



PROGRAMLAMA TEMELLERİ

1. TEMEL KAVRAMLAR

Öğr. Gör. Fevzi ÖZEK

2019-20

Kaynaklar

- Algoritmayı Anlamak Nirvana Yayınları
Mustafa EKER
- Algoritma Geliştirme ve Prog. a Giriş Seçkin
Fahri Vatansever
- 3. İnternet, C ve Algoritma ile ilgili kitaplar

Dersin Değerlendirilmesi

- Yarıyıl İçi = Arasınav x 0,50 + Devam x 0,50
- Başarı Notu = Yarıyıl İçi x 0,30 + Final x 0,70

Geçme Notu: 50

Devam sınırı: 5 hafta

Dersin İçeriği

1. Temel Kavramlar
2. Problem Çözme, Algoritma ve Akış Diyagramı
3. C Programlama Dili
 1. Dilin tarihçesi ve özellikleri
 2. Değişkenler, Sabitler ve Operatörler,
 3. Giriş-Çıkış Fonksiyonları,
 4. Karar Yapıları,
 5. Döngüler,
 6. Diziler,
 7. Alt Programlar,

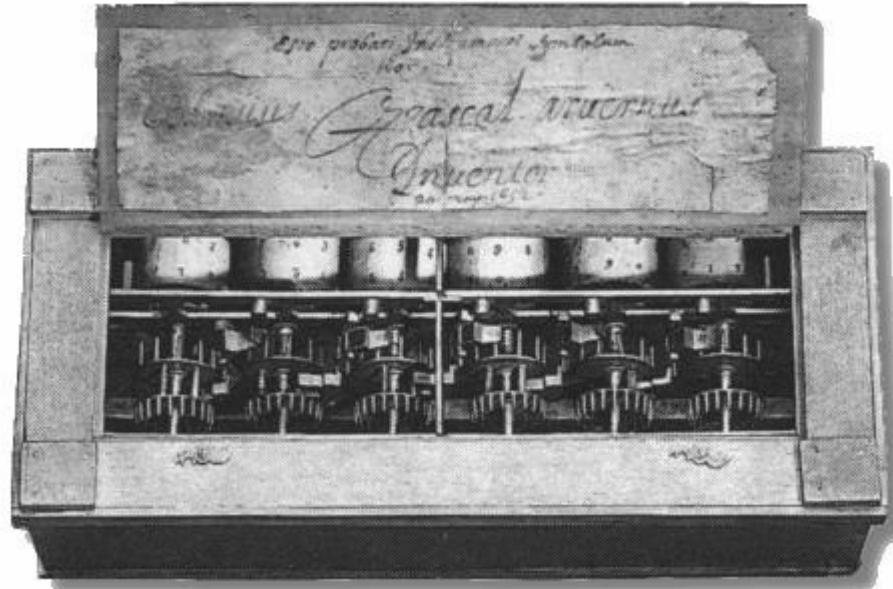
1. Temel Kavramlar

- Bilgisayar: Verilen bilgileri saklayan, gerekiğinde bu bilgileri hızlı bir şekilde istenilen amaca uygun kullanmayı sağlayan/şleyen, mantıksal ve aritmetiksel işlemleri çok hızlı biçimde yapan bir makinedir.

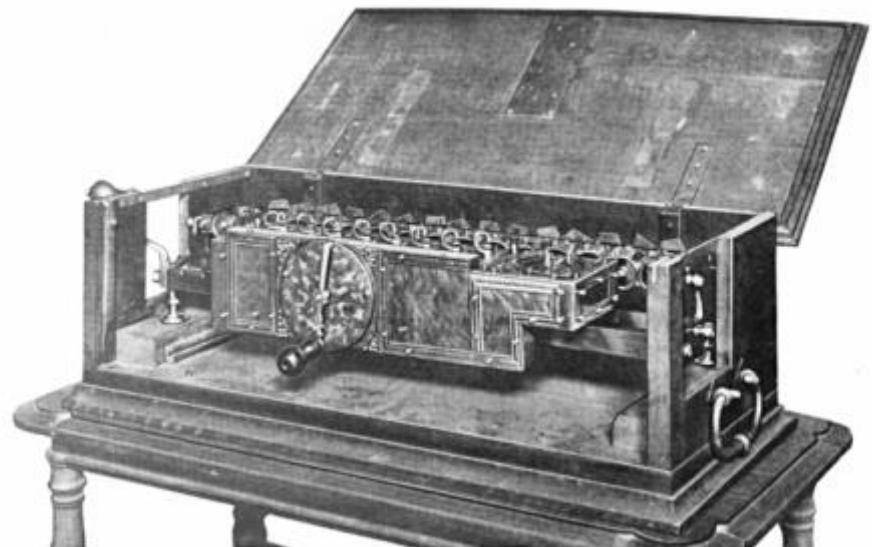
1. Temel Kavramlar (Bilgisayar Tarihçesi)

Mekanik Çağ

Blaise Pascal (1642)
Vites tabanlı toplama makinası



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



1. Temel Kavramlar (Bilgisayar Tarihçesi)

Mekanik Çağ



The Difference Engine (1822)
Charles Babbage

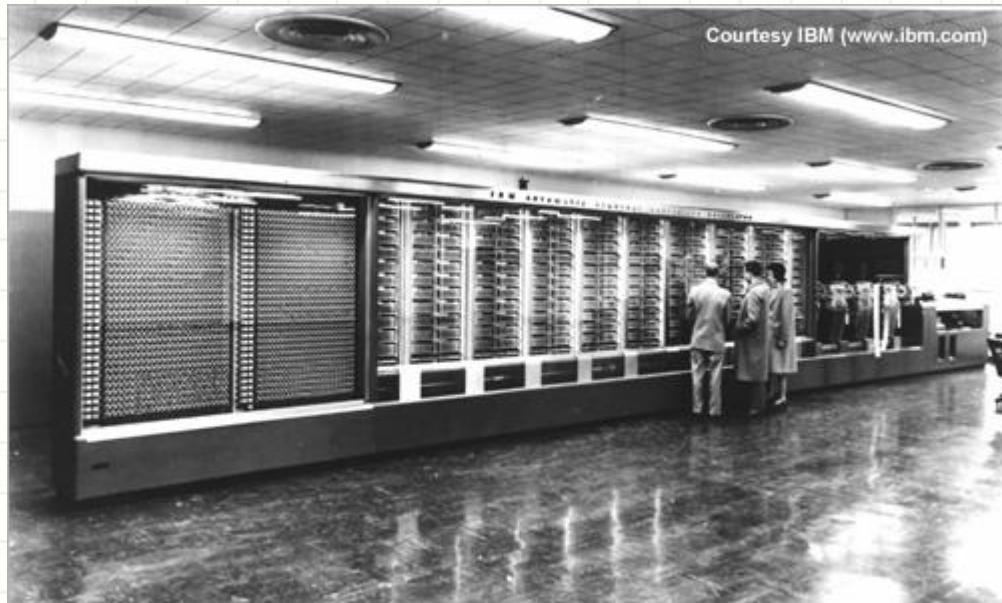


Joseph Jacquard (1810)
Bilgisayar tabanlı halı dokuma makinesi

1. Temel Kavramlar (Bilgisayar Tarihçesi)

Elektro-mekanik Çağ (1840 – 1940)

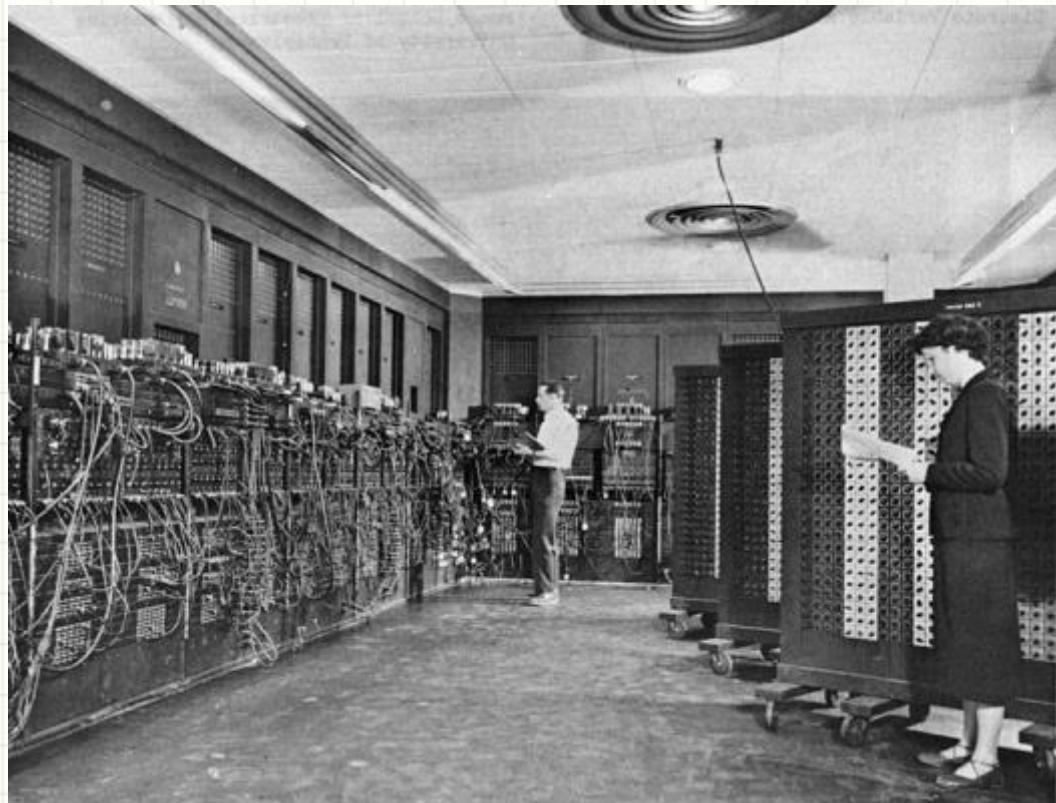
Howard Aiken + IBM
+ Harvard (1930)
Veri depolama:
Mekanik röle telefon
anahtarları (switch)
Girdi: Punch Card



1. Temel Kavramlar (Bilgisayar Tarihçesi)

Elektronik Çağ (1940 – Bugün)

Savaş sırasında bilgisayar konusundaki çalışmalar çok daha hızlı bir şekilde geliştirilmiştir. John von Neumann, 1946 yılında, ilk bilgisayar olarak kabul edilen Eniac'ı geliştirmiştir.



1. Temel Kavramlar (Bilgisayar Tarihçesi)

Elektronik Çağ (1940 – Bugün)

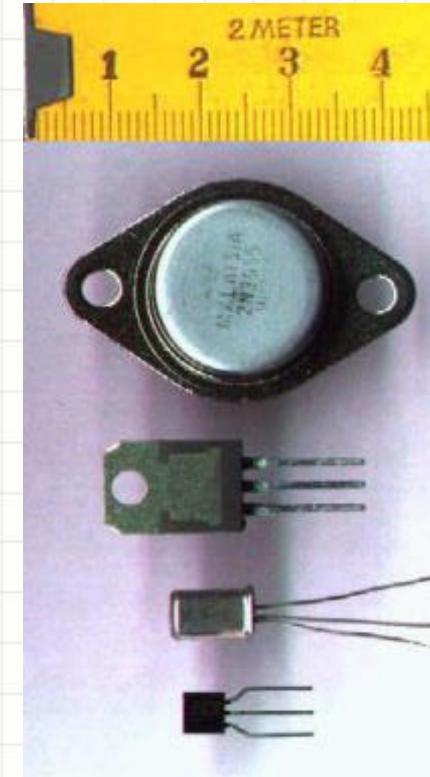
- 1000 metre kare alan
- 30 ton
- vacuum tüpleri kullanıyordu >17,000
- Karar verebiliyordu: ilk gerçek bilgisayar
- Programlama kablo temaslari ve switch ayarları ile yapıliyordu.

1. Temel Kavramlar (Bilgisayar Tarihçesi)

Elektronik Çağ (1940 – Bugün)

Bell laboratuarlarında yarıiletkenler konusundaki çalışmalar sonucunda, Walter Brattain, John Bardeen ve William Shockley tarafından tranzistör icat edilmiştir. 1950 yılında bu yeni devre elemanın patenti alınmış ve 1951 yılında da Allentown Pennsylvania'da ticari olarak üretilmeye başlanmıştır.

Tranzistörün icadı elektronikte devrim niteliğindedir.



1. Temel Kavramlar (Bilgisayar Tarihçesi)

Elektronik Çağ (1940 – Bugün)

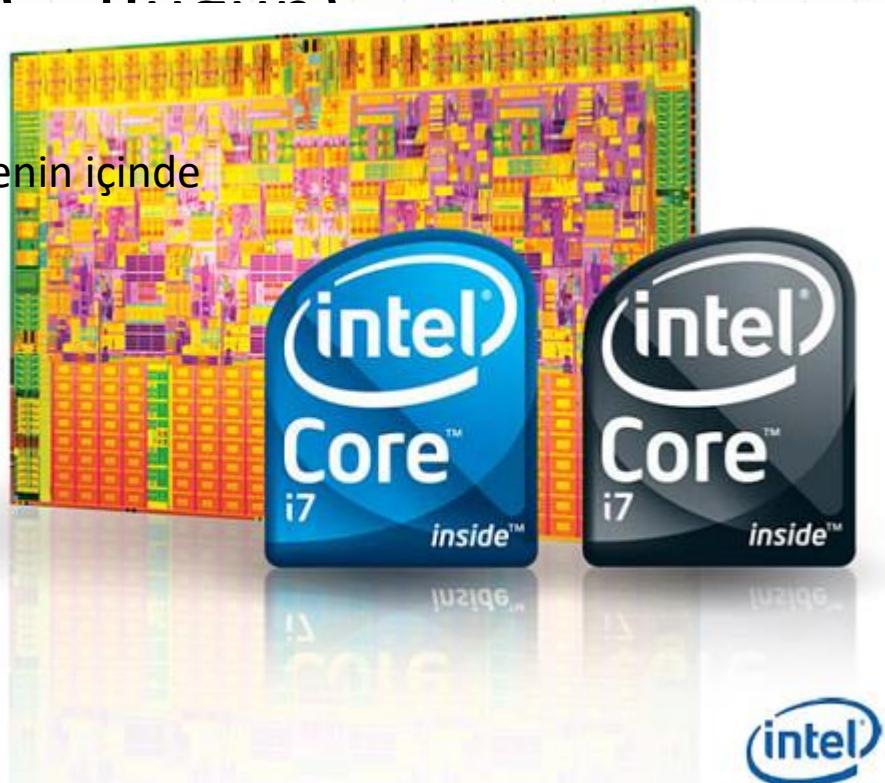
- 1950'li yıllarda yapılan araştırmalar sonucunda çok sayıda tranzistör, diyon ve kapasiteden oluşan devrelerin bir bütün olarak gerçekleştirilmesi yolu bulunmuştur. Böylece ortaya tümdevreler veya entegre devreler (integrated circuit) çıkmıştır.
- Jack Kilby, 1958 yılında, Texas Instruments firmasında ilk tümdevreyi gerçekleştirmiştir.



1. Temel Kavramlar (Bilgisayar Tarihçesi)

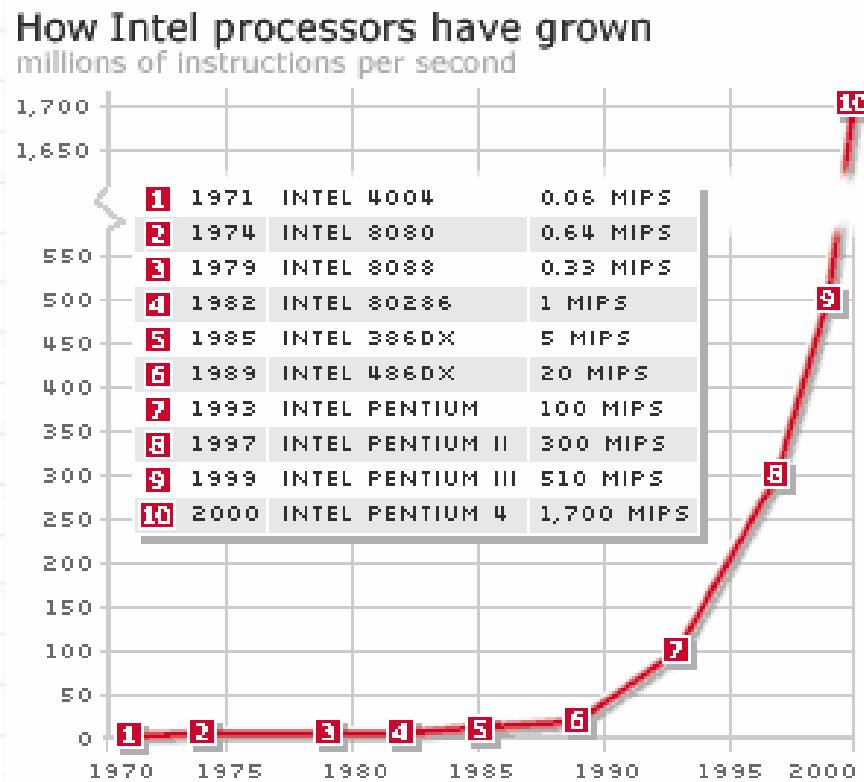
Elektronik Çağ (1940 - 1970)

781 Milyon tranzistör bir tümdevrenin içinde



1. Temel Kavramlar (Bilgisayar Tarihçesi)

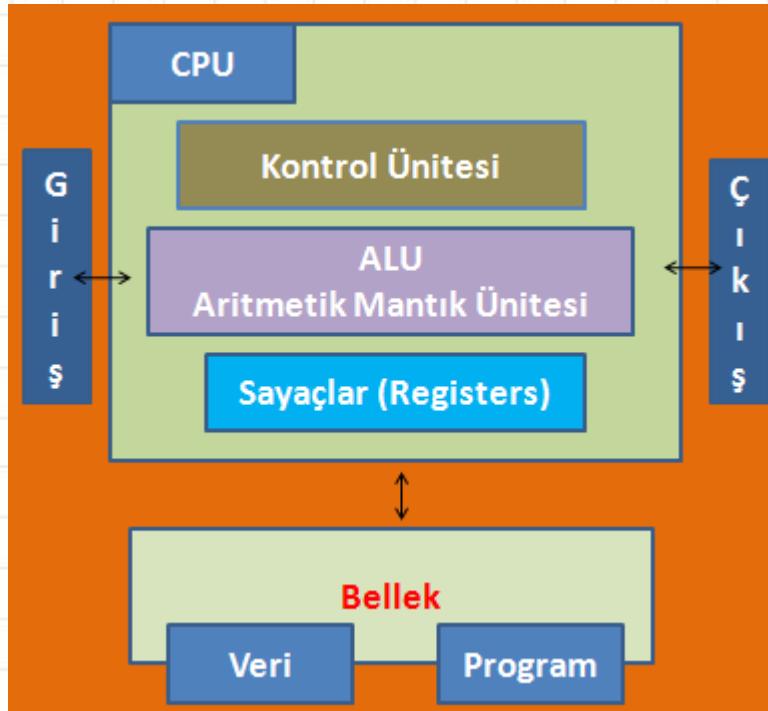
Elektronik Çağ (1940 – Bugün)



Her 18-24 ayda
bir işlemci gücü
ikiye katlanıyor.

