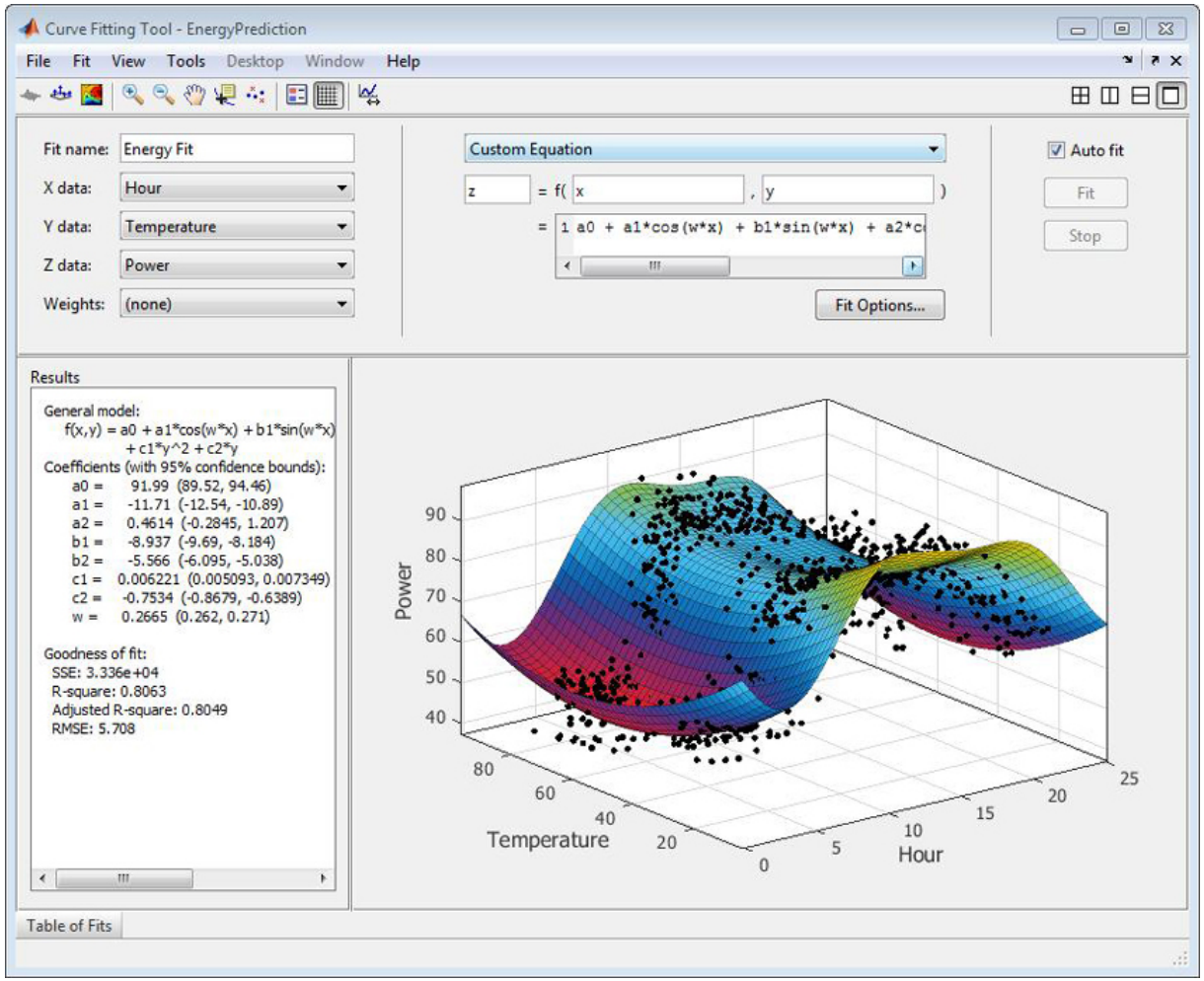
3D MAX


Edit Tools Group Views Create Modifiers Animation Graph Editors Rendering Customize MAXScript Plugins Help
Graphite Modeling Tools Freeform Selection
Polygon Modeling
Sketchup Importer
Import Sketchup File
Register
About
[+] [Top] [Wireframe]
[-] [Front] [Wireframe]
[+] [Left] [Wireframe]
[+] [Perspective] [Smooth + Highlights]
Photometric
Object Type
Target Light
Free Light
Name and Color
None Selected
scaImportMacro
Welcome to M
Add Time Tag
Auto Key
Set Key
Key Filters...
3:38 PM

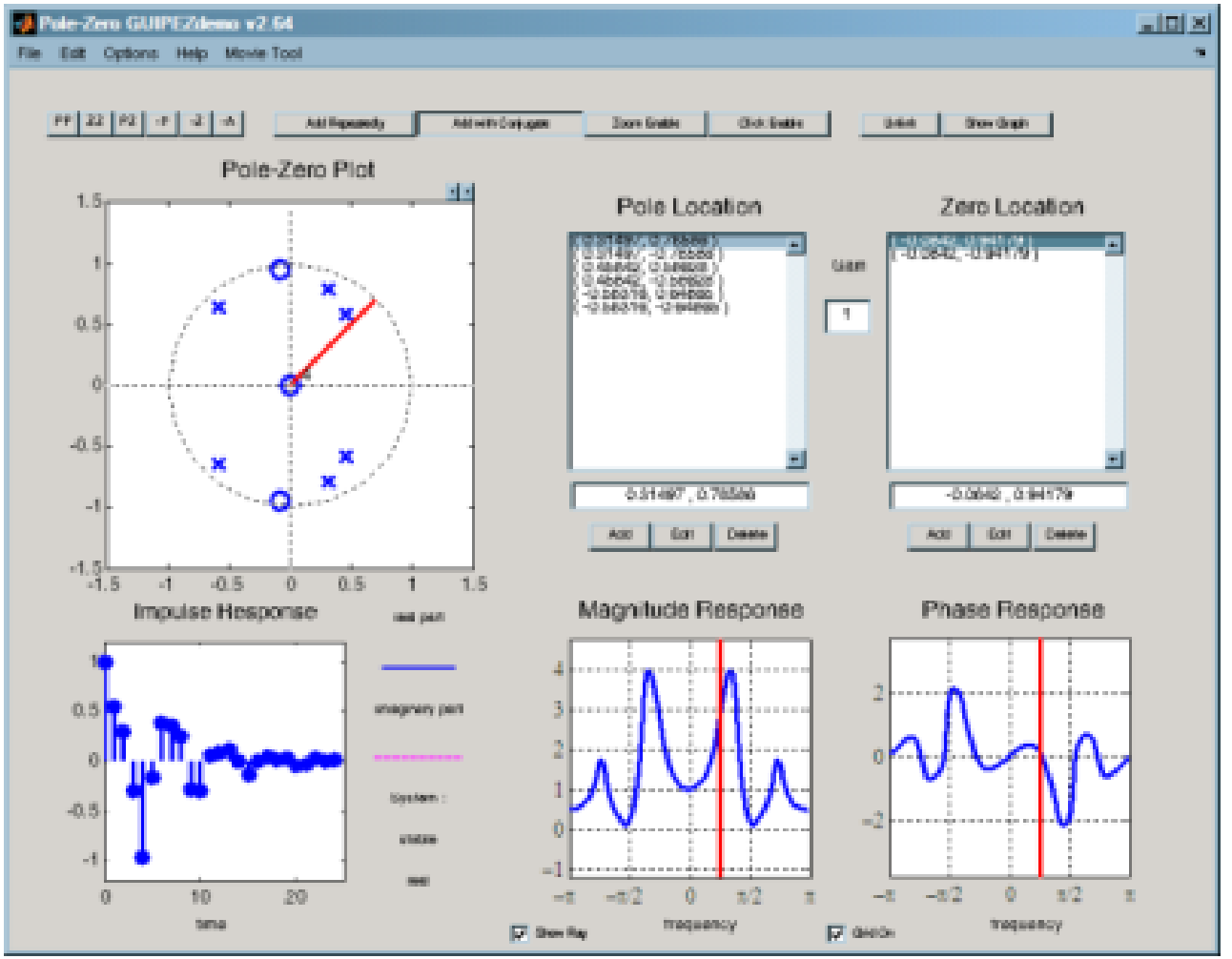

[+] [Perspective] [Shaded]
Google


Folk-Body-Head-EyeTracking-3-1.max - 3ds max 6 - Stand-alone License
File Edit Tools Group Views Create Modifiers Character reactor Animation Graph Editors Rendering Customize MAXScript Help
Top
Right
User
Camera02
folk_LSHEAD
Modifier List
LSH Animation
Mesh
Add Sequence
Insert Sequence
folk (0,20035)
Properties
Up
Down
Delete
Reload
Enable Pupil Control
Time in Frames
Use All Vertices
Eye Tracking
Target Object
Sphere01
Head Reference Object
99 / 230
0 20 40 60 80 100 120 140 160 180 200 220
X: Y: Z:
Auto Key
Selected
Set Key
Key Filters...
Click or click-and-drag to select o
Add Time Tag

MATLAB


Curve Fitting Tool - EnergyPrediction
File Fit View Tools Desktop Window Help
Fit name: Energy Fit
X data: Hour
Y data: Temperature
Z data: Power
Weights: (none)
Custom Equation
z = f(x , y)
= 1 a0 + a1*cos(w*x) + b1*sin(w*x) + a2*c
Fit Options...
Auto fit
Fit
Stop
Results
General model:
f(x,y) = a0 + a1*cos(w*x) + b1*sin(w*x) + c1*y^2 + c2*y
Coefficients (with 95% confidence bounds):
a0 = 91.99 (89.52, 94.46)
a1 = -11.71 (-12.54, -10.89)
a2 = 0.4614 (-0.2845, 1.207)
b1 = -8.937 (-9.69, -8.184)
b2 = -5.566 (-6.095, -5.038)
c1 = 0.006221 (0.005093, 0.007349)
c2 = -0.7534 (-0.8679, -0.6389)
w = 0.2665 (0.262, 0.271)
Goodness of fit:
SSE: 3.336e+04
R-square: 0.8063
Adjusted R-square: 0.8049
RMSE: 5.708
Power
90
80
70
60
50
40
Temperature
80
60
40
20
Hour
0
5
10
15
20
25
Table of Fits

MATLAB


File Edit Options Help Movie Tool
Pole-Zero Plot
Pole Location
Zero Location
Gain
Impulse Response
Magnitude Response
Phase Response
time
frequency
frequency

Program: belirli bir işi gerçekleştirmek için gerekli komutlar dizisi olarak tanımlanabilir.

Programlama: Bir programı oluşturabilmek için gerekli komutların belirlenmesi ve uygun biçimde kullanılmasıdır.

Programlama Dilleri: Bir programın oluşturulmasında kullanılan komutlar, tanımlar ve kuralların belirtildiği programlama araçlarıdır.

Yazılım: Belirli bir amacı sağlayan, program yada programlar ve ilgili dokümantasyonlardır.

Makine dili **(birinci seviye)**

- Bilgisayarın ana dilidir.
- İkicil (binary) kodlardan oluşur (0'lar ve 1'ler)
  - **Örn. 0110 1001 1010 1011**
- Bilgisayarın anlayabildiği tek dildir.

Assembly Dili **(ikinci seviye)**

- Makine diline birebir çevrilebilir
- Makine dilinden daha kolay anlaşılabilir (ama çok da değil)
  - **Örn. ADD X Y Z**
- Assembler – assembly dilini makine diline çeviren program

Procedural diller **(üçüncü seviye)**

- Bir komut pek çok makine dili komutuna karşılık gelir
- Programlarda bilgisayarın işlem akışını adım adım tasarlayabilirsiniz.
- İnsan diline daha çok benzer; bilinen kelimeleri kullanır
- Örnek: C, C++, Java, Fortran, QuickBasic
- Derleyici (compiler) – programın tümünü assembly veya makine diline çevirir (C++, Pascal, Ada).
- Interpreter – program çalıştırıldığında adım adım programı makine koduna çevirir (Basic, Javascript, LISP)

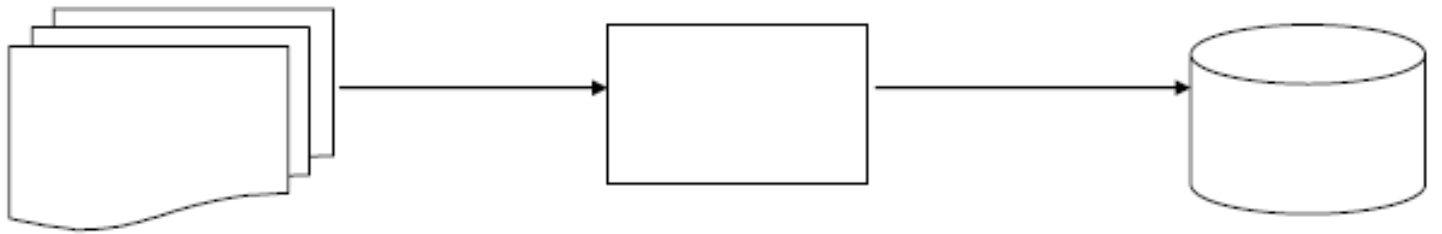
Nonprocedural Diller **(dördüncü seviye)**

- Kullanıcının sadece gerekli sorguyu göndermesi sonuca ulaşması için yeterlidir.
- Örnek: – veritabanı sorgulama dili- SQL
- Teknik olmayan insanlar tarafından da kullanılabilir.

Natural Language Programming Languages **(beşinci seviye (akıllı diller))**

- İnsan dilini programlama diline çevirir.
- Oldukça karmaşık ve yenidir.

- Dilden dile çevrim



Programlama dili ile yazılan program (kaynak kodu)(**source code**)

Çevirici program

- **Assembler**
- **Compiler**
- **Interpreter**

Makine diline Çevrilmiş kod (**object code**)

CPU tarafından işlenir

Makine Dilinde Çarpma İşlemi

```
                00000000   00000100   0000000000000000
01011110   00001100   11000010   0000000000000010
                11101111   00010110   0000000000000101
                11101111   10011110   0000000000001011
11111000   10101101   11011111   0000000000010010
                01100010   11011111   0000000000010101
11101111   00000010   11111011   0000000000010111
11111100   10101101   11011111   0000000000011110
00000011   10100010   11011111   0000000000100001
11101111   00000010   11111011   0000000000100100
01111110   11110100   10101101
11111000   10101110   11000101   0000000000101011
```

Bir Assembly programı Örneği:

```
LDI temp,0x80 ; Analog Comparator disabled
OUT ACSR,temp
LDI temp,0x00
OUT DDRB,temp ; PORTB giriş
LDI temp,0b01110000 ; PD0,PD1,PD2,PD3 inputdiğerleri output
OUT DDRD,temp
LDI temp,0b01000000 ; initPORTD
OUT PORTD,temp
CLR hat1_time_out ;ilk değerleri atama bölümü
CLR hat2_time_out
CLR temp
LDI ZH,0 ;hat1 temp buffer'ı boşalt
LDI ZL,hat1_temp_adres
ST Z,temp
LDI YH,0 ;hat2 temp buffer'ı boşalt
LDI YL,hat2_temp_adres
ST Y,temp
LDI XH,0
```

İlk programın, Ada Lovelacetarafından Charles Babbage'ın tanımlamış olduğu "Analytical Engine" i ile Bernoullisayılarının hesaplanmasına ilişkin makalesinde olduğu söylenmektedir. Bu nedenle ilk gerçek anlamdaki programlama dillerinden birinin adı Ada Lovelace anısına ADA olarak isimlendirilmiştir.1940 larda ilk modern bilgisayar ortaya çıktığında, programcılar yalnızca assembly dili kullanarak yazılım yapabiliyorlardı.

1950 –1960

•FORTRAN (1955), the"**FOR**mula **TRAN**slator

•LISP, the"**LIS**t **P**rocessor",

•COBOL, the **CO**mmon **B**usiness **O**riented **L**anguage

•ALGOL **Algo**rithmic **L**anguage

- 1962 - APL
- 1964 - BASIC
- 1964 - PL/I
- 1970 - Pascal → Yapısal programlama
- 1970 - Forth
- 1972 - C
- 1972 - Prolog
- 1978 - SQL → Nesne yönelimli dillerin ortaya çıkışı
- 1983 - Ada
- 1983 - C++
- 1987 - Perl

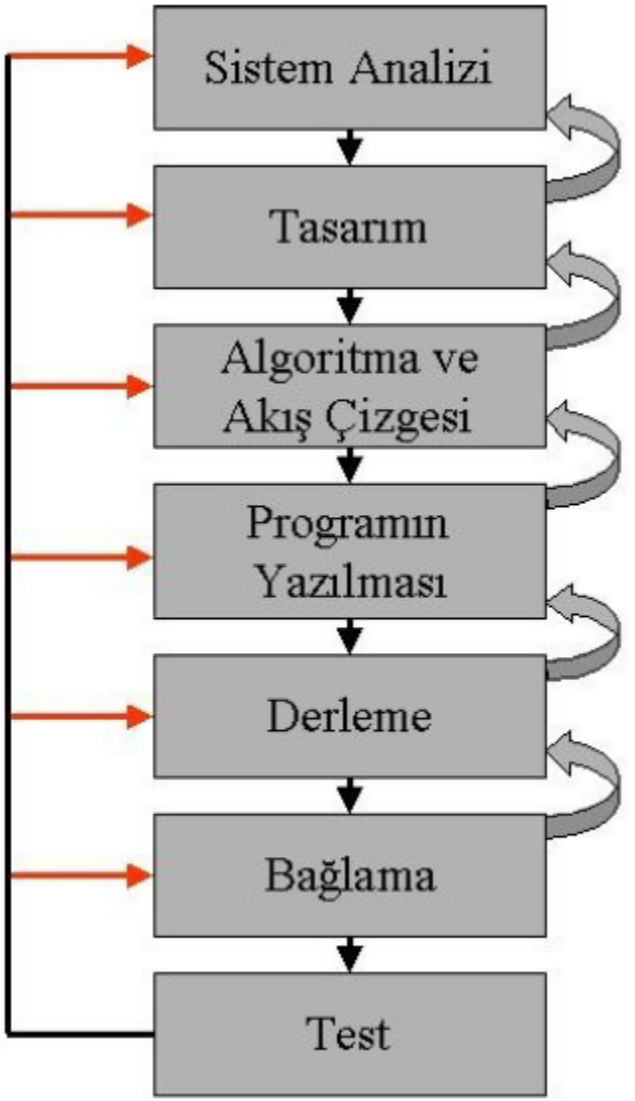
1990 lar, Internet

- 1991 - Python
- 1991 - Java
- 1995 - PHP
- 2000 - C#

Tamamı nesne yönelimli dillerdir. Yeni programlama kavramlarından ziyade, programlamanın kolaylaşmasını ve taşınabilirliği amaçlamaktadırlar

Yazılım Geliştirme

Yazılım Geliştirilirken Bir Programcı ve Yazılım Gurubunun takip edeceği adımlar şu şekildedir.



Sistem Analizi : Sorunun çözülebilmesi için tamamen anlaşılmasını sağlayan çalışmalardır.

Tasarım : İsteklerle ilgili olarak belirlenen bir takım çözümlerin tanımlanmasıdır.

Programlama Stili : Her yiğidin yoğurt yiyişi farklıdır. Aynı şekilde her programcı programındaki mantığı farklı kurar bu her programcının kendine özgün bir stili var anlamına gelir. Ancak bunun yanında Her programcının programın sağlığı bakımından dikkat etmesi gereken şeyler vardır. Örneğin kodlar açık olmalıdır. Kullanılan değişkenler kullanıldıkları amacı anlatır tarzda isimlendirilmelidir. Program içi dokümantasyona mutlaka önem verilmelidir.

Algoritma : Çözümün adımlarla ifade edilmesidir.

Akış Çizgesi : Algoritmanın şekillerle ifade edilmesidir.

Programlama Dili Seçimi : Çözümün netleşmesinden sonra yapılacak işlemleri kolay bir şekilde bilgisayar ortamına aktaracak dilin seçilmesidir. Önemli olan bu dilin özelliklerinin programcı tarafından iyi bilinmesidir.

