

BLM 1551 BİLGİSAYAR BİLİMLERİNE GİRİŞ 1, Gr.2
Yrd.Doç.Dr. Yunus Emre SELÇUK
EYLÜL 2015

GENEL BİLGİLER

KAYNAK KİTAP

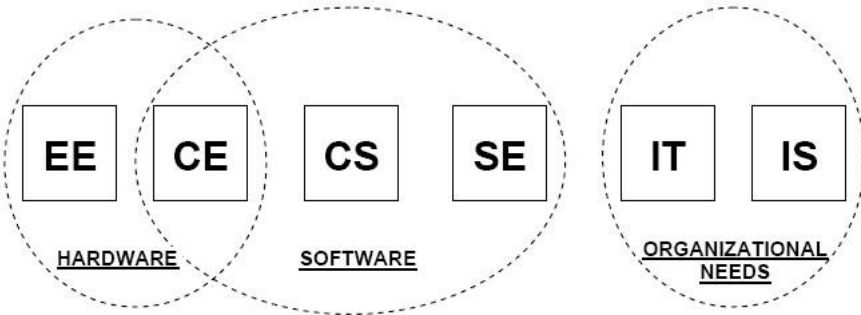
- Önerilebileceğim özel bir kitap bulunmamaktadır.
- Herhangi bir yapısal programlama dilini giriş düzeyinden anlatan bir kitap işinize yarayabilir.
- Bilgisayar ve bilişim kültürünüzü genişletmek üzere çeşitli dergileri takip edin.
- Önceden programlama deneyimi olanlar küçük bir avantaja sahip.
- Bol bol pratik yapın!
 - Fahri Vatansever, "Algoritma Geliştirme ve Programlamaya Giriş". Seçkin Yayıncılık (Akış şemalarında kısmi uyumsuzluklara dikkat).
- ce.yildiz alanındaki sayfamı takip edin.

DEĞERLENDİRME

- 1. Ara Sınav: %25, 6 Kasım 2015 Cuma
- 2. Ara Sınav: %25, 11 Aralık 2015 Cuma
- Lab çalışmaları: %10, en erken 9 Ekim Cuma(!)
- Final sınavı: %40
- Not için değil, öğrenmek için çalışın. Not nasılsa kazanılır.
- Yeni uygulamalar:
 - Vize haftası
 - Ortak sınavlar

1

MESLEK HARİTASI



- IT: Kurum ve şirketlerin günlük bilgi-işlem gereksinimleri ile orta-uzun vadeli gereksinimlerini karşılamak.
- IS: Bilgi teknolojileri ve işletme/yönetim süreçlerini en uygun şekilde birleştirmek.

2

VERİ, BİLGİ VE BİLGİSAYAR

- Bilgisayar:
 - Kendisine verilmiş verileri,
 - Aldığı komutlar dizisine göre, : PROGRAM
 - İşleyen makinedir.
- Veri (Data): Olgu ve kavramlarla ilgili nicelik/sayı
- Bilgi (Information): Düzenli ve anlamlı gerçekler ve ilkeler
- Veri vs. Bilgi:
 - Veri: 19°C hava sıcaklığı
 - Veri: 3,5kg. ağırlık
 - Veri: Ham gerçekler
 - Bilgi: Bugün hava mevsim normallerinde
 - Bilgi: Yenidoğan uygun ağırlıkta
 - Bilgi: Yorumlanmış gerçekler

3

BİLGİSAYARIN TEMEL ÇALIŞMA İLKELERİ

- Program: Bilgisayara bir iş yaptırmak için verilen komutlar dizisi
- Temel işlem birimi: Transistörler
 - Yarı iletken elemanlardır: 3 bacağından birine uygulanan akım, diğer iki bacağından geçen akımı değiştirir.
 - 1950'ler: Vakum tüpleri
 - 1980'ler: Tümleşik devrelerde silikon malzemeleri ile
 - 2008: 214mm² alanda 820 milyon transistör (Intel Core 2 Extreme QX9650)

4

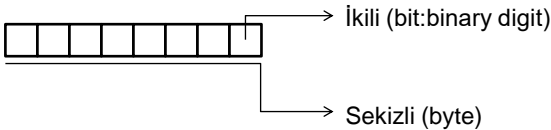
BİLGİSAYARLARIN TARİHÇESİ

- İlk Bilgisayarlar:
 - Bilgisayar teriminin anlamı zamanla değiştiğinden, ilk bilgisayara örnek vermek zordur.
 - Abaküsü de bir hesaplama aygıtı olarak ele alabiliriz!
 - 1250: Diyarbakır'lı El Ceziri'nin gökcisimsel saati
 - İlk programlanabilir analog bilgisayar
 - El Ceziri daha çok ilk robotları ile tanınır.
 - 1837: Charles Babbage, İngiltere. "Analytical Engine"
 - Tasarladı ancak gerçeklemedi: Sınırlı maddi kaynak ve "şurasını da geliştirmek" güdüsü nedeniyle!
 - 1843: Augusta Ada Byron, İngiltere: Bu aygıtla Bernoulli sayılarını hesaplayan programı hazırladı
 - 1941: Zuse Z3 bilgisayarı, Almanya
 - Elektro-mekanik, delikli kartlar ile programlanır, 1 ton ağırlığında
 - 1945: ENIAC, ABD
 - Elektronik, kablolar ve anahtarlarla programlanır, 27 ton ağırlığında