1. Temel Kavramlar (Bilgisayar Mimarisi)

Von- Neumann mimarisi



Harward mimarisi



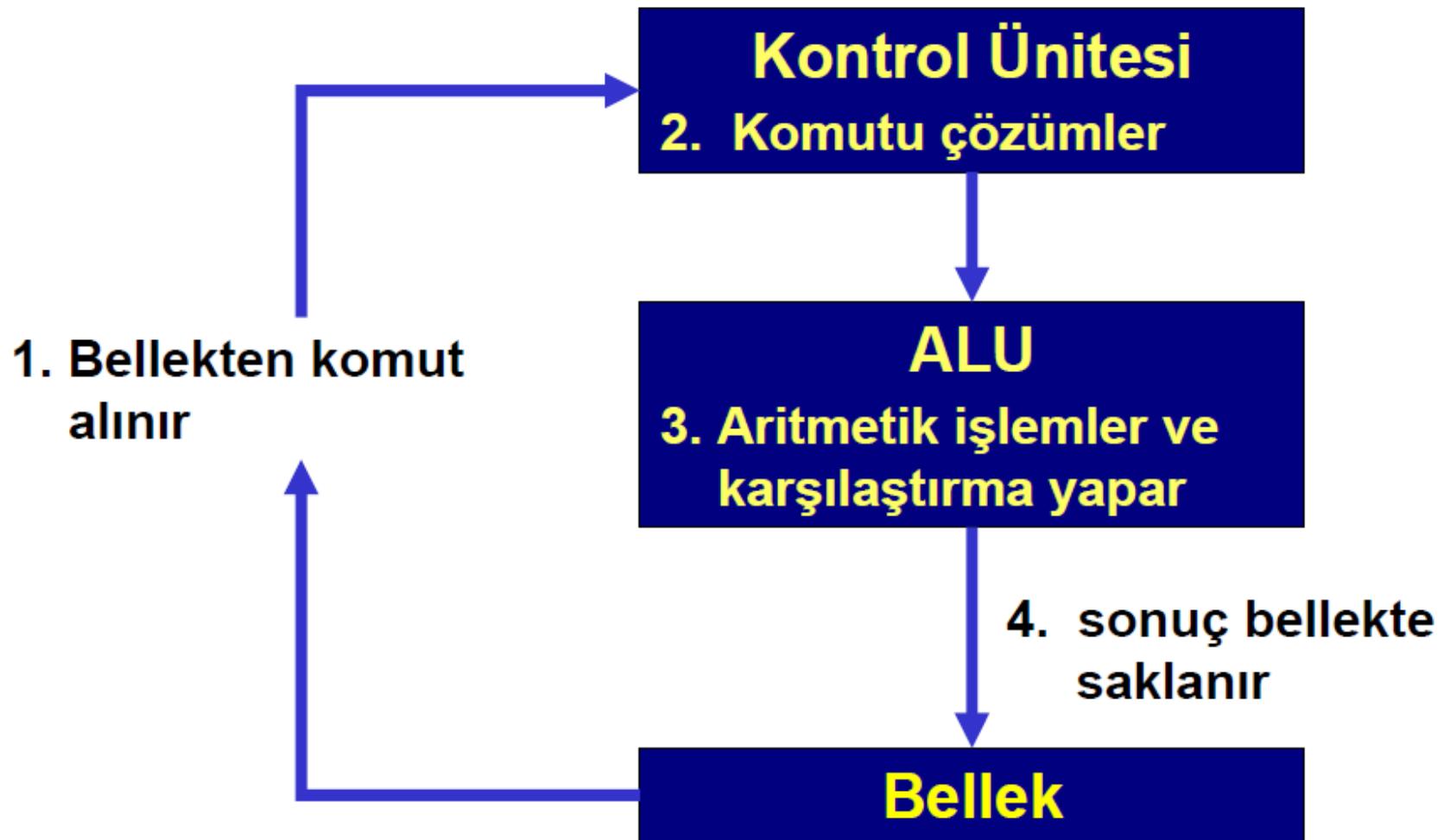
1. Temel Kavramlar (Bilgisayar Sistemi)



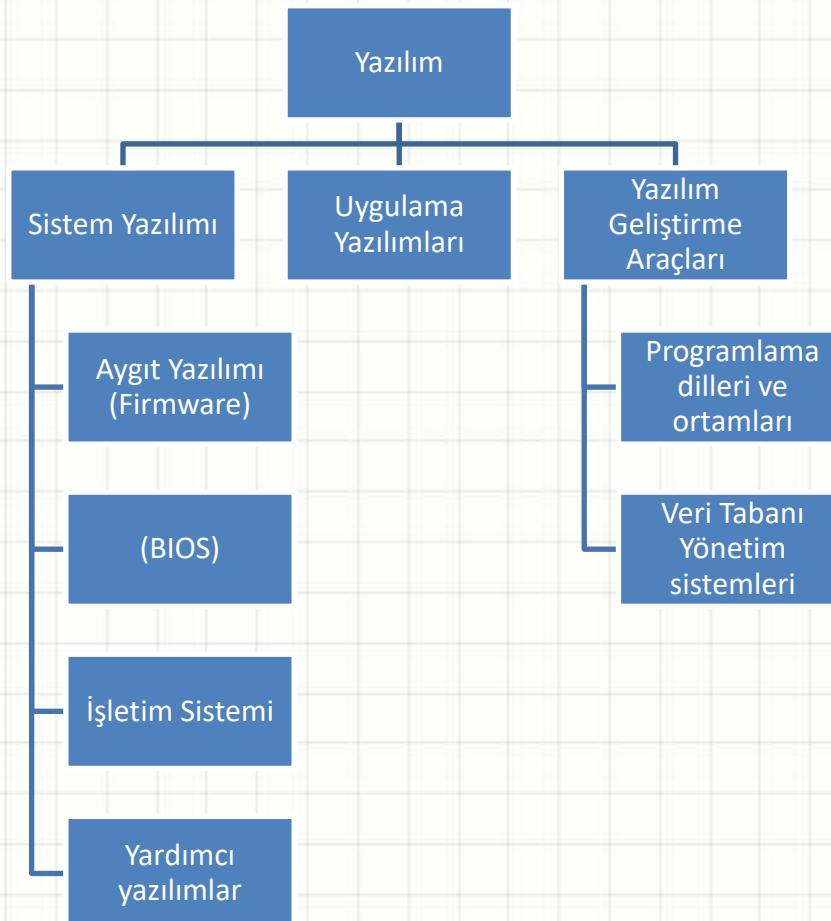
1. Donanım: fiziksel aygıtlardır.
2. Yazılım: yapılması gereken işleri yapabilmek için donanıma komutlar veren programlar topluluğudur.

1. Temel Kavramlar (Donanım)

CPU'daki Komut Döngüsü



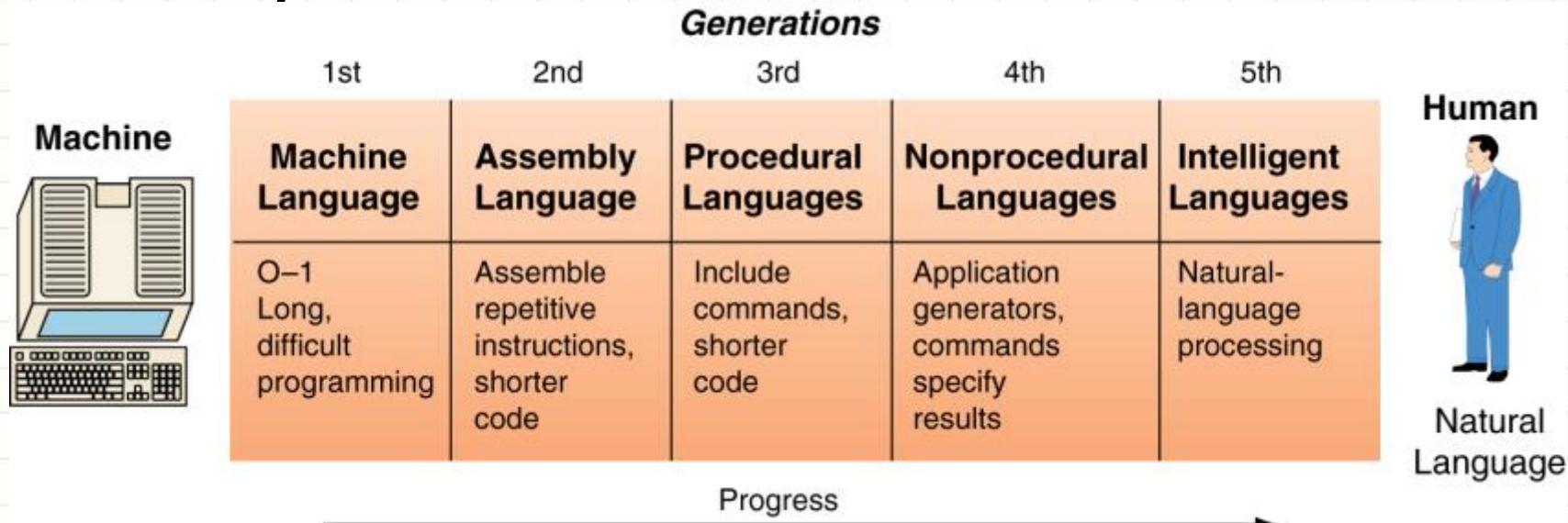
1. Temel Kavramlar (Yazılım)



1. Temel Kavramlar

- **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.

1. Temel Kavramlar (Programlama Dilleri)



- Makine dili (**birinci seviye**)
binary kodlardan oluşur (0'lar ve 1'ler)

Örn. 0110 1001 1010 1011

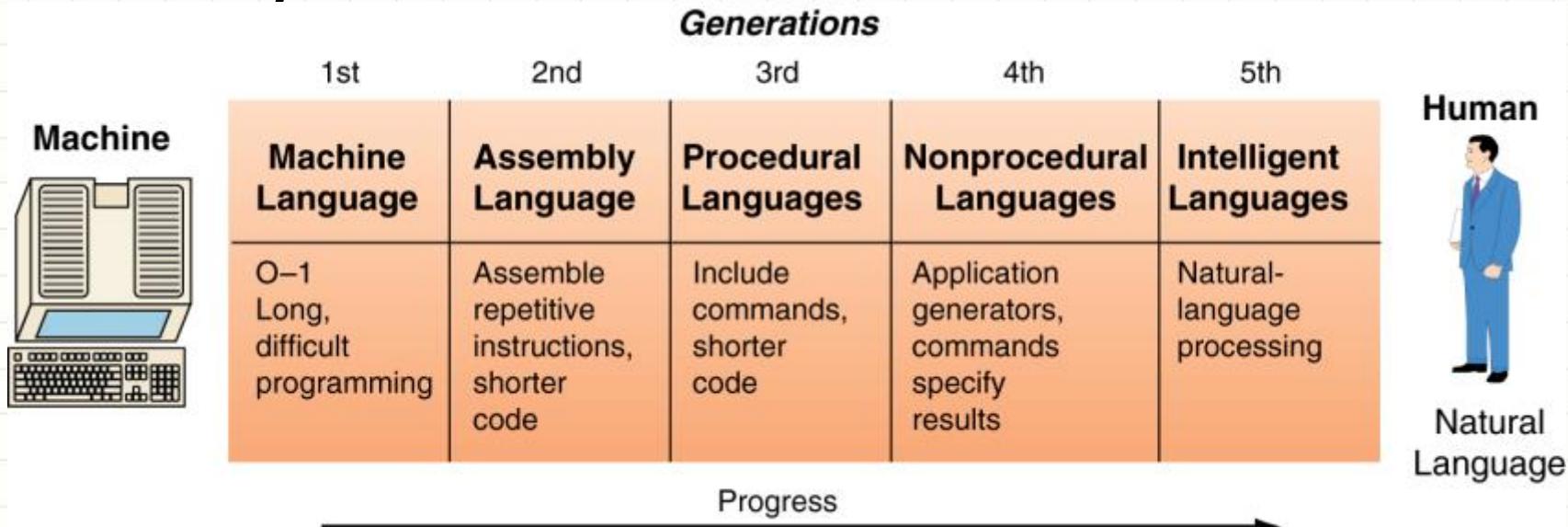
- Assembly Dili (**ikinci seviye**)

Makine diline kolay çevrilebilir.
Makine dilinden daha kolay
anlaşılabilir.

Örn. ADD X Y Z

- 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

1. Temel Kavramlar (Programlama Dilleri)



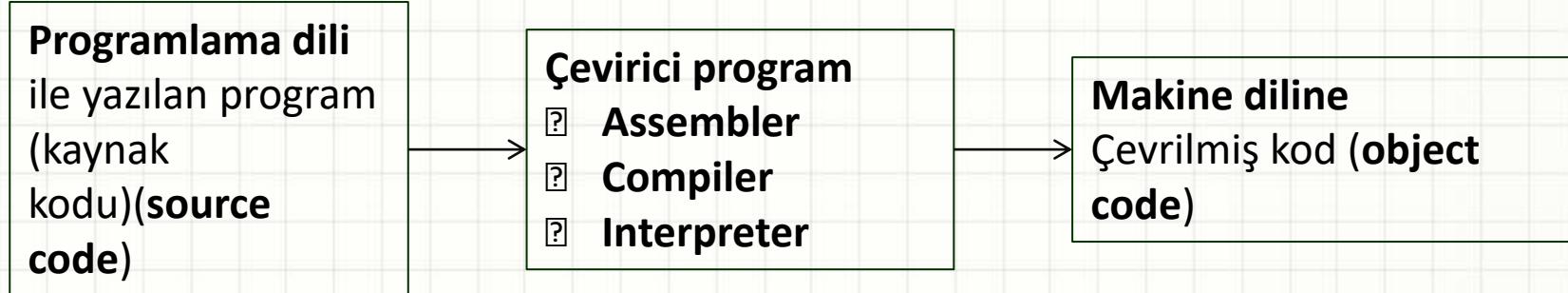
- Nonprocedural Diller (**dördüncü seviye**)

Kullanıcının sadece gerekli soruyu göstermesi sonuca ulaşması için yeterlidir.

Örnek: – veritabanı sorgulama dili- SQL
Teknik olmayan insanlar tarafından da kullanılabilir.

- Natural Language Programming Languages (**beşinci seviye (aklılı diller)**)
İnsan dilini programlama diline çevirir.
Oldukça karmaşık ve yenidir.

1. Temel Kavramlar (Derleme)



1. Temel Kavramlar

Programlama Dillerinin Tarihçesi

1950 – 1960

- FORTRAN (1955), the "**FOR**mul**A** **T**RAN**S**lator
- LISP, the "**L**I**S**t Processor",
- COBOL, the **C**Ommon **B**usiness **O**riented **L**anguage
- ALGOL **A**lgorithmic **L**anguage

1. Temel Kavramlar

Programlama Dillerinin Tarihçesi

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

1990 lar, Internet

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

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

1. Temel Kavramlar

Program Yazma Araçları

Editörler

Program kodlarını yazmak için kullanılan, metin düzenleyicilerdir.

Program kodları saf metin biçiminde yazıldığından, herhangi bir metin düzenleyicisi, program yazılımı için kullanılabilir.

IDE (Tümleşik geliştirme ortamı)

Tümleşik geliştirme ortamları, Genellikle derleyicileri –bağlayıcıları ortam içinden kullanabilmeyi yada derleyici ve bağlayıcıya ortam içinden erişme yollarını sağlarlar.

Derleyiciler: (Compiler) 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.

1. Temel Kavramlar

Program Yazma Araçları

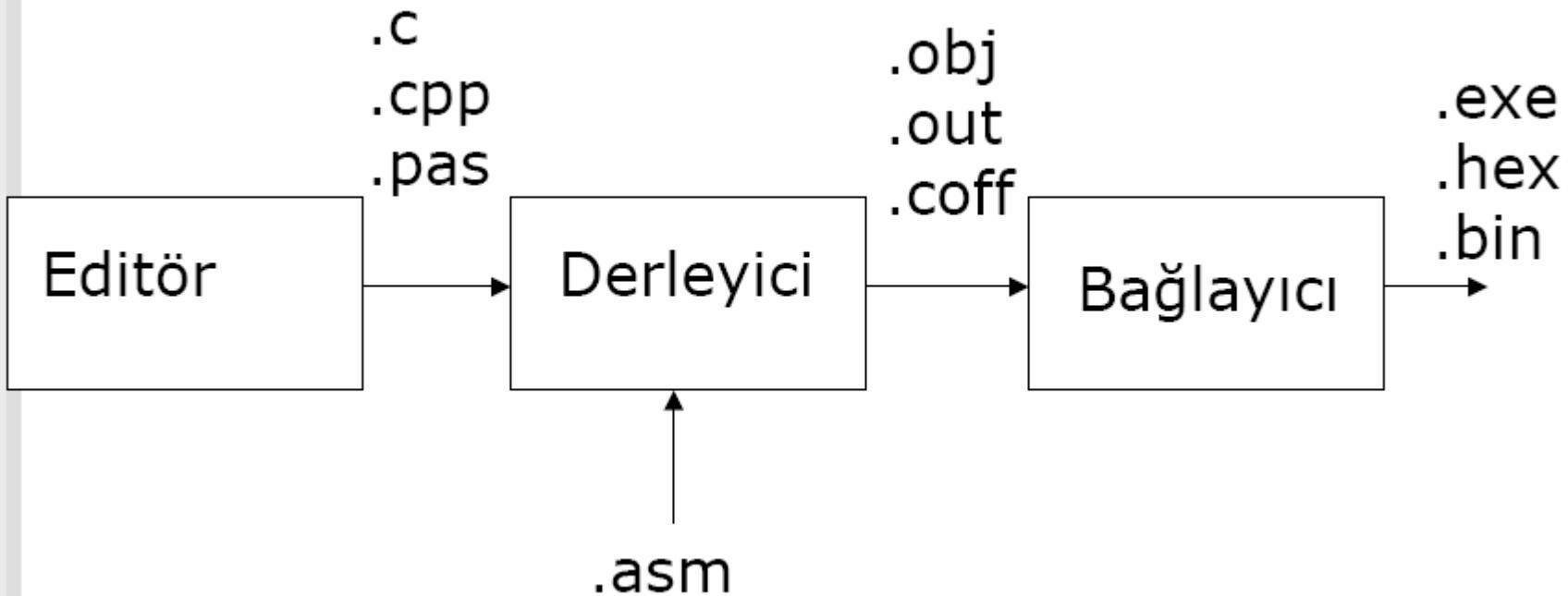
Bağlayıcılar: (linker) Bir bağlayıcı, derleyici tarafından derlenmiş olan OBJ program kodlarını uygun bellek bölgelerine yerleştirerek, değişkenlerin ve sabitlerin bellek atamalarını ve ilklemelerini gerçekleyerek tek bir çalıştırılabilir program elde eden bir yazılımıdır.

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.

1. Temel Kavramlar

Program Yazma Araçları





PROGRAMLAMA TEMELLERİ

2. PROBLEM ÇÖZME VE ALGORITMA

Öğr. Gör. Fevzi ÖZEK

2019-20

2. Problem Çözme ve Algoritma

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 parçaaya 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 dışında bırakmayacak kadar kusursuz ve eksiksiz olsun.

2. Problem Çözme ve Algoritma

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)**

2. Problem Çözme ve Algoritma

Yazılım Geliştirme Aşamaları

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

2. Problem Çözme ve Algoritma

Yazılım Geliştirme Aşamaları

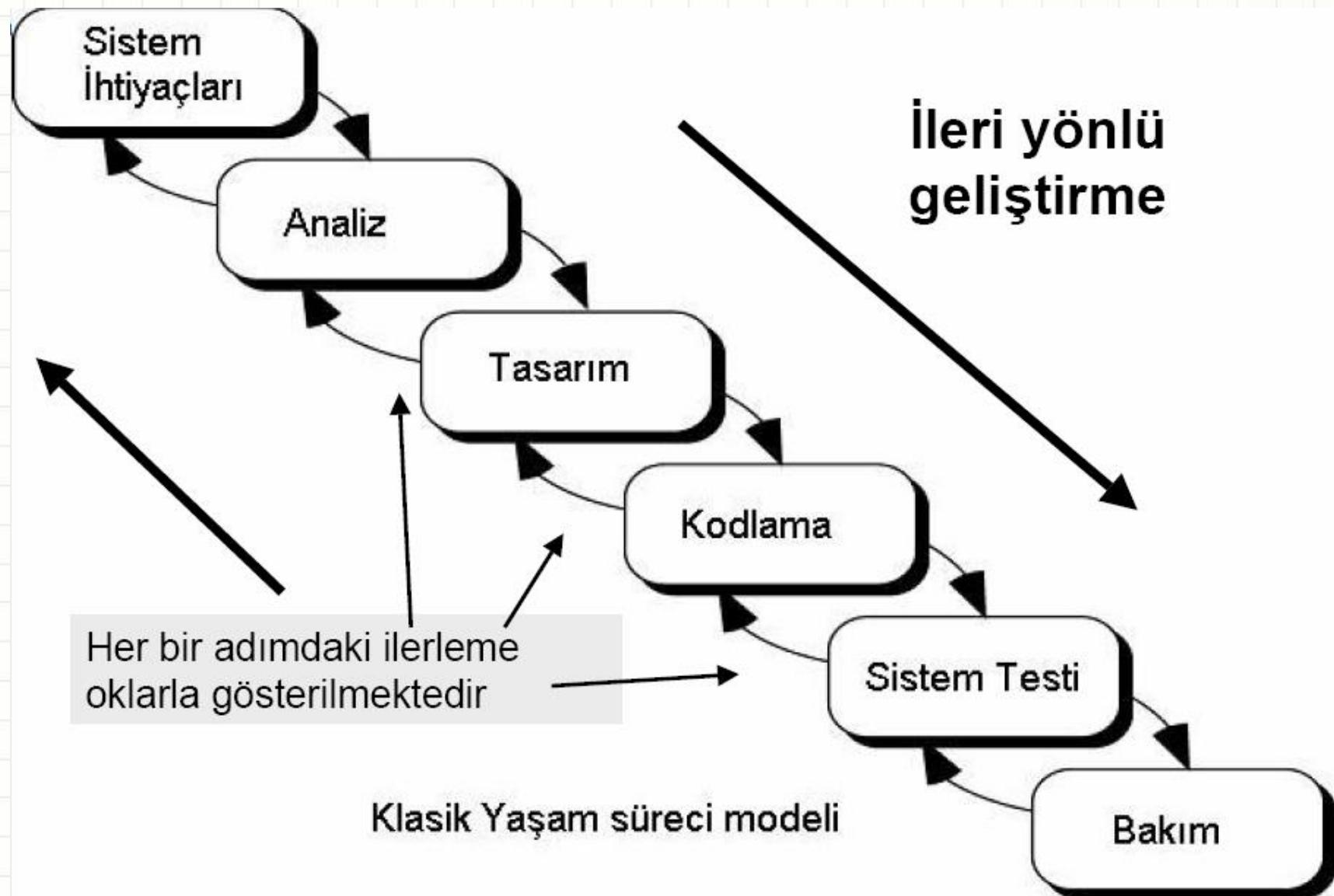
5. Test etme: Program değişik girdiler ile çalıştırılarak ürettiği sonuçlar kontrol edilerek test işlemi gerçekleştirilir. Sonuçlar bekendiği gibi ise , programın doğru çalıştığı kanıtlanmış olunur; değilse doğru çalışmayan kısımları tespit edilerek düzelttilir.

6. Derleme

7. Belgeleme: Bütün bu çalışmaların belli bir dosyalama sistemi içinde belgeler halinde saklanmasıının sağlandığı aşamadır.