Programın Yazılması : Seçilen Programlama dilinin kuralları kullanılarak program yazılmaya başlanır. bu amaçla çoğunlukla sade bir metin editörü kullanılır. Bazı durumlarda Syntax highlighting denilen bir özelliğe sahip olan daha akıllı editörler de kullanılabilir. Bazen de editör ile Programlama dilinin derleyicisinin, bağlayıcısının hatta hata ayıklayıcısının iç içe bulunduğu IDE (Integrated Development Environment) denilen türde derleyiciler kullanılır.

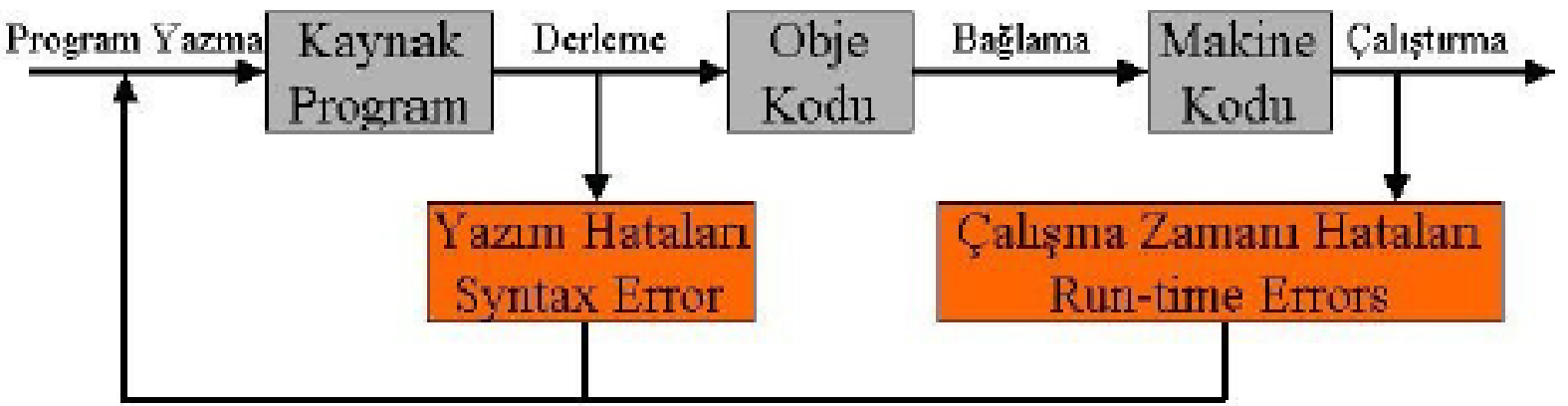
Derleme : Programlama Dili ile yazılmış programın yazım hatalarının olup olmadığının kontrol edilmesini ve ara kod olarak Obje kodun üretilmesini sağlama adımıdır.

Bağlama : Derlenmiş ara kod diğer kütüphane ve parça programlarla birleştirilerek Makine dilinde programın oluşturulması adımıdır. Ancak bazı IDE ortamlarda ve derleyicilerde Derleme ve Bağlama bir bütündür ve beraberce halledilirler. Programcının ayrıca bir bağlama işlemi yapması gerekmez işlemi yapması gerekmez.

Çalıştırma : Oluşturulan Makine dili Programının çalıştırılması adımıdır. Yukarıdaki adımların hepsi yolunda gittiyse program sorunsuz olarak çalışabilmelidir.

Test : Programın Mantıksal olarak test edilmesini sağlar ve içerik olarak her ihtimal için doğru sonuçlar üretip üretmediğini kontrol etmenizi sağlar.

Yaşam Döngüsünün Sağlanması : Yukarıdaki Akış Çizgesi dikkat edilirse aslında bir döngüdür. Hatta test aşamasında sorun çıkmazsa bile Sorunun tanımında yani ihtiyaçlarda bazı değişiklikler olursa adımlar baştan aşağı tekrar incelenmek zorunda kalınır. Bu çizgeye bir Yazılımımın Yaşam Döngüsü de denilebilir. Bu çizimde Yazılımın Bakım süreci göz önüne alınmamıştır.



Yazılım Geliştirme Araçları

- Editörler-Tümleşik geliştirme ortamları(IDE)
- Derleyicilerle birlikte kullanılır
- Derleyiciler-Bağlayıcılar (Compilers–Linkers)
- Yorumlayıcılar (Interpreter)

Editörler

Program kodlarını yazmak için kullanılan, metin düzenleyicilerdir. Program kodları saf metin biçiminde yazıldığından, herhangi bir metin düzenleyicisi, program yazılımı için kullanılabilir.

- Kodlamadaki hatalar görülmez.
- Anahtar kelimeler, fonksiyonlar ve parametreleri, vb.. Tanımlar ayrı renklendirilmediğinden kod yazmak daha zordur.
- Breakpoint, yada watch gibi, hata ayıklama unsurları yoktur.
- Notepad, Wordpad.. editör olarak kullanılabilir.
- Program derleme ve bağlama işlemi editör dışında genellikle komut satırı üzerinde yapılır.

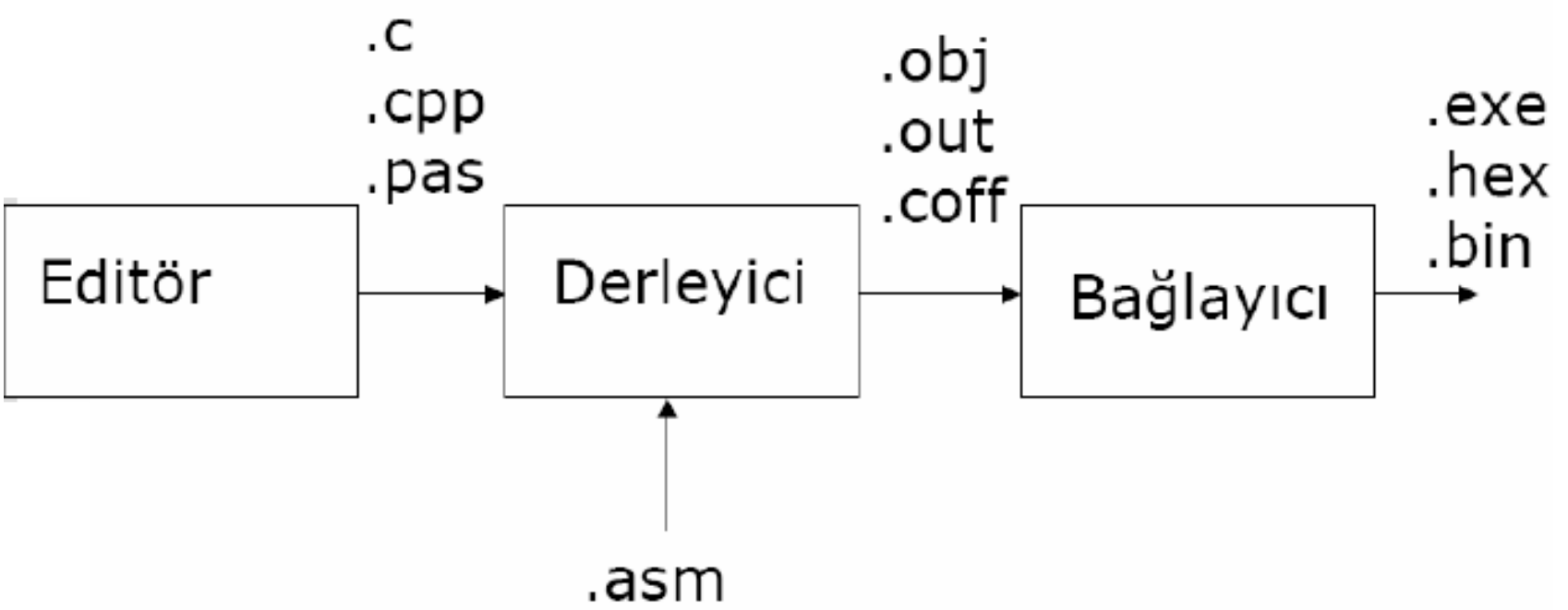
IDE (Tümleşik geliştirme ortamı)

Tümleşik geliştirme ortamları, Genellikle derleyicileri –bağlayıcıları ortam içinden kullanabilmeyi yada derleyici ve bağlayıcıya ortam içinden erişme yollarını sağlarlar (Makefilevs..). Bunun yanı sıra;

- Derleyici ve bağlayıcı tümleşik olan yapılarda Hata ayıklama, Gözlem penceresi gibi bileşenler mevcuttur.
- Yazım işlemini kolaylaştıracak vurgulamalar ve uyarılar mevcuttur.
- Derleyici ve bağlayıcı parametreleri menülerden ayarlanabilir.
- Yardımlar mevcuttur.
- Her yazılım dilinin kendi IDE si mevcuttur. Ancak bazı IDE'ler birden fazla yazılım dili için ortam sağlayabilir.

Derleyiciler: Bir derleyici, bir metin editörü yada IDE üzerinde yazılan program kodlarını, makinenin anlayabileceği OBJ kodlara dönüştüren bir uygulama yazılımıdır. Derleyicilerin birçoğu, Program dilinin yanısıra makine dilinin(assembly) de kullanılmasına izin verir.

Bağlayıcılar: Bir bağlayıcı, derleyici tarafından derlenmiş olan OBJ program kodlarını uygun bellek bölgelerine yerleştirerek, değişkenlerin ve sabitlerin bellek atamalarını ve ilklemelerini gerçekleyerek tek bir çalıştırılabilir program elde eden bir uygulama yazılımıdır (windows için exe dosya).
Örnek derleyiciler ve bağlayıcılar:
MS VC++ 6.0 için CL.exe derleyici, Link.exe bağlayıcı
Keil uVision 8051 için c51.exe derleyici, Ld51.exe bağlayıcı
gcc.exe açık kaynaklı ücretsiz bir derleyici ve bağlayıcı


.c
.cpp
.pas
.obj
.out
.coff
.exe
.hex
.bin
Editör
Derleyici
Bağlayıcı
.asm

YORUMLAYICILAR (INTERPRETERS)

Yorumlayıcılar, program kodunu bir bütün olarak değerlendirmez. Program kodunu satır, satır yorumlayarak çalıştırırlar. Bu nedenle günümüzde derleyicilere göre daha kısıtlı uygulamalara sahiptirler, internet uygulamaları ve bilimsel alanda yaygın kullanılmaktadırlar.

MATLAB

Elektronik sistemlerde dört farklı sayı sistemi kullanılır. Bunlar;

i) İkili(Binary) Sayı Sistemi
ii) Onlu(Decimal) Sayı Sistemi
iii) Onaltılı(Heksadecimal) Sayı Sistemi
iv) Sekizli(Oktal) Sayı Sistemi

İkili binary sistemi

Bu sistemde **0** ve **1** olmak üzere 2 tane sembol vardır ve bu sebeple ikili sayı sistemi denir. Her birine bir "dijit" denir ve bir biti temsil eder. BIT ifadesi de **Bİ**nary digi**T**'ten gelmektedir. Elektronik sistemlerde ***0 volt*** →lojik **0**, ***5 volt veya 3.3 volt*** →lojik **1** değeri ile ifade edilir. Sayı tabanı 2'dir.

Basamak	7	6	5	4	3	2	1	0
Sayı	1	0	1	1	1	0	0	1
	MSB							LSB

Binary sayılar yazılırken en sağdaki basamağa en düşük değerlikli bit (Least Significant Bit-LSB), en soldaki basamağa en yüksek değerlikli bit (Most Significant Bit-MSB) adı verilir.

Bu sistemde tüm sayısal değerler 0 ve 1'ler ile ifade edilir.

Onluk	Birary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

İkili sayı sisteminde tümleyen işlemleri:

İkili sayı sistemde çıkarma ve mantık işlemlerini daha basit hane getirmek amacıyla tümleyenler kullanılır.

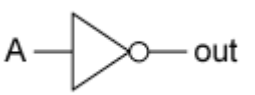
Ayrıca negatif sayıların elektronik olarak saklanmasında tümleyen yöntemleri kullanılır.

Tümleyen işlemleri iki şekilde yapılabilir; Bunlar 1'e tümleyen ve 2'ye tümleyendir.

1'e Tümleyenin Bulunması

Bire tümleyen için Binary bir sayının her biti terslenir(0→1 ve 1→0 yapılır). Bu işlem elektronik devrelerde olarak değil kapıları ile yapılır.

Sayı	0	1	11	110	1110 0101
Bire Tümleyen	1	0	00	001	0001 1010



2'ye Tümleyenin Bulunması

I.Yol

Binary bir sayıyının 2'ye tümleyenini elde etmek için önce 1'e tümleyenine 1 eklenir ve işlem sonucunda sola taşma olursa soldan bir bit silinir.

Sayı	0	1	11	110	1110 0101
Bire Tümleyeni	1	0	00	001	0001 1010
İkiye Tümleyeni	~~1~~0	1	01	010	0001 1011

II.Yol

Sağdan sola doğru rastlanan ilk 1 ve öncesindeki 0'lar aynen yazılır, 1'in solundaki bitler terslenerek yazılır.

Sayı	0	1	11	110	1110 0101
İkiye Tümleyeni	0	1	01	010	0001 1011

Hexadecimal sayılar (Hex)

Bilgisayar sistemlerinde uzun bit dizilerini temsil etmek zor olacağı için yazım biçimi olarak hexadecimal sayılar tercih edilmektedir. Hex sayılar 16 lık sayılardır. Herbir Nibble bir Hex sayı ile temsil edilebilir. Böylelikle ikili sayının yazım uzunluğu 4 te bir digite düşecektir.

Hex sistemde sayılar 16 sembolden oluşur ve aşağıdaki gibidir.

0000	0	0100	4	1000	8	1100	C
0001	1	0101	5	1001	9	1101	D
0010	2	0110	6	1010	A	1110	E
0011	3	0111	7	1011	B	1111	F

Örnek: 0011 1010 = 3A Hex, 1110 0101 = E5 Hex
0101 1101 1100 1001 0110 0111 =5DC967 Hex

Bir çok kodlama türü olmasına karşın dünyada bilgisayar ortamlarında ANSI tarafından 1963 yılında standartlaştırılan **ASC**II(**A**merican **N**ational**C**ode for Information Interchange) kodlaması yoğun olarak kullanılmaktadır. Ancak günümüzde , ASCII kodları çok dilliği sağlayabilmek için yetersiz kaldığından UNICODE kodlaması yaygınlaşmaktadır. Ancak pek çok uygulamada ASCII kodlaması hala geçerliliğini korumaktadır.

ASCII temel olarak 7 bit' tir. 127 karakterden oluşur. Ama Extended kısmıyla birlikte 8 bit kullanılmaktadır. Ancak genişletilmiş kısımdaki semboller yazılım ortamına göre değişebilmektedir.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	*[NULL]*	32	20	*[SPACE]*	64	40	@	96	60	`
1	1	*[START OF HEADING]*	33	21	!	65	41	A	97	61	a
2	2	*[START OF TEXT]*	34	22	"	66	42	B	98	62	b
3	3	*[END OF TEXT]*	35	23	#	67	43	C	99	63	c
4	4	*[END OF TRANSMISSION]*	36	24	$	68	44	D	100	64	d
5	5	*[ENQUIRY]*	37	25	%	69	45	E	101	65	e
6	6	*[ACKNOWLEDGE]*	38	26	&	70	46	F	102	66	f
7	7	*[BELL]*	39	27	'	71	47	G	103	67	g
8	8	*[BACKSPACE]*	40	28	(	72	48	H	104	68	h
9	9	*[HORIZONTAL TAB]*	41	29	)	73	49	I	105	69	i
10	A	*[LINE FEED]*	42	2A	*	74	4A	J	106	6A	j
11	B	*[VERTICAL TAB]*	43	2B	+	75	4B	K	107	6B	k
12	C	*[FORM FEED]*	44	2C	,	76	4C	L	108	6C	l
13	D	*[CARRIAGE RETURN]*	45	2D	-	77	4D	M	109	6D	m
14	E	*[SHIFT OUT]*	46	2E	.	78	4E	N	110	6E	n
15	F	*[SHIFT IN]*	47	2F	/	79	4F	O	111	6F	o
16	10	*[DATA LINK ESCAPE]*	48	30	0	80	50	P	112	70	p
17	11	*[DEVICE CONTROL 1]*	49	31	1	81	51	Q	113	71	q
18	12	*[DEVICE CONTROL 2]*	50	32	2	82	52	R	114	72	r
19	13	*[DEVICE CONTROL 3]*	51	33	3	83	53	S	115	73	s
20	14	*[DEVICE CONTROL 4]*	52	34	4	84	54	T	116	74	t
21	15	*[NEGATIVE ACKNOWLEDGE]*	53	35	5	85	55	U	117	75	u
22	16	*[SYNCHRONOUS IDLE]*	54	36	6	86	56	V	118	76	v
23	17	*[ENG OF TRANS. BLOCK]*	55	37	7	87	57	W	119	77	w
24	18	*[CANCEL]*	56	38	8	88	58	X	120	78	x
25	19	*[END OF MEDIUM]*	57	39	9	89	59	Y	121	79	y
26	1A	*[SUBSTITUTE]*	58	3A	:	90	5A	Z	122	7A	z
27	1B	*[ESCAPE]*	59	3B	;	91	5B	[	123	7B	{
28	1C	*[FILE SEPARATOR]*	60	3C	<	92	5C	\	124	7C	|
29	1D	*[GROUP SEPARATOR]*	61	3D	=	93	5D	]	125	7D	}
30	1E	*[RECORD SEPARATOR]*	62	3E	>	94	5E	^	126	7E	~
31	1F	*[UNIT SEPARATOR]*	63	3F	?	95	5F	_	127	7F	*[DEL]*

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

Matematiksel işlemlerin öncelik sırası ?

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

NOT: Bilgisayar diline kodlanmış bir matematiksel ifadede, aynı önceliğe sahip işlemler mevcut ise bilgisayarın bu işlemleri gerçekleştirme sırası soldan sağa(baştan sona) doğrudur.

Örneğin ; Y=A*B/C
Önce A*B işlemi yapılacak, ardından bulunan sonuç C ye bölünecektir.

MATLAB İSLEÇLER

İşlem yapmayı sağlayan sembollerdir.

- +, -, *, /\
- ^ (üs alma, a^b) : 2^3
- ‘‘ (Tek tırnak arası) : ‘Ali‘(Metin girişlerinde kullanılır)
- ‘: Transpoze
- ( ): 2*(3-4) (İşlem sırasını belirler)
- [ ] : Dizi gösteriminde kullanılır.
- =: x=3
- ==: x==K
- ? : Programın işleyişini etkilemeyen açıklama satırlarını gösterir.
- ! : DOS moduna geçer.

Aritmetik işlemlerde, işleç öncelik sırası (precision) vardır.
dot(x,y)=sum(x.*y)

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

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

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

Mantıksal işlem	Matematiksel sembol	Komut
Ve	.	And
Veya	+	Or
değil	‘	Not

“ve,veya,değil “ operatörleri hem matematiksel işlemlerde hem de karar ifadelerinde kullanılırlar.

- Bütün şartların sağlatılması isteniyorsa koşullar arasına VE
- Herhangi birinin sağlatılması isteniyorsa koşullar arasına VEYA
- Koşulu sağlamayanlar isteniyorsa DEĞİL

  mantıksal operatörü kullanılır.

Mantıksal Işlemler

Örnek; Bir işyerinde çalışan işçiler arasından yalnızca yaşı 23 üzerinde olup, maaş olarak asgari ücret alanların isimleri istenebilir.

Burada iki koşul vardır ve bu iki koşulun da doğru olması gerekir. Yani;

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

1.KOŞUL *2.KOŞUL*

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

Örnek; Bir sınıfta Bilgisayar dersinden 65 in üzerinde not alıp, Türk Dili veya Yabancı Dil derslerinin herhangi birinden 65 in üzerinde not alanların isimleri istenmektedir.

Burada 3 koşul vardır ve Bilgisayar dersinden 65 in üzerinde not almış olmak temel koşuldur. Diğer iki dersin notlarının herhangi birinin 65 in üzerinde olması gerekir.