5

GÜNÜMÜZ BİLGİSAYARLARI

- Bilgisayarlarla yapılan işlemlerin temeli: İkili Aritmetik
- Sayıların gruplanması: Bit'ler ve Byte'lar / İkili ve Sekizli
 - İkili: Tek basamak, 0 veya 1 değeri alabileceği için bu şekilde adlandırılmış.
 - Sekizli: Sekiz basamağın yan yana gelmesi ile.
 - DİKKAT: İkili tek basamaktır, sekizli sekiz basamaktır!

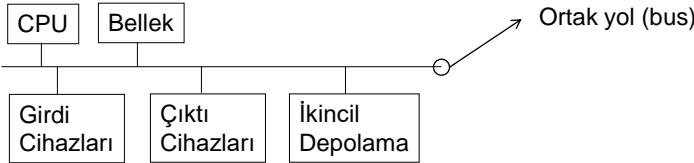


- Temel mantıksal işlemler: ve / veya / değil : AND / OR / NOT

X	Y	X ve Y	X veya Y	değil X
0	0	0	0	1
0	1	0	1	
1	0	0	1	0
1	1	1	1	

6

GÜNÜMÜZ BİLGİSAYARLARI

- Bilgisayarların temel yapısı:

```
graph TD; CPU[CPU] --- Bus[Ortak yol (bus)]; Bellek[Bellek] --- Bus; Bus --- Girdi[Girdi Cihazları]; Bus --- Cikti[Çıktı Cihazları]; Bus --- Depolama[İkincil Depolama];
```
- CPU: Central Processing Unit / Merkezi İşlem Birimi
 - İşlemlerin yürütüldüğü birim.
 - Diğer bilgisayar bileşenlerini denetler ve yönlendirir.
 - Bu işi kendisine verilmiş olan komutlar doğrultusunda yapar.
 - Çeşitli alt birimler içerir (ayrıntıları bu dersin konusu değildir).

7

GÜNÜMÜZ BİLGİSAYARLARI

- CPU'nun alt birimleri:
 - Denetim birimi (Control Unit): Program komutlarını yürütmez, yürütülmesi için tüm sistemi idare eder.
 - Aritmetik lojik birim (ALU): Tamsayı ve mantıksal işlemleri yürüten birim.
 - Yazmaçlar (Register): Veri veya komutlar için sınırlı kapasitede ancak çok hızlı geçici saklama birimleri.
 - Kayar noktalı hesaplama birimi (FPU): Gerçek sayılar ile işlem yapılan birim. Uzun süredir CPU ile tümleşiktir.
 - Cep bellek (Cache): Ana bellek ile işlemci arasındaki geçici saklama alanı.
 - GPU: Son zamanlarda bazı işlemcilerde tümleşik olarak gelmektedir.
- Bellek: Veriler ve komutlar bellekte saklanır.
 - Temel bellek çeşitleri:
 - RAM: Random Access Memory
 - ROM: Read-Only Memory (Normal şartlarda içeriği değiştirilemez)

8

GÜNÜMÜZ BİLGİSAYARLARI

- Girdi cihazları:
 - Bilgisayara veri girişi için kullanılan cihazlardır.
 - Tuş takımı, fare, tarayıcı, vb.
- Çıktı cihazları:
 - Bilgisayarın kullanıcıya veri ve/veya bilgi sunması için kullanılan cihazlardır.
 - Ekran, yazıcı, ses kartı, vb.
- Ortak yol (bus):
 - Şekilde gördüğümüz 5 türden olan çeşitli bileşenlerin birbirleri ile haberleşebilmesini sağlar.
 - Bu yoldan veri, denetim ve adresleme sinyalleri aktarılır.
 - Adresleme: Veri bir birime aktarılacak, tamam, ama o birimdeki hangi alana aktarılacak?