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

2. Problem Çözme ve Algoritma



2. Problem Çözme ve Algoritma

Programlama Hataları

- 1. yazım hataları
- 2. mantık hataları
- 3. girdi/çıktı hataları
- 4. Çalışma zamanı hataları

2. Problem Çözme ve Algoritma

Algoritma

- Algoritma, herhangi bir sorunun çözümü için izlenecek yol anlamına gelmektedir.
- Diğer bir deyişle algoritma, verilerin, bilgisayara hangi çevre biriminden girileceğinin, problemin nasıl çözüleceğinin, hangi basamaklardan geçirilerek sonuç alınacağıının, sonucun nasıl ve nereye yazılıcağıının sözel olarak ifade edilmesi biçiminde tanımlanabilir.
- Algoritma hazırlanırken, çözüm için yapılması gereklı işlemler, öncelik sıraları gözönünde bulundurularak ayrıntılı bir biçimde tanımlanmalıdır.

2. Problem Çözme ve Algoritma

Algoritmayı hazırlarken uyulacak kurallar

1. Başla ile başlar.
2. Dur ile biter.
3. Tüm adımlar kesin ve net olmalıdır.
4. Tüm ihtimaller dikkate alınmalı
5. En kısa yol tercih edilmeli

2. Problem Çözme ve Algoritma

Örnek Algoritma

Klavyeden girilen iki sayının toplamını hesaplayıp yazdırın algoritma

- | | |
|----------------|-------------------------------|
| 1. başla | 1. başla |
| 2. A, B yi gir | 2. A yi gir |
| 3. $T=A+B$ | 3. B yi gir |
| 4. T yi yaz | 3. A ve B yi topla T ye aktar |
| 5. Dur | 4. T yi yaz |
| 1. başla | 5. Dur |
| 2. Gir A, B | |
| 3. $A+B>T$ | |
| 4. Yaz T | |
| 5. Dur | |

2. Problem Çözme ve Algoritma

AKİŞ ÇİZELGESİ

- Algoritmanın görsel/şekilsel olarak ortaya koyulmasıdır. Problemin çözümü için yapılması gerekenleri, başından sonuna kadar, geometrik şekillerden oluşan simgelerle gösterir.
- Programın saklanacak esas belgeleri olan akış şemalarının hazırlanmasına, sorun çözümlenmesi sürecinin daha kolay anlaşılır biçimde getirilmesi, iş akışının kontrol edilmesi ve programın kodlanması kolaylaştırılması gibi nedenlerle başvurulur.

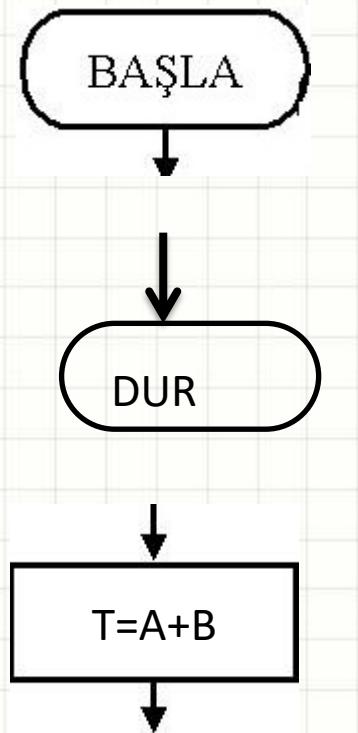
2. Problem Çözme ve Algoritma

AKİŞ ÇİZELGESİ

- Uygulamada çoğunlukla, yazılacak programlar için önce programın ana adımlarını (bölgümlerini) gösteren genel bir bakış akış şeması hazırlanır. Daha sonra her adım için ayrıntılı akış şemalarının çizimi vardır.
- En basit şekliyle dikdörtgen kutulardan ve oklardan oluşur. Akış şeması sembollerini ANSI (American National Standards Institute) standarı olarak belirlenmiş ve tüm dünyada kullanılmaktadır.

2. Problem Çözme ve Algoritma

Akış çizelgesi



Başla

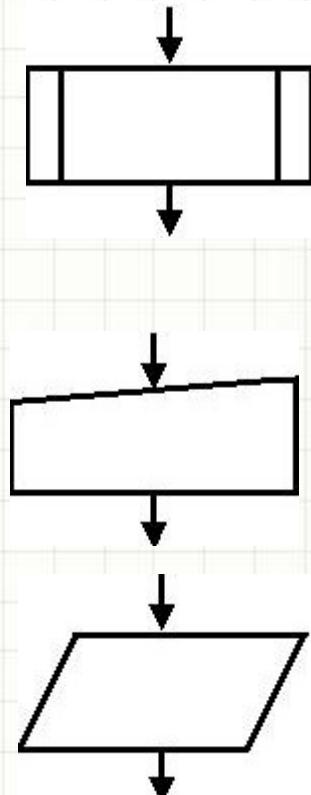
Dur

İşlem

Altprogram

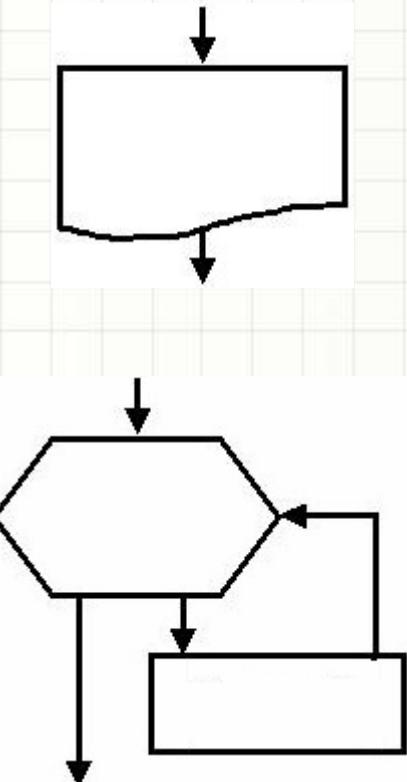
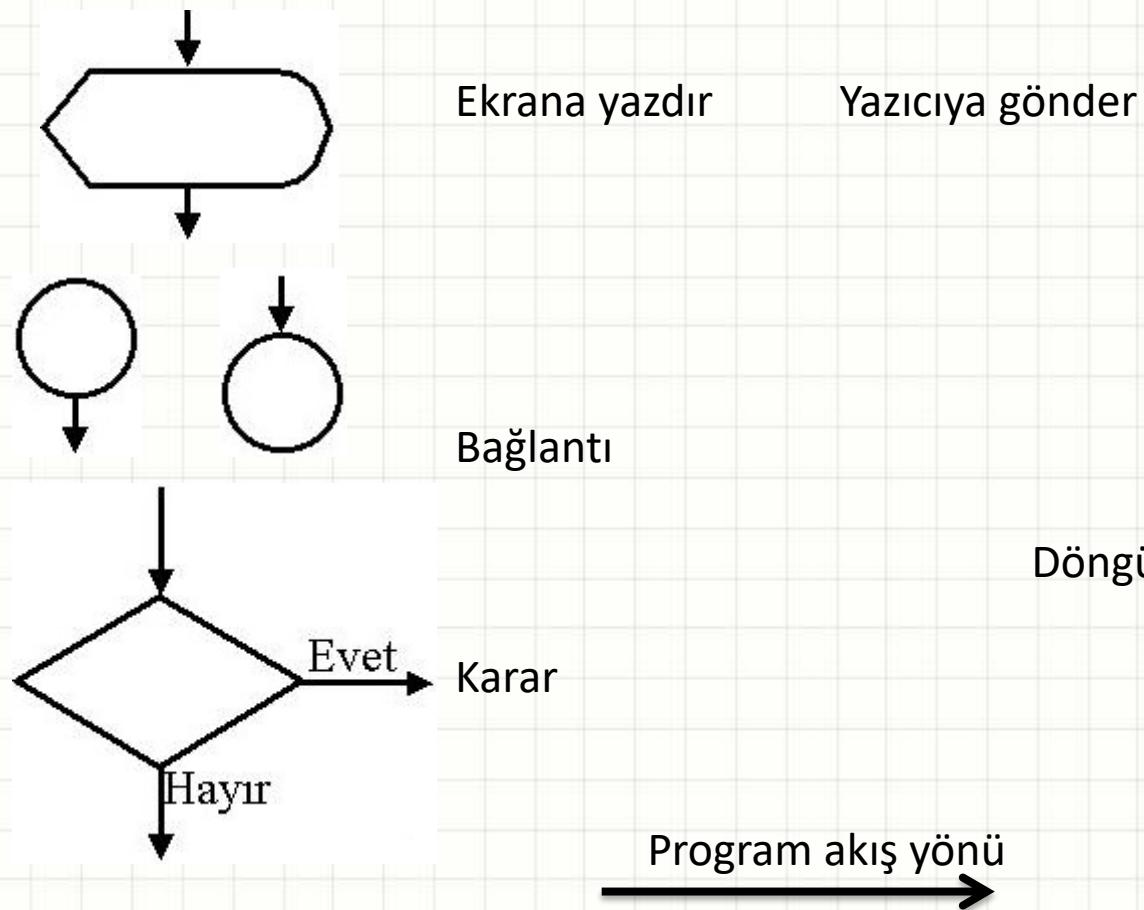
Klavyeden giriş

Giriş ya da çıkış



2. Problem Çözme ve Algoritma

AKİŞ ÇİZELGESİ

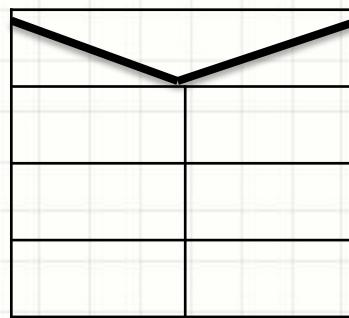


2. Problem Çözme ve Algoritma

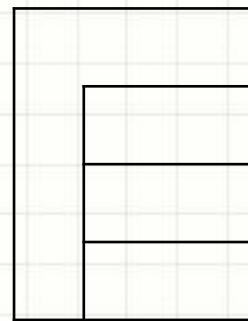
Kutu Diyagram



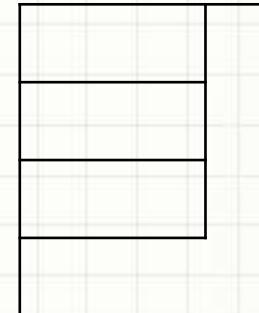
Sıralı işlem



Karar



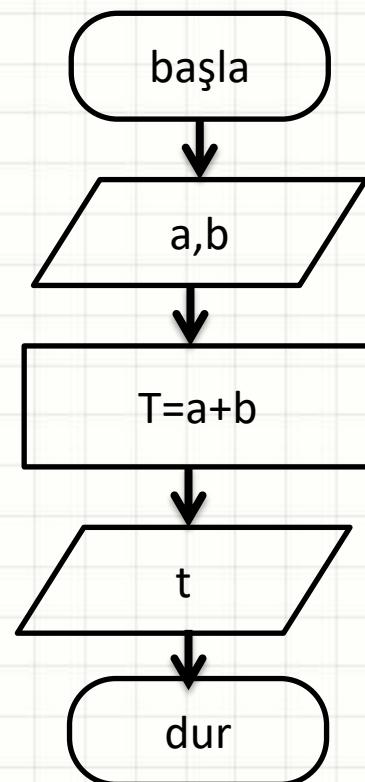
Döngü



2. Problem Çözme ve Algoritma

Örnek Akış çizelgesi

Klavyeden girilen iki sayının toplamını hesaplayıp yazdırın akış çizelgesi



2. Problem Çözme ve Algoritma

Günlük hayattan örnekler

Çay demleme?

2. Problem Çözme ve Algoritma

Günlük hayattan örnekler

Çay demleme?

1. başla
2. çaydanlığa su koy
3. demliğe çay koy
4. çaydanlık ve demliği ocağa koy
5. ocağı yak
6. bekle
7. çaydanlık kaynadı mı? E:8 H: 6
8. Çaydanlıktan demliğe su koy
9. çaydanlığa eksilen su kadar su ilave et
10. bekle
11. demlendi mi? E:12 H: 10
12. Ocağı kapat.
13. Servis yap.
14. dur

2. Problem Çözme ve Algoritma

Günlük hayattan örnekler
bölünmüş yolda karşıdan karşıya geçiş?

2. Problem Çözme ve Algoritma

Günlük hayattan örnekler
böülünmüş yolda karşıdan karşıya geçiş?

1. başla
2. sağa-sola bak
3. araba var mı? e:4 h:6
4. bekle
5. 2 ye git
6. refüje yürü
7. sağa-sola bak
8. yol boş mu? e:11 h:9
9. bekle
10. 7 e git
11. karşıya yürü
12. dur

2. Problem Çözme ve Algoritma

Günlük hayattan örnekler
sabit telefonla görüşme?

2. Problem Çözme ve Algoritma

Günlük hayattan örnekler

sabit telefonla görüşme?

1. başla
2. ahizeyi kaldır
3. çevir sesi var mı? e:7 h:4
4. ahizeyi kapat.
5. bekle.
6. 2 ye git
7. numarayı çevir
8. numara düştü mü? e:9 h:4
9. meşgul mü? e:4 h:10
10. cevap veren var mı? e:11 h:4
11. görüşmeyi yap.
12. ahizeyi kapat.
13. dur

2. Problem Çözme ve Algoritma Sayısal İşlemler

Tanımlayıcı

- Programcı tarafından oluşturulur.
- Programdaki değişkenleri, sabitleri, kayıt alanlarını, özel bilgi tiplerini vb adlandırmak için kullanılan kelimelerdir.
- Tanımlayıcılar, yerini tuttukları ifadelere çağrışım yapacak şekilde seçilmelidir.

Değişken

- Programın her çalıştırılmasında, farklı değerler alan bilgi/bellek alanlarıdır.
- Değişken isimlendirilmeleri, yukarıda sayılan tanımlayıcı kurallarına uygun biçimde yapılmalıdır.
- Dinamik/statik değişken, global/yerel değişken?

2. Problem Çözme ve Algoritma Sayısal İşlemler

Sabit

- Programdaki değeri değişmeyen ifadelere “sabit” denir. “İsimlendirme kuralları”na uygun olarak oluşturulan sabitlere, sayısal veriler doğrudan; alfa sayısal veriler ise tek/çift tırnak içinde aktarılır.

2. Problem Çözme ve Algoritma Sayısal İşlemler

Sabit

- Programdaki değeri değişmeyen ifadelere “sabit” denir. “İsimlendirme kuralları”na uygun olarak oluşturulan sabitlere, sayısal veriler doğrudan; alfa sayısal veriler ise tek/çift tırnak içinde aktarılır.

2. Problem Çözme ve Algoritma Sayısal İşlemler

Aktarma/atama

- Herhangi bir bilgi alanına, veri yazma; herhangi bir ifadenin sonucunu başka bir değişkende gösterme vb görevlerde “aktarma” operatörü kullanılır.

değişken= değer | değişken | fonksiyon | aritmetik/mantıksal ifade

- Değişken yazan kısım herhangi bir değişken ismidir.
- = sembolü, aktarma operatöründür ve sağdaki ifadenin/işlemin sonucunu soldaki değişkene aktarır. Bu durumda değişkenin eğer varsa bir önceki değeri silinir.

2. Problem Çözme ve Algoritma Sayısal İşlemler

Aktarma/atama örnekleri

a=5

b=a+5

a=b-3

a+b=c

c=a+b

c=carp(a,b)

a=a+5

a=b>c

2. Problem Çözme ve Algoritma

Sayısal İşlemler

Aritmetik İşlemler

İşlem	Matematik	Algoritma	C
Toplama	+	+	+
Çıkarma	-	-	-
Çarpma	. Veya x	*	*
Bölme	/ veya -	/	/
Üs alma	üst yazı	^	Yok
Mod	Mod	mod	%

2. Problem Çözme ve Algoritma

Sayısal İşlemler

Öncelik sırası

- ()
- ^
- / *
- - +

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

2. Problem Çözme ve Algoritma

Sayısal İşlemler

Öncelik sırası örnekler

$$X = \frac{a+b-c+2abc-7}{a+b^2-c^3}$$

$$Z = \frac{b}{c} + 2ac - \frac{2}{a+b}$$

$$V = \frac{a^2 + b^2}{2ab}$$

2. Problem Çözme ve Algoritma Sayısal İşlemler

Öncelik sırası örnekler

Örnek 1: $a=4$ $b=6$ $c=8$ $d=10$

$$x_1 = c * d / (a * d) + b + c * d / a$$

$$x_2 = c * d / a * d + b + c * d / a$$

$$x_3 = c * d / a * d + (b + c) * d / a$$

2. Problem Çözme ve Algoritma Sayısal İşlemler

Öncelik sırası örnekler

Örnek 1 cevap: $a=4$ $b=6$ $c=8$ $d=10$

$$x_1 = 8 * 10 / (4 * 10) + 6 + 8 * 10 / 4$$

$$x_1 = 8 * 10 / 40 + 6 + 8 * 10 / 4$$

$$x_1 = 80 / 40 + 6 + 80 / 4$$

$$x_1 = 2 + 6 + 20$$

$$x_2 = 8 * 10 / 4 * 10 + 6 + 8 * 10 / 4$$

$$x_2 = 8 * 2.5 * 10 + 6 + 20$$

$$x_2 = 20 * 10 + 6 + 20$$

$$x_2 = 200 + 26$$

2. Problem Çözme ve Algoritma Sayısal İşlemler

Öncelik sırası örnekler

Örnek 1 cevap: $a=4 \ b=6 \ c=8 \ d=10$

$$x_3 = 8 * 10 / 4 * 10 + (6 + 8) * 10 / 4$$

$$x_3 = 200 \quad + 14 \quad * 10 / 4$$

$$x_3 = 200 \quad + 140 / 4$$

$$x_3 = 200 \quad + 35$$

$$x_3 = 235$$

2. Problem Çözme ve Algoritma Sayısal İşlemler

Öncelik sırası örnekler

Örnek 2: $a=8$ $b=8$ $c=2$ $d=4$ $e=2$

$$x_1 = (a+b)/c * e^2 + (a+d)/(c^2+e)$$

$$x_2 = a+b/c * e^2 + a+d/c^2+e$$

$$x_3 = a+b/(c * e^2) + (a+d)/(c^2+e)$$

$$x_4 = (a+b)/c * e^2 + a+d/c^2+e$$

$$x_5 = (a+b)/(c * e^2) + (a+d)/(c^2+e)$$

2. Problem Çözme ve Algoritma

Sayısal İşlemler

Öncelik sırası örnekler

Örnek 3: $a=9$ $b=16$

$$x_1 = a + b^{1/2}$$

$$x_2 = (a+b)^{1/2}$$

$$x_3 = a + b^{(1/2)}$$

$$x_4 = (a+b)^{(1/2)}$$

2. Problem Çözme ve Algoritma

Sayısal İşlemler

Öncelik sırası örnekler

Örnek 3 cevap: a=9 b=16

$$x_1 = a + b^{1/2}$$

$$x_2 = (a+b)^{1/2}$$

$$x_3 = a + b^{(1/2)}$$

$$x_4 = (a+b)^{(1/2)}$$

2. Problem Çözme ve Algoritma Sayısal İşlemler

Öncelik sırası örnekler

Örnek 4: $a=1$ $b=2$ $c=3$ $d=4$ $e=-2$

$$x_1 = a + d/b + d^2 + 2 * a * b * c / d + e$$

$$x_2 = (a+b) / c + d^2 + 2 * a * b * c / (d+e)$$

$$x_3 = a + b / (c+d)^2 + 2 * a * b * c / d + e$$

$$x_4 = (a+b) / (c+d)^2 + 2 * a * b * c / (d+e)$$

2. Problem Çözme ve Algoritma

Sayısal İşlemler

Karşılaştırma İşlemleri

İşlem	Mat.	Algoritma	C
Eşit mi	=	=	= =
büyüktür	>	>	>
küçüktür	<	<	<
Büyük eşittir	\geq	\geq	\geq
Küçük eşittir	\leq	\leq	\leq
Eşit değildir	\neq	\neq	!=

2. Problem Çözme ve Algoritma Sayısal İşlemler

Karşılaştırma İşlemleri örnekler

2. Problem Çözme ve Algoritma

Sayısal İşlemler

Mantıksal İşlemler

- Bütün şartların sağlatılması isteniyorsa koşullar arasında VE
- Herhangi birinin sağlatılması isteniyorsa koşullar arasında VEYA
- Koşulu sağlamayanlar isteniyorsa önüne DEĞİL mantıksal operatörü kullanılır.

İşlem	Mat.	Algoritma	C
Değil	'	Değil	!
Ve	x	Ve	&&
veya	+	veya	

2. Problem Çözme ve Algoritma

Sayısal İşlemler

Mantıksal İşlemler

Doğruluk tablosu

x	y	X ve y	X veya y	Değil x
F	F	F	F	T
F	T	F	T	T
T	F	F	T	F
T	T	T	T	F

2. Problem Çözme ve Algoritma Sayısal İşlemler

Mantıksal İşlemler

Örnekler

1. Bir işyerinde çalışan işçiler arasından yalnızca yaşı 23 üzerinde olup, maaş olarak asgari ücret alanların isimleri istenebilir.
2. Bir sınıfta Bilgisayar dersinden 65 in üzerinde not alıp, Türk Dili veya Yabancı Dil derslerinin herhangi birinden 65 in üzerinde not alanların isimleri istenmektedir.

2. Problem Çözme ve Algoritma Sayısal İşlemler

Örnekler

1. klavyeden iki sayı gir. Büyük olan? (= dikkate alma)

2. klavyeden bir sayı gir. tek mi çift mi?

3. klavyeden iki sayı gir. Büyük olan? (= se "eşit" yaz)

2. Problem Çözme ve Algoritma Sayısal İşlemler

Örnekler

1. klavyeden iki sayı gir. Büyük olan? (= dikkate alma)

- | | |
|--|---------------------------|
| 1. Başla | 1. Başla |
| 2. A, b yi gir | 2. A, b yi gir |
| 3. ($a > b$) ? E: a yi yaz H: b yi yaz | 3. ($a > b$) ? E: 4 H:6 |
| 4. Dur | 4. a yi yaz |
| | 5. 7 ye git |
| | 6. B yi yaz |
| | 7. Dur |

2. Problem Çözme ve Algoritma Sayısal İşlemler

Örnekler

1.klavyeden bir sayı gir. tek mi çift mi?