Eğer ;

Bilg.Not>65 ve (TDili not>65 veya YDil not>65) ismi Yaz

Karşılaştırma İşlemleri

Bilgisayar temel matematiksel işlemlerin yanında karşılaştırma/karar işlemlerini de yapabilir. Yani iki büyükten hangisinin büyük veya küçük olduğu, iki değişken değerinin birbirine eşit olup olmadığı, alfabetik olarak önce veya sonra mı olduğu gibi konularda karar verebilir.

Algoritma çalışmasını inceleyelim

```
1. Başla.
2. A sayısını gir
3. B sayısını 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 1.13'te, girilen değişik sayılarla oluşan program çıktıları verilmektedir.

Tablo 1.13: Örnek-1.8'deki algoritmanın uygulama sonuçları

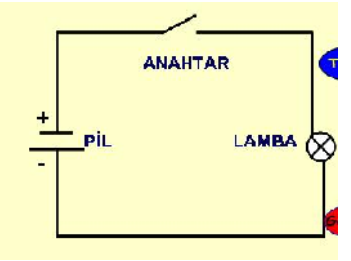
Girilen A sayısı	Girilen B sayısı	Ekrana yazılan sonuç
3	7	B sayısı, A sayısından büyüktür.
33	11	A sayısı, B sayısından büyüktür.
99	99	İki sayı birbirine eşittir.

Algoritma çalışmasını inceleyelim

Örnek-1.9: Aşağıdaki algoritmanın çalışmasını inceleyiniz.

```
1. Başla.
2. Bir büyük harf veya kelime (A) gir
3. İkinci büyük harfi veya kelimeyi (B) 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
```

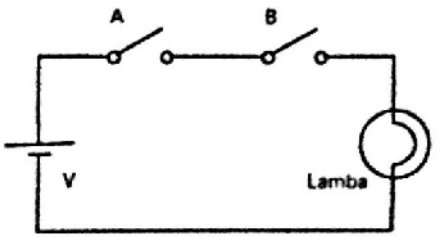
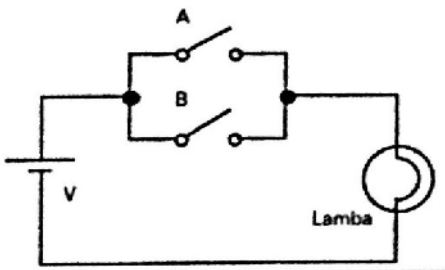
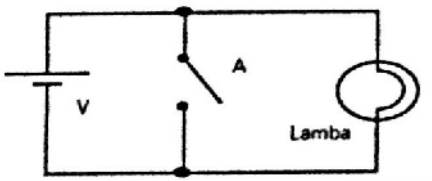
Verilen algoritmada; klavyeden iki karakter veya kelime girilmekte ve birbiriyle karşılaştırılmaktadır. Aslında bu karşılaştırma; karakterler arasında değil, karakterlerin ASCII kodları (bkz. Ek-A) arasındadır. Örneğin klavyeden "A" karakterine basıldığında, bilgisayar ASCII kod tablosundaki karşılığı 65 tamsayısının ikili tabandaki karşılığıyla işlemlerini gerçekleştirir (Tablo 1.14).



Tablo 1.17: Temel mantıksal işlem karşılıkları

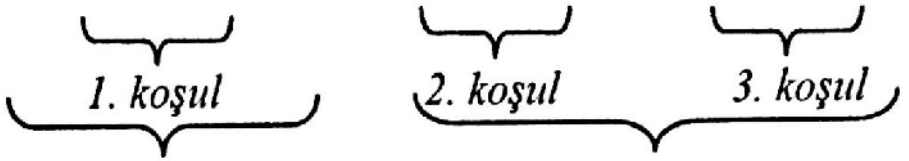
Mantıksal işlem	Komut olarak	Matematiksel sembol olarak
VE	AND	.
VEYA	OR	+
DEĞİL	NOT	'

Tablo 1.18: Temel mantıksal işlemler

İşlem	Elektriksel 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.

```
Eğer  BN>65   VE   (TDN>65 VEYA YDN>65) ise Yaz İsim
```



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

şeklinde yazılabilir. Örnek uygulama değerleri Tablo 1.22'de verilmektedir.

Tablo 1.22: Örnek-1.11 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

ALGORİTMA

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

"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.

Problem Çözme

Problem Çözme Tekniği (Descartes'e göre):

1. Doğruluğu kesin olarak kanıtlanmadıkça, hiçbir şeyi doğru olarak kabul etmeyin; tahmin ve önyargılardan kaçının.
2. Karşılaştığınız her güçlüğü mümkün olduğu kadar çok parçaya bölün.
3. Düzenli bir biçimde düşünün; anlaşılması en kolay olan şeylerle başlayıp yavaş yavaş daha zor ve karmaşık olanlara doğru ilerleyin.
4. Olaya bakışınız çok genel, hazırladığınız ayrıntılı liste ise hiçbir şeyi dışarıda bırakmayacak kadar kusursuz ve eksiksiz olsun.

Problem çözme sırası

1. **Problemi anlama** (Understanding, Analyzing),
2. **Bir çözüm yolu geliştirme** (Designing),
3. **Algoritma ve program yazma** (Writing),
4. **Tekrar tekrar test etme** (Reviewing)

Problem Çözme

Bir problemi çözmek için yazılacak programda, genel olarak, aşağıdaki yazılım geliştirme aşamaları uygulanmalıdır.

Yazılım Geliştirme Aşamaları

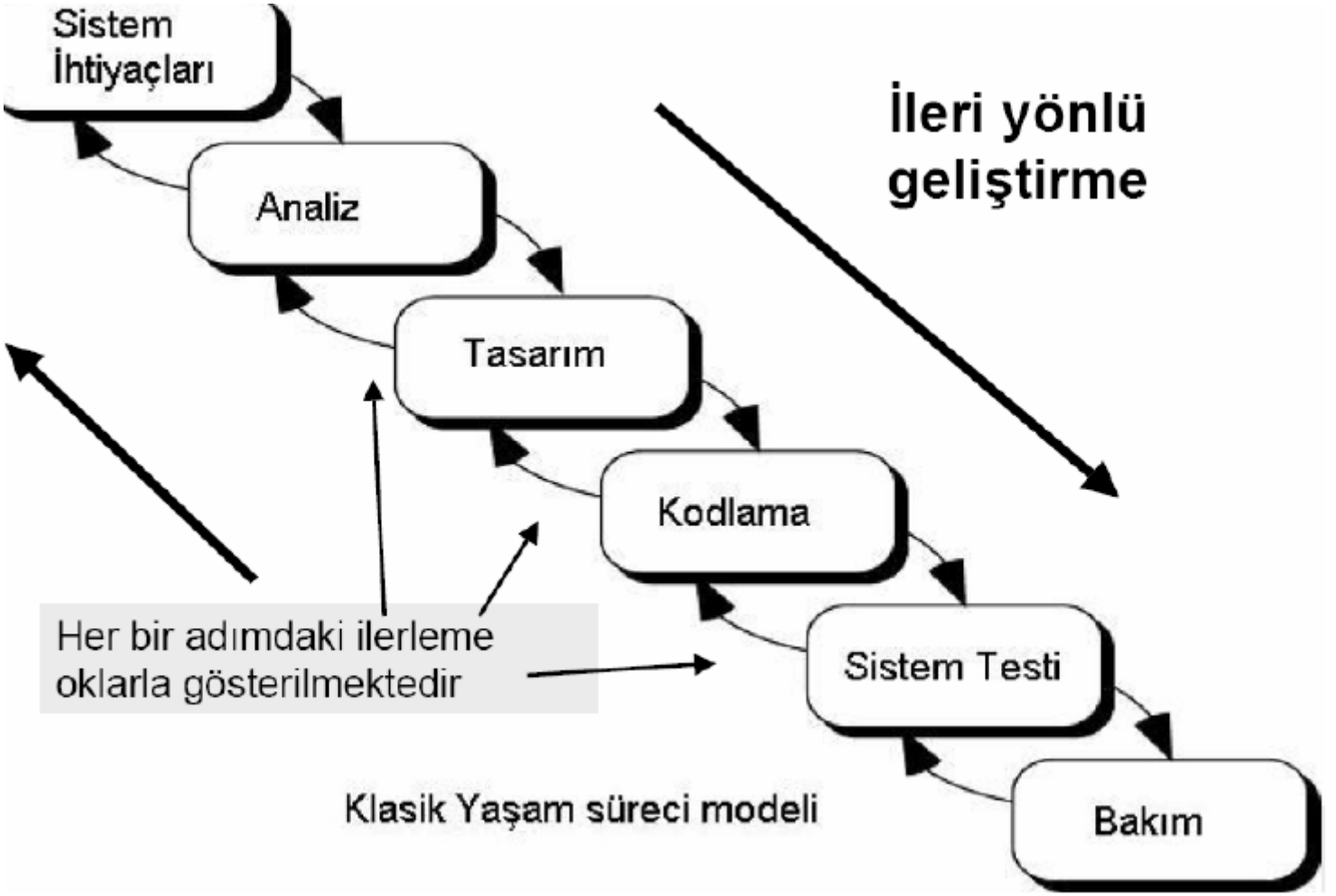
1. Problemin Analizi: Problemin tam olarak ne olduğunun anlaşılmasıdır. Bu nedenle, problemin çözümünden neler beklendiği ve yaratacağı çözümün girdi ve çıktılarının neler olacağı kesin olarak belirlenmelidir.
2. Tasarım: Problemi çözmek için kullanılacak çözüm adımlarını gösteren bir liste yapılması gereklidir. Bir problemin çözüm adımlarını gösteren bu listeye algoritma denir. Böyle bir liste tasarlanırken, ilk önce problemin ana adımları çıkarılır; daha sonra her adım için, gerekiyorsa, daha ayrıntılı bir çözüm tasarlanır.
3. Kodlama: Kağıt üzerinde geliştirilen algoritma, programcının tercih ettiği bir programlama diline çevrilir.

Problem Çözme

4. Test etme: Program değişik girdiler ile çalıştırılarak ürettiği sonuçlar kontrol edilerek test işlemi gerçekleştirilir. Sonuçlar beklendiği gibi ise , programın doğru çalıştığı kanıtlanmış olunur; değilse doğru çalışmayan kısımları tespit edilerek düzeltilir.

5. Belgeleme: Bütün bu çalışmaların belli bir dosyalama sistemi içinde belgeler halinde saklanmasının sağlandığı aşamadır.

6. Bakım: Programın güncel koşullara göre yeniden düzenlenmesini içeren bir konudur. Oluşan hataların giderilmesi,, yeni eklemeler yapılması ya da programın teknolojisinin yenilenmesi gibi işlemlerdir.



Klasik Yaşam süreci modeli

Problem Çözme

Bir problem çözülürken biri algoritmik, diğeri herustic(sezgisel) olarak adlandırılan iki yaklaşım vardır.

Algoritmik yaklaşımda, çözüm için olası yöntemlerden en uygun olanı seçilir ve yapılması gerekenler adım adım ortaya konulur.

Herustic yaklaşımda ise, çözüm açıkça ortada değildir. Tasarımcının deneyimi, birikimi ve o andaki düşüncesine göre problemi çözecek bir şeylerin şekillendirilmesiyle yapılır. Program tasarımcısı, algoritmik yaklaşımla çözemediği, ancak çözmek zorunda olduğu problemler için bu yaklaşımı kullanır.

Algoritmik Yaklaşım

Algoritma, herhangi bir sorunun çözümü için izlenecek yol anlamına gelmektedir. Çözüm için yapılması gereken işlemler hiçbir alternatif yoruma izin vermeksizin sözel olarak ifade edilir. Diğer bir deyişle algoritma, verilerin, bilgisayara hangi çevre biriminden girileceğinin, problemin nasıl çözüleceğinin, hangi basamaklardan geçirilerek sonuç alınacağının, sonucun nasıl ve nereye yazılacağının sözel olarak ifade edilmesi biçiminde tanımlanabilir.

Algoritma hazırlanırken, çözüm için yapılması gerekli işlemler, öncelik sıraları gözönünde bulundurularak ayrıntılı bir biçimde tanımlanmalıdırlar.

Örnek 1: Verilen iki sayının toplamının bulunmasının algoritması aşağıdaki gibi yazılır.

Adım 1 – Başla
Adım 2 – Birinci Sayıyı Oku
Adım 3 – İkinci Sayıyı Oku
Adım 4 – İki Sayıyı Topla
Adım 5 – Dur

Algoritmik Yaklaşım

Algoritmalar iki farklı şekilde kağıt üzerinde ifade edilebilirler;

1. Pseudo Code (*Kaba Kod* veya *Yalancı Kod* veya *Sözde Kod*), bir algoritmanın yarı programlama dili kuralı, yarı konuşma diline dönük olarak ortaya koyulması/ tanımlanmasıdır. Bu şekilde gösterim algoritmayı genel hatlarıyla yansıtır.

2. Akış şeması, algoritmanın görsel/şekilsel olarak ortaya koyulmasıdır. Problemin çözümü için yapılması gerekenleri, başından sonuna kadar, geometrik şekillerden oluşan simgelerle gösterir.

Sözde (Pseudo) Kod

Sözde programlar, doğrudan **konuşma dilinde** ve programlama mantığı altında, **eğer, iken** gibi koşul kelimeleri ve **> = <** gibi ifadeler ile beraber yazılır. İyi bir biçimde yazılmış, sözde koddan, programlama diline kolaylıkla geçilebilir.

Örnek: Verilen bir sıcaklık derecesine göre suyun durumunu belirten bir sözde program yazınız.

- Örnek Giriş/Çıkış
  - Bu Program, Sıcaklığa göre suyun durumunu gösterir
  - Su, Buz, Buhar
  - ----------
  - Lütfen derece cinsinden sıcaklığı giriniz: 140
  - BUHAR elde edeceksiniz.

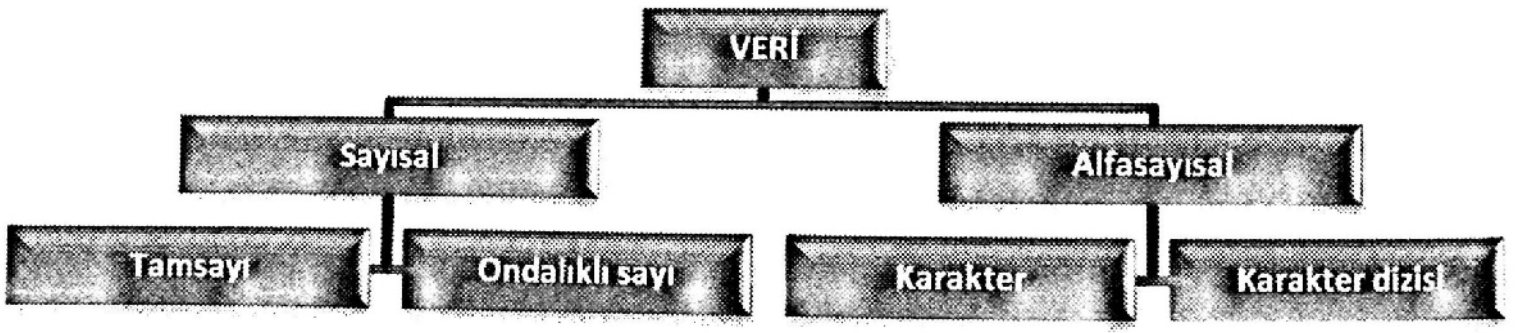
Sözde (Pseudo) Kod

İstenilen programın Pseudo Kodu:

1. Program açıklama mesajı yaz.
2. Kullanıcın sıcaklığı girmesi için bir uyarı mesajı yaz.
3. Girilen Sıcaklığı Oku.
5. Eğer Sıcaklık < 0 ise Durum="Buz"
6. Eğer Sıcaklık>= 100 ise Durum="Buhar"
7. Değilse Durum ="Su"
8. Sonucu Yaz.

A. Veri

Bilgisayarlarda işlenen tüm bilgiler "*veri*" olarak adlandırılırlar ve genel olarak iki gruba ayrılırlar: sayısal ve alfasayısal



Bilgisayarda işlenen veriler

Tablo 2.1. Algoritmalarda/akış diyagramlarında kullanılan [illegible]

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

a. ***Sayısal veriler***: Sayısal ağırlığı (değeri) olan verilerdir. Tamsayılar, ondalıklı sayılar vb. bu grubun içine girer. Sayısal veriler bilgisayara herhangi bir tabanda veya üstel biçimde aktarılabilirler (girilebilirler).

Not

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

Sayıüs veya sayıEüs anlamı sayı. $10^{üs}$ dür

Örneğin `12.34E2=12,34.10²=1234` veya `1024e-3=1024.10⁻³=1,024` tür.

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.

Aşağıda sayısal ve alfasayısal verilere ilişkin birkaç örnek verilmiştir.

Sayısal veri	Alfasayısal veri
2004	Bursa
-555	a
6.78	;
2013	"2013"

B. Tanımlayıcı

Programı yazan kişi tarafından düşünülüp oluşturulan ve programdaki değişkenleri, sabitleri, paragrafları, kayıt alanlarını, özel veri/bilgi tiplerini, alt programları vb. adlandırmak için kullanılan kelimelere/ifadelere "*tanımlayıcı*" denir. Prensip olarak; programdaki tanımlayıcı kelimelerin, yerini tuttukları ifadelere çağrışım yapacak şekilde seçilmesi/oluşturulması daha uygundur. Böylece diğer programcıların kodları incelemesi, anlaması ve geliştirmesi/düzenlemesi daha da kolaylaşacaktır. Bu kelimeler 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. Tanımlayıcı isimleri, harf veya alt çizgi ile başlayabilir.
v. Tanımlayıcı ismi, rakamla başlamamalı veya sadece rakamlardan oluşmamalıdır (sayı olmamalıdır).
vi. Tanımlayıcı ismi, ilgili programlama dilinin komutu veya saklı (anahtar) kelimelerinden olmamalıdır.

C. Değişken

Programın her çalıştırılmasında, farklı değerler alabilen/aktarılabilen bellek/veri/bilgi alanları, "*değişken*" olarak adlandırılır.

"İsimlendirme kurallarına" uygun olarak oluşturulan değişken isimlerinin; yerini tuttukları/aldıkları verilere çağrışım yapacak şekilde olması, programın anlaşılırlığı açısından önemlidir. Örneğin; bir kişinin adının aktarıldığı değişken '`Ad`', adı ve soyadının aktarıldığı '`Adsoy`', telefonunun aktarıldığı '`tel`', ev adresinin aktarıldığı '`EvAdres`', işyeri adresinin aktarıldığı '`isadres`', vb. şeklinde oluşturulabilir.

Örnek-2.2:

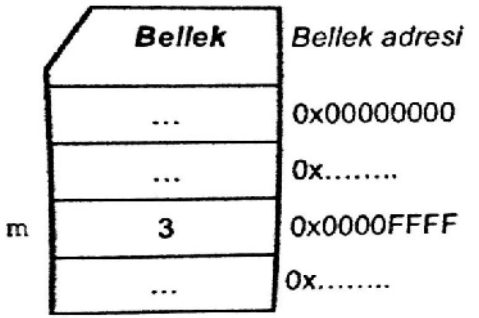
```
1. Başla
2. Öğrencinin numarasını (No) gir
3. Öğrencinin adını ve soyadını (Adsoy) gir
4. Öğrencinin vize notunu (Vize) gir
5. Öğrencinin final notunu (Final) gir
6. Ort = 0.3*Vize + 0.7*Final
7. Numara (No) ve ortalamayı (Ort) yaz
8. Dur
```

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 2.3'te verilmektedir.

Tablo 2.3: Örnek-2.2 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 bildirilmektedir. Değişken bildirimi yapıldığında; veri türüne göre bellekte yer açılmakta ve program içinde bunlara de-



Şekil 2.3: Bellek ve değer

Yukarıda değişken konusu incelenmiştir. Işaretçiler 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 oluşturulmaktadır.

Örnek-2.3: Yarıçapı girilen dairenin çevresiyle alanını hesaplayıp yazdıran programın algoritması.

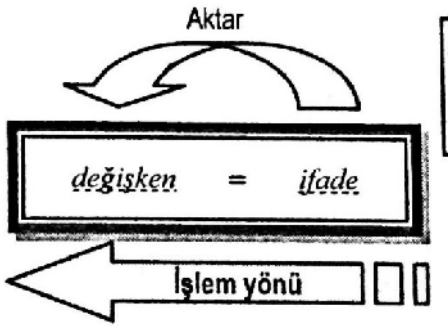
```
1. Başla
2. pi_sayisi=3.14
3. Dairenin yarıçapını (r) gir
4. cevre=2*pi_sayisi*r
5. alan=pi_sayisi*r*r
6. Çevre(cevre) ve alanı(alan) yaz
7. Dur
```

E. Atama/aktarma

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