1. Başla
2. A yi gir
3. $(a \bmod 2=0)$? E: "çift" yaz H: "tek" yaz
4. Dur

2. Problem Çözme ve Algoritma Sayısal İşlemler

Örnekler

3. klavyeden iki sayı gir. Büyük olan? (= se "eşit" yaz)

1. Başla
2. A, b yi gir
3. ($a > b$) ? E: 4 H:6
4. a yi yaz
5. 10 ye git
6. ($A = b$)? E: 7 h:9
7. "eşittir" yaz
8. 10 ye git
9. B yi yaz
10. dur

2. Problem Çözme ve Algoritma Sayısal İşlemler

ödev 1: iki sayıdan küçük olanı bul.

ödev 2: bir sayı giriliyor. pozitif-negatif-0 mı? bul.

2. Problem Çözme ve Algoritma Sayısal İşlemler

Örnekler

4. klavyeden üç sayı gir. en Büyük olan? (= dikkate alma)

ödev 3: üç sayıdan en küçüğü?

ödev 4: üç sayıdan ortancayı?

ödev 5: beş sayıdan en büyüğü?

2. Problem Çözme ve Algoritma Sayısal İşlemler

- | | |
|--------------------|--------------------------------------|
| 1. Başla | 1. Başla |
| 2. a, b, c yi gir | 2. a, b, c yi gir |
| 3. A>b ? E: 4 h:9 | 3. (A>b) ve (A>c) ? E: 4 h:6 |
| 4. A>c ? E: 5 h:7 | 4. A yi yaz |
| 5. A yi yaz | 5. 7 e git |
| 6. 11 e git | 6. B>c ? E: b yi yaz h: c yi yaz |
| 7. C yi yaz | 7. dur |
| 8. 11 e git | |
| 9. B>c ? E: 10 h:7 | 1. Başla |
| 10. B yi yaz | 2. a, b, c yi gir |
| 11. dur | 3. (A>b) ve (A>c) ? E: A yi yaz H:4 |
| | 4. (b>a) ve (b>c) ? E: b yi yaz H:5 |
| | 5. (c>b) ve (c>a) ? E: c yi yaz H:6 |
| | 6. dur |

2. Problem Çözme ve Algoritma Sayısal İşlemler

Örnekler

5. klavyeden üç basamaklı sayı girilsin. (sayı 3 basamaklı değilse tekrar girdirilsin.) Bu sayının basamaklarını ayrı ayrı yazdırın algoritma/akış çizelgesi?

ödev 6. klavyeden beş basamaklı sayı girilsin. (sayı 5 basamaklı değilse tekrar girdirilsin.) Bu sayının basamaklarını ayrı ayrı yazdırın algoritma/akış çizelgesi?

2. Problem Çözme ve Algoritma Sayısal İşlemler

Örnekler

6. gün/ay/yıl cinsinden iki tarih bilgisi gir. bu iki tarih arasındaki farkı aynı cinsten bulup ekrana yazdırın alg. ve akış çizelgesi? 1 ay=30 gün 1yıl=12 ay

ödev 7: saat, dakika, saniye, salise cinsinden iki zaman bilgisi girilmektedir. Bu iki zaman arasındaki farkı aynı cinsten bulup yazdırın...?

ödev 8: Klavyeden bir çocuğun doğum tarihi gün/ay/yıl olarak girilmektedir. Çocuğun okul çağına girip girmediğini ekrana yazdırın algoritma/akış çizelgesi? (60-66 ay isteğe bağlı, 66 aydan büyük okul çağы)



PROGRAMLAMA TEMELLERİ

3. C PROGRAMLAMA DİLİ

Öğr. Gör. Fevzi ÖZEK

2019-20

3. C Programlama Dili

C dili, 1972'de bu (B programlama) çalışmaların izinde yine Bell Laboratuarlarında Dennis Ritchie tarafından DEC PDP-11 bilgisayarlarında geliştirilmiştir. C, BCPL ve B dillerinin önemli bir çok kavramını kullanırken, veri yazımı ve daha bir çok güçlü özellikleri de içerir. C, genel anlamda bir işletim sistemi olan UNIX' in geliştirilmesinde kullanılmasıyla ün kazanmıştır. Bugün, bütün yeni işletim sistemleri C ve/veya C++ ile yazılmaktadır. Geçen yirmi yıl içinde C, bütün bilgisayarlar için uygun hale getirilmiştir. C, donanımdan bağımsızdır. Bu yüzden C'de dikkatli bir biçimde yazılmış bir program her bilgisayara *taşınabilir*.

3. C Programlama Dili

yazım kuralları:

1. programlar bloklardan oluşur. { ile başlar, } ile biter.
2. komutlar ; ile biter.
3. Programda kullanılacak değişkenler mutlaka tanımlanır.
4. kütüphane dosyaları mutlaka eklenmelidir.
5. büyük küçük harf ayımı vardır.

3. C Programlama Dili

Örnek program

```
/* şekil 2.1:fig02_01.c
C ile ilk program */
#include<stdio.h>
#include<conio.h>
#include<locale.h>
int main ()
{
setlocale(0, " Turkish ");
printf ( "C'ye hoş geldiniz!\n" );
getch();
return 0;
}
```

3. C Programlama Dili

c programının yapısı: (Tanımlamalar-İşlemler)

TANIMLAMALAR

1. Açıklama

/* açıklamalar1

açıklama 2

açıklama 3 */ başlık

ya da

// açıklama

2. #include // kütüphane dosyasını ekle/çağır

3. #define; // tanımlama

4. #undef; // tanımlamayı kaldırma

5. typedef; // yeni tip tanımlama

3. C Programlama Dili

c programının yapısı:

TANIMLAMALAR

6. struct; // kayıt tipi (yapısal tip) tanımlama
7. const; // sabit tanımlama

8. tip değişkenler; // değişken tanımlama

HEM İŞLEM HEM TANIMLAMA

9. tip fonk_adı(parametreler) //alt

program/fonksiyon bildirimi

{

}

3. C Programlama Dili

c programının yapısı:

İŞLEMLER

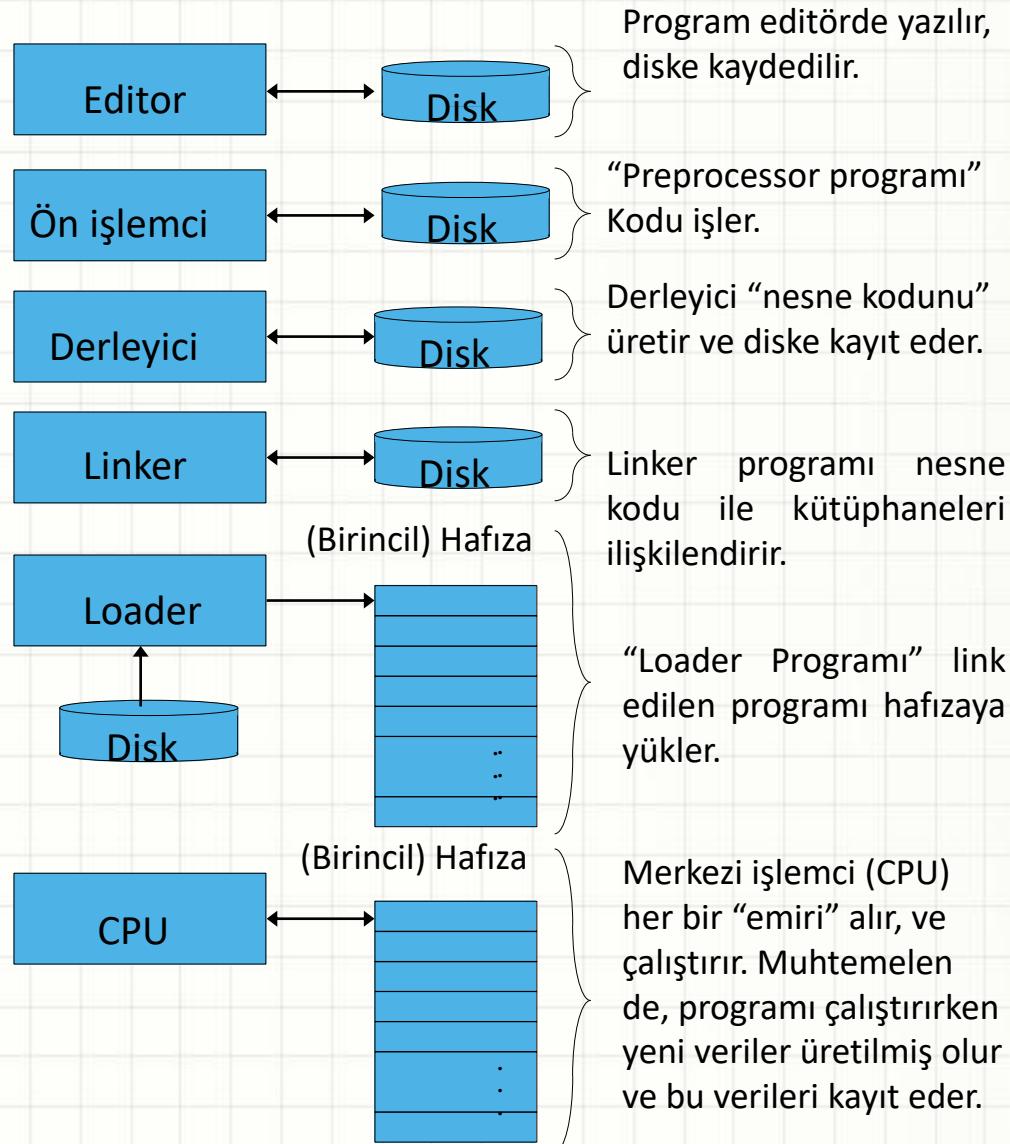
```
10. main()      // ana program  
{
```

```
}
```

C Programlama Ortamının temel elemanları

- C Programının Safhaları:

1. *Yazmak (Edit)*
2. *Ön işleme (Preprocess)*
3. *Derlemek (Compile)*
4. *Link (Link)*
5. *Yüklemek (Load)*
6. *Çalıştırmak (Execute)*



C'de Değişken Tipleri

- Değişkenler değerleri saklamak için kullanılan isimlendirilmiş alanlardır.
- Her basit bir değişken tipi belli olan tek bir değeri saklayabilir.
- C dilinde değişkenler kullanılmadan önce mutlaka tanımlanmalıdır.
- değişken isminde a..z, A..Z, 0..9 ve _ kullanılabilir.
- Ama rakamla başlamaz.

C'de Değişken Tipleri

- Genelde değişken isimleri küçük harfle, sembolik sabitler büyük harf ile gösterilir.
- C büyük küçük harf duyarlılığı olan bir dildir.
- C' de değişken tanımlarken C de rezervli olan isimler kullanılmamalıdır.
- özel karakterler ve tr harfler bulunmaz.
- Değişken isimleri en fazla 31 karakter olabilir.

Özel Amaçlı Sözcükler

auto	double	if	static
break	else	int	struct
case	entry	long	switch
char	extern	register	typedef
const	enum	return	union
continue	float	sizeof	unsigned
default	for	signed	void
do	goto	short	while

C'de Değişken Tipleri

C'DE TEMEL DEĞİŞKEN TİPLERİ

TAMSAYI (Integer)

- short int
- int
- long int
- unsigned int
- unsigned short int
- unsigned long int

GERÇEL SAYILAR (Real Numbers)

- float
- double
- long double

Karakter (Character)

- Karakter

TİP	DEKLARASYON	printf();	scanf();	Minimum	Maksimum	Byte
Karakter	char degisken;	printf("%c",degisken);	scanf("%c",°isken);	-128	127	1
Kısa Tam Sayı	short degisken;	printf("%d",degisken);	scanf("%d",°isken);	-32768	32767	2
Tamsayı	int degisken;	printf("%d",degisken);	scanf("%d",°isken);	-32768	32767	2
Uzun Tamsayı	long int degisken;	printf("%ld",degisken);	scanf("%ld",°isken);	-2147483648	2147483647	4
İşaretsiz Tamsayı	unsigned int degisken;	printf("%u",degisken);	scanf("%u",°isken);	0	65535	2
İşaretsiz Uzun Tamsayı	long unsigned degisken;	printf("%lu",degisken);	scanf("%lu",°isken);	0	4294967295	4
Virgüllü Sayı	float degisken;	printf("%f",degisken);	scanf("%f",°isken);	1,17549e-38	3,40282e+38	4
Uzun Virgüllü Sayı	double degisken;	printf("%lf",degisken);	scanf("%lf",°isken);	2,22504e-308	1,79769e+308	8

C'de Değişken Tipleri

örnek:

Bildirimler:

```
char a;           // tek karakterlilik bilgi
char ad[20];     //20 karakterlik bilgi, cümle ya da kelime
double y;
char ch;
double y=4.687;
int v,f;
float ort;
char bn[2];
long nufus;
unsigned adet;
long double ortalama;
```

Atamalar:

```
y=4.687;
ch='f';  ch=102;
```

Atama

değişken = değişken operatör deyim;

şeklinde yazılabilir. Burada operatör, +, -, *, / ya da % gibi tekli operatörler olabilir.

örnek:

a=7;

b=7.25;

c=a+b;

x+y=z;

a=b=c=d=8;

Operatörler

Aritmetik Operatörler

Tekli (unary) eksi (-): Sayıyı negatif hale getirir: -3 -9

Tekli (unary) artı (+): Sayıyı pozitif hale getirir: $+5$ $+7.8$

Çıkarma Operatörü (-): İki değerin birbirinden çıkarılmasını sağlar, $13 - 1 \rightarrow 12$
 $7 - 9 \rightarrow -2$ $2.9 - 0.3 \rightarrow 2.6$

Toplama Operatörü (+): İki değerin toplanmasını sağlar,

$$3 + 1 \rightarrow 4$$

$$5 + 2 \rightarrow -3$$

$$1.1 + 0.3 \rightarrow 1.4$$

Bölme Operatörü (/): Bir değerin diğer bir değere bölünmesini sağlar,
 $5 / 2 \rightarrow 2$ $-3.0 / 2 \rightarrow -1.5$ $6 / 2 \rightarrow 3$

Operatörler

Aritmetik Operatörler

Mod Operatörü (%): İki tam sayı değerinin birbirine bölünmesinden kalan değeri verir. Sadece tam sayı değerleri için tanımlıdır.

$$5\%2 \rightarrow 1$$

$$10\%3 \rightarrow 1$$

$$4\%2 \rightarrow 0$$

$$\begin{array}{r} 5 & | & 2 \\ - & 4 & \\ \hline & 2 & \longrightarrow 5/2 \\ & 1 & \searrow \quad 5\%2 \end{array}$$

Operatörler

Aritmetik Operatör Kuralları

- İki operatör yan yana kullanılamaz. $(2+3)$ geçersiz
- İki tamsayı işleminin sonucu tamsayıdır. $2+3 \rightarrow 5$ $5/2 \rightarrow 2$
- Sayılardan birisi reel ise sonuç reel sayıdır.
 $2.0+3 \rightarrow 5.0$ $5/2.0 \rightarrow 2.5$
- İşlem sırası parantez kullanılarak belirtilebilir.
- Parantez kullanıldığı durumlarda, işlem içten dışa doğru ilerler.
- Parantezlerin olmadığı durumda öncelik tablosu geçerlidir

Operatörler

Aritmetik Operatörlerin Öncelik Sırası

Öncelik sırası	Operatör	Özellik
En yüksek	()	İçten dışa
	- +	Tekli operatör (sağdan sola)
	* / %	İkili operatör (soldan sağa)
En düşük	+ -	İkili operatör (soldan sağa)

Operatörler

Aritmetik Operatörlerin Öncelik Sırası

$9 / 2 * 4 . 2 + 5 / 2 - 1 . 1$ işleminin sonucunu bulalım.

$$\underline{9 / 2} * 4 . 2 + 5 / 2 - 1 . 1$$



$$\underline{4 * 4 . 2} + 5 / 2 - 1 . 1$$



$$16 . 8 + \underline{5 / 2} - 1 . 1$$



$$\underline{16 . 8 + 2} - 1 . 1$$



$$\underline{18 . 8 - 1 . 1}$$



$$17 . 7$$

İsim Sabitleri

```
#define sabit_adı değer
```

Örnek:

Pi sayısını isim sabiti olarak tanımlayan komutu yazalım.

```
#define PI 3.1415
```

Veri Tipi Dönüşümü

Değişkenlerin değerlerinin veya sabitlerin veri tiplerinin başka veri tiplerine dönüştürülmesi veri tipi dönüşümü olarak adlandırılır.

Otomatik Veri Tipi Dönüşümü

```
double r=0.5, p=5.2, s;  
int i=15, q=10, w;  
char ch;
```

```
s = i/q;          /*s 1.0 değerini alır. */  
w = r * p;        /*w 2 değerini alır. */  
ch=5*i;           /*ch 75 değerini alır */
```

Veri Tipi Dönüşümü

Tanımlanan Veri Tipi Dönüşümü

(istenilen_veri_tipi) değişken_ismi

```
int sayil,sayı2;  
double bolum;  
sayil=2;  
sayı2=4;  
bolum=sayı1/sayı2; /*bolum 0.0 değerini alır */  
bolum=(double)sayı1/(double)sayı2;  
                                /*bolum 0.5 değerini alır */  
sayıl=(int) 3.6;      /*sayıl 3 değerini alır */
```

Operatörler

İşlemli atama:

int c=3, d=5, e=4, f=6, g=12; olduğunu kabul edelim

ATAMA OPERATÖRÜ	ÖRNEK	AÇIKLAMA	ATAR
<code>+=</code>	<code>c += 7</code>	<code>c = c + 7</code>	<code>c' ye 10' u</code>
<code>-=</code>	<code>d -= 4</code>	<code>d = d - 4</code>	<code>e' ye 1'i</code>
<code>*=</code>	<code>e *= 5</code>	<code>e = e * 5</code>	<code>e' ye 20' yi</code>
<code>/=</code>	<code>f /= 3</code>	<code>f = f / 3</code>	<code>f' e 2' yi</code>
<code>%=</code>	<code>g %= 9</code>	<code>g = g % 9</code>	<code>g' ye 3'ü</code>

Arttırma ve Azaltma Operatörleri

OPERATÖR	ÖRNEK DEYİM	AÇIKLAMA
++	$++a$	a'yı bir arttır ve a'nın yeni değerini a'nın içinde bulunduğu deyimde kullan
++	$a++$	a'nın değerini a'nın içinde bulunduğu deyimde kullan ve daha sonra a'yı bir arttır
--	$-b$	b'yi bir azalt ve b'nin yeni değerini b'nin içinde bulunduğu deyimde kullan
--	$b--$	b'nın değerini b'nın içinde bulunduğu deyimde kullan ve daha sonra b'yı bir arttır

Arttırma ve Azaltma Operatörleri

```
#include <stdio.h>
void main ( )
{
    int c = 5;
    printf( "%d\n", c );
    printf( "%d\n", c++ ); /* Ön artırma */
    printf( "%d\n\n", c );

    c = 5;
    printf( "%d\n", c );
    printf( "%d\n", ++c ); /* Son artırma */
    printf( "%d\n", c );
}
```

İlişkisel işleçler(operator) : iki değer arasındaki ilişkiyi test etmek için kullanılır.

<u>İşleç</u>	<u>Anlamı</u>	
>	büyük	
\geq	büyük - eşit	$x=8, y=5$ için
$=$	eşit	$x > y$ Doğru
<	küçük	$x < y$ Yanlış
\leq	küçük - eşit	$x \neq y$ Doğru
\neq	eşit değil	

Mantıksal İşleçler : İki mantıksal ifade arasındaki ilişki üzerindeki ilişkide kullanılır.

! DEĞİL (NOT)

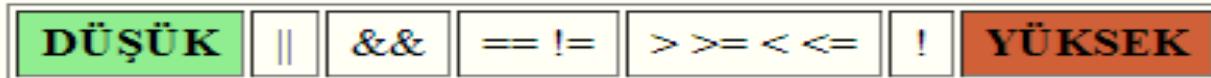
&& VE (AND)

(X>0) && (X>Y)

|| VEYA (OR)

$$(X>0) \quad || \quad (Y>0)$$

İLİŞKİSEL ve MANTIKSAL OPERATÖR ÖNCELİK SIRASI



İfadelerde matematiksel ve mantıksal işlemler bittikten sonra ilişki test edilir.

X=50, Y=80, Z=45 için

((X / 4 + Y / 4 + Z / 2) >= 50) && (Z >= 50)

OPERATÖR ÖNCELİK SIRASI

DÜŞÜK

|

^

&

<< >>

+ -

* / %

! ~ - ++ --

()

YÜKSEK

Çıktı Fonksiyonu – printf()

printf() değişkenlerin veya ifadelerin değerlerinin ekranda gösterilmesini sağlar. (stdio.h)

printf("format dizgisi", çıktı listesi);

Örnek:

```
int x = 75;  
printf("%d", x);
```

printf(biçim,değerler/değişkenler);

Örnek: printf("Bu bir
ciktidir.");

Çıktı: Bu bir ciktidir.

Çıktı: 75

Çıktı Fonksiyonu – printf()

Örnek:

```
#include <stdio.h>
int main(void)
{ printf("gecen ogrenci sayisi");
  printf("=30,");
  printf(" kalan ogrenci sayisi=");
  printf("10");
  return(0);
}
```

Çıktı:

gecen ogrenci sayisi=30, kalan ogrenci sayisi=10

Çıktı Fonksiyonu – printf()

Çıktıların ayrı satırlarda gösterilmek isteniyorsa yeni satır karakteri '\n' kullanılmalıdır.

Örnek:

```
printf("Bu 1. satır. \nBu 2. satır.");
```

Çıktı: Bu 1. satır.
 Bu 2. satır.

Çıktı Fonksiyonu - printf()

```
printf ("x=%d y=%d", x, y);
```

Yer belirleyici	Veri tipi
%c	Karakter
%d	Tamsayı
%e	Bilimsel gösterim (scientific notation)
%f %lf	Reel sayı (decimal, floating point)
%g	%e ve %f'den hangisi daha kısa ise onu kullanır
%s	Dizgi (string)
%u	İşaretsiz ondalık (Unsigned decimal)
%x	Hexadecimal

Çıktı Fonksiyonu - printf()

Formatlı Çıktı

	Örnek	Çıktı
%nd	printf ("%4d", 33);	33
%nc	printf ("%3c", 'M');	M
%ns	printf ("%10s", "Merhaba");	Merhaba
%n.mf	printf ("%f", 12.236);	12.236000
%n.me	printf ("%10.3e", -0.0536);	-5.350e-02

Çıktı Fonksiyonu - printf()

Ters Eğik Çizgi Karakter Sabitleri (\)

Kod	Açıklama
\b	Geriye doğru boşluk (backspace)
\f	Form besleme (form feed)
\n	Yeni satır
\r	Satır başı (carriage return)
\t	Sekmə (horizontal tab)
\'	Tek tırnak karakteri
\0	Boş (null)

Çıktı Fonksiyonu - printf()

Ters Eğik Çizgi Karakter Sabitleri (\)

```
printf ("%s \b%s", "Merhaba", "Nasilsin?");
```

Geriye doğru bir boşluk ver

MerhabNasilsin?

```
printf ("%s \t\t\t%s", "Merhaba", "Nasilsin?");
```

Üç sekme kadar ilerle

Merhaba

Nasilsin

Çıktı Fonksiyonu – printf()

Diğer Çıktı Fonksiyonları

puts(değişken);

Bir stringi ekrana yazdırır. (stdio.h)

putchar(değişken);

Bir karakteri ekrana yazdırır. (stdio.h)

örnek:

```
#include <stdio.h>
#include <conio.h>
main()
{
    char c,ad[10];
    c='x';
    printf("adı gir:"); scanf("%s",ad);
    putchar(c);printf("\n");
    puts(ad);
    getch();
}
```

Girdi Fonksiyonu - scanf()

scanf() fonksiyonu kullanıcı tarafından veri girişinin yapılmasını ve bu verilerin girdi listesinde belirtilen değişkenlerde saklanması sağlayan bir fonksiyondur. (stdio.h) Ayrıntılar printf ile aynıdır.

```
scanf("format_dizgisi", girdi_listesi);
```

```
int a,b;  
...  
scanf("%d%d", &a, &b);
```

Girdi: 3 5

3

a

5

b

Girdi Fonksiyonu - scanf()

örnek:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a,b,c,d,e,f;
    printf("bir tamsayı giriniz:");scanf("%u",&a);
    printf("araya bir boşluk bırakarak iki tamsayı giriniz:");
    scanf("%d %d",&b,&c);
    printf("araya bir , bırakarak iki tamsayı giriniz:");
    scanf("%d,%d",&d,&e);
    f=a+b+c+d+e;
    printf("sayıların toplamı=%d\n",f);
    getch();
}
```

Girdi Fonksiyonu - scanf()

örnek:

```
#include <stdio.h>
#include <conio.h>
main()
{
    float a,b,c;
    printf("bir ondalıklı sayı giriniz:");scanf("%f",&a);
    printf("bir ondalıklı sayı giriniz:");scanf("%3f",&b);
    c=a+b;
    printf("sayıların toplamı=%f\n",c);
    getch();
}
```

Diğer Girdi Fonksiyonları

gets (değişken) ;
string girmek için kullanılır. (stdio.h)

değişken=getch () ;
Enter tuşuna basmadan Bir karakter girmek için kullanılır.
Karakter ekranda görünmez. (conio.h)

değişken=getche () ;
Enter tuşuna basmadan Bir karakter girmek için kullanılır.
Karakter ekranda görünür. (conio.h)

kbhitz () ;
klavyeden bir tuşa basılıp basılmadığını öğrenmek için
kullanılır. (conio.h)

Diğer Girdi Fonksiyonları

Örnek:

```
#include <stdio.h>
#include <conio.h>
main()
{
    char k1[25],k2[25],a,b;
    printf("bir cümle giriniz:"); gets(k2);
    printf("aynı cümleyi giriniz:");scanf("%s",k1);
    printf ("birinci girilen %s \n",k2);
    printf ("birinci girilen %s \n",k1);
    a=getch();
    b=getche();
    printf("\n %c %c",a,b);
    getch();
}
```



PROGRAMLAMA TEMELLERİ

3. C PROGRAMLAMA DİLİ

KARAR YAPıLARI

Öğr. Gör. Fevzi ÖZEK

2019-20

Karar Yapıları

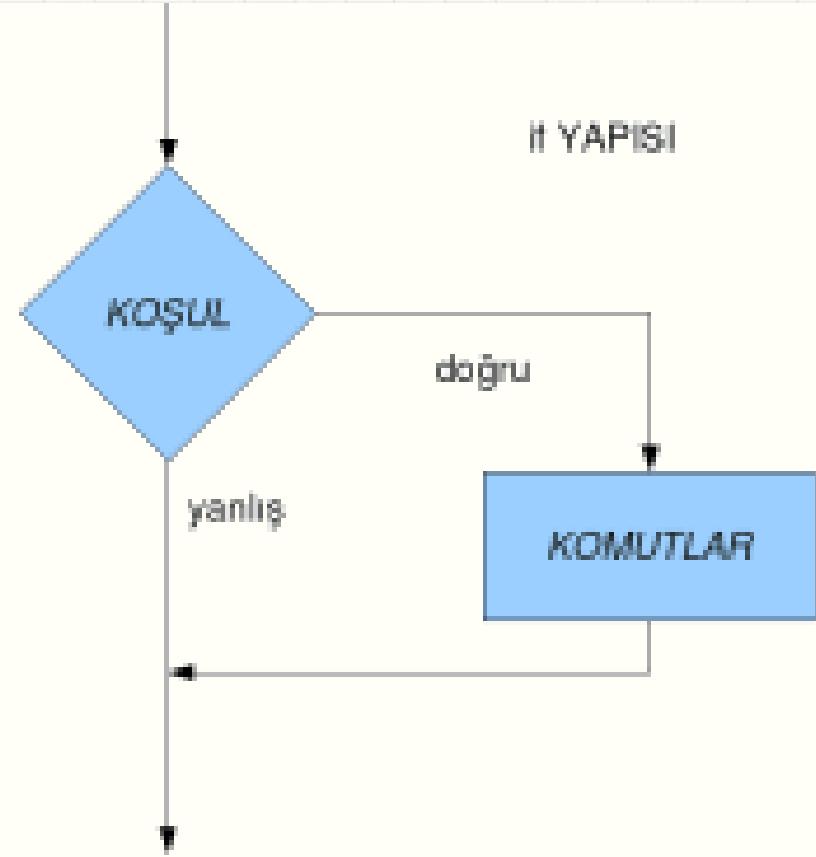
Birkaç seçenekten birini seçmek veya bir deyimin bir koşula bağlı olarak işlemek için kullanılır.

Basit if Deyimi

Mantıksal ifade doğru ise blok doğru ise sonraki blok yürütülür.

```
if (şart) komut;
```

```
if (şart)
{ komut1;
  komut2;
  komut3;
  komut4;
}
```



Örnek: Klavyeden girilen karakterin rakam olduğunu tespiti.

```
char c;  
c = getch();  
if ((c>='0') && (c<='9'))  
    printf("Rakam girdiniz.");
```

örnek: gün/ay/yıl cinsinden iki tarih bilgisi gir. bu iki tarih arasındaki farkı aynı cinsten bulup ekrana yazdırın c programı? 1 ay=30 gün 1yıl=12 ay

```
#include <stdio.h>
#include <conio.h>
main()
{
    int g1,a1,y1,g2,a2,y2,g,a,y;
    printf ("1. tarihi gün ay yıl olarak araya boşluk koyarak
gir:");
    scanf("%d %d %d",&g1,&a1,&y1);
    printf ("2. tarihi gün ay yıl olarak araya boşluk koyarak
gir:");
    scanf("%d %d %d",&g2,&a2,&y2);
```

```
if (g1<g2)
{
    a1--;
    g1+=30;
}
if (a1<a2)
{
    y1--;
    a1+=12;
}
g=g1-g2; a=a1-a2; y=y1-y2;
printf("%d yıl %d ay %d günlüğünüz.",y,a,g);
getch();
}
```

ödev 8: klavyeden girilen sayı negatifse ekrana negatif, pozitifse pozitif yazdırın
c programını yazınız.

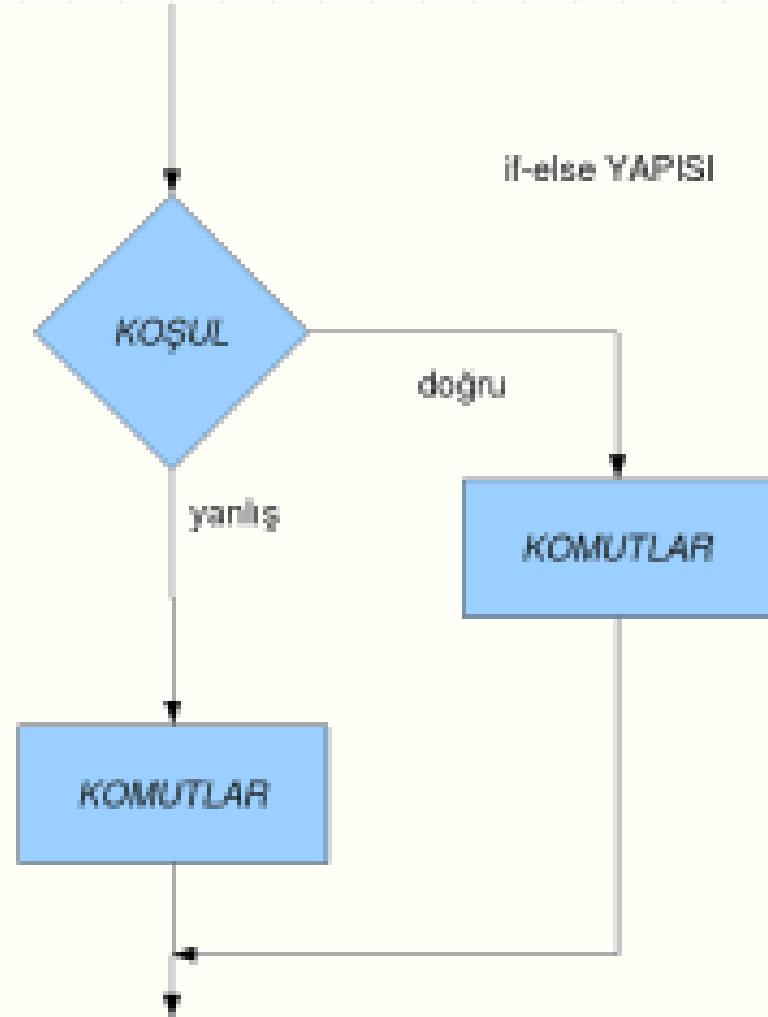
ödev 9: klavyeden üç sayı girilmektedir. en büyüğü bulup yazdırın c programını yaz.

if-else Deyimi

Mantıksal ifade doğru ise blok doğru, yanlış ise else sözcüğünden sonraki blok yürütülür.

```
if (şart) komut1;  
else komut2;
```

```
if (şart)  
{ komutlar1 }  
else  
{ komutlar2 }
```



? Şartlı işlem operatörü

Mantıksal ifade doğru ise ?
Dan sonraki, yanlış ise :
dan sonraki ifade
yürütülür.

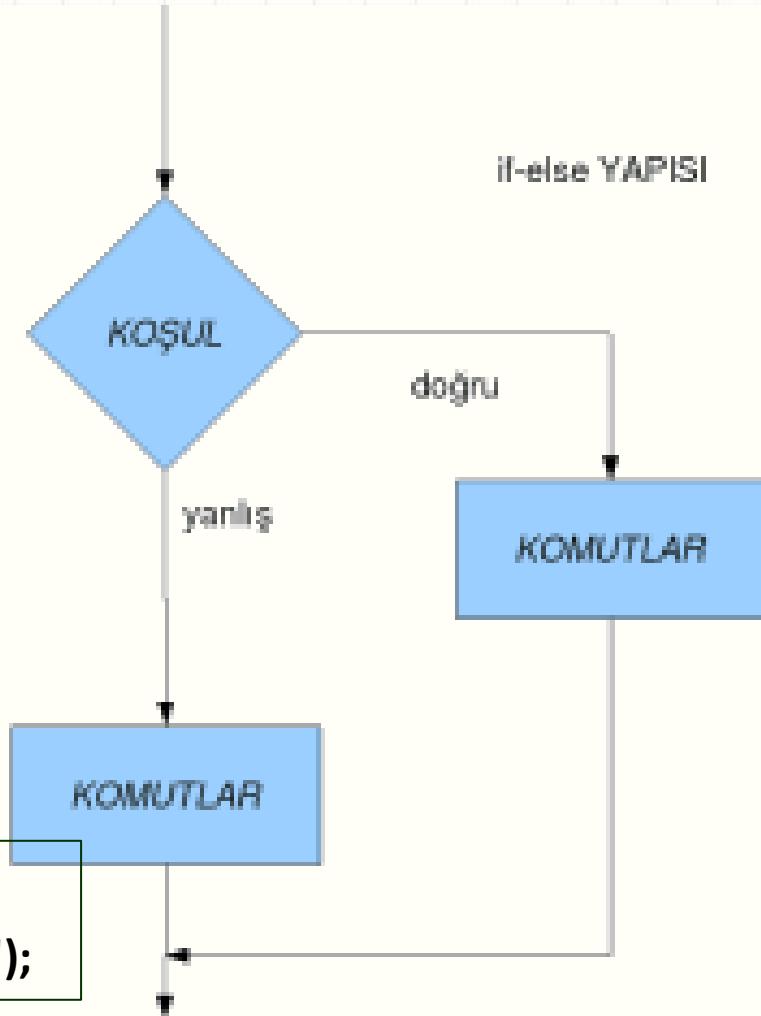
```
sart? ifade2 : ifade3;
```

örnek:

```
y=x>100 ? 5 :20;
```

Örnek:

```
( i % 2 == 1 ) ? printf("Tek"): printf("Çift");
```



Örnek : Girilen sayının tek/çift olduğunu yazan program

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i;
    scanf("%d", &i);
    if ( i % 2 == 1)
        printf("Tek");
    else
        printf("Çift");
    getch();
}
```

Örnek : Girilen iki sayıdan büyük olanı yazan program (eşitliği gözardı et)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a,b;
    printf ("sayıları gir:");scanf("%d %d",&a,&b);
    if(a>b) printf("%d daha büyüktür,\n",a);
    else printf("%d daha büyüktür,\n",b);
    getch();
}
```

ödev 10: Klavyeden bir çocuğun doğum tarihi gün/ay/yıl olarak girilmektedir. Çocuğun okul çağına girip girmedğini ekrana yazdırın c programı? (66 aydan küçükse okul çağы değil, 66 aydan büyük okul çağы)

ödev 11: klavyeden girilen bir sayının pozitif/negatif durumunu ekrana yazdırın c programı.

İç içe if

1. durum:

```
if (şart1) k1;  
else if (şart2) k2;  
    else if (şart3) k3;  
        else k4;
```

2. durum:

```
if (şart1) if (şart2) if (şart3) i4;  
    else i3;  
        else i2;  
            else i1;
```

3. durum:

```
if (şart1) if (şart2) kom1;  
    else kom2;  
else if (şart3) kom3;  
    else kom4;
```

Örnek : klavyeden iki sayı gir. Büyük olan? Eşitse "eşit" yaz.

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a,b;
    printf ("1. sayıyı gir:");
    scanf("%d",&a);
    printf ("2. sayıyı gir:");
    scanf("%d",&b);
    if(a>b) printf("%d daha büyüktür,\n",a);
    else if (b>a) printf("%d daha büyüktür,\n",b);
        else printf("eşittir");
    getch();
}
```

Örnek : Girilen üç sayıdan en küçüğünün bulunması

1. yol:

```
scanf("%d%d%d", &s1, &s2, &s3);
if ((s1<s2) && (s1<s3)) ek =s1;
else if (s2<s3)      ek =s2;
else ek = s3;
printf('En küçük olanı = %d", ek);
```

2. yol

```
scanf("%d%d%d", &s1, &s2, &s3);
if (s1<s2) if (s1<s3) ek =s1;
else ek =s3;
else if (s2<s3) ek = s2;
else ek=s3;
printf('En küçük olanı = %d", ek);
```

Pivot kullanarak çözme:

```
scanf("%d %d %d", &s1, &s2, &s3);
```

```
ek = s1;
```

```
if (ek>s2) ek = s2;
```

```
if (ek>s3) ek = s3;
```

```
printf('En küçük olanı = %d", ek);
```

Örnek : İkinci dereceden denklemin köklerinin bulunması.

```
if (delta<0) printf("Gerçel kök yoktur.\n");
else if (delta>0)
{
    x1 = (-b + sqrt(delta)) / (2 * a);
    x2 = (-b - sqrt(delta)) / (2 * a);
    printf("Birinci kök = %f\n" , x1);
    printf("ikinci kök      = %f\n" , x2);
}
else
{
    x = (-b ) / (2.0 * a);
    printf("tek kök var\n");
    printf("kök= %f\n" , x);
}
```

ödevler:

12. klavyeden girilen 5 sayıdan en büyüğü?

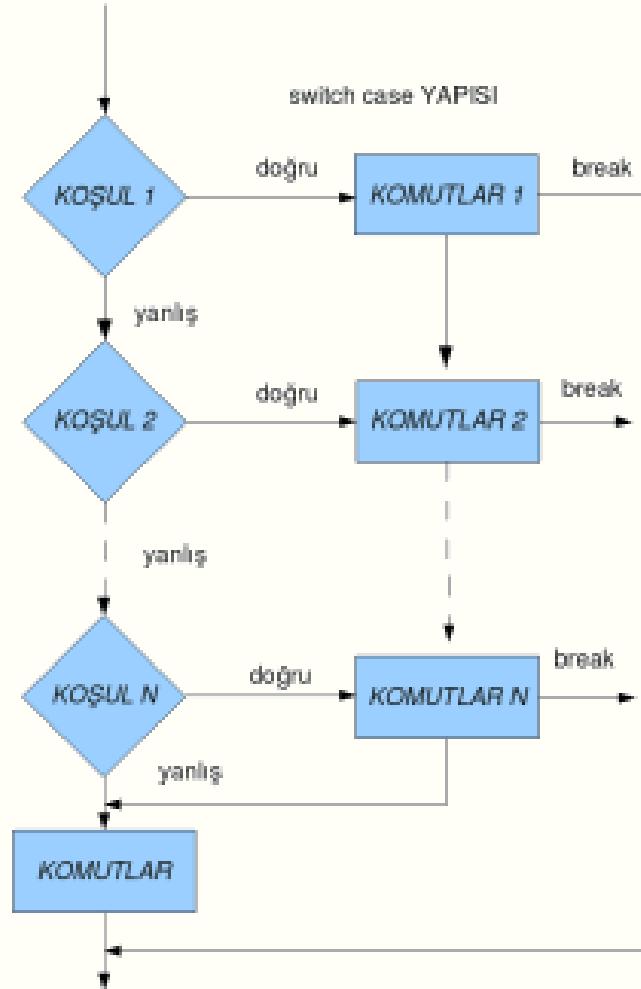
13. klavyeden bir derse ait ders notu
girilmektedir. Uygun ifadeyi yazdırın c
programı?

14. klavyeden işlem seçimi için bir karakter ve
işlem için 2 tane sayı gir.

+ ise toplam, - ise fark, x ise çarpım, / ise
bölmü hesapla ve yaz.

switch Deyimi

```
switch(<seçici>) {  
    case değer1 :  
        komutlar1;  
        break;  
    case değer2 :  
        komutlar2;  
        break;  
    .  
    default :  
        komutlarX;  
}
```



switch Deyimi

Seçicinin aldığı değere eşit seçenekin olup olmadığına bakar.

Var ise o noktadan sonraki deyimler yürütülür.
switch deyiminin sonuna gelindiğinde veya **break** deyimi ile karşılaşıldığında yürütme işlemi durur.

Programın akışı switch deyimini izleyen deyim ile devam eder.

switch Deyimi

```
switch(i) {  
    case 1 : printf("Bir");  
    case 2 : printf("İki");  
    default : printf("Hiçbiri");  
}
```

i=1 ise çıkış

Bir İki Hiçbiri

i=2 ise çıkış

İki Hiçbiri

Sorunu ortadan kaldırma için her durum için break deyimi eklenmeli.

switch Deyimi

Seçici **Ordinal** tiplerden biri olmalıdır.

(Ordinal tip: tüm değerleri listelenebilinen veri tipleri - integer, char).

Seçici ile seçenekler aynı tipte olmalıdır.

Default kısmı seçimliktir.

Seçeneklerin hiçbirini uygun değil ise yürütülür.

switch Deyimi

switch deyimi yerine if deyimi kullanılabilir.