```
değişken = ifade
```

satırında '*değişken*' ile gösterilen kısım, herhangi bir değişkenin/sabitin 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 değişkenin - varsa - bir önceki değeri (eski değeri) silinir



İşlem adımları:

I. Sağdaki işlemi yap veya sağdaki ifadeyi gerçekleştir.
II. Bulunan sonucu, soldaki değişkene aktar.

değişken ← *ifade*

Şekil 2.4: Aktarma işlemi

Tablo 2.4: 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

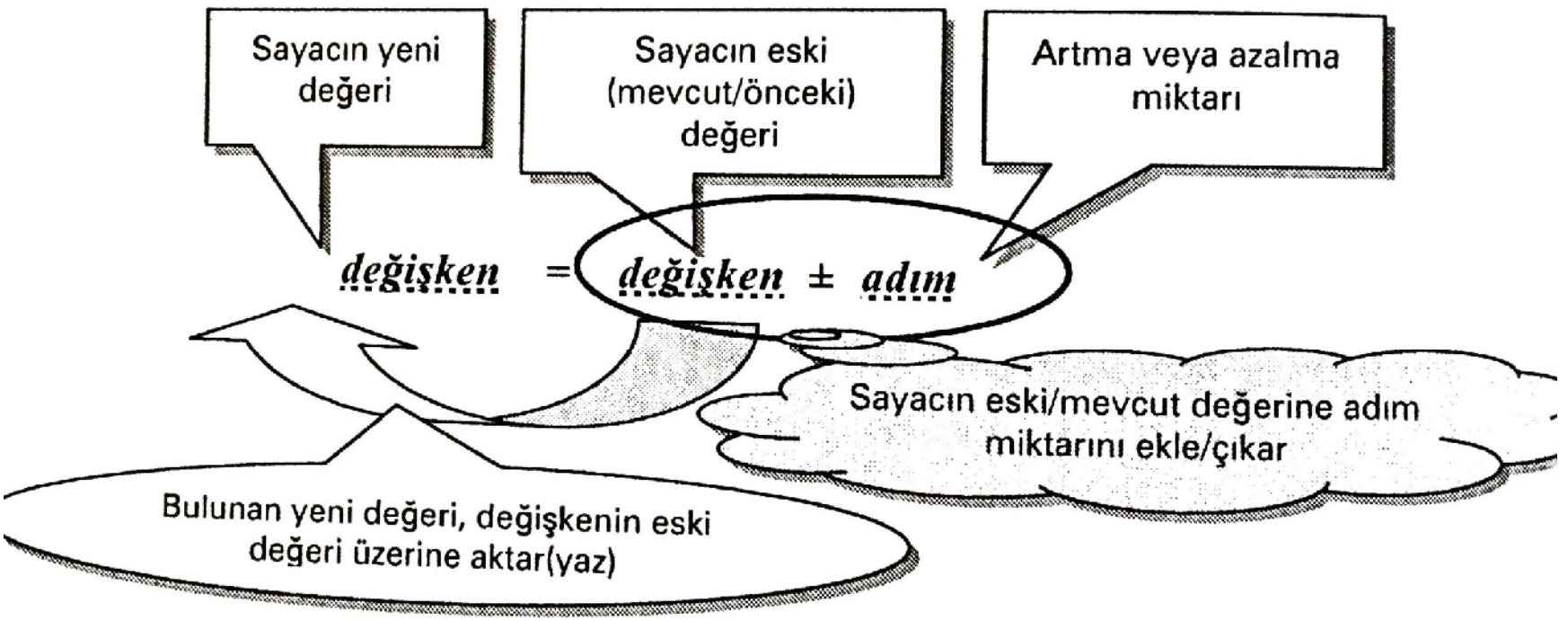
Aktarma işlemiyle ilgili örnekler Tablo 2.4'te görülmektedir. Burada her ifade 3 kez arka arkaya çalıştırılmaktadır. Bulunan her yeni sonuçta, eski sonucun hiçbir etkisi yoktur. Çünkü bulunan her yeni sonuç; eskisinin üzerine yazılmakta, dolayısıyla onu silmektedir.

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.

```
Say = Say + 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.



Sayaç işlemi

```
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 2.6 Örnek-2.5'teki 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

- Döngü değişkenine başlangıç değeri verilir
- Döngünün artma veya azalma miktarı belirlenir
- Döngünün bitiş değeri belirlenir
- Eğer döngü, karar/karşılaştırma ifadeleriyle oluşuyorsa, döngü değişkeni, döngü içinde adım miktarı kadar arttırılmalı / azaltılmalıdır.

Örnek-2.6: Aşağıdaki algoritma 1-10 arası tek sayıların toplamını hesaplamakta ve çalışma prensibi Tablo 2.7'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          Döngü
7. Git 4
8. Yaz T
9. Dur
```

Tablo 2.7 Örnek-2.6'daki 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 benzemektedir. 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.

N adet sayının ortalaması, bu sayıların toplamının N 'e bölünmesi ile 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 2.8'de 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 2.8 Örnek-2.7 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-2.8: Klavyeden girilen N sayısının faktöriyelini hesaplayan programın algoritması.