Ancak **switch** deyimi programı daha okunabilir kıldığı için gerekli olduğu durumlarda kullanılmalıdır.

switch Deyimi

örnek: klavyeden ilköğretimden bir derse ait ders notu girilmektedir. Uygun ifadeyi yazdırın c programı?

```
#include <stdio.h>
#include <conio.h>
main()
{
    int nt;
    printf ("ders notunu gir:");scanf("%d",&nt);
    switch(nt)
    {
        case 5: printf("pekiyi\n"); break;
        case 4: printf("iyi\n"); break;
        case 3: printf("orta\n"); break;
        case 2: printf("geçer\n"); break;
        case 1: printf("başarısız\n"); break;
        default: printf("geçersiz not girildi.\n");
    }
}
```

switch Deyimi

örnek: klavyeden bir sayı gir. girilen sayı tek/çift yazdırın programı switch case yapısı ile yaz.

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a,birler;
    printf ("bir sayı gir:");scanf("%d",&a);
    birler=a % 10;
    switch(birler)
    {
        case 0:
        case 2:
        case 4:
        case 6:
        case 8: printf("çift\n"); break;
    }
}
```

```
case 1:  
    case 3:  
    case 5:  
    case 7:  
    case 9: printf("tek\n");  
}  
getch();  
}
```

switch Deyimi

örnek: klavyeden işlem seçimi için bir karakter ve işlem için 2 tane sayı gir.

+ ise toplam, - ise fark, x ise çarpım, / ise bölümü hesapla ve yaz.

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a,b; char sec;
    printf ("birinci sayıyı gir:");scanf("%d",&a);
    printf ("ikinci sayıyı gir:");scanf("%d",&b);
    printf ("seçiminizi gir:");sec=getche();
    printf("\n");
```

```
switch(sec)
{
    case '+': printf("%d+%d=%d\n",a,b,a+b);
                break;
    case '-': printf("%d-%d=%d\n",a,b,a-b);
                break;
    case 'x': printf("%dx%d=%d\n",a,b,a*b);
                break;
    case '/': printf("%d/%d=%f\n",a,b,(float)a/b);
                break;
    default: printf("hatalı seçim.\n");
}
getch();
```

Örnek : 16'lık sistemdeki rakamın 10'luk sistemdeki karşılığı (**char tipinin sayı gibi davranışsı**).

```
switch(c) {  
    case '0':  
    case '1':  
    ...  
    case '9': i = c - '0'; break;  
    case 'a':  
    case 'A': i = 10; break;  
    ...  
    case 'f':  
    case 'F': i = 15; break;  
}
```

Ödev 15: klavyeden geometrik şekil tipi için bir seçim karakteri gir. seçime göre geometrik şeklin gerekli bilgilerini klavyeden girmeyi sağlayan ve alanı hesaplayan programı yaz.

D: dikdörtgen

U: üçgen

R: Daire

Y: yamuk

Hiçbirisi: hatalı seçim



PROGRAMLAMA TEMELLERİ

3. C PROGRAMLAMA DİLİ

DÖNGÜLER

Öğr. Gör. Fevzi ÖZEK

2019-20

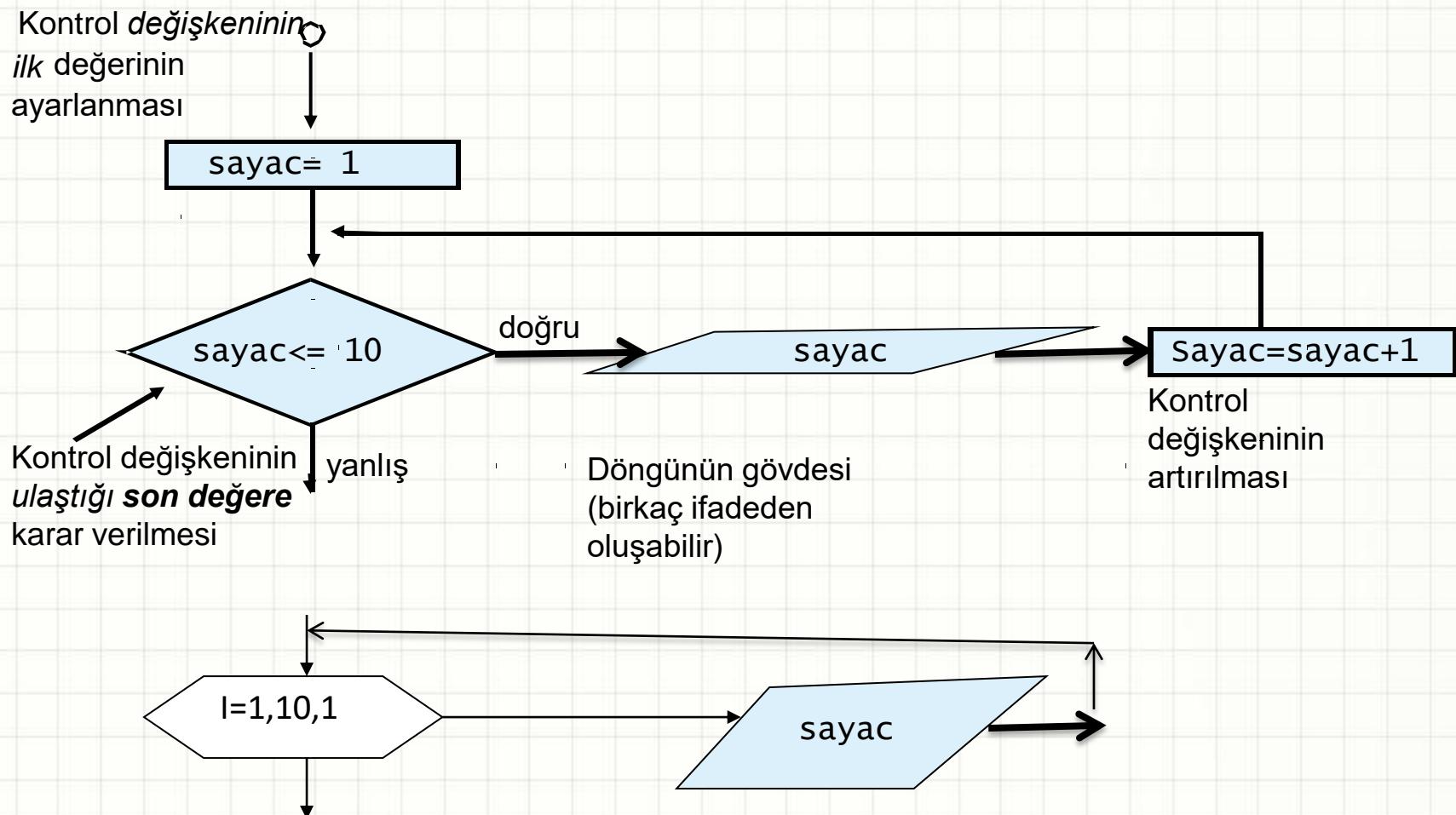
Giriş

- Bu bölümde öğrenecekleriniz
 - Döngü kontrol yapıları
 - for
 - do...while
 - break ifadesi
 - Belli kontrol yapılarından derhal ve hızlı bir biçimde çıkmak için kullanılır
 - continue ifadesi
 - Bir döngü gövdesinin geri kalan kısmını atlayarak döngünün diğer kısımlarının çalışmasını sağlar

Döngülerin Temelleri

- Döngü
 - Döngü devam koşulları doğru (true) kaldığı sürece bilgisayar bir grup emri defalarca çalıştırır
- Belirli Sayıda Tekrarlayan Döngü
 - Belirli döngü: döngünün kaç defa çalıştırılacağı önceden bilinir
 - Tekrarların sayısını saymak için kontrol değişkeni kullanılır
- Belirsiz Döngü
 - Belirsiz döngü
 - Tekrar sayısının belli olmadığı durumlarda kullanılır
 - Nöbetçi değer veri girişinin sonlandığını belirtir
 - Bir şart sağlandığı sürece devam eder/bir şart sağlandığında sona erer.

Belirli Sayıda Tekrarlı Döngüler



Belirli Sayıda Tekrarlı Döngüler

- Sayaç Kontrollü Döngü gereksinimleri
 - Kontrol değişkeninin ismine (veya döngü sayacına)
 - Kontrol değişkeninin ilk değerine
 - Kontrol değişkeninin döngü içinde artırılarak ya da azaltılarak değiştirilmesine
 - Kontrol değişkeninin son değerini kontrol edecek bir koşula (döngünün devam edip etmeyeceğini belirtmek için)

for Döngü Yapısı

```
for keyword           Control variable name           Final value of control variable  
for ( counter = 1; counter <= 10; ++counter )  
Initial value of control variable           Loop-continuation condition           Increment of control variable
```

for Döngü Yapısı

- for döngüsünün biçimi

*for (sayac değişkenine ilk değeri atama; döngü devam koşulu; artırım)
ifade*

- Örnek:

```
for( int sayac= 1; sayac<= 10; sayac++ )  
    printf( "%d\n", sayac);
```

- 1 den 10 kadar olan tamsayıları ekrana basar

for Döngü Yapısı

- for yapısı aşağıdaki gibi while yapısı biçimine çevirilebilir:

kontrol değişkenine ilk değeri atama;
while (*döngü devam koşulu*) {
 ifade;
 artırıım;
}

- Kontrol değişkenlerine ilk değeri atama ve artırıım
 - Virgülle ayrılmış listeler şeklinde olabilir
 - Örnek:

```
for (int i = 0, j = 0; j + i <= 10; j++,  
     i++)  
    printf( "%d\n", j + i );
```

for Döngü Yapısıyla İlgili Notlar ve Gözlemler

- Aritmetik ifadeler

- İlk değeri verme, döngü devam koşulu ve artırma deyimleri aritmetik operatörler içerebilir. Örneğin, $x \cdot 2$ ve $y + 10$ olsun

```
for ( j = x; j <= 4 * x * y; j += y / x )
```

yukarıdaki for döngüsü ile aşağıdaki for döngüsü eşdeğerdir.

```
for ( j = 2; j <= 80; j += 5 )
```

- for ifadesi hakkında notlar:

- Arttırma negatif olabilir (azaltma)
 - Eğer döngü devam koşulu en baştan yanlışsa (false)
 - for yapısının gövdesi tümden atlanır
 - for yapısından sonraki ilk satır çalıştırılır
 - Kontrol değişkeni
 - Döngü gövdesinde sıkılıkla yazdırılır ya da işlemlere sokulur

Örnek:

örnek: 1 den 5 e kadar sayıları ekrana yazdırın program.

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i;
    for (i=1;i<=5;i++) printf("i=%d\n",i);
    getch();
}
```

Örnek:

örnek: klavyeden girilen 10 tane sayının ortalaması?

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i,t=0,a;
    float ort;
    for (i=1;i<=10;i++)
    {
        printf("%dinci sayıyı gir",i); scanf("%d",&a);
        t+=a;
    }
    ort=(float)t/10;
    printf("ortalama=%f\n",ort);
    getch();
}
```

Örnek:

örnek: klavyeden girilen sayının faktoriyeli?

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

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

```
main()
```

```
{
```

```
    int i,n,f=1;
```

```
    printf("faktoriyeli hesaplanacak sayıyı gir:"); scanf("%d",&n);
```

```
    for (i=1;i<=n;i++) f*=i;
```

```
    printf("faktoriyel=%d\n",f);
```

```
    getch();
```

```
}
```

break ve continue İfadeleri

- break
 - while, for, do...while veya switch ile kullanıldığında o yapıdan çıkışı sağlar
 - Program yapıdan sonraki ilk ifadeyi çalıştırarak devam eder
 - break ifadesinin genel kullanımları
 - Döngüden istenen anda çıkmak
 - switch yapısında olduğu gibi belli bir kısmından kurtulmaktadır

break ve continue İfadeleri

- continue
 - while, for veya do...while yapıları içinde çalıştığında döngü gövdesinin kalan kısmını atlar
 - Döngü sıradaki tekrar ile devam eder
 - while ve do...while
 - Döngü devam koşulu continue ifadesinden hemen sonra değerlendirilir
 - for
 - Arttırma deyimi çalıştırılır, sonra da döngü devam koşulu kontrol edilir

break ve continue örnek

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i;
    for (i=1;i<=10;i++)
    {
        if (i==5) continue;
        printf("i=%d\n",i);
    }
    printf("döngü sonu i=%d\n",i);
    getch();
}
```

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i;
    for (i=1;i<=10;i++)
    {
        if (i==5) break;
        printf("i=%d\n",i);
    }
    printf("döngü sonu i=%d\n",i);
    getch();
}
```

Mantık Operatörleri

- `&&` (mantıksal ve)
 - İki koşul da aynı anda doğru (true) ise doğru dönürür
- `||` (mantıksal veya)
 - İki koşuldan en az biri doğru (true) ise doğru döndürür
- `!` (mantıksal DEĞİL)
 - Doğruyu yanlış, yanlışı doğruya çevirir
 - Tekli operatördür, sadece tek bir işleneni vardır
- **Useful as conditions in loops**

İfade	Sonuç
<code>true && false</code>	<code>false</code>
<code>true false</code>	<code>true</code>
<code>!false</code>	<code>true</code>

Ödevler:

ödev 16. klavyeden N değeri girilmektedir. 1 den N e kadar sayıların toplamını, 1 den N e kadar çift sayıların toplamını, 1 den N e kadar tek sayıların toplamını ayrı ayrı bulup ekrana yazdırın c programını yaz.

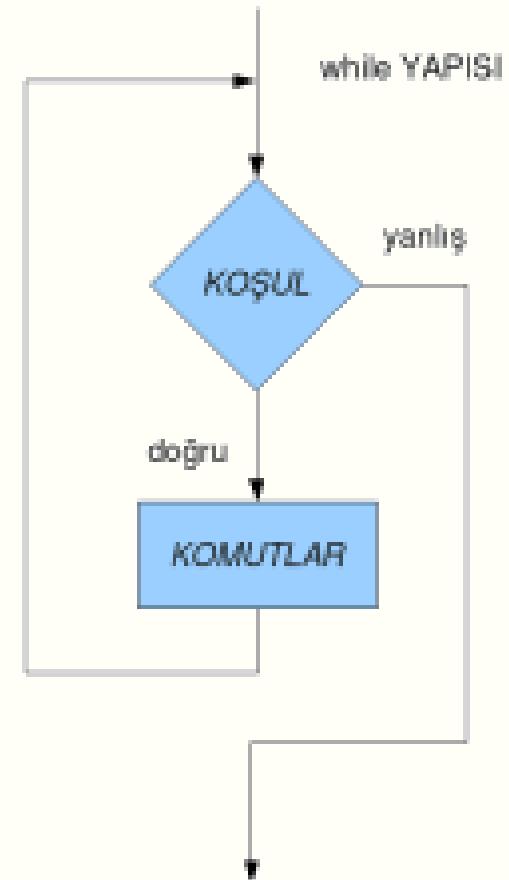
ödev 17. Klavyeden N ve R değerleri girilmektedir. $C(N,R)$ yi hesaplayan programı yazınız.

ödev 18. Klavyeden taban (x) ve kuvvet (y) girilmektedir. x üzeri y yi hesaplayan programı yazınız.

while

while(**koşul**) komutlar;

Mantıksal ifade doğru olduğu sürece komutlar yürütülür. Eğer yanlış ise kontrol bir sonraki deyime geçer.



while örnek

örnek: klavyeden sayılar girilmektedir. 0 girilene kadar girilen sayıların ortalaması?
(0 hariç)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i=0,t=0,a=5;
    float ort;
    while(a!=0)
    {
        printf("sayıyı gir. bitirmek için 0 gir.");
        scanf("%d",&a);
        t=t+a; i++;
    }
    ort=(float)t/(float)(i-1);
    printf("ortalama=%0.3f\n",ort);
    getch();
}
```

while örnek

örnek: klavyeden sayılar girilmektedir. 0 girilene kadar girilen sayıların ortalaması?
(0 hariç) break li çözüm

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

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

```
main()
```

```
{
```

```
    int i=0,t=0,a;
```

```
    float ort;
```

```
    while(0==0)
```

```
{
```

```
        printf("sayıyı gir. bitirmek için 0 gir.");
```

```
        scanf("%d",&a);
```

```
        if (a==0) break;
```

```
        t=t+a; i++;
```

```
}
```

```
    ort=(float)t/(float)i;
```

```
    printf("ortalama=%0.3f\n",ort);
```

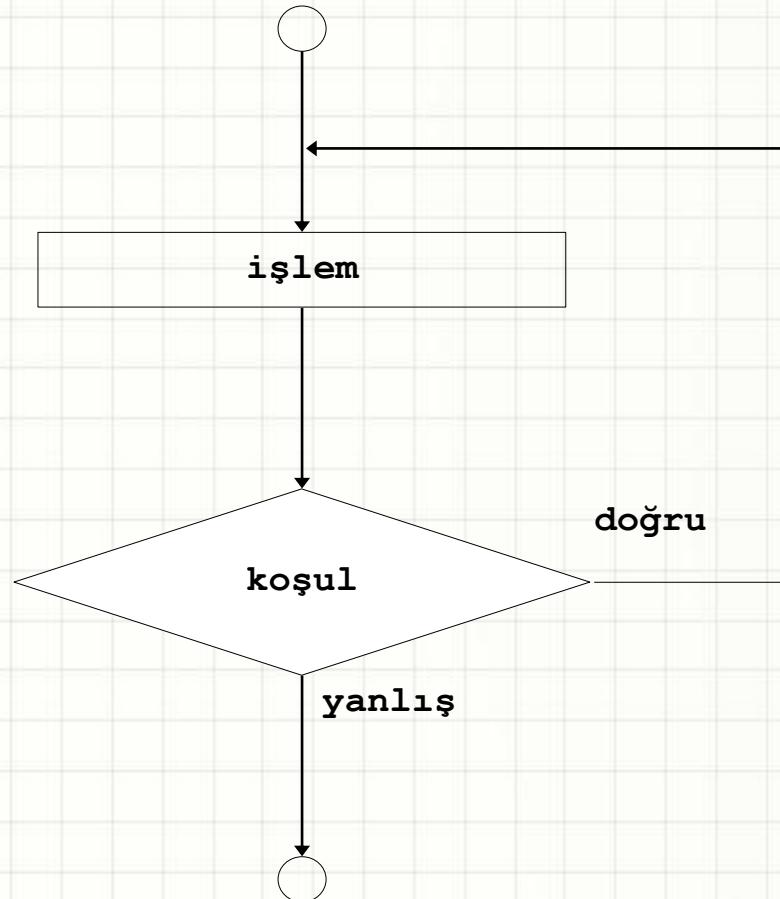
```
    getch();
```

```
}
```

do/while Döngü Yapısı

- do...while döngüsü
 - while yapısına oldukça benzer
 - Döngü devam koşulu döngü gövdesi çalıştırıldıktan sonra kontrol edilir
 - Döngünün gövdesi en az bir kere çalıştırılır
 - Biçim:

```
do {  
    ifade;  
} while (koşul);
```



do/while Döngü Yapısı

- Örnek (sayıcı değişkeninin değeri 1 olsun):

```
do {  
    printf( "%d ", sayıcı );  
} while (++sayıcı <= 10);
```

- 1 den 10 a kadar olan tamsayıları sayıları ekrana basar

do/while örnek

Örnek:klavyeden sayılar girilmektedir. 0 girilene kadar girilen sayıların ortalaması? (0 hariç)

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i=0,t=0,a;
    float ort;
    do{
        printf("sayıyı gir. bitirmek için 0 gir.");
        scanf("%d",&a);
        t=t+a; i++;
    }while(a!=0);
    ort=(float)t/(float)(i-1);
    printf("ortalama=%0.3f\n",ort);
    getch();
}
```

do/while örnek

Örnek: klavyeden sayılar girilmektedir. 0 girilene kadar girilen sayıların ortalaması? (0 hariç) break li çözüm

```
#include <stdio.h>
#include <conio.h>
main()
{
    int i=0,t=0,a;
    float ort;
    do{
        printf("sayıyı gir. bitirmek için 0 gir.");
        scanf("%d",&a);
        if (a==0) break;
        t=t+a; i++;
    }while(5==5);
    ort=(float)t/(float)i;
    printf("ortalama=%0.3f\n",ort);
    getch();
}
```

Ödevler:

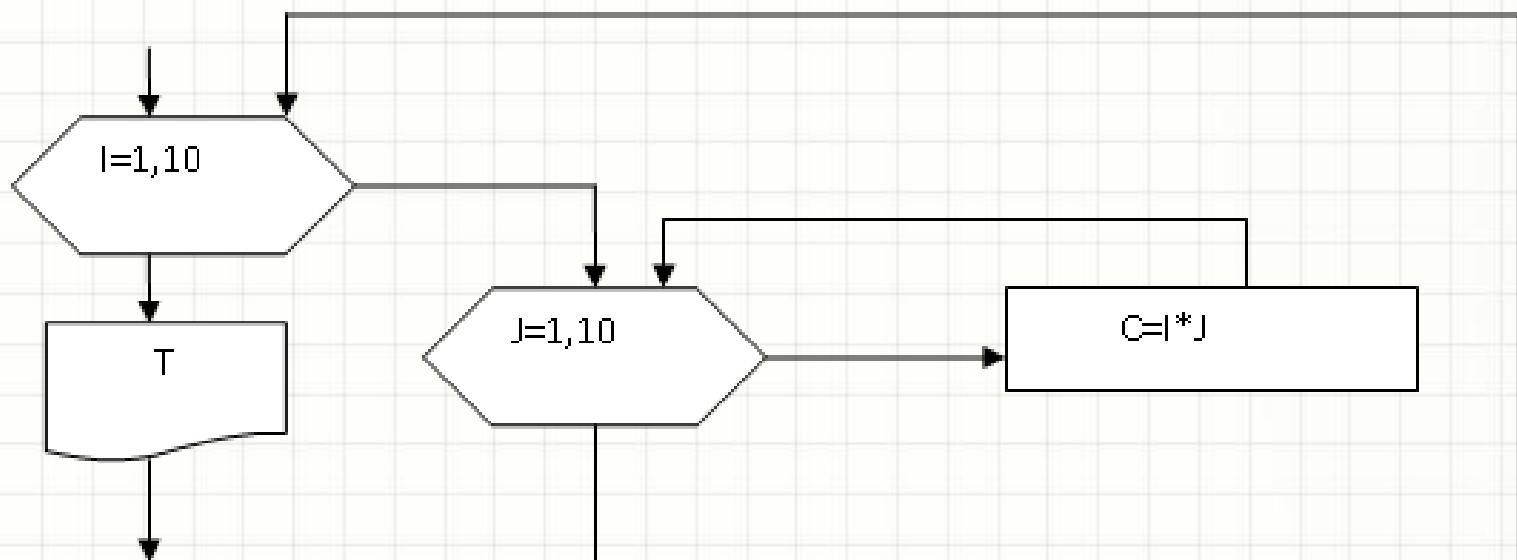
ödev 19: klavyeden sayılar girilmektedir. 0 girilene kadar teklerin ve çiftlerin ayrı ayrı ortalamasını bulup yazdırın (0 hariç) programı yazınız.

ödev 20: klavyeden sayılar girilmektedir. 0 girilene kadar teklerin toplamını ve çiftlerin çarpımını bulup yazdırın (0 hariç) programı yazınız.

ödev 21: klavyeden bir N sayısı gir. N sayısı asal mı değil mi? Ekrana yazdırın.

İçinde Döngüler

- Bir döngü içinde bir ya da daha fazla döngünün bulunması.



İçinde Döngüler örnek

Örnek: 1 den 10 a kadar sayıların çarpım tablosu?

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

İçinde Döngüler örnek çıktılar

Örnek 1: aşağıdaki ekran çıktısını veren program?

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

1 2 3 4 5

Örnek 2: aşağıdaki ekran çıktısını veren program?

1 1 1 1 1

2 2 2 2 2

3 3 3 3 3

4 4 4 4 4

5 5 5 5 5

6 6 6 6 6

Örnek 3: aşağıdaki ekran çıktısını veren program?

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

1 2 3 4 5 6

Örnek 4:

1 2 3 4 5 6

1 2 3 4 5

1 2 3 4

1 2 3

1 2

1

İç içe Döngüler ödevler

ödev22: aşağıdaki ekran çıktısını veren program?

6
6 5
6 5 4
6 5 4 3
6 5 4 3 2
6 5 4 3 2 1

ödev23: aşağıdaki ekran çıktısını veren program?

1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
6 5 4 3 2 1

ödev24: aşağıdaki ekran çıktısını veren program?

6 5 4 3 2 1
6 5 4 3 2 1
6 5 4 3 2 1
6 5 4 3 2 1
6 5 4 3 2 1

ödev25: aşağıdaki ekran çıktısını veren program?

6
5 5
4 4 4
3 3 3 3
2 2 2 2 2
1 1 1 1 1 1

İçinde Döngüler

örnek: $t=1!+2!+3!+\dots+n!$

Ödev 26: $c=1*(1+2)*(1+2+3)*\dots*(1+2+3+\dots+n)$

Ödev 27: $t=1+2*2+3*3*3+\dots+n*n*n*\dots*n$



PROGRAMLAMA TEMELLERİ

3. C PROGRAMLAMA DİLİ

DİZİLER

Öğr. Gör. Fevzi ÖZEK

2019-20

Diziler

- Giriş
- Diziler
- Dizilerin Bildirimleri
- Dizilerin Kullanımlarına Örnekler
- Çok Boyutlu Diziler
- Dizileri Sıralamak
- Dizilerde Arama Yapmak

Amaçlar

- Bu Bölümde Öğreneceklerimiz:
 - Dizi veri yapısını tanıtmak
 - Dizilerin değerlerini depolama, sıralama ve listeleri arama ile değer tabloları oluşturmada kullanımlarını anlamak
 - Bir dizinin nasıl bildirileceğini, bir diziye nasıl ilk değer atanacağını ve dizideki bağımsız elemanların nasıl çağrılabileceklerini anlamak
 - Dizileri fonksiyonlara geçirebilmek.
 - Temel sıralama tekniklerini anlamak
 - Çok boyutlu dizileri bildirebilmek ve kullanabilmek

Giriş

- Diziler
 - Birbirleriyle ilişkili veri yapıları
 - Statik yapılar- programın çalışma süresi boyunca hep aynı boyutta kalırlar.
- Dizi bir kümedir. Aynı tipte verilere tek bir isimle erişmek için kullanılır.
- Bir dizinin bütün elemanları bellekte peş peşe saklanır.
- Diziler bir veya çok boyutlu olabilirler.
 - Tek boyutlu diziler vektör,
 - iki boyutlu diziler matris olarak anılır.

Diziler

- Dizi içindeki elemanı veya konumunu belirtme
 - Dizi Adı
 - Pozisyon Numarası
- Biçim:
Diziadı[pozisyon numarası]
 - Dizinin ilk elemanı
her zaman sıfır'inci elemandır
 - C dizisinin n. elemanı:
 $c[0], c[1] \dots c[n - 1]$

1. Eleman	Dizi[0]
2. Eleman	Dizi[1]
3. Eleman	Dizi[2]
4. Eleman	Dizi[3]
5. Eleman	Dizi[4]
6. Eleman	Dizi[5]

Diziler

- Dizi elemanları normal değişkenler gibidir.

```
c[ 0 ] = 3;  
printf( "%d", c[ 0 ] );
```

- Eğer $x = 3$ e eşitse, belirteçteki bu işlemi yapar

```
c[ 5 - 2 ] == c[ 3 ] == c[ x ]
```

Dizilerin Bildirimleri

- Dizileri tanımlayacağımız zaman,
 - Adı
 - Dizi Tipi
 - Eleman Sayısı
 - Dizitipi Diziadı[Elemansayısı];
 - Örnekler:

```
int c[ 10 ];  
float myDizi[ 3284 ];
```
- Aynı anda birden fazla dizi bildirimi,
 - Aynı değişken tipleri için, tanımlama biçimleri aynı
 - Example:

```
int b[ 100 ], x[ 27 ];
```

Dizileri Kullanan Örnekler

- Atama değerleri

```
int n[ 5 ] = { 1, 2, 3, 4, 5 };
```

- Eğer dizideki elemanların sayılarından daha az atama değeri varsa, kalan elemanlar 0 değerine atanır

```
int n[ 5 ] = { 0 }
```

- Bütün elemanlar 0

- Dizideki eleman sayılarından fazla atama değeri varsa yazım hatası oluşacaktır.

- Dizinin boyutu belirtilmezse, dizinin boyutu atama listesindeki eleman sayısı olur

```
int n[ ] = { 1, 2, 3, 4, 5 };
```

- 5 atama değeri, n Dizisinin boyutunu 5 olarak belirler

Dizileri Kullanan Örnekler

Karakter Dizileri

- String “birinci”, gerçekten karakterlerden oluşmuş statik bir dizidir.
- Karakter Dizisi bir string kullanarak ilk değerlere atanabilir
 - `char string1[] = "birinci";`
 - Null karakterin, karakter sabiti olarak gösterimi '\0'
 - `string1` gerçekten 7 elemanlıdır
 - Az önceki bildirim

```
char string1[] = { 'b', 'i', 'r', 'i', 'n', 'c', 'i', '\0' };
```

- Bağımsız karakterlere dizi belirteci gösterimiyle erişebiliriz.
`string1[3] is character 'i'`
- Dizi adı dizinin başlangıç adresidir, ‘&’ kullanımı gereksizdir
 - `scanf("%s", string2);`
 - Boşluk karakteri girilinceye kadar karakter okur
 - Dizinin sonundan öteye de yazabilir, dizinin büyüklüğüne dikkat etmez.

Örnekler

Örnek: Dizinin elemanlarına değer atama

```
#include<stdio.h> main()
{
    int a[10];
    int i;
    for (i=0; i<=9; i++)
        a[i] = (i+1)*(i+1);
    for (i=0; i<=9; i++)
        printf("%d . elemanın değeri = %d\n", i, a[i]);
}
```

Örnekler

Klavyeden 10 sayı okuyup. Bunları tersten yazdırma.

```
#include <stdio.h>
main()
{
    int a[10]; int i;
    for (i=0; i<=9; i++) { printf("%d. Sayıyı
gir",i);
    scanf("%d", &a[i]);
}
printf("\n-----\n");
for (i=9; i>=0; i--)
    printf("%d . sırada girilen sayı = %d\n", 9-i, a[i]);
}
```

Örnekler

n elemanlı bir diziye (n en fazla 30 olabilir) sayı giridiren, tek olanları t, çift olanları c dizisine atayan ve bu iki diziyi ekrana yazdır.

```
#include <conio.h>
#include <stdio.h>
main()
{
    int a[30],t[30],c[30],i,j=0,k=0,n;
    printf("dizinin eleman sayısını gir(en fazla 30)=");
    scanf("%d",&n);
    for (i=0;i<=(n-1);i++)
    {
        printf("%dinci elemanı gir:",i);
        scanf("%d",&a[i]);
        if (a[i]%2==0) c[k++]=a[i];
        else t[j++]=a[i];
    }
    for (i=0;i<=(j-1);i++) printf("%d     inci      tek
eleman=%d\n",i,t[i]);
    for (i=0;i<=(k-1);i++) printf("%d     inci      çift
eleman=%d\n",i,c[i]);
    getch();
}
```

Örnekler

fibonacci sayılarını n elemanlı ($n \geq 3$) bir diziye atayan ve diziyi yazdırın program?

```
#include <conio.h>
#include <stdio.h>
main()
{
    int a[100],i,n;
    a[0]=a[1]=1;
    printf("dizinin eleman sayısını gir(en az 3
en fazla 100)=");
    scanf("%d",&n);
    for (i=2;i<=n-1;i++) a[i]=a[i-1]+a[i-2];
    for (i=0;i<=n-1;i++) printf("%d ",a[i]);
    getch();
}
```

Ödevler

ödev26: n elemanlı bir diziye değer girişi yaptıran ve bu dizinin elemanlarını tersten başka bir diziye atayıp bu diziyi yazdır?

ödev27: n elemanlı bir diziye değer girişi yaptıran ve bu dizinin standart hatasını bulup ekrana yazdırın program?

formül tahtada:

ödev28: klavyeden bir sayı gir. bu sayıyı ikilik (binary) sisteme çevir sonucu yazdırın program?

Çok Boyutlu Diziler

Çok boyutlu bilgileri veya veri tablolarını saklamak için kullanılır.

İki boyutlu diziler daha sık kullanılır.

Örneğin; yıllara ve aylara göre enflasyon rakamının takibi, matematikteki matris işlemlerinin gerçekleştirilmesi, öğrenciler ve aldıkları derslerin takibi.

Çok Boyutlu Diziler

Tanımlama Biçimi:

1. Tip DegiskenAdi[indis1][indis2]...[indisn]

float dizi[5][12]

Dizi iki boyutlu olup satır yılları, sutun ise ayları gösterir

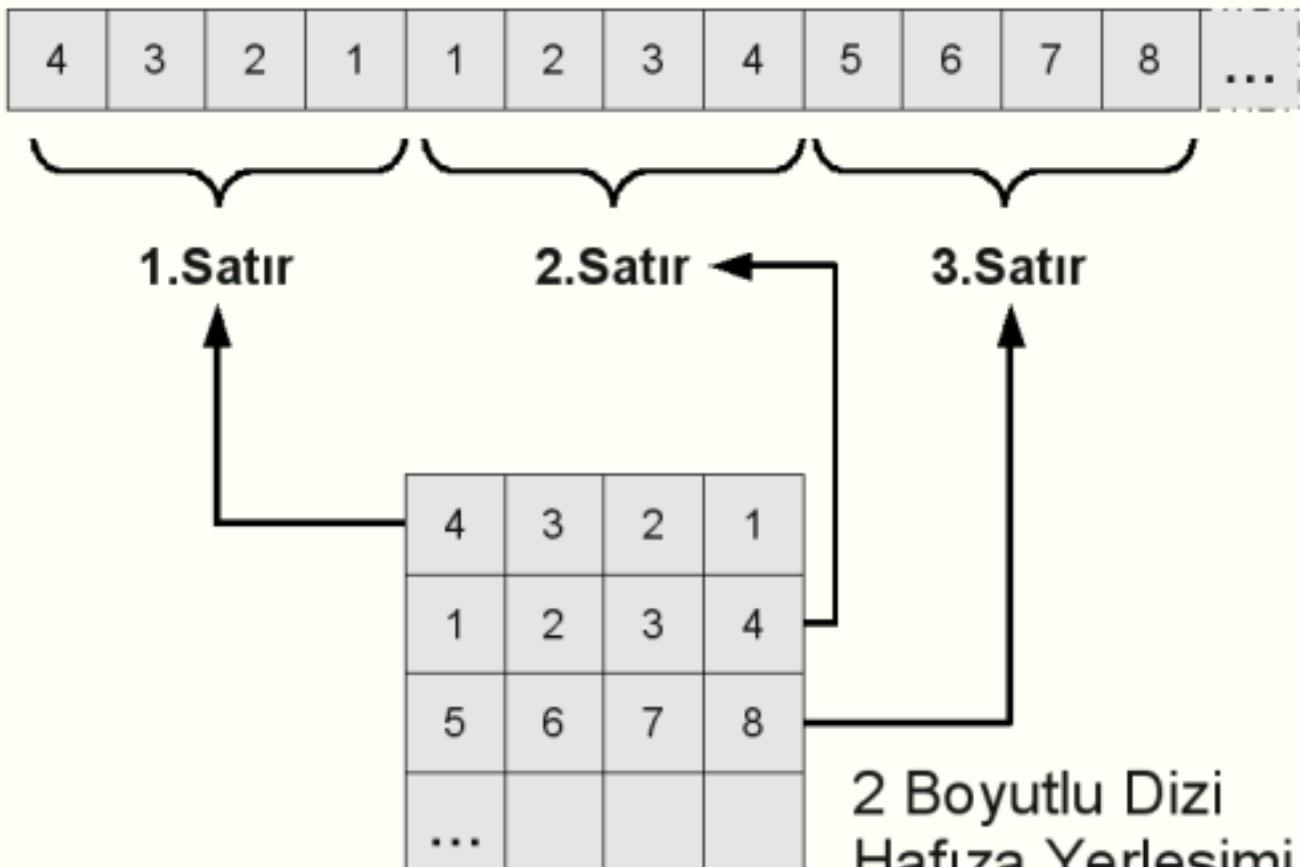
	1/ay	2/ay	3/ay	4/ay	5/ay	6/ay	7/ay	8/ay	9/ay	10/ay	11/ay	12/ay
1												
2												
3												
4												
5												

8 Sınav

5 Öğrenci

80	76	58	90	27	60	85	95
60	59	75	80	82	79	64	87
77	...						
				...	67	60	84

5.Öğrencinin 6.sınavı



Çok Boyutlu Diziler

- Çoklu belirteçli Diziler
 - Satırlar ve sütunlarla gösterilir ($m \times n$ Dizisi)
 - Matris gibi: önce satır belirtilir, sonra sütun

	sütun 0	sütun 1	sütun 2	sütun 3
Satır 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Satır 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Satır 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Dizi adı *Satır belirteci* *Sütün belirteci*

Çok Boyutlu Diziler

- İlk değer atama
 - `int b[2][2] = { { 1, 2 }, { 3, 4 } };`
 - Parantezler içinde satırlara göre guruplandırılarak değer atama
 - Yeterli değer yoksa, belirtilmemiş elemanlar 0 olarak atanır
`int b[2][2] = { { 1 }, { 3, 4 } };`
- Elemanların bildirimi
 - Önce satırı belirle, sonra sütunu
`printf("%d", b[0][1]);`

Çok Boyutlu Diziler örnek

1-den 10 a kadar çarpım tablosunu ekrana yazdırın.

```
#include <conio.h>
#include <stdio.h>
main()
{
    int a[10][10],i,j;
    for (i=0;i<=9;i++)
        for (j=0;j<=9;j++) a[i][j]=(i+1)*(j+1);
    for (i=0;i<=9;i++)
    {
        for (j=0;j<=9;j++) printf("%4d   ",a[i][j]);
        printf("\n\n");
    }
    getch();
}
```

Dizileri Fonksiyonlara Geçirmek

- Daha sonra anlatılacak
- Fonksiyon Prototipi

```
void modifyDizi( int b[], int Diziboyutu );
```

— Prototipte parametre isimleri görecelidir

- int b[], int [] şeklinde de yazılabilir
- int Diziboyutu basitçe int olarak yazılabilir

Dizileri Fonksiyonlara Geçirmek

- Dizileri Geçirmek...
 - Bir dizi argümanını fonksiyona geçirerek için, dizinin ismini parantez kullanmadan belirtmeliyiz...

```
int myDizi[ 24 ];  
myFunction( myDizi , 24 );
```

 - Genellikle Dizileri fonksiyonlara geçirirken boyutları da geçirilir
 - Diziler referansa göre çağrıma yöntemiyle geçirilir
 - Dizinin adı ilk dizinin ilk elemanın adresidir
 - Fonksiyon dizinin nerede tutulduğunu bilir
 - Dizinin orijinal hafıza konumunda elemanlarını modifiye eder
- Dizinin Elemanlarını Geçirmek
 - Değere göre çağrılarla geçirme
 - Altsimgi (subscripted name) isimlerini fonksiyona geçirme (i.e., `myDizi[3]`)

Dizileri Sıralama

- Veri sıralama
 - Bilgisayar uygulamalarında önemlidir
 - Hemen hemen tüm organizasyonlar bazı verileri sıralamak zorundadırlar.

Ayrıntılar .pdf dosyasında

Dizileri Sıralama Exchange Sort

45 21 3 16 8

1. tur

a₁>a₂=>e yerdeğiştir 1. geçiş

21 45 3 16 8

a₁>a₃=>e y.d 2. geçiş

3 45 21 16 8

a₁>a₄=>h işlem yok 3. geçiş

a₁>a₅=>h i.y 4. geçiş

3 45 21 16 8

2. tur

a₂>a₃=>e y.d

3 21 45 16 8

a₂>a₄=>e y.d

3 16 45 21 8

a₂>a₅=>e y.d

3 8 45 21 16

3. tur

a₃>a₄=>e y.d

3 8 21 45 16

a₃>a₅=>e y.d

3 8 16 45 21

4. tur

a₄>a₅=>e. y.d

3 8 16 21 45

Dizileri Sıralama Bubble Sort

45 21 3 16 8

1.tur

a1>a2=>e y.d.

21 45 3 16 8

a2>a3=>e y.d

21 3 45 16 8

a3>a4=>e y.d

21 3 16 45 8

a4>a5=>e y.d

21 3 16 8 45

2. tur

a1>a2=>e y.d

3 21 16 8 45

a2>a3=>e y.d

3 16 21 8 45

a3>a4=>e y.d

3 16 8 21 45

3. tur

a1>a2=>h i.y

a2>a3=>e y.d

3 8 16 21 45

4. tur

a1>a2=>h i.y

3 8 16 21 45

Dizileri Sıralama Selection Sort

45 21 3 16 8

1. tur

k=1

ak>a₂=>e k=2

ak>a₃=>e k=3

ak>a₄=>h işlem yok 3. geçiş

ak>a₅=>h i.y 4. geçiş

a₁-ak yd.

3 21 45 16 8

2. tur

k=2

ak>a₃=>h i.y

ak>a₄=>e k=4

ak>a₅=>e k=5

a₂-ak y.d

3 8 45 16 21

3. tur

k=3

ak>a₄=>e k=4

ak>a₅=>h. i.,y

a₃-ak

3 8 16 45 21

4. tur

k=4

ak>a₅=>e k=5

a₄-ak yd.