Algoritmada, klavyeden N sayısı girilerek faktöriyeli hesaplanmaktadır. N=5 için programın çalışması Tablo 2.9'da verilmektedir.

```
1. Başla
2. N sayısını gir
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 2.9: Örnek-2.8 için uygulama sonuçları

Eski S	Yeni S	Eski Fak	Yeni Fak	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 hazırlarken aşağıda verilen adımlar izlenebilir.

Algoritma hazırlama kuralları

Algoritma hazırlama kuralları

i. Yapılacak iş/çözülecek problem iyice irdelenir. Tüm olasılıklar gözden geçirilir.
ii. En az 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 işlemler/kullanılacak yöntemler açık şekilde verilir.
vii. Bulunan sonuçlar görüntülenir veya belirli ortamlarda saklanır.

Algoritma hazırlama kuralları (özet)

i. Tanımlayıcı isimleri belirlenir.
ii. Veri girişleri yapılır veya çevre birimlerden okutulur.
iii. Yapılacak işlemler ve yöntemler yazılır.
iv. Sonuçlar yazdırılır veya çevre birimlerde saklanır.

Algoritmanın/akış diyagramının avantajları

i. Program yazmayı kolaylaştırır.
ii. Hatalı kodlama oranını azaltır.
iii. Program yazımı için geçen süreyi kısaltır.
iv. İşlem akışını açık bir şekilde gösterdiğinden program kontrolünü ve hata takibini kolaylaştırır.
v. Sonradan yapılacak düzenlemelerde kolaylıklar sağlar.

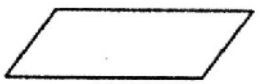
AKIŞ DİYAGRAMI

A. Başla/Dur

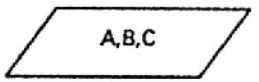
Şekil 3.1: Başla/Dur şekilleri

'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



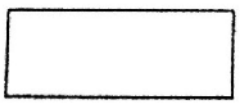
Şekil 3.2: Veri girişi şekli



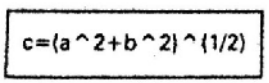
Şekil 3.3: Veri girişi örneği

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' yazıyorsa 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 çalıştırılırsa; 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

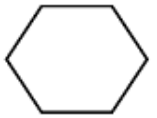
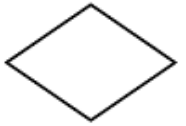
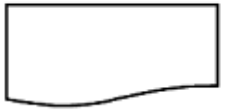
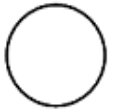
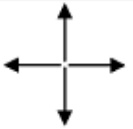


Şekil 3.5: İşlem örneği

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.

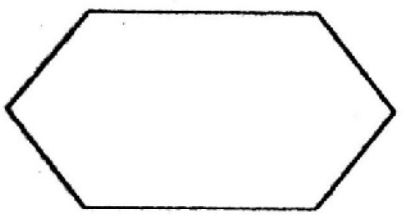
AKIŞ DİYAGRAMI

Programlamanın temeli olan algoritmanın, özel geometrik şekillerle çizilmiş gösterimine ‘akış diyagramı’ denir. Yazılımcı; programını her hangi bir dilde kodlamadan önce onun algoritmasını yazabilir veya akış diyagramını çizebilir.

Sembol	Sembolün Adı ve Anlamı
	**Elips:** Akış diyagramının başlangıç ve bitiş yerlerini gösterir. Başlangıç simgesinden çıkış oku vardır. Bitiş simgesinde giriş oku vardır.
	**Paralel Kenar:** Programa veri girişi için kullanılır.
	**Dikdörtgen:** Aritmetik işlemler ve her türlü atama işlemlerinin temsil edilmesi için kullanılır.
	**Altıgen:** Program içinde belirli blokların art arda tekrar edileceğini gösterir.
	**Eşkenar Dörtgen:** Karar verme işlemini temsil eder.
	**Belge:** Ekrana veya yazıcıya bilgi çıkışı için kullanılır.
	**Daire:** Birleştirici veya bağlantı noktalarını temsil eder.
	**Oklar:** Diyagramın akış yönünü, yani herhangi bir adımdaki işlem tamamlandıktan sonra hangi adıma gidileceğini gösterir.


Başla/Bitir
Döngü
Genel
Girdi/Çıktı
Manyetik
Disk
Genel İşlem
Yordam
Çağırma
Denetim
(Karar)
Akış
Yönü
Yazıcı Çıktısı
Bağlaç
Görüntü Çıktısı
Sayfa
Bağlacı
Ele ile Girdi
(Klavye)

D. Döngü



Döngü şekli

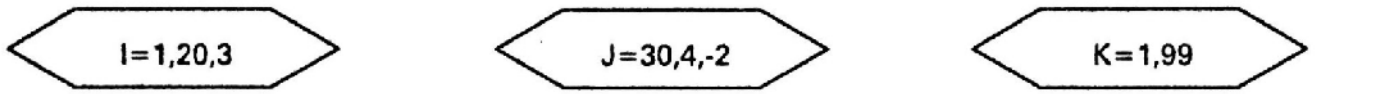
Birçok programda; belirli işlem veya işlem blokları aynı veya farklı/ardışık değerlerle ya da bazı koşullar sağlanıncaya kadar tekrarlanır. Bu tekrarlamalı işlemler '*döngü*' (*çevrim*) olarak isimlendirilir. Döngü şeklinin içine; döngü (çevrim, kontrol) değişkeninin başlangıç-bitiş- adım değerleri yazılır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

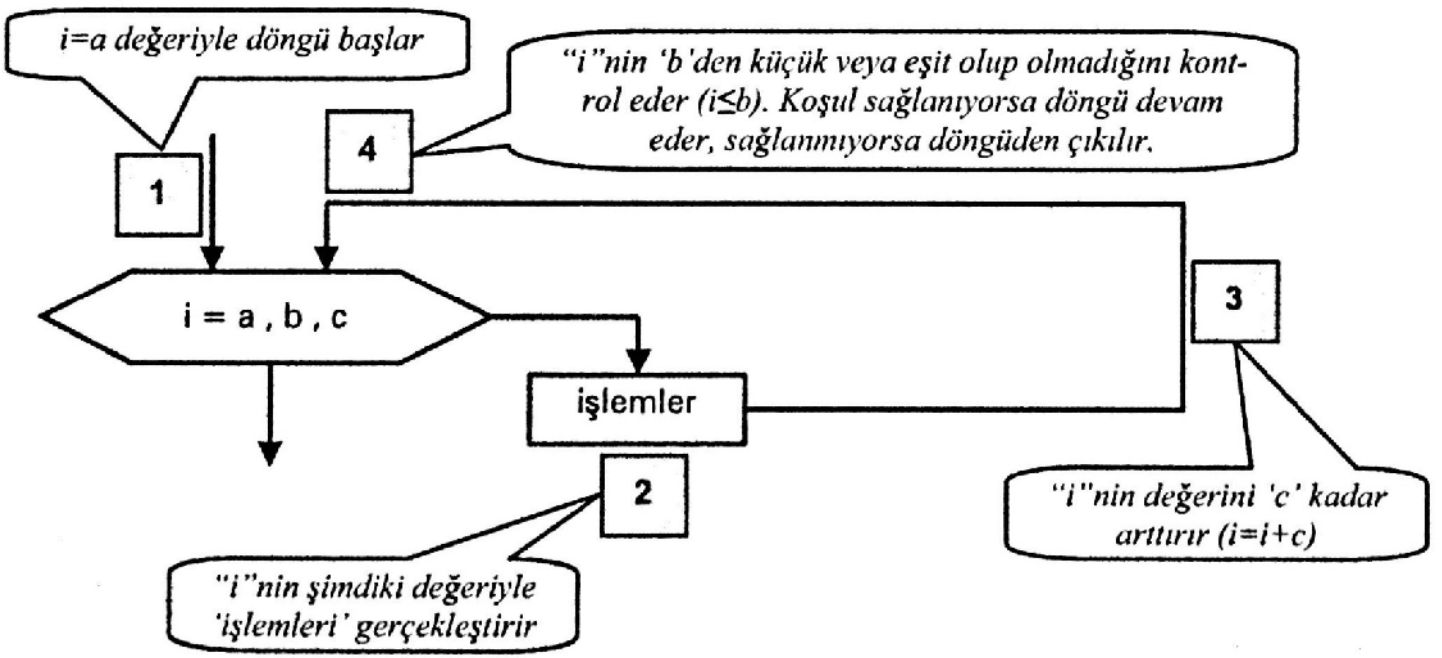


1'den 20'ye kadar 3'er artan 'I' döngüsü

30'dan 4'e kadar 2'şer azalan 'J' döngüsü

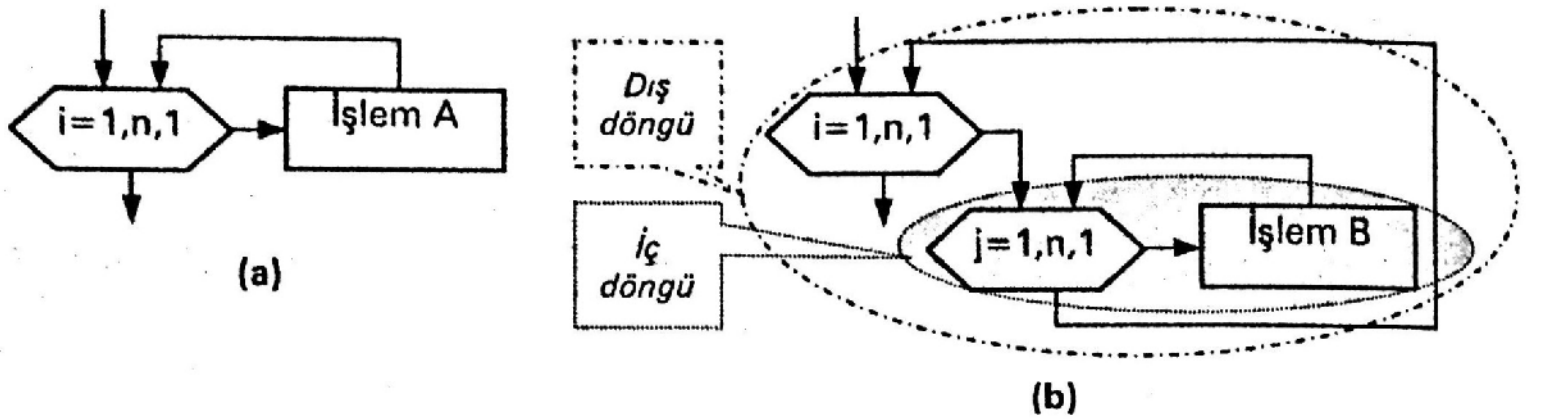
1'den 99'a kadar 1'er artan 'K' döngüsü

Şekil 3.7: Döngü örnekleri



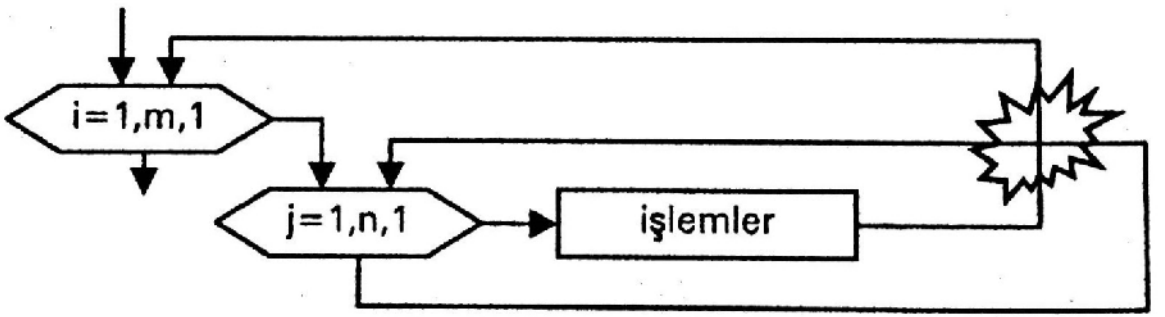
Uygun değerlerle(a<b ve c>0) örnek artan bir döngünün çalışma mantığı

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.



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

Döngü Yapısı

Bu yapı kullanılırken, döngü sayacı, koşul bilgisi ve sayacın artım bilgisi verilmelidir. Döngü sayacı kullanılmıyorsa sadece döngüye devam edebilmek için gerekli olan koşul bilgisi verilmelidir.

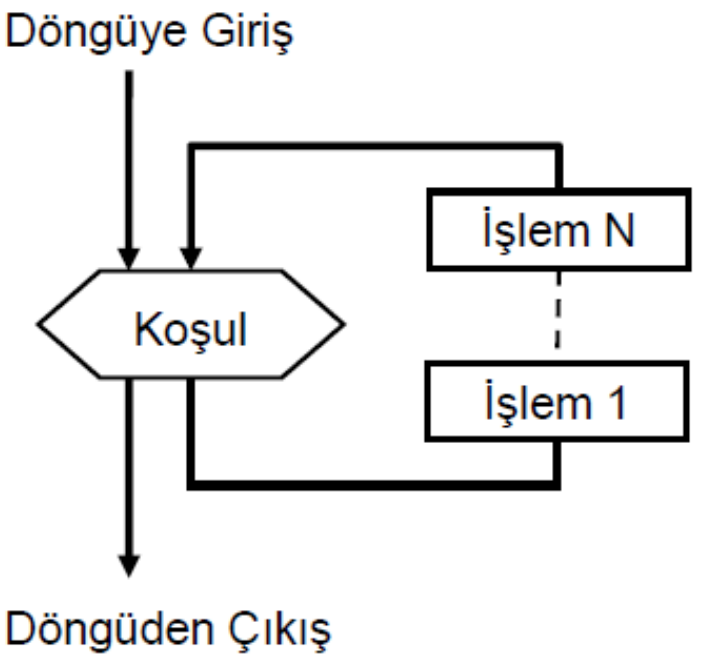
Genel olarak çoğu programlama dilinin döngü deyimleri ;

- While
- Do- while
- For

gibi yapılar üzerine kurulmuştur. Farklı dillerde bu yapılara farklı alternatifler olsa da döngülerin çalışma mantığı genel olarak benzerdir.

While Deyimi

Koşul daha çevrim içerisine girmeden sınanır. Koşul olumsuz olduğunda çerime hiç girilmez ve döngü içerisinde yapılması gerekenler atlanır.

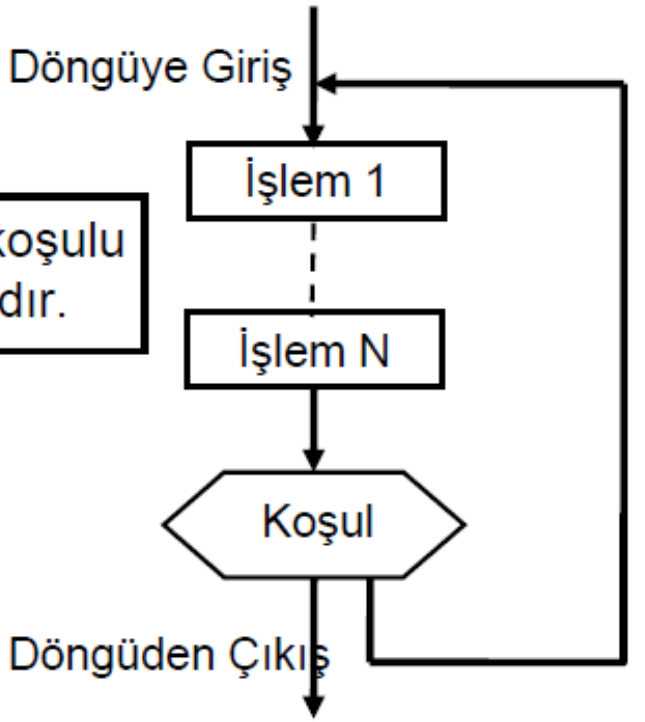


Çevrim içerisinde koşulu etkileyen kod olmalıdır.

Do-While Döngüsü

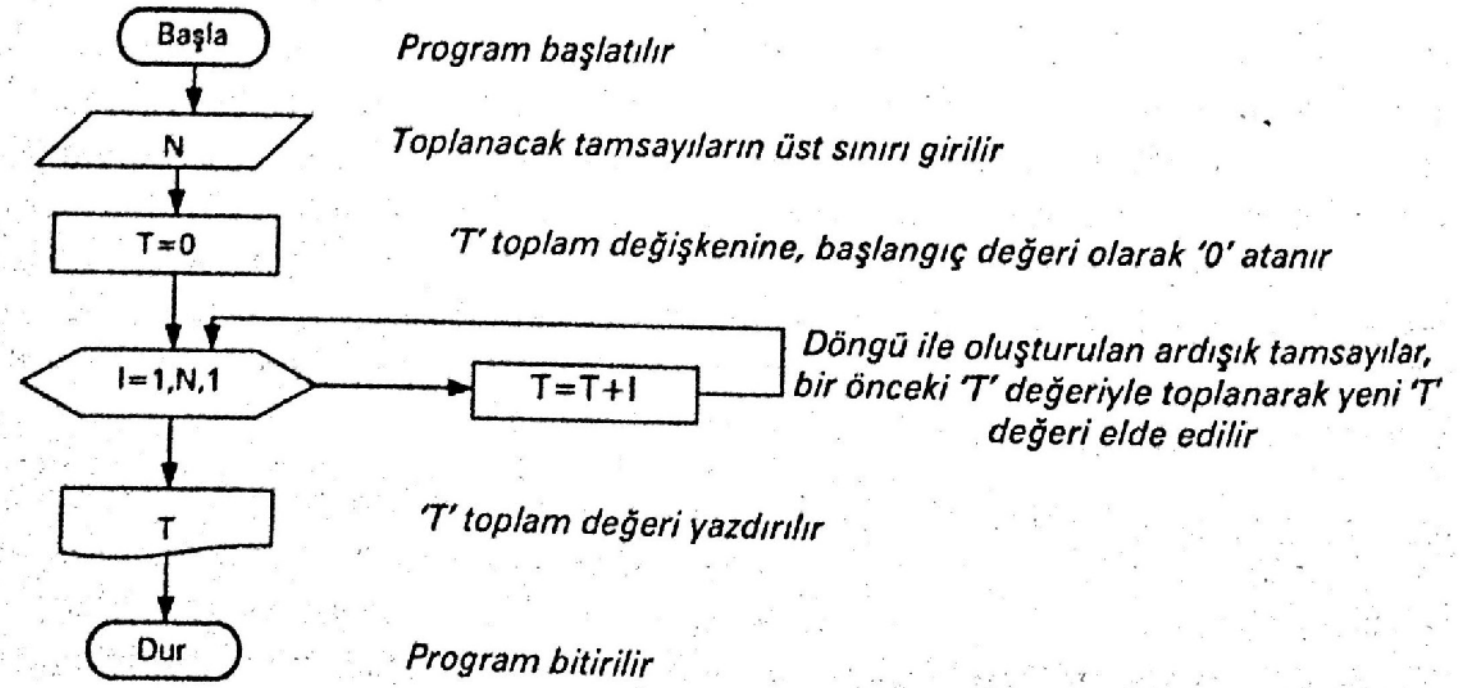
- Bu döngü deyiminde, çevrim en az bir defa olmak üzere gerçekleşir. Çünkü koşul sınaması döngü sonunda yapılmaktadır. Eğer koşul sonucu olumsuz ise bir sonraki çevrime geçilmeden döngüden çıkılır. Çevrimin devam edebilmesi için her döngü sonunda yapılan koşul testinin olumlu sonuçlanması gerekir.

Çevrim içerisinde koşulu etkileyen kod olmalıdır.



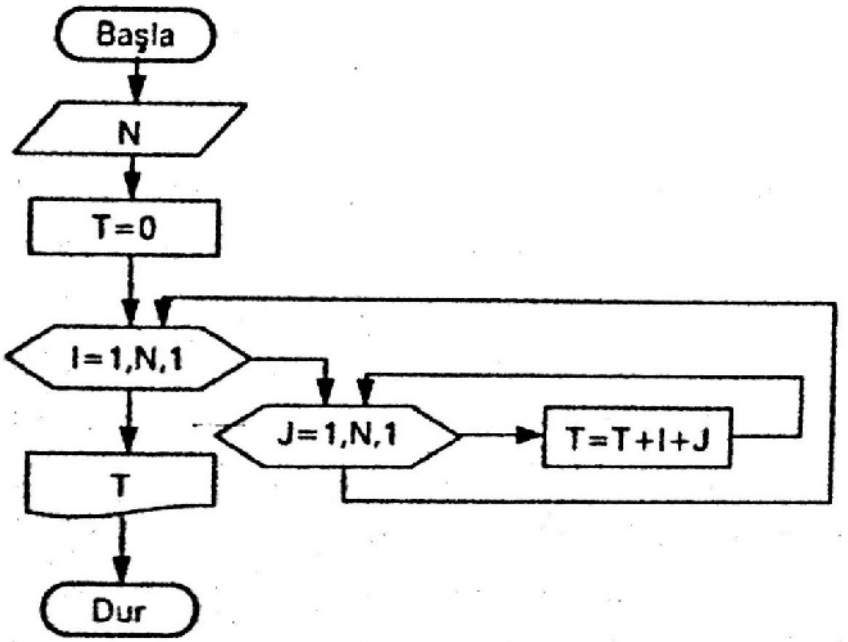
Örnek-3.1: 1'den N'e kadar tamsayıların toplamını hesaplayan programın akış diyagramı.

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.



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

İç içe döngülerde, dıştaki döngünün her çalıştırılma adımında içteki döngü baştan başlayarak işlenir. 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.



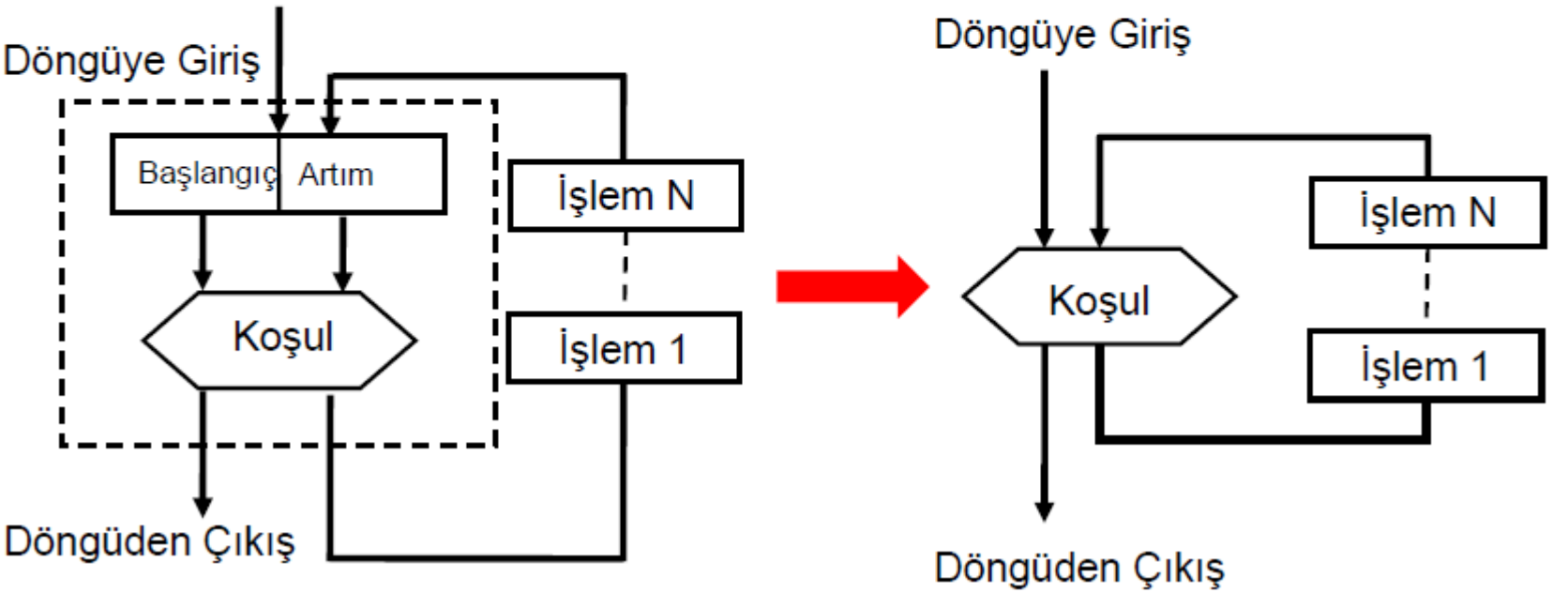
Tablo 3.4: Örnek-3.2'nin çalışma prensibi

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

For Döngüsü

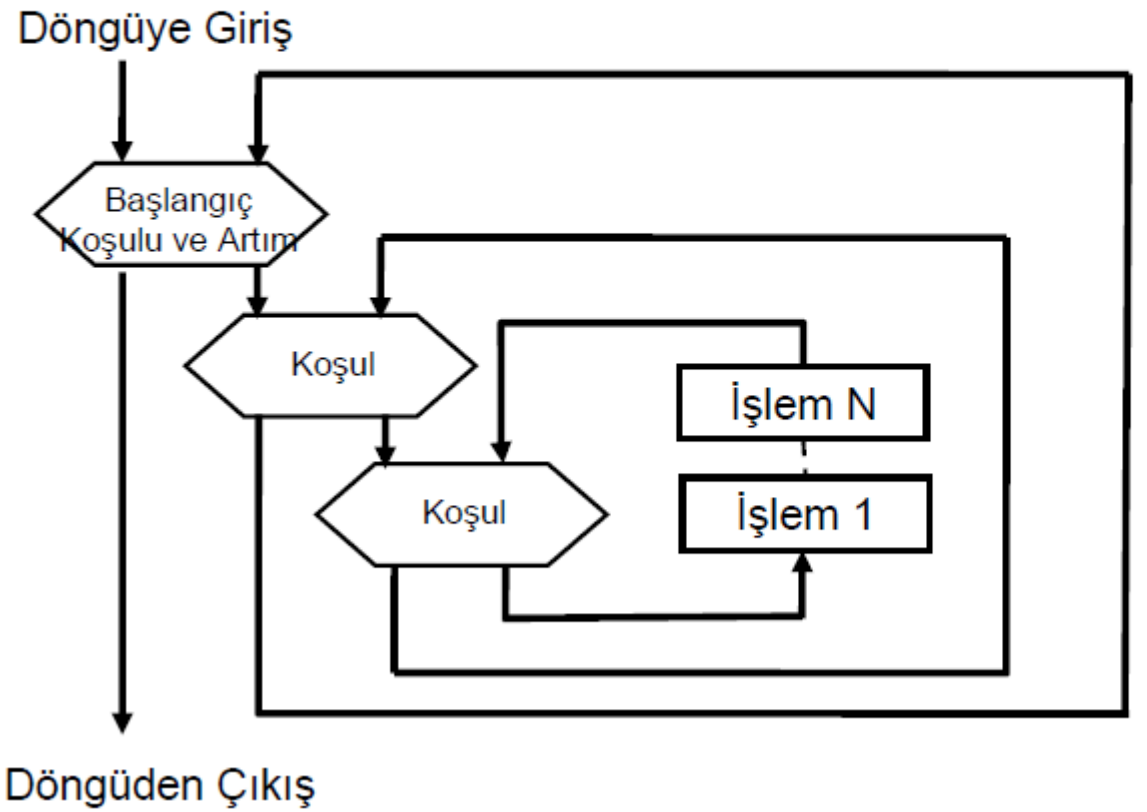
Diğer deyimlerden farklı olarak, döngü sayacı doğrudan koşul parametreleri düzeyinde verilir.

Döngü girmeden önce sayaç değişkenine başlangıç değeri atanmakta ve daha sonra koşula bakılmaktadır. Döngü içerisinde belirtilen işlemler yapıldıktan sonra sayaç değişkeni arttırılmaktadır.



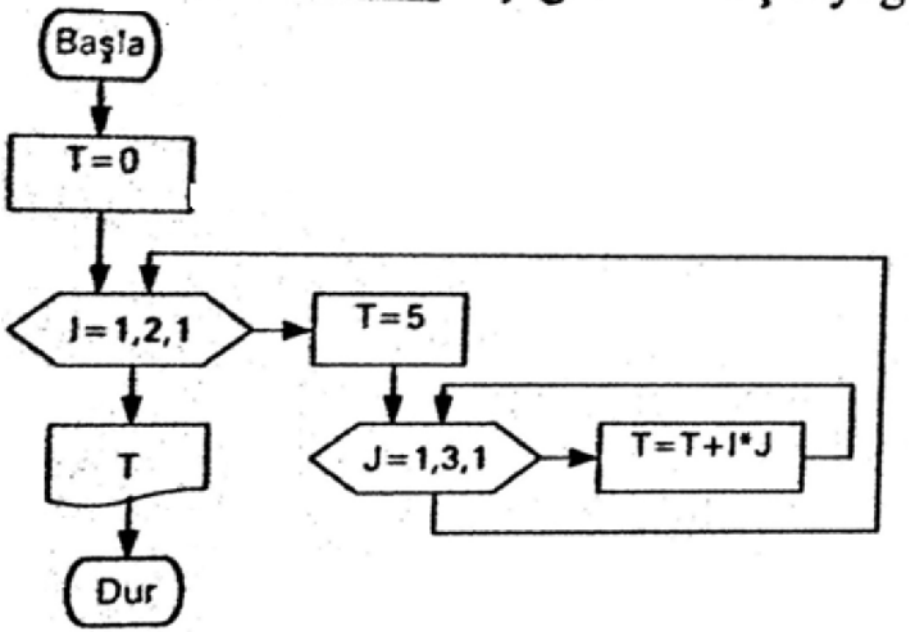
İç içe döngülerin kullanılması

İçiçe döngü kurulurken en önemli unsur, içteki döngü sonlanmadan bir dıştaki döngüye geçilmemesidir. Diğer bir deyişle döngüler birbirlerini kesmemelidir.



En içteki döngü bir dıştaki döngünün her adımında N kez tekrarlanır.

Örnek-3.4: Aşağıdaki akış diyagramının sonucunu bulunuz.



Örnek-3.4'teki akış diyagramının çalışma prensibi

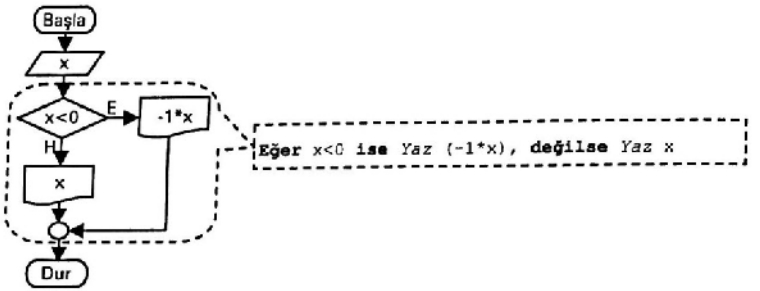
I	J	Eski T	Yeni T
1	1	5	5+1*1=6
1	2	6	6+1*2=8
1	3	8	8+1*3=11
2	1	5	5+2*1=7
2	2	7	7+2*2=11
2	3	11	11+2*3=17

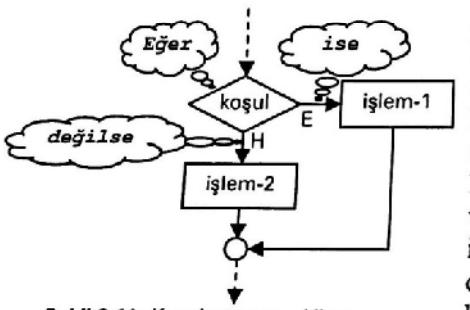
Örnek-3.6: Klavyeden girilen bir sayının mutlak değerini ekrana yazdıran programın akış diyagramının incelenmesi.

Herhangi bir x sayısının $|x|$ mutlak değeri

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

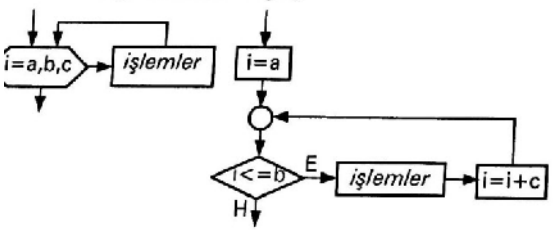
şeklinde matematiksel olarak ifade edilebilir.





Şekil 3.11: Karşılaştırma şekli ve algoritmadaki karşılığı

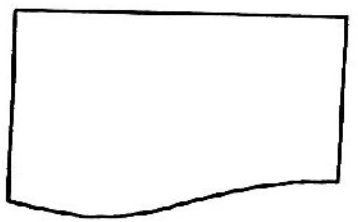
Akış diyagramından da görüldüğü gibi karşılaştırma şeklinin algoritmadaki ifade veya programlama dillerindeki komut olarak karşılığı "*Eğer*"; 'E' (evet) (doğru ise, koşul sağlanıyorsa) kolu üzerindeki akışın karşılığı "*ise*" ve 'H' (hayır) (doğru değilse, yanlış ise, koşul sağlanmıyorsa) kolu üzerindeki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

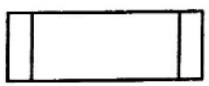
Örnek-3.7: Tablo 3.8'de yazdırma/çıktı örnekleri yer almaktadır.

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

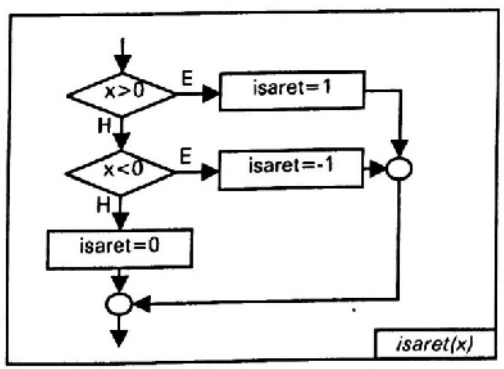
Büyük programlar, birçok küçük programın (modülün) birleşiminden meydana gelmektedir. Benzer şekilde küçük programlarda da bazı işlemler ayrı olarak tanımlanıp (alt program, fonksiyon vb.) kullanılmaktadırlar. Böylece belirli işlem veya işlemlerin yapılması bir defa kodlanarak parametresiz veya parametreli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

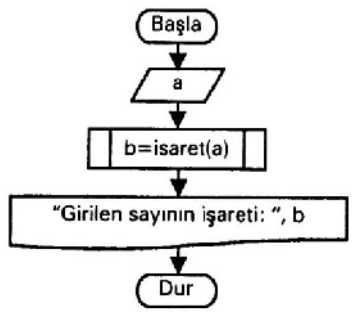


Şekil 3.14 Önceden tanımlı işlem

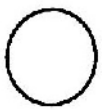
Önceden tanımlı işlemler, isimleri ve varsa parametreleriyle birlikte Şekil 3.14'tekinin içine yazılır.

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





H. Bağlantı



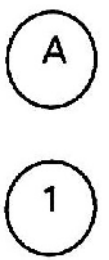
Şekil 3.15 Bağlantı şekli

Bağlantı şekli (daire) (Şekil 3.15), genel anlamda işlem akışlarını birleştiren bir yer olup aşağıdaki amaçlar için kullanılır:

i. Farklı yerlere dallanan işlem akışlarını toplamak.

ii. Akış diyagramı bir sayfaya sığmadığı zaman diğer sayfadaki akış diyagramı ile bağlantı kurmak.

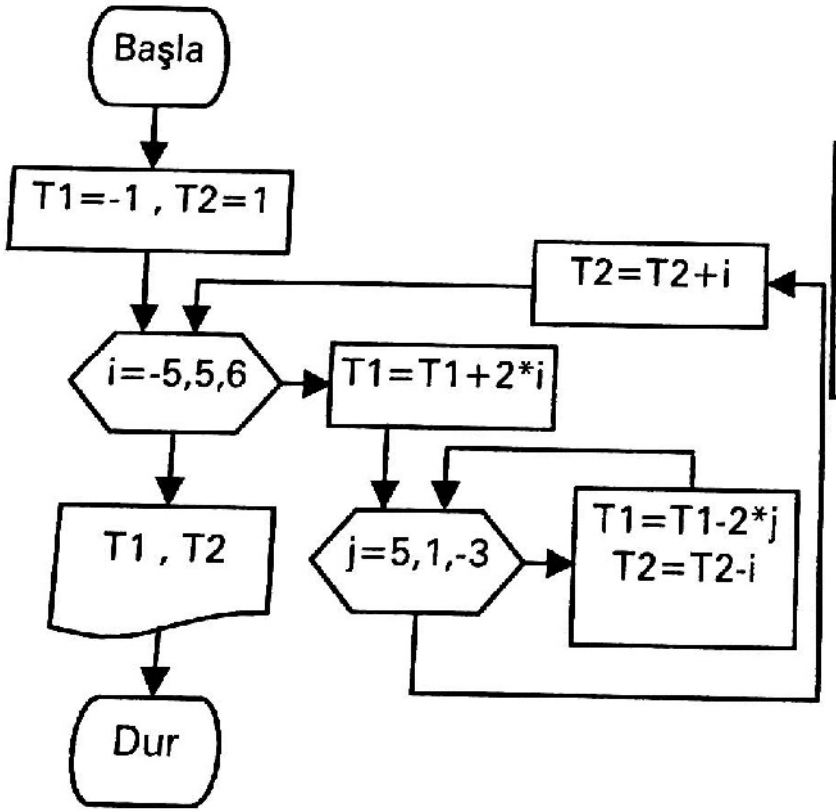
iii. Parça parça çizilen akış diyagramları arasında bağlantı kurmak.



Şekil 3.16 Bağlantı yapma

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

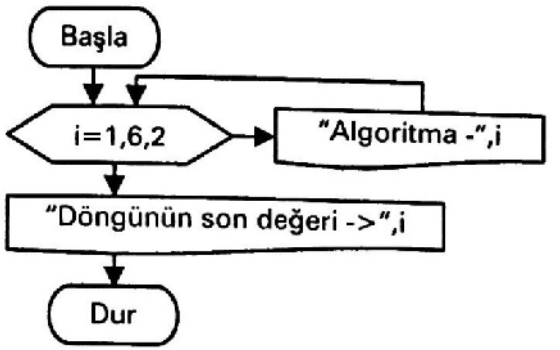
Örnek-3.9: Aşağıdaki akış diyagramının sonuçlarını elde ediniz.



Tablo 3.9: Örnek-3.9'un çalıştırma adımlarının sonuçları

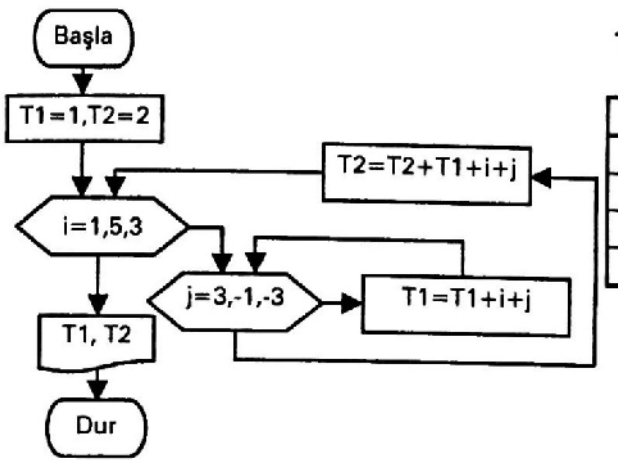
i	j	Eski T1	Yeni T1	Eski T2	Yeni T2
-5	5	~~-1~~ -11	-21	1	6
	2	-21	-25	6	~~11~~ 6
1	5	~~-25~~ -23	-33	6	5
	2	-33	**-37**	5	**4 5**

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




```
Algoritma - 1
Algoritma - 3
Algoritma - 5
Döngünün son değeri -> 7
```

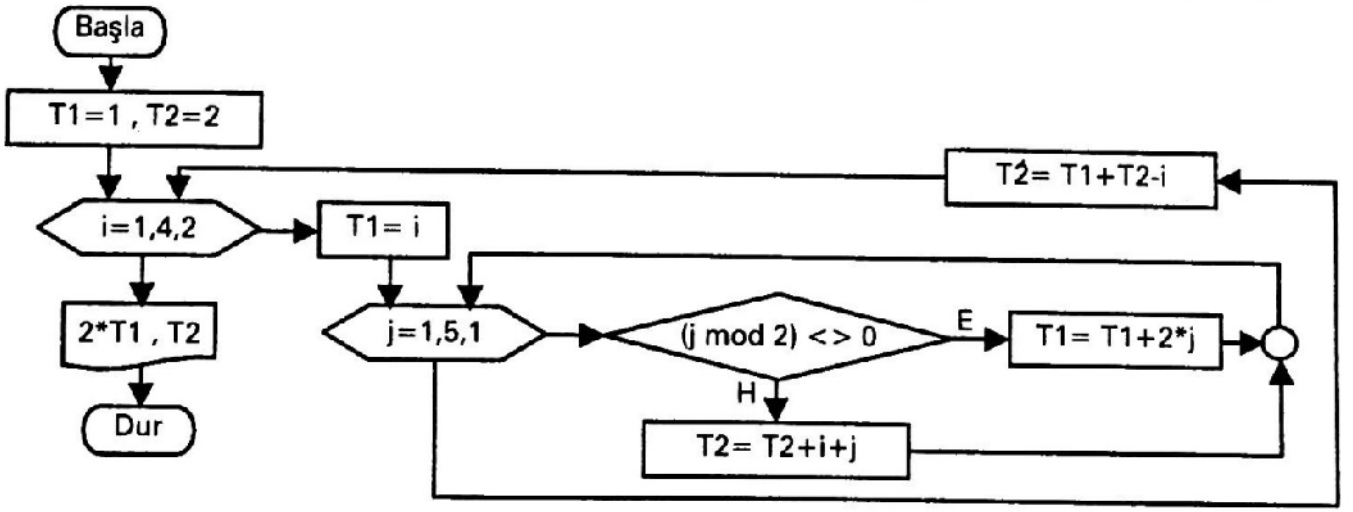
Örnek-3.11: Aşağıdaki akış diyagramının sonuçlarını elde ediniz.



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+1-3=6
4	3	6	6+4+3=13	6	-
	0	13	13+4+0=17	6	6+17+4-3=24

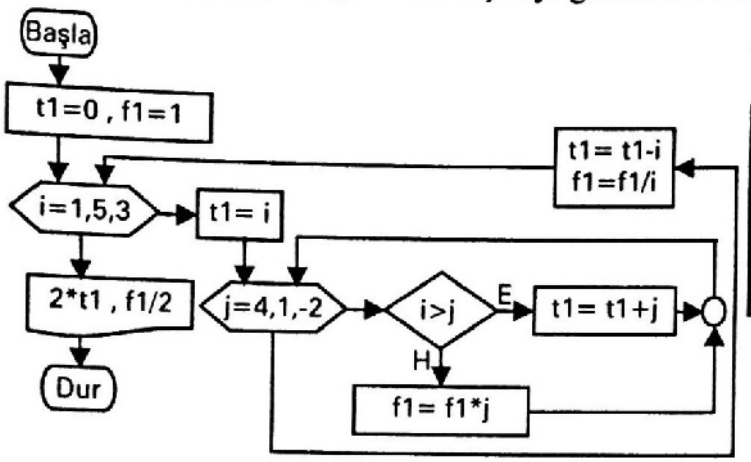
Örnek-3.12: Aşağıdaki akış diyagramının ekran çıktısını elde ediniz.



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.

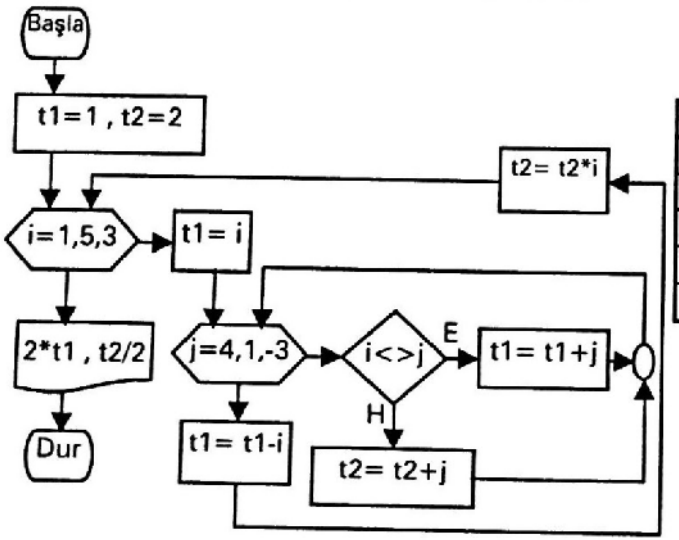


Tablo 3.12: Örnek-3.13'ün çalıştırma adımlarının sonuçları

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.

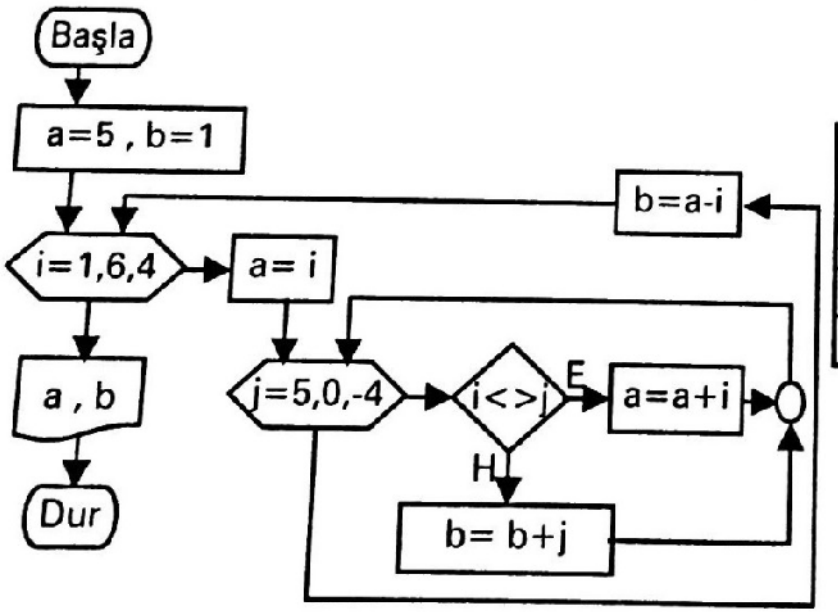


Tablo 3.13: Örnek-3.14'ün çalıştırma adımlarının sonuçları

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**

PROGRAMLAMA DİLLERİ

PROGRAMLAMA DİLLERİNDE KULLANILAN VERİ TİPLERİ

Bilgisayarda işlenen veriler 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	Basic	Pascal	C / C++ / C# / Java
10 (decimal)	-	-	-
16 (hexadecimal)	&h*sayı* veya &H*sayı*	$*sayı*	0x*sayı* veya 0X*sayı*
Üstel	e*üs* veya E*üs*	e*üs* veya E*üs*	e*üs* veya E*üs*

Tablo 4.2: Programlama dillerinde alfasayısal verilerin gösterilmesi/aktarılması

Alfasayısal veri	Basic	Pascal	C / C++ / C# / Java
Karakter	Çift tırnak içinde	Tek tırnak içinde	Tek tırnak içinde
Karakter dizisi	Çift tırnak içinde	Tek tırnak içinde	Çift tırnak içinde

A. Basic'te Kullanılan Veri Tipleri

Basic (GWBasic, QBasic) programlama dillerinde işlenen verileri temsil eden değişkenleri ve tiplerini önceden bildirmeye gerek yoktur. Aktarılacak/tutulacak olan verinin tipi, değişkenin hemen yanına yazılan özel simgelerle belirtilir.

1. Sayısal Veri Tipleri

Sayısal veriler iki gruba ayrılır: tamsayılar ve ondalıklı sayılar.

a. Tamsayı Veri Tipleri

Basic'te tamsayı veri tipleri, aktarılabilecek tamsayı sınırları farklı olmak üzere tamsayı ve uzun tamsayı olarak ikiye ayrılır (Tablo 4.3).

Tablo 4.3: Basic'teki tamsayı veri tipleri

Veri tipi	Aktarılabilecek en küçük değer	Aktarılabilecek en büyük değer	Belirtme simgesi
Tamsayı	-32 768	32 767	%
Uzun tamsayı	-2 147 483 648	2 147 483 647	&

b. Ondalıklı Sayı Veri Tipleri

Ondalıklı sayılar da tek duyarlıklı ondalıklı sayı ve çift duyarlıklı ondalıklı sayı olmak üzere ikiye ayrılırlar (Tablo 4.4).

Tablo 4.4: Basic'teki ondalıklı sayı veri tipleri

Veri tipi	İşaret	Aktarılabilecek en küçük değer	Aktarılabilecek en büyük değer	Üstel gösterim	Belirtme simgesi	Basamak duyarlılığı
Tek duyarlıklı	+	2.802597E-45	3.402823E+38	E	!	6-7
	-	-3.402823E+38	-2.802597E-45			
Çift duyarlıklı	+	4.940656458412465D-324	1.79769313486231D+308	D	#	14-15
	-	-1.79769313486231D+308	-4.940656458412465D-324			

Not

Sayısal verinin aktarılacağı değişkenin yanında herhangi belirtme simgesi yoksa tek duyarlıklı ondalıklı sayı kabul edilir.

Aşağıdaki programda "π" sayısı, 22/7 ile değişik sayısal veri tiplerinde he- saplattırılmaktadır. Özellikle ondalıklı sayı veri tiplerinde virgülden sonraki basamak sayısına dikkat ediniz.

BASIC

```
10 REM *** Sayısal veri tipleri ***
20 PRINT "22/7 işleminin sonucu:"
30 A%=22/7:PRINT "Tamsayı veri tipinde =>                    ",A%
40 A=22/7:PRINT "Tek duyarlıklı veri tipinde =>              ",A
50 A#=22/7:PRINT "Çift duyarlıklı veri tipinde =>            ",A#
60 END
```

```
22/7 işleminin sonucu:
Tamsayı veri tipinde =>          3
Tek duyarlıklı veri tipinde =>   3.142857
Çift duyarlıklı veri tipinde =>  3.142857074737549
```

2. Alfasayısal Veri Tipleri

Alfasayısal veri aktarımı için, değişken isminden sonra '$' simgesi kullanılır ve aktarılan veri çift tırnak (" ") içinde verilir (Tablo 4.5).

Tablo 4.5: Basic'teki alfasayısal veri tipleri

Veri tipi	Aktarılabilecek en küçük değer	Aktarılabilecek en büyük değer	Belirtme simgesi
Alfasayısal	0 karakter	32 767 karakter	$

BASIC

```
10 REM *** Alfasayısal veri kullanımı ***
20 A$="Seçkin Yayınevi"
30 PRINT A$
40 END
```

B. Pascal'da Kullanılan Veri Tipleri

Pascal programlama dilindeki (Turbo Pascal, Borland Pascal) değişkenleri ve veri tiplerini, kullanmadan önce bildirmek (tanımlamak) gerekmektedir.

1. Sayısal Veri Tipleri

a. Tamsayı Veri Tipleri

Sadece tam kısmı bulunan sayısal değerler için kullanılan bu veri tipleri, verinin büyüklüğüne göre farklı şekillerde isimlendirilirler (Tablo 4.6).

Tablo 4.6: Pascal'daki tamsayı veri tipleri

Veri tipi	Aktarılabilecek en küçük değer	Aktarılabilecek en büyük değer	İşaret	Bellekte kapladığı alan (byte)
Byte	0	255	Yok	1
Shortint	-128	127	Var	1
Integer	-32 768	32 767	Var	2
Word	0	65 535	Yok	2
Longint	-2 147 483 648	2 147 483 647	Var	4

b. Ondalıklı Sayı Veri Tipleri

Hem tam hem de ondalıklı kısmı bulunan sayısal değerler için kullanılan bu veri tipleri, yine sayısal verinin büyüklüğüne/duyarlılığına göre farklı şekillerde tanımlanırlar (Tablo 4.7).

Tablo 4.7: Pascal'daki ondalıklı sayı veri tipleri

Veri tipi	Alt sınır	Üst sınır	Bellekte kapladığı alan (Byte)	Duyarlılık (hane)
Real	$2,9.10^{-39}$	$1,7.10^{38}$	6	11-12
Single	$1,5.10^{-45}$	$3,4.10^{38}$	4	7-8
Double	5.10^{-324}	$1,7.10^{308}$	8	15-16
Extended	$3,4.10^{-4932}$	$1,1.10^{4932}$	10	19-20
Comp	$-9,2.10^{63}$	$9,2.10^{63}$	8	19-20

Basic'teki duyarlılıkları göstermek amacıyla "π" sayısı için yazılan programın benzeri Pascal dilinde de kodlanarak aşağıda verilmektedir.

PASCAL

```
{$E+,N+}
Program Duyarlilik;
Uses Crt;
Var
    a:Real; b:Single; c:Double; d:Extended; e:Comp;
Begin
    Clrscr;Writeln('Ondalıklı Sayılarda Duyarlılık:');
    Writeln('==============================');
    Writeln;Writeln('Pi sayısının değeri:');Writeln;
    a:=Pi;b:=Pi;c:=Pi;d:=Pi;e:=Pi;
    Writeln('Reel tipinde: ',a:1:20);
    Writeln('Single tipinde: ',b:1:20);
    Writeln('Double tipinde: ',c:1:20);
    Writeln('Extended tipinde: ',d:1:20);
    Writeln('Comp tipinde: ',e:1:20);
    Readln;
End.
```

Tablo 4.4: Basic'teki ondalıklı sayı veri tipleri

Veri tipi	İşaret	Aktarılabilecek en küçük değer	Aktarılabilecek en büyük değer	Üstel gösterim	Belirtme simgesi	Basamak duyarlılığı
Tek duyarlıklı	+	2.802597E-45	3.402823E+38	E	!	6-7
	-	-3.402823E+38	-2.802597E-45			
Çift duyarlıklı	+	4.940656458412465D-324	1.79769313486231D+308	D	#	14-15
	-	-1.79769313486231D+308	-4.940656458412465D-324			

Not

Sayısal verinin aktarılacağı değişkenin yanında herhangi belirtme simgesi yoksa tek duyarlıklı ondalıklı sayı kabul edilir.

Aşağıdaki programda "π" sayısı, 22/7 ile değişik sayısal veri tiplerinde he- saplattırılmaktadır. Özellikle ondalıklı sayı veri tiplerinde virgülden sonraki basamak sayısına dikkat ediniz.

BASIC

```
10 REM *** Sayısal veri tipleri ***
20 PRINT "22/7 işleminin sonucu:"
30 A%=22/7:PRINT "Tamsayı veri tipinde =>                    ",A%
40 A=22/7:PRINT "Tek duyarlıklı veri tipinde =>              ",A
50 A#=22/7:PRINT "Çift duyarlıklı veri tipinde =>            ",A#
60 END
```