3 8 16 21 45

Dizileri Sıralama örnek

10 elemanlı bir diziye değer girdiren ve dizinin elemanlarını küçükten büyüğe doğru sıralayıp ekrana yazdırın programı yazınız. (ex sort)

```
#include<conio.h>
#include<stdio.h>
#include<locale.h>
main()
{
    setlocale(0,"Turkish");
    int a[10];
    int i,j,yd;
    for (i=0; i<=9; i++)
    {
        printf("%d . elemanın değerini girin:",i+1);scanf("%d",&a[i]);
    }
```

Dizileri Sıralama örnek

```
for (i=0; i<=8; i++)
    for (j=i+1;j<=9;j++)
        if(a[i]>a[j])
            {yd=a[i];a[i]=a[j];a[j]=yd;}
    for (i=0; i<=9; i++)
        printf("%d . elemanın değeri = %d\n", i+1, a[i]);
    getch();
}
```

Dizilerde Arama: Dizide arama yapmak

- Bir *key değeri* için dizide arama
- Lineer arama
 - Basit
 - Key değeri ile dizinin her değerini karşılaştırır
 - Küçük ve sıralanmamış diziler için kullanışlı

ÖRNEK: 10 elemanlı bir dizide klavyeden girilen bilgiyi arayan program:

```
for (i=0;i<=9;i++)
```

```
    if (ara==a[i]) { k=i; break;}
```

```
if(k==-1) printf("Aranan eleman bulunamadı.");
```

```
else printf("Aranan eleman %d. sırada  
bulundu.",k);
```

Dizilerde Arama: Dizide arama yapmak

- Binary search
 - Sıralanmış Diziler içindir
 - key ile orta elementi karşılaştırır
 - Eğer eşitse, arama değeri bulunmuştur
 - Eğer key < orta element, dizinin ilk yarısında arar
 - Eğer key > orta element , Son yarısında arama yapar
 - İşlem tekrarlanır
 - Çok hızlı; en fazla n adımda, $2^n >$ eleman sayısı
 - 30 elemanlı Dizide en fazla 5 adım sürer
 - $2^5 > 30$ öyleyse, en fazla 5 adıms

ÖRNEK: 7 elemanlı bir dizide 6'ı arayan program:

```
int dizi[7] = {1, 3, 4, 7, 10, 12, 15};
```

```
int aranan = 12;
```

```
int bas = 0;
```

```
int son = 6;
```

```
int i;
```

```
while (bas != son)
```

```
{ i = (bas + son)/2;
```

```
    if (dizi[i] == aranan) break;
```

```
    else if (dizi[i] > aranan)
```

```
        son = i-1;
```

```
    else bas = i+1;
```

```
}
```



PROGRAMLAMA TEMELLERİ

3. C PROGRAMLAMA DİLİ

FONKSIYONLAR

Öğr. Gör. Fevzi ÖZEK

2019-20



Fonksiyon Tanımı

Değer Döndürmeyen Fonksiyonlar

Değer Döndüren Fonksiyonlar

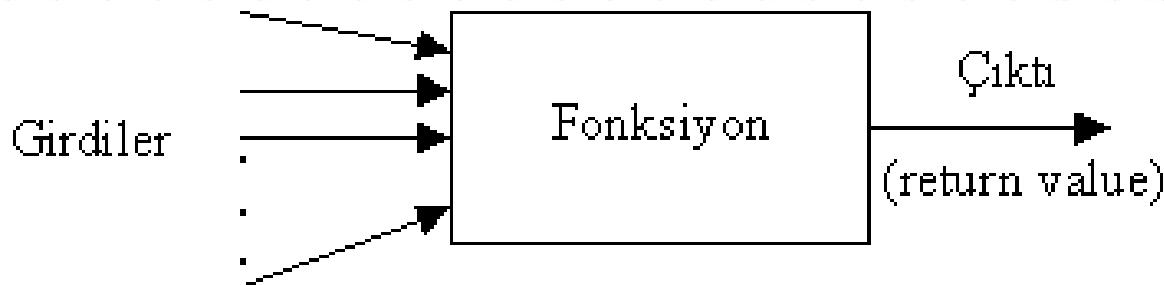
Çok Parametreli Fonksiyonlar

Değişken Kapsamları

Çok Fonksiyonlu Programlar

Fonksiyon Kavramı

Fonksiyon, belirli sayıda verileri kullanarak bunları işleyen ve bir sonuç üreten komut grubudur. Her fonksiyonun bir adı ve fonksiyona gelen değerleri gösteren parametreleri (bağımsız değişkenleri) vardır. Genel olarak bir fonksiyon aşağıdaki gibi bir kutu ile temsil edilir:



Fonksiyon Kavramı

Fonksiyonların girdilerine *parametre* denir.

Bir fonksiyon bu parametreleri alıp bir işleme tabi tutar ve bir değer hesaplar.

Bu değer, *çıktı* veya *geri dönüş değeri (return value)* olarak adlandırılır.

Unutmayın ki, bir fonksiyonun kaç girişi olursa olsun sadece bir çıkışı vardır.

Fonksiyon Kavramı

C Programlama Dili, kullanıcısına bu türden fonksiyon yazmasına izin verir. C dilinde hazırlanan bir fonksiyonun genel yapısı şöyledir:

```
FonksiyonTipi FonksiyonAdı(parametre listesi)
parametrelerin tip bildirimleri
{
    Yerel değişkenlerin bildirimi ...
    fonksiyon içindeki deyimler veya diğer fonksiyonlar
    ...
    return geri dönüş değeri;
}
```

Fonksiyon Kavramı

Örneğin iki sayının toplamının hesaplayacak bir fonksiyon şöyle tanımlanabilir:

```
/* klasik biçim */  
int topla(x, y)  
int x,y  
{  
    int sonuc;  
    sonuc = x + y;  
    return sonuc;  
}
```

```
/* modern biçim */  
int topla(int x, int y) /* modern biçim */  
int topla(int x, int y)  
{  
    int sonuc;  
    sonuc = x + y;  
    return sonuc;  
}
```

Fonksiyon Kavramı

Bu örnekte, fonksiyonun *kimlik kartı!* ve kutu gösterimi şöyledir:

- Fonksiyon tipi: int
- Fonksiyon adı : topla
- parametreler : x ve y
- geri dönüş değeri: $x+y$



Fonksiyon Kavramı

Her üç program parçasında da **return** (geri dönüş) deyimi kullanılmaktadır.

Bu deyim C programlama dilinin anahtar sözcüklerinden biridir ve fonksiyon içerisinde sonucu, kendisini çağrıran yere göndermek için kullanılır.

Yani topla fonksiyonu herhangi bir programın içerisinde kullanıldığında, fonksiyonun üreteceği sonuç **return** deyiminden sonra belirtilen değişken veya işlem olacaktır.

Fonksiyon bildiriminde, fonksiyona girdi olarak kullanılan değişkenlere **parametre** denir.

```
// fonksiyon bildiriliyor...
int topla(int x, int y) // buradaki x ve y parametre
{
    return (x+y);
}
```

```
// fonksiyon çağrılmıyor...
t = topla(9, 6);           // burada 9 ve 6 parametre
```

Bir fonksiyon bildirimi iki türlü olabilir:

1. Ana programdan önce:

...

```
int topla(int x, int y) /* fonksiyon */  
{ ... }
```

...

```
main()  
{ ... }
```

2. Ana programdan sonra: Bu durumda fonksiyon örneği ana programdan önce bildirilmek zorundadır.

...

```
int topla(int x, int y); /*  
fonksiyon */
```

...

```
main()  
{ ... }
```

...

```
int topla(int x, int y)  
{ ... }
```

Geri Dönüş Değerleri

return anahtar sözcüğünün iki önemli işlevi vardır:

1. fonksiyonun geri dönüş değerini oluşturur.
2. fonksiyonu sonlandırır.

return (a + b / c);

return 10;

return topla(a, b) / 2;

void Fonksiyonlar

- Bir fonksiyonun her zaman geri dönüş değeri olması gerekmekz.
- Bu durumda ***return*** deyimi kullanılmayabilir.
- Eğer bu anahtar kelime yoksa, fonksiyon ana bloğu bitince kendiliğinden sonlanır.
- Böyle fonksiyonların tipi ***void (hükümsüz)*** olarak belirtilmelidir.
- Bu tip fonksiyonlar başka bir yerde kullanılırken, herhangi bir değişkene atama yapılması söz konusu değildir.

Yapısal Programlama

Program içinde birden fazla fonksiyon tanımlayıp kullanmak mümkündür.

Ana program

```
main()
{
```

```
    ...
    fonk1() i çağır ←
```

```
    ...
    fonk2() yi çağır ←
```

```
    ...
    fonk3() ü çağır ←
```

```
    ...
    .
    .
}
```

```
fonk1()
{
    ...
}
```

```
fonk2()
{
    ...
}
```

```
fonk3()
{
    ...
}
```

ana programdan fonksiyona
değer aktarımı yok.

```
#include <stdio.h>
#include <conio.h>
void topla();
main()
{
    topla();
    getch();
}
void topla()
{
    int x,y,t;
    printf("birinci sayıyı gir:"); scanf("%d",&x);
    printf("ikinci sayıyı gir:"); scanf("%d",&y);
    t=x+y;
    printf("sonuç=%d\n",t);
}
```

ana programdan fonksiyona global değişkenlerle değer aktarımı

```
#include <stdio.h>
#include <conio.h>
void topla(); // void topla(void);
int x,y,t;
main()
{
    printf("birinci sayıyı gir:"); scanf("%d",&x);
    printf("ikinci sayıyı gir:"); scanf("%d",&y);
    topla();
    printf("sonuç=%d\n",t);
    getch();
}

void topla() // void topla(void)
{
    t=x+y;
}
```

ana programdan fonksiyona parametrelerle değer aktarımı

```
#include <stdio.h>
#include <conio.h>
void topla(int x,int y);
int t;
main()
{
    int a,b;
    printf("birinci sayıyı gir:"); scanf("%d",&a);
    printf("ikinci sayıyı gir:"); scanf("%d",&b);
    topla(a,b);
    printf("sonuç=%d\n",t);
    getch();
}
void topla(int x,int y)
{
    t=x+y;
}
```

ana programdan fonksiyona parametrelerle
değer aktarımı ve return ile geri dönüş

```
#include <stdio.h>
#include <conio.h>
int topla(int,int);

main()
{
    int a,b;
    printf("birinci sayıyı gir:"); scanf("%d",&a);
    printf("ikinci sayıyı gir:"); scanf("%d",&b);
    printf("sonuç=%d\n",topla(a,b));
    getch();
}

int topla(int x,int y)
{
    return(x+y);
}
```

Örnek: geometrik şekillerin seçimini ve alan hesabını fonksiyonlarla yazınız.

Sec : seçim yaptıracak. (1-5)

Kare :

Dikdortgen

Ucgen

Daire

Yamuk

```
#include <stdio.h>
#include <conio.h>
int secim();
int kare(int);
int dikdortgen(int,int);
float ucgen(int,int);
float daire(int);
float yamuk(int,int,int);
main()
{
    int sec;
    sec=secim();
```

```
switch(sec)
{
    case 1:
        int x;
        printf("Kenar değerini gir:"); scanf("%d",&x);
        printf("alan=%d\n",kare(x));
        break;
    case 2:
    case 3:
    case 4:
    case 5:
    default:
        printf("geçersiz seçim\n");
}
getch();
```

```
int secim()
{
    int s;
    printf ("kare için 1\n");printf ("dikdörtgen için 2\n");
    printf ("üçgen için 3\n");printf ("daire için 4\n");printf
    ("yamuk için 5\n");
    printf("seçiminiz="); scanf("%d",&s);
    return(s);
}

int kare(int a)
{ return(a*a); }

float ucgen(int a,int h)

{ }

float daire(int r)

{ }

float yamuk(int a,int t,int h)

{ }
```

Örnek: klavyeden iki sayı girilmektedir. Bu iki sayıdan büyüğü bulan büyük, küçüğü bulan küçük isimli iki fonksiyonla ana programı yazınız.

Buyuk
kucuk

```
#include <stdio.h>
#include <conio.h>
int buyuk(int,int);
int kucuk(int,int);
main()
{
    int a,b;
    printf("birinci sayıyı gir:"); scanf("%d",&a);
    printf("ikinci sayıyı gir:"); scanf("%d",&b);
    printf("en büyük=%d\n",buyuk(a,b));
    printf("en küçük=%d\n",kucuk(a,b));
    getch();
}
```

```
int buyuk(int x,int y)
{
    return(x>y?x:y);
}

int kucuk(int x,int y)
{
    if (x<y) return(x);
    else return(y);
}
```

- ödev 29. klavyeden girilen iki sayı ve bir seçimi kullanarak toplama, çıkarma, çarpma ve bölme işlemlerinden birinin sonucunu yazdırınan programı fonksiyonla gerçekleştiriniz.
- ödev 30. klavyeden üç sayı girilmektedir. En büyük, en küçük ve ortancayı bulup yazdırınan programı fonksiyonlarla gerçekleştiriniz.

örnek: klavyeden bir sayı gir. 1 den bu sayıya kadar sayıların toplamını ekrana yazdırın programı fonksiyon kullanarak yazınız?

```
#include <stdio.h>
#include <conio.h>
int topla(int);
main()
{
    int n;
    printf("sayıyı gir:"); scanf("%d",&n);
    printf("toplam=%d\n",topla(n));
    getch();
}
```

```
int topla(int x)
{
    int t=0,i;
    for (i=1;i<=x;i++) t=t+i;
    return(t);
}
```

- ödev 31: klavyeden bir sayı gir. Girilen sayının faktoriyel?
- ödev 32: klavyeden iki sayı gir. kombinasyonu?

Dizileri Fonksiyonlara Geçirmek

- Dizileri Geçirmek...
 - Bir dizi argümanını fonksiyona geçirebilmek için, dizinin ismini parantez kullanmadan belirtmeliyiz...

```
int myDizi[ 24 ];  
myFunction( myDizi , 24 );
```

 - Genellikle Dizileri fonksiyonlara geçirirken boyutları da geçirilir
 - Diziler referansa göre çağrıma yöntemiyle geçirilir
 - Dizinin adı ilk dizinin ilk elemanın adresidir
 - Fonksiyon dizinin nerede tutulduğunu bilir
 - Dizinin orijinal hafıza konumunda elemanlarını modifiye eder
- Dizinin Elemanlarını Geçirmek
 - Değere göre çağrılarla geçirme
 - Altsimge (subscripted name) isimlerini fonksiyona geçirme (i.e., `myDizi[3]`)

Dizileri Fonksiyonlara Geçirmek

- Fonksiyon Prototipi

 Tip modifyDizi(int b[], int Diziboyutu);

- Prototipte parametre isimleri görecelidir

- int b[], int [] şeklinde de yazılabilir

- int Diziboyutu basitçe int olarak yazılabilir.

örnek: klavyeden eleman sayısı ve dizi elemanları gir. dizinin ortalaması?

```
#include <stdio.h>
#include <conio.h>
float ortalama(int [],int);
main()
{
    int a[30],i,n;
    printf("1 den büyük 30 dan küçük sayı gir:"); scanf("%d",&n);
    for (i=0;i<=n-1;i++)
    {
        printf("değeri gir:"); scanf("%d",&a[i]);
    }
    printf("ortalama=%f\n",ortalama(a,n));
    getch();
}
```

örnek: klavyeden eleman sayısı ve dizi elemanları gir. dizinin ortalaması? (devam)

```
float ortalama(int dizi[],int x)
{
    int i,t=0;
    for (i=0;i<=x-1;i++) t=t+dizi[i];
    return((float)t/x);
}
```

```
#include<stdio.h>
void kareleri(int []);
main()
{
    int a[10];
    int i;
    for (i=0; i<=9; i++) a[i] = i + 1 ;
    printf("Dizinin elemanlarının değerleri\n"); for
    (i=0; i<=9; i++) printf("%d      ",a[i]);
    kareleri(a);
    printf("\n\nKare alma işlemi sonrası dizinin elemanlarının değerleri\n");
    for (i=0; i<=9; i++)    printf("%d      ",a[i]);
}
void kareleri(int a[])
{
    int i;
    for (i=0; i<=9; i++)    a[i] = a[i] * a[i];
}
```

Örnek: 10 elemanlı dizinin elemanlarının karelerini alarak diziyi değiştiren program.

ödev 33: dizinin eleman sayısı ve elemanları klavyeden gir. en büyük elemanı, en büyük elemanın sıra numarasını bulan program.

Gir (dizi,n) enbüyük(dizi,n) enküçük(dizi,n)

ödev 34: standart sapmayı bulan programı fonksiyonlarla yap.

Gir (dizi,n) ssapma(dizi,n)

ödev 35: dizinin eleman sayısı ve elemanları klavyeden gir. Diziye küçükten büyüğe sıralayıp ekrana yazdırın program.

Gir (dizi,n) Yaz (dizi,n) Sırala(dizi,n)

Ödev 36: 10 kişilik bir sınıfın öğrencilerinin sınav sonuçlarını okuyup aşağıdaki işlemleri yapan programı fonksiyonlarla yazınız.

1. not bilgilerin saklanacağı veri yapısını belirle ve tipi tanımla
2. notları girdiren gir(dizi,n)
3. Ortalamayı bulan ortalama(dizi,n)
4. 50'den küçük değerlerin sayısını bulan kbul(dizi,n)
5. 50'den büyük değerlerin sayısını bulan bbul(dizi,n)
6. en yüksek notu bulan ebuyuk(dizi,n)

ÖzYineleme (Rekürsif)

- ÖzYineleme
 - Kendi kendini çağıran fonksiyonlar
 - Temel durumda çözülebilir
 - Bir problemi parçalara ayırır
 - Ne yapacağını bildiği parça
 - Ne yapacağını bilmediği parça
 - Ne yapacağını bilmediği parça orijinal probleme benzer
 - Fonksiyon ne yapacağını bilmediği parçayı çözmek için kendisinin bir kopyasını başlatır.
 - En sonunda temel durum çözülür
 - Tıkalı yerleri yakalar, yukarı doğru çalışır ve bütün problemi çözer

ÖzYineleme (Rekürsif)

- Örnek: faktoriyel
 - $5! = 5 * 4 * 3 * 2 * 1$
 - Dikkat!!!
 - $5! = 5 * 4!$
 - $4! = 4 * 3! \dots$
 - Faktoriyel yineleme ile de çözülebilir
 - Temel durumu çöz ($1! = 0! = 1$) sonra yerine koy
 - $2! = 2 * 1! = 2 * 1 = 2;$
 - $3! = 3 * 2! = 3 * 2 = 6;$



```
int factorial(int n){  
    int result;  
    if (n <= 1)  
        result = 1;  
    else  
        result = n *  
            factorial(n - 1);  
    return result;  
}  
  
int main(void) {  
    ...  
    k = factorial(4);  
    ...  
}
```

Örnek: faktoriyel

```
#include<stdio.h>
int faktoriyel( int );
main( void )
{
    printf( "%d\n", faktoriyel(5) );
}

int faktoriyel( int sayı )
{
    if( sayı<= 1 ) return (1);
    else return (sayı * faktoriyel( sayı-1 ));
}
```

Örnek:

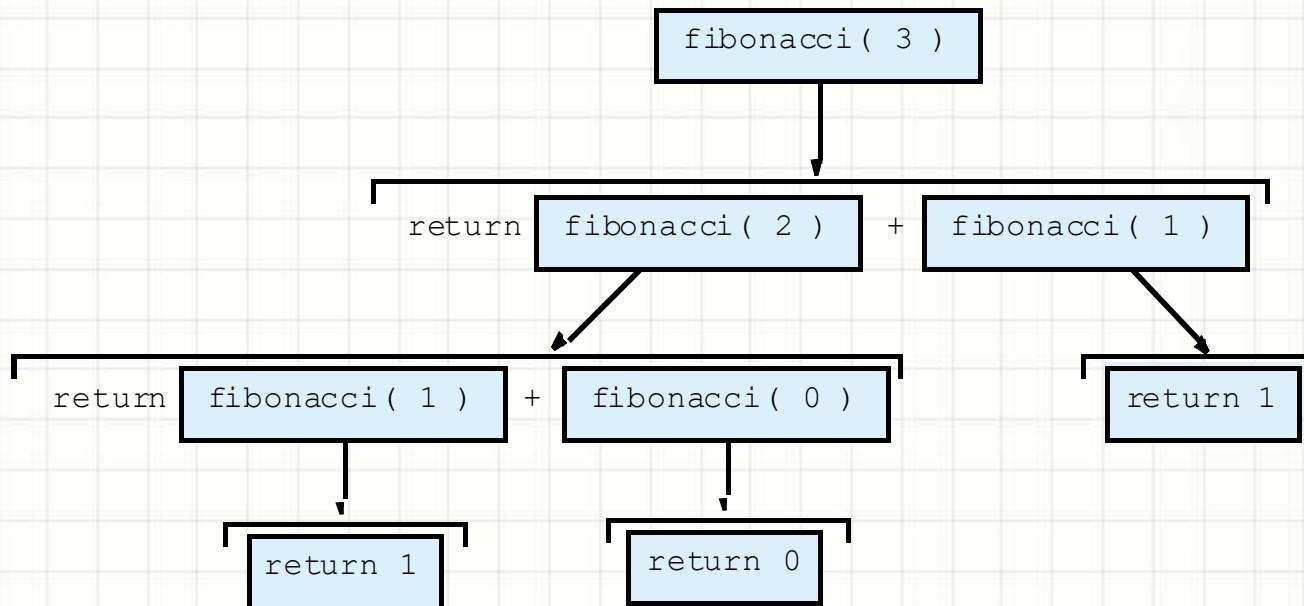
Fibonacci Serisi

- Fibonacci serisi: 0, 1, 1, 2, 3, 5, 8...
 - Her sayı kendinden önceki iki sayının toplamıdır
 - Yinelemeyle çözülebilir:
 - $\text{fib}(n) = \text{fib}(n - 1) + \text{fib}(n - 2)$

```
long fibonacci( long n )
{
    if (n == 0 || n == 1) // base case
        return n;
    else
        return fibonacci( n - 1 ) +
               fibonacci( n - 2 );
}
```

örnek:

Fibonacci Serisi



Örnek:

Fibonacci Serisi

```
#include<stdio.h>
int fibonacci( int );
main()
{
    int i; // Fibonacci serisinin ilk 10 elemani
    for( i = 0; i <= 10; i++ ) printf( "%d    ",fibonacci( i ) );
}

int fibonacci( int x )
{
    if( x<=1 ) return 1;
    else return (fibonacci(x - 1 ) + fibonacci( x - 2 )) ;
}
```

ÖzYineleme vs. Döngü

- Tekrarlama
 - Tekrar : döngü yapısı
 - Yineleme: tekrarlı fonksiyon çağrıır
- Sınır
 - Tekrar: döngü koşul şartı yanlış oluncaya..
 - Yineleme: temel durum gerektirir
- Her ikisi de sonsuz döngüye sahip olabilir
- Denge
 - İyi yazılım mühendisliği(yineleme) ve performans(tekrar) arasında seçim yapmak...

ödev 37: x^y kendisine gelen taban ve kuvveti kullanarak hesaplayan öz yinelemi fonksiyonu yazınız.

ödev 38: Kendisine verilen iki sayının OBEB (Ortak Bölenlerin En Büyübü) değerini hesaplayıp, geriye döndüren fonksiyonu yazınız.

ödev 39: Kendisine verilen iki sayının OKEK (Ortak Katların En Küçüğü) değerini hesaplayıp, geriye döndüren fonksiyonu yazınız.