```
22/7 işleminin sonucu:
Tamsayı veri tipinde =>          3
Tek duyarlıklı veri tipinde =>   3.142857
Çift duyarlıklı veri tipinde =>  3.142857074737549
```

BASIC

```
10 REM *** Sayısal veri tipleri ***
20 PRINT "22/7 işleminin sonucu:"
30 A%=22/7:PRINT "Tamsayı veri tipinde =>                    ",A%
40 A=22/7:PRINT "Tek duyarlıklı veri tipinde =>              ",A
50 A#=22/7:PRINT "Çift duyarlıklı veri tipinde =>            ",A#
60 END
```

```
22/7 işleminin sonucu:
Tamsayı veri tipinde =>          3
Tek duyarlıklı veri tipinde =>   3.142857
Çift duyarlıklı veri tipinde =>  3.142857074737549
```

2. Alfasayısal Veri Tipleri

Alfasayısal veri aktarımı için, değişken isminden sonra '$' simgesi kullanılır ve aktarılan veri çift tırnak (" ") içinde verilir (Tablo 4.5).

Tablo 4.5: Basic'teki alfasayısal veri tipleri

Veri tipi	Aktarılabilecek en küçük değer	Aktarılabilecek en büyük değer	Belirtme simgesi
Alfasayısal	0 karakter	32 767 karakter	$

BASIC

```
10 REM *** Alfasayısal veri kullanımı ***
20 A$="Seçkin Yayınevi"
30 PRINT A$
40 END
```

B. Pascal'da Kullanılan Veri Tipleri

Pascal programlama dilindeki (Turbo Pascal, Borland Pascal) değişkenleri ve veri tiplerini, kullanmadan önce bildirmek (tanımlamak) gerekmektedir.

1. Sayısal Veri Tipleri

a. Tamsayı Veri Tipleri

Sadece tam kısmı bulunan sayısal değerler için kullanılan bu veri tipleri, verinin büyüklüğüne göre farklı şekillerde isimlendirilirler (Tablo 4.6).

Tablo 4.6: Pascal'daki tamsayı veri tipleri

Veri tipi	Aktarılabilecek en küçük değer	Aktarılabilecek en büyük değer	İşaret	Bellekte kapladığı alan (byte)
Byte	0	255	Yok	1
Shortint	-128	127	Var	1
Integer	-32 768	32 767	Var	2
Word	0	65 535	Yok	2
Longint	-2 147 483 648	2 147 483 647	Var	4

A. Basic'te Kullanılan Veri Tipleri

Basic (GWBasic, QBasic) programlama dillerinde işlenen verileri temsil eden değişkenleri ve tiplerini önceden bildirmeye gerek yoktur. Aktarılacak/tutulacak olan verinin tipi, değişkenin hemen yanına yazılan özel simgelerle belirtilir.

1. Sayısal Veri Tipleri

Sayısal veriler iki gruba ayrılır: tamsayılar ve ondalıklı sayılar.

a. Tamsayı Veri Tipleri

Basic'te tamsayı veri tipleri, aktarılabilecek tamsayı sınırları farklı olmak üzere tamsayı ve uzun tamsayı olarak ikiye ayrılır (Tablo 4.3).

Tablo 4.3: Basic'teki tamsayı veri tipleri

Veri tipi	Aktarılabilecek en küçük değer	Aktarılabilecek en büyük değer	Belirtme simgesi
Tamsayı	-32 768	32 767	%
Uzun tamsayı	-2 147 483 648	2 147 483 647	&

b. Ondalıklı Sayı Veri Tipleri

Hem tam hem de ondalıklı kısmı bulunan sayısal değerler için kullanılan bu veri tipleri, yine sayısal verinin büyüklüğüne/duyarlılığına göre farklı şekillerde tanımlanırlar (Tablo 4.7).

Tablo 4.7: Pascal'daki ondalıklı sayı veri tipleri

Veri tipi	Alt sınır	Üst sınır	Bellekte kapladığı alan (Byte)	Duyarlılık (hane)
Real	$2,9.10^{-39}$	$1,7.10^{38}$	6	11-12
Single	$1,5.10^{-45}$	$3,4.10^{38}$	4	7-8
Double	5.10^{-324}	$1,7.10^{308}$	8	15-16
Extended	$3,4.10^{-4932}$	$1,1.10^{4932}$	10	19-20
Comp	$-9,2.10^{63}$	$9,2.10^{63}$	8	19-20

Basic'teki duyarlılıkları göstermek amacıyla "π" sayısı için yazılan programın benzeri Pascal dilinde de kodlanarak aşağıda verilmektedir.

PASCAL

```
{$E+,N+}
Program Duyarlilik;
Uses Crt;
Var
    a:Real; b:Single; c:Double; d:Extended; e:Comp;
Begin
    Clrscr;Writeln('Ondalıklı Sayılarda Duyarlılık:');
    Writeln('==============================');
    Writeln;Writeln('Pi sayısının değeri:');Writeln;
    a:=Pi;b:=Pi;c:=Pi;d:=Pi;e:=Pi;
    Writeln('Reel tipinde: ',a:1:20);
    Writeln('Single tipinde: ',b:1:20);
    Writeln('Double tipinde: ',c:1:20);
    Writeln('Extended tipinde: ',d:1:20);
    Writeln('Comp tipinde: ',e:1:20);
    Readln;
End.
```

1. Sayısal Veri Tipleri

a. Tamsayı Veri Tipleri

Tamsayılar için mevcut veri tipleri ve sınırları Tablo 4.22'de verilmektedir.

Tablo 4.22: Java'daki tamsayı veri tipleri

Veri tipi	Aktarılabilecek en küçük değer	Aktarılabilecek en büyük değer	Bellekte kapladığı alan (byte)
byte	-128	127	1
short	-32768	32767	2
int	-2147483648	2147483647	4
long	-9223372036854775808	9223372036854775807	8

b. Ondalıklı Sayı Veri Tipleri

Ondalıklı sayılar için Tablo 4.23'teki veri tipleri kullanılabilir.

Tablo 4.23: 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 4.24) aynen C/C++/C# programlarında olduğu gibi bir boyutlu dizi olarak işlenebilmektedirler.

Tablo 4.24: Java'daki alfasayısal veri tipleri

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

Tablo 4.25'te Basic, Pascal, C, C++, C# ve Java'da kullanılan temel veri tipleri karşılaştırmalı olarak verilmektedir.

Tablo 4.25: Basic, Pascal, C, C++, C# ve Java'daki temel veri tipleri

Veri tipi	Basic	Pascal	C	C++	C#	Java
Mantıksal	-	boolean	-	bool	bool	boolean
Tamsayı	-	integer	int	int	int	int
Ondalıklı sayı	-	real	float	float	float	float
Karakter	-	char	char	char	char	char
Karakter dizisi	-	string	char	char	string	string

III. PROGRAMLAMA DİLLERİNİN GENEL YAPILARI

Bilgisayarda program yazmak için geliştirilen dillerden herbiri farklı komut ve yapıya sahip olmasına rağmen, diller arasında büyük benzerlikler mevcuttur. Bu kısımda, sonraki bölümlerde yazılacak olan programlara temel oluşturması amacıyla Basic, Pascal, C, C++, C# ve Java dillerinin genel yapıları incelenecektir.

A. BASIC Dilinin Yapısı

'BASIC' ismi, "**B**eginner's **A**ll-Purpose **S**ymbolic **I**nstruction **C**ode" (Yeni başlayanların her/tüm-amaçlı sembolik öğretim kodu) kelimelerinin baş harflerinden türetilmiş olup, 1964 yılında New Hampshire (ABD)'deki Dartmounth College'de çalışmalar yapan *John George KEMENY* ve *Thomas Eugene KURTZ* tarafından geliştirilmiş olan en basit genel amaçlı yüksek seviyeli programlama dillerindendir. BASIC programlama dili daha sonraları MS-BASIC, GWBASIC ve QBASIC isimleri altında geliştirilerek kullanılmıştır. QBASIC veya QB, Microsft'un geliştirmiş olduğu Microsoft QuickBASIC'in kısaltılmış ifadesidir. Öğrenilmesi, uygulanması ve program kodlanması son derece kolay olan BASIC'in görsel programlama dili Visual Basic, geniş bir kullanım alanına sahiptir. Hatta Windows işletim sistemlerinin değişik sürümlerindeki bazı bölümleri Visual Basic ile kodlanmıştır.

BASIC dilinin temel yazım özellikleri

i. En temel, öğrenilmesi ve kod yazılması en kolay programlama dillerindendir.

ii. Karmaşık bir program yapısına sahip değildir.

iii. Program yazımı son derece kolay olup herhangi bir kalıpla sınırlı değildir.

iv. Komutlar, işlem çözümüne göre sırayla, alt alta satırlara yazılırlar. Komutlar, aralarına iki nokta üst üste konularak, aynı satıra da kodlanabilirler. Bir satıra, en fazla 255 karakter yazılabilir.

v. Komutlar, büyük harflerle yazılırlar. Kullanıcı küçük harflerle komut girse bile; BASIC editöründe, otomatik olarak büyütülmektedir.

vi. Yazılımın başında değişkenleri bildirme zorunluluğu yoktur. Değişken ismi en az 1, en fazla 40 karakter olabilmektedir.

B. PASCAL Dilinin Yapısı

Fransız matematikçi ve filozof Blaise Pascal'ın adı verilen ve ALGOL diline dayanan bu programlama dili, 1971 yılında İsviçre Federal Teknoloji Enstitüsü'nden (ETH-Zürih) *Dr. Niklaus WIRTH* tarafından geliştirilmiştir. Günümüzdeki en popüler görsel programlama dillerinden olan Delphi, Pascal tabanlı olup diğer adı "Visual Pascal"dır. Pascal programlama dili için Turbo Pascal, Borland Pascal, Free Pascal, Dev Pascal gibi birçok derleyiciler geliştirilmiştir. Komutları ve program yapısı konuşma diline çok yakın olan Pascal; tüm dünyada genel kabul görmüş ve programlama eğitiminde kullanılan yüksek seviyeli dillerden birisidir. Programlama mantığına en uygun dillerden birisi olan Pascal ile tasarlanan algoritmalar, kolaylıkla kodlanabilir. Esnek ve fonksiyonel bir yapısı olan, güçlü komut kümesine sahip bu dildeki programlar, birçok altprogramın birleşiminden oluşturulabilmektedir.

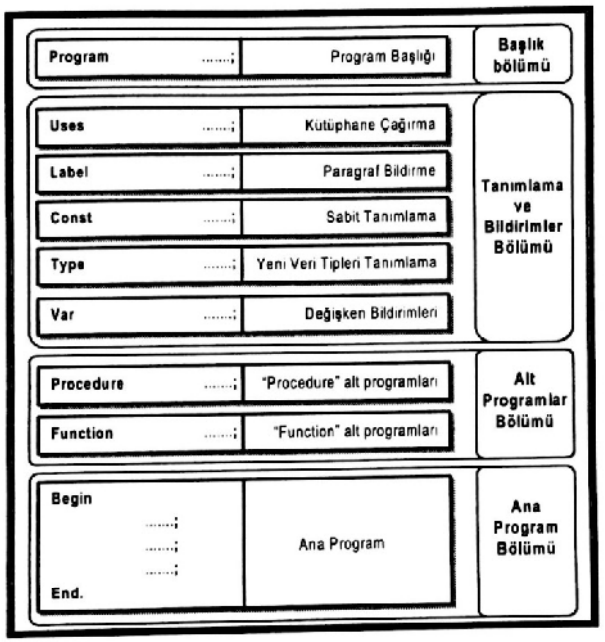
Pascal programları genel olarak dört ana bölümden oluşur:

i. *Program başlığı*

ii. *Tanımlama ve bildirimler bölümü*

iii. *Alt programlar bölümü*

iv. *Ana program bölümü*



1. Program Başlığı

İsimlendirme kurallarına uygun olarak yazılan ve kullanımı zorunlu olmayan bu ifade, genellikle program hakkında açıklayıcı bilgi içerir.

Kullanım şekli:

```
Program   program adı;
```

2. Tanımlama ve Bildirimler Bölümü

a. Uses

Yapısal dillerde (Pascal, C, C++ vb.); komutlar, belirli kütüphanelerde bulunmaktadır. Programlar derlenirken sadece, yazılımın başında çağırılan/belirtilen/aktifleştirilen kütüphanelerdeki komutlar yüklenerek çalıştırılırlar. Dolayısıyla yazılımda hangi komutlar kullanılacaksa, bunların bulunduğu kütüphaneler aktifleştirilmelidir. Aksi durumda; derleyici, programı çalıştırdığında

yazım hataları verecektir. Program başlığından sonraki satırda yer alan "Uses" ifadesi, kullanılacak kütüphaneleri (Tablo 4.26) bildirir.

Kullanım şekli:

```
Uses    kütüphane-1 , kütüphane-2 , kütüphane-3;
```

Tablo 4.26: Pascal'daki bazı kütüphaneler

Kütüphane	İçeriği
Crt	Temel işlemler ve ekran komutları
Graph	Grafik komutları
Dos	DOS işletim sistemiyle ilgili komutlar
Printer	Yazıcı ile ilgili komutlar

b. Label

Eğer programdaki işlem akışı normal seyrini izlemeyip belirli paragraflara (bloklara) dallanacaksa/yönlenecekse, gidilecek yerleri tanıtıcı isimler gerekmektedir. Pascal programlarında; işlem akışının geçeceği bu paragraflar (bloklar) "Label" ile bildirilir ve dallanma 'Goto' komutuyla yapılır. Basic'te satır numaraları kullanıldığından, paragraf tanımlamaya gerek kalmamaktadır.

Kullanım şekli:

```
Label    paragraf ismi 1 ,  paragraf ismi 2;
```

PASCAL

```
Program Paragraf_tanimlama;
Uses Crt;
Label geri;
Var   a:integer;
Begin
     ClrScr;
geri:Write('İki haneli bir tamsayı giriniz: ');
     Readln(a);a:=abs(a);
     if ((a<10) or (a>99)) then goto geri;
     Writeln; Write('Girilen sayının onlar basamağı: ',
                    a div 10,' ve birler basamağı: ',a mod 10);
     Readln;
End.
```

```
İki haneli bir tamsayı giriniz: 5
İki haneli bir tamsayı giriniz: 125
İki haneli bir tamsayı giriniz: 67

Girilen sayının onlar basamağı: 6 ve birler basamağı: 7
```

c. Const

Birçok programda, sabit değerler kullanılır. Pascal programları için kullanılacak olan sayısal veya alfasayısal sabitler, "`Const`" bloğunda tanımlanırlar. Sabit tanımlama iki şekilde yapılabilir:

- **Tip belirtmeden:** Veri tipi verilmeden yapılan bu sabit atamada; atanan sabit veri, program boyunca *değiştirilemez*.

b. Function

Ana programa yardımcı işlemlerin yapıldığı diğer bir alt program türü de "function"dır. Procedure'ler ana programa birden fazla sonuç gönderirken "function"lar tek sonuç gönderir. Tipi olan "function", tek sonuç üreten parametreli procedure'e eşdeğerdir.

Kullanım şekli:

```
Function   Fonksiyon adı (Parametreler:veri tipi):veri tipi;
Begin
    .....
    .....   } işlemler
    .....
End;
```

```
End.
```

```
Varsayılan alan: 12.560

Yeni yarıçap: 3
Hesaplanan alan: 28.260
```

d. Type

Programcıların özel veri tiplerini veya veri alanlarını tanımladıkları yerdir.

- **Özel veri tipi tanımlama:**

Kullanım şekli:

```
Type  Özel veri tipi adı = Standart tipler cinsinden tanımı;
```

Örneğin;

```
Type
     Kelime15=String[15];
     Dizi20=Array[1..20] of Integer;
```

- **Veri/kayıt alanı tanımlama:**

Kullanım şekli:

```
Type   Veri alanı adı = Record
          veri alt alanı 1 : veri tipi 1;
          ..............................
          veri alt alanı N : veri tipi N;
End;
```

Örneğin;

```
Type    Ogrenci=Record
        AdSoyad:String[25]; Numara:Longint;
        Vize,Final,Ort:Integer;
End;
```

Yukarıda 'Ogrenci' adlı kayıt/veri alanı tanımlanmıştır. Bu alanda "String" tipinde '*AdSoyad*', "Longint" tipinde '*Numara*' ve "Integer" tipinde '*Vize*', '*Final*' ve '*Ort*' değişkenleri vardır. İlgili alt alanlara değer aktarmak/erişmek için; bu veri tipinde tanımlanan değişken ile alt alan adı arasına nokta konulur.

e. Var

Pascal programlarında kullanılacak olan tüm değişkenler ve bunların veri tipleri, "Var" bloğunda bildirilir.

Kullanım şekli:

```
Var     değişken 1 : veri tipi;
        değişken 2 , değişken 3 : veri tipi;
```

3. Alt Programlar Bölümü

Karmaşık ve büyük programlar, birçok alt programın birleşiminden meydana gelir. Ana programa yardımcı işlemler veya ana programda aynı veya farklı değerlerle tekrar eden işlem blokları, alt program şeklinde oluştururlar. Alt programlar, ana programdan isim ve varsa parametreleri verilerek çağırılırlar. Alt programdaki işlemler bitince, işlem akışı; ana programdaki çağırıldığı satırdan devam eder.

a. Procedure

Pascal'daki "`procedure`" alt programları iki şekilde kullanılabilir:

- ***Parametresiz procedure:*** Kısa programlar için uygun olan bu tarzda, ana programdaki değişkenler ile çağırılan alt programdaki değişkenler aynıdır.

Kullanım şekli:

```
Procedure  Alt program ismi ;
Yerel bildirimler
Begin
    .....
    .....   } işlemler
    .....
End;
```

- ***Parametreli procedure:*** Uzun programlar için uygun olan bu tarzda, ana programdaki değişkenler ile çağırılan alt programdaki değişkenler farklıdır. Ana programdaki ile alt programdaki değişkenlerde veri aktarımı otomatik olarak gerçekleşir. Ana program ile alt program arasındaki parametre aktarımına göre de "parametreli procedure"ler ikiye ayrılırlar:

- *Yarım aktarımlı*: Sadece ana programdan alt programa parametre değeri aktarımı vardır. Alt program parametre bildiriminde "`Var`" deyimi yoktur.
- *Tam aktarımlı:* Hem ana programdan alt programa, daha sonra da alt programdan ana programa parametre değeri aktarımı vardır. Alt program parametreleri bildirilirken önlerinde "`Var`" deyimi kullanılır.

Kullanım şekli:

```
Procedure  Alt program ismi (Var Parametreler : veri tipi );
Yerel bildirimler
Begin
    .....
    .....   } işlemler
    .....
End;
```

Aşağıdaki programlarda iki sayının karesini alan alt program, birincisinde parametresiz ikincisinde de parametreli olarak tanımlanmıştır.

PASCAL

```
Program Parametresiz;
Uses Crt;
Var  x,y:Integer;
Procedure Kare;
Begin
    x:=x*x;y:=y*y;
End;
Begin
    ClrScr;Write('Karesi alınacak birinci sayı = ');Readln(x);
    Write('Karesi alınacak ikinci sayı = ');Readln(y);
    Kare; Writeln;Writeln('Birinci sayının karesi =',x);
    Writeln('İkinci sayının karesi =',y); Readln; End.
```

PASCAL

```
Program Parametreli;
Uses Crt;
Var x,y:Integer;
Procedure Kare(Var z,t:Integer);
Begin
    z:=z*z; t:=t*t;
End;
Begin
    ClrScr;Write('Karesi alınacak birinci sayı = ');Readln(x);
    Write('Karesi alınacak ikinci sayı = ');Readln(y);
    Kare(x,y); Writeln;Writeln('Birinci sayının karesi =',x);
    Writeln('İkinci sayının karesi =',y); Readln; End.
```

```
Karesi alınacak birinci sayı = 5
Karesi alınacak ikinci sayı = 6

Birinci sayının karesi =25
İkinci sayının karesi =36
```

b. Function

Ana programa yardımcı işlemlerin yapıldığı diğer bir alt program türü de "function"dır. Procedure'ler ana programa birden fazla sonuç gönderirken "function"lar tek sonuç gönderir. Tipi olan "function", tek sonuç üreten parametreli procedure'e eşdeğerdir.

Kullanım şekli:

```
Function   Fonksiyon adı (Parametreler:veri tipi):veri tipi;
Begin
    ..........................................
    ..........................................   } işlemler
    ..........................................
End;
```

PASCAL

```
Program Fonksiyon;
Uses Crt;
Var x,y:Integer;
Function Pisagor(a,b:Integer):Real;
Begin
    Pisagor:=sqrt(a*a+b*b);
End;
Begin
    ClrScr;Write('Birinci dik kenar = ');Readln(x);
    Write('İkinci dik kenar = ');Readln(y);
    Writeln('Hipotenüs =',Pisagor(x,y):0:2); Readln;
End.
```

```
Birinci dik kenar = 3
İkinci dik kenar = 4
Hipotenüs =5.00
```

4. Ana Program Bölümü

Tüm işlemlerin kontrol edildiği ve alt programların çağırıldığı programın temel bloğudur.

Not

Programlama dillerinde, ana programda yapılan bildirimler (*genel bildirimler*) tüm program boyunca geçerli iken (kullanılabilirken), alt programda yapılan bildirimler (*yerel bildirimler*) sadece o alt programda geçerlidir.

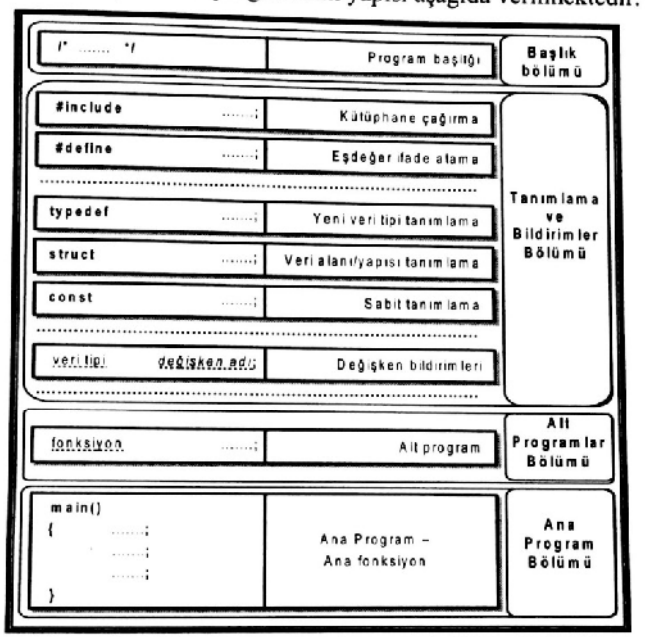
C. C Dilinin Yapısı

C programlama dili; 1972 yılında AT&T Bell Telefon Laboratuvarları'ndan *Dennis RITCHIE* tarafından UNIX işletim sistemiyle birlikte kullanılmak üzere oluşturulmuş genel amaçlı bilgisayar programlama dili olup 1970'te *Ken THOMPSON* tarafından geliştirilen "B" dili üzerine kurulmuştur. "B" dilinin tasarımında ise *Martin RICHARDS* tarafından 1967'de geliştirilen "BCPL" dilinden esinlenilmiştir. Günümüzde de çok popüler olan C ve C++ programla-

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. Bir satıra, en fazla 1023 karakter yazılabilir.
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.

En genel hatlarıyla bir C programının yapısı aşağıda verilmektedir.



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. Kitapta, önişlemci komutlarından sadece "`include`", "`define`" ve "`undef`"; yeni veri tipleri tanımlama ("`typedef`"), yapı/kayıt alanları oluşturma ("`struct`") ve değişken/sabit bildirimleri özetlenecektir.

a. include

Pascal programlarındaki gibi; C'de de komutlar/fonksiyonlar, belirli kütüphanelerde toplanmıştır. Kullanılacak komutlar için "*başlık dosyaları*" olarak adlandırılan ve bunları içeren kütüphaneler (Tablo 4.27) çağırılmalıdır.

Kullanım şekli:

```
#include < kütüphane adı >
```

Tablo 4.27: 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
```

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 birleşimindir. 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 değeri 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`" (yani hiçbir şey geri gönderme, sıfır geri gönder) kullanılmıştır. Programcı, geri dönen değer hakkında bilgilendirme/uyarı mesajı (`Function should return a value`) almamak için bu iki yoldan birini tercih edebilir.

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.

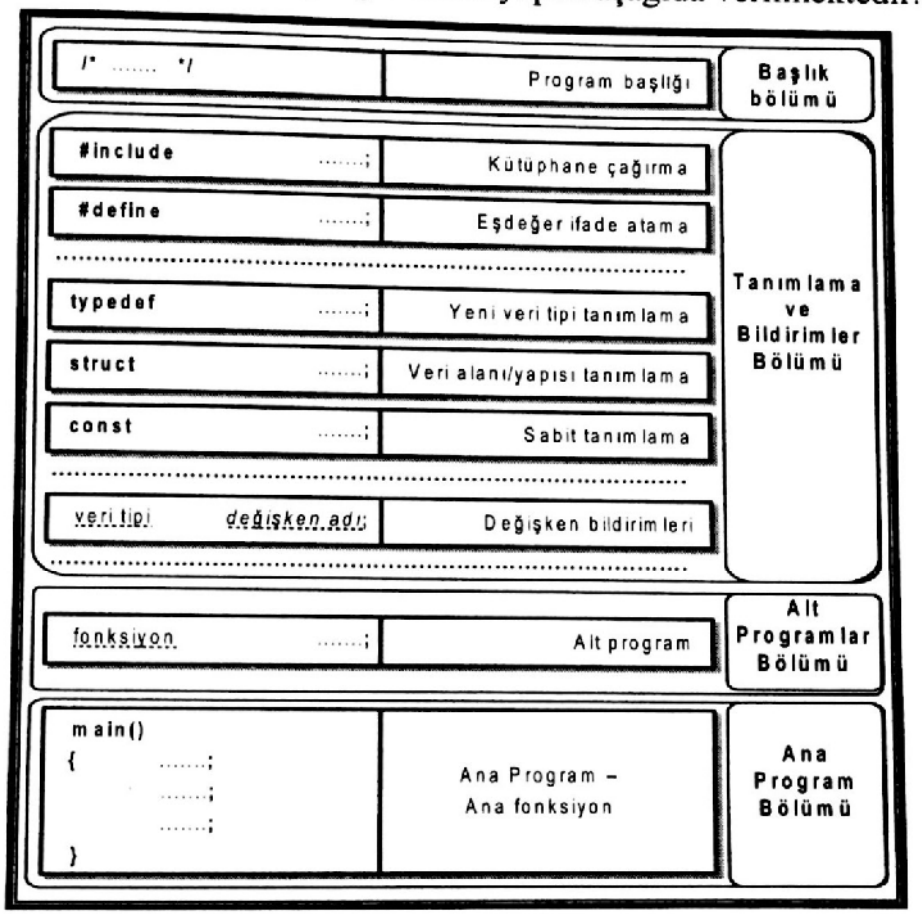
Kullanım şekli:

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

D. C++ Dilinin Yapısı

1979 yılında Bell Laboratuvarı araştırmacılarından Bjarne Stroustrup 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.

En genel hatlarıyla bir C programının yapısı aşağıda verilmektedir.



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. Kitapta, önişlemci komutlarından sadece "include", "define" ve "undef"; yeni veri tipleri tanımlama ("typedef"), yapı/kayıt alanları oluşturma ("struct") ve değişken/sabit bildirimleri özetlenecektir.

bildirimler yapılır. Burada yine C'de anlatılan ve daha sonraki bölümlerde yeterli olacak olan "include", "define" ve "undef" önişlemci komutları (Tablo 4.28); yeni veri tipleri tanımlama ("typedef"), yapı oluşturma ("struct") ve değişken/sabit bildirimleri özetlenecektir.

Tablo 4.28: 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

Pascal ve C programlarında olduğu gibi C++ programlarında da ilgili kütüphane (başlık/kitaplık) dosyalarının bildirilmesi gerekmektedir. C kütüphane dosyalarına ilave olarak bazı C++ kütüphane dosyaları Tablo 4.29'da 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 4.29: 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
```

Kullanım şekli:

```
#include <  kütüphane adı  >
```

Tablo 4.29: 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 2    veri alt alanı 2;
          ..............................
          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);
}
long sayi;
int main ()
{    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 fonksiyon*"u (*rekürsif fonksiyon*) tanımlanmıştır. Fonksiyonun tipi, yani çalışması sonucu oluşan verinin tipi uzun tamsayı (`long`) ve giriş parametresi de uzun tamsayı (`long`) cinsinden '`x`' dir. Fonksiyon; kedisine gelen uzun tamsayıyı yinelemeli olarak çarpa-

rak, bulduğu faktöriyel sonucunu ve işlem akışını "`return`" komutu ile çağırılan yere gönderir.

4. Ana Program Bölümü

Temel işlem ve kontrollerin bulunduğu ana program kısmı, C++'da da C'deki gibi "`main()`" ana fonksiyonudur.

Kullanım şekli:

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

E. 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 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) ismini verdikleri dili geliştirmeye başladılar. Ancak Temmuz 2000 yılında marka nedenlerinden dolayı Microsoft bu dilin ismini C# olarak değiştirdi.

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.

Bir C# programının genel yapısı aşağıda verilmektedir.

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 dâhil 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  ;
```

```
//  .....                                   // 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          }
        }
}
```

3. namespace

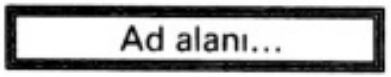
.NET Framework'te çok sayıda sınıf, yapı, arabirim vb. bir araya getirilerek ad alanlarında toplanmakta ve programlarda referans verilerek kullanılmaktadırlar. Yani C# programları, ad alanları kullanılarak düzenlenmekte ve programın hem iç hem de dış organizasyonunda yer almaktadırlar.

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 oluşacaktır. 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.

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



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.

F. Java Dilinin Yapısı

Sun Microsystems mühendislerinden James Gosling tarafından geliştirilmeye başlanmış olan Java'nın bir genel amaçlı, nesne yönelimli ve yüksek seviyeli programlama dili olarak ilk sürümü 1996 yılında piyasaya sürülmüştür. 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.

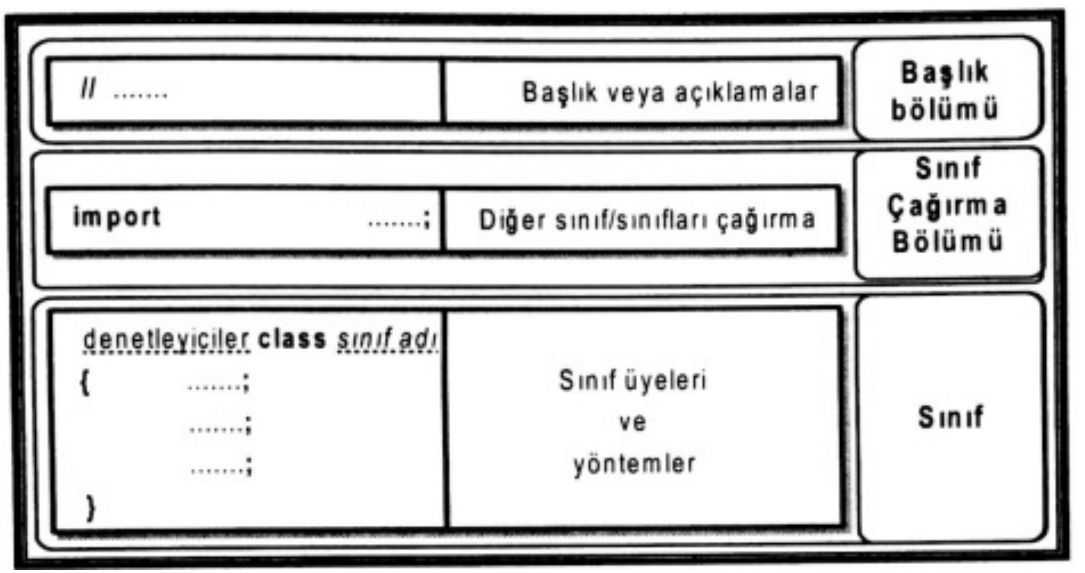
Java programları değişik şekillerde çalıştırılabilirler:

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. En basit haliyle bir Java program yapısı aşağıdadır.



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 4.30) toplanmışlardır. Diğer sınıfların, ilgili sınıfta (programda) kullanılabilmesi için önceden çağırılması gerekir. Pascal, 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 4.30: 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;
```

Kitaptaki Basic, Pascal ve C programlarının yazımında Microsoft QBasic (www.microsoft.com) (en az 1.1 sürümü), Borland Pascal (www.borland.com) (en az 7.0 sürümü) ve Borland/Turbo C (www.borland.com) (Turbo C++'ın en az 3.0 sürümü) üretici sayfalarındaki veya internette (www.qb64.net, www.freepascal.org, vb.) mevcut olan birçok yazılım geliştirme editörü kullanılabilir. C++ programlarının yazımında Bloodshed Software tarafından geliştirilen ve GPL lisansı (GNU General Public License) ile dağıtılan açık kaynak kodlu Dev-C++ editörü (en az 4.9.9.2 sürümü) kullanılabilir. 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. 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.

BASIC

```
10 REM Merhaba
20 CLS
30 PRINT "Merhaba"
40 END
```

Basic programlarında herhangi bir yapı kalıbı olmadığı için, işlemler/komutlar sırayla yazılır. Program satırlarına istenirse satır numaraları verilir. Kullanımı zorunlu olmayan açıklamalar ve başlık ifadeleri "REM" deyiminden sonra yapılır. "CLS" komutu ekranı temizlerken, "PRINT" komutu da; kendisinden sonra gelen çift tırnak içindeki alfasayısal veriyi veya değişkenin içeriğini ekrana yazdırır. "END" deyimiyle program sonlanır.

PASCAL

```
Program Merhaba;
Uses Crt;
Begin
     ClrScr; Writeln('Merhaba'); Readln;
End.
```

Pascal programlarında kullanımı zorunlu olmayan başlıklar, "Program" deyiminden sonra yazılır. Daha sonra programda kullanılacak olan komutları barındıran kütüphaneler "Uses" ile çağırılır. Ana program "Begin"-"End." bloğu arasına yazılır. "ClrScr" komutu ("Crt" içindedir) ekranı temizler, "Writeln" komutu parantez içindeki tek tırnak arasında verilen alfasayısal veriyi veya değişken içeriğini yazdırır. "Readln" komutu, Pascal programlarında klavyeden bilgi girişi için kullanılır. Ayrıca sonuç ekranını, "ENTER" tuşuna basılıncaya kadar bekletir.

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(); }
     }
}
```

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.

Son olarak; Pascal, C, C++, C# ve Java arasındaki bazı yapı benzerlikleri karşılaştırmalı olarak Tablo 4.31'de verilmektedir.

Tablo 4.31: Pascal, C, C++ ve Java arasındaki bazı yapı benzerlikleri

Kısım	Pascal	C	C++	C#	Java
Program başlığı	program ...	/* ... */	// ... veya /* ... */	// ...	// ...
Kütüphane çağırma	uses	#include ...	#include ...	using ...	import ...
Değişken bildirme	var *değişken adı* : *tip*;	*tip değişken adı* ;	*tip değişken adı* ;	*tip değişken adı* ;	*tip değişken adı* ;
Bloklar	begin ... end;	{ ... }	{ ... }	{ ... }	{ ... }
Komut sonlandırma	;	;	;	;	;
Açıklamalar	{ ... }	/* ... */	// ... veya /* ... */	// ...	// ...
Fonksiyonlar	function *ad*(*parametreler*:*tip*):*tip*; begin ..... end;	*tip ad* (*parametreler*) *tip parametreler;* { ..... }	*tip ad* (*parametreler*) { ..... }	class *ad* { ..... }	*denetleyiciler tip yöntemin adı* (*parametreler*) { ..... }
Ana program	begin ... end.	main() { ... }	main() { ... }	main()	main()

I. GİRİŞ

Algoritması 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, programlama dilleri arasında program yapısında ve komut yazımında biraz farklılıklar oluşmaktadır. Ancak bu farklılıkları kavramak ve gidermek oldukça basittir. Bu nedenle programcılığın özünde "*algoritma geliştirme*" vardır. 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 arttırır.

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: Basic, Pascal, C, C++, C# (konsol) ve Java. 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.

Basic, Pascal, C, C++, C# ve Java'daki operatörlerin 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**	**Basic**	**Pascal**	**C**	**C++**	**C#**	**Java**
Matematiksel operatörler							
^	*Üs alma*	^					
*	*Çarpma*	*	*	*	*	*	*
/	*Bölme*	/	/	/	/	/	/
+	*Toplama*	+	+	+	+	+	+
-	*Çıkarma*	-	-	-	-	-	-
.	*Ondalık ayıracı*	.	.	.	.	.	,
	Mod alma (kalan)			%	%	%	%
	Arttırma			++	++	++	++
	Azaltma			--	--	--	--
	Topla ve aktar			+=	+=	+=	+=
	Çıkar ve aktar			-=	-=	-=	-=
	Çarp ve aktar			*=	*=	*=	*=
	Böl ve aktar			/=	/=	/=	/=
Karşılaştırma operatörleri							
=	*Eşittir*	=	=	==	==	==	==
<>	*Eşit değildir*	<>	<>	!=	!=	!=	!=
<	*Küçüktür*	<	<	<	<	<	<
>	*Büyüktür*	>	>	>	>	>	>
>=	*Büyük eşittir*	>=	>=	>=	>=	>=	>=
<=	*Küçük eşittir*	<=	<=	<=	<=	<=	<=
Mantıksal operatörler							
.	*DEĞİL*	NOT	not	!	!	!	!
.	*VE*	AND	and	&&	&&	&&	&&
+	*VEYA*	OR	or	||	||	||	||
Alfasayısal operatörler							
+	*Birleştirme*	+	+	+	+	+	+
İşaretçiler							
	Adres		@	&	&	&	
	Adresteki değer		^	*	*	*	
Genel operatörler							
=	*Aktarma/atama*	=	:=	=	=	=	=
()	*Parantez*	()	()	()	()	()	()

Tablo 5.2: Akış diyagramı şekilleri ve programlama dillerindeki temel karşılıkları

Şekil	Basic	Pascal	C	C++	C#	Java
Başla	Açıklamalar	Açıklamalar Bildirimler	Açıklamalar Bildirimler	Açıklamalar Bildirimler	Açıklamalar Bildirimler	Açıklamalar Bildirimler
değişken	**INPUT** *değişken*	**Readln(** *değişken***)**	**scanf(***biçim ifadesi* **,** *&değişken***)**	**cin>>***değişken*	*değişken* **=Console.ReadLine();**	*değişken* **=nextInt();** *değişken* **=nextLine();** ...
değişken	**PRINT** *değişken*	**Write(** *değişken***)**	**printf(***biçim ifadesi* **,** *değişken***)**	**cout<<***değişken*	**Console.Write(***değişken***)**	**System.out.print(***değişken***)**
işlem	*işlem*	*işlem*	*işlem*	*işlem*	*işlem*	*işlem*
değişken=başla,dur,adım	**FOR** *değişken=başla,dur, adım*	**For** *değişken* **:=** *başla* **to** *dur* **do**	**for** **(***başla, şart, adım***)**	**for** **(***başla, şart, adım***)**	**for** **(***başla, şart, adım***)**	**for** **(***başla, şart, adım***)**
koşul / E / H / işlem1 / 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)* **then** *işlem1;* **else** *işlem2;*	**if** *(koşul)* **then** *işlem1;* **else** *işlem2;*
Dur	**END**	**End.**				

II. VERİ GİRİŞ KOMUTLARI – "Gir/Oku"

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

KAYNAKLAR

Kaynaklar

1. Ayten U. E. Algoritma ve programlama
2. Ahmet Topçu, Visual Basic 6 ile görsel programlama dili, Eskişehir Osman Gazi Üniversitesi, Ders Notları
3. Johnson B., Professional Visual Studio 2012, John-Willey and Sons, 2013
4. Volkan Aktaş, Visual Basic .Net 10, KODLAB, e-book., 2010.