9

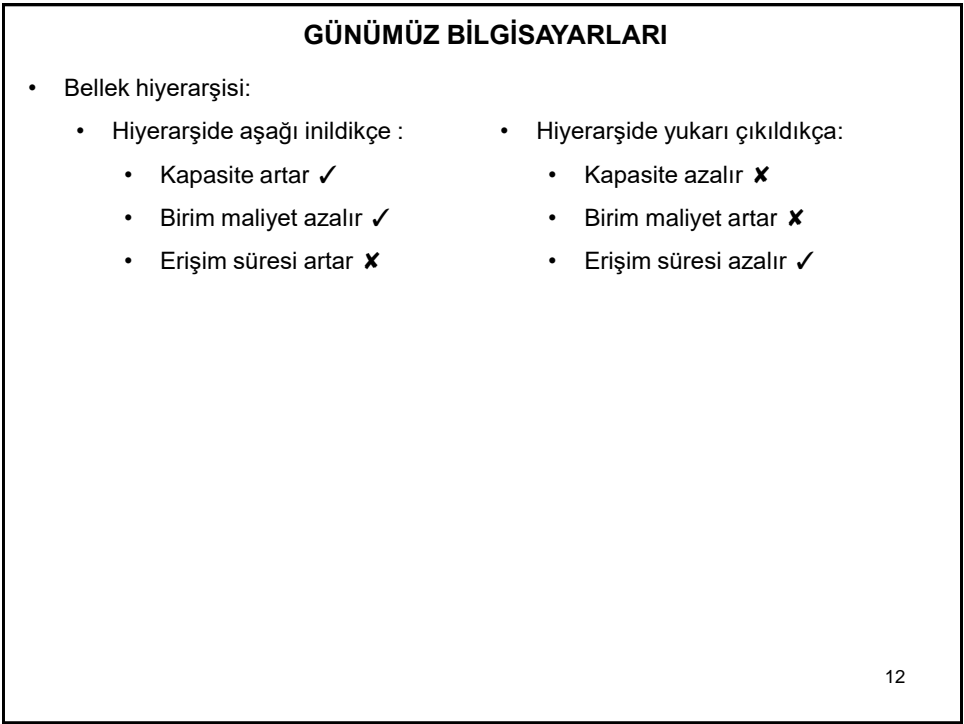
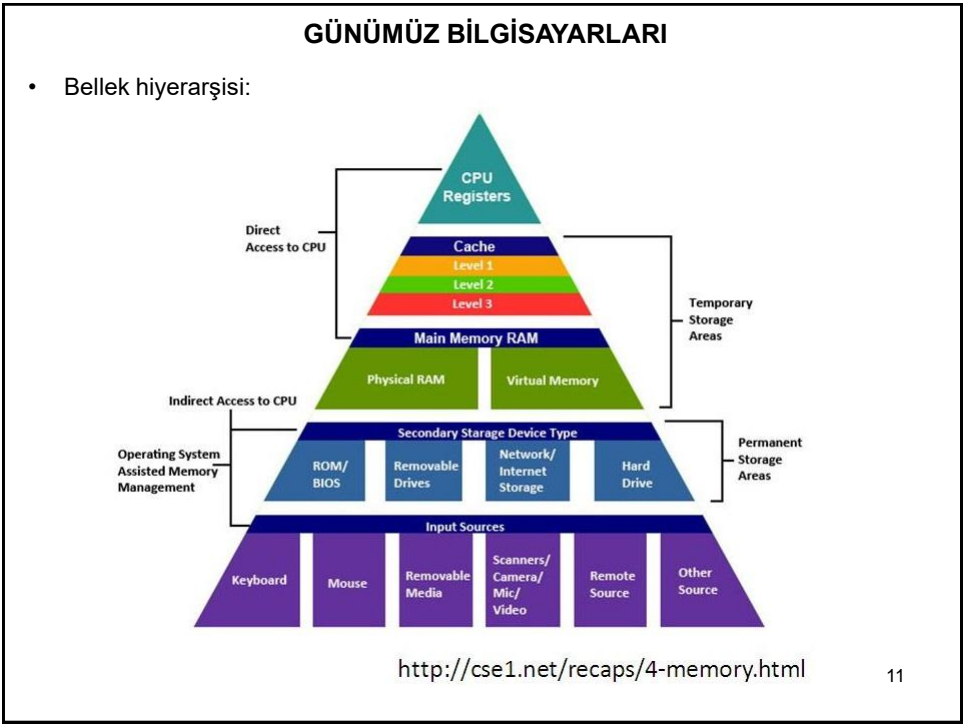
GÜNÜMÜZ BİLGİSAYARLARI

- İkincil saklama ortamları
 - Sabit disk: Manyetik (Yeni SSD diskler RAM bellek tabanlı)
 - CD/DVD: Optik
 - CD/DVD-ROM: Sadece okuma, CD/DVD±RW: Okuma ve yazma.
 - ROM bellek ile CD/DVD'yi karıştırmayın!
- Benzetim:
 - CPU: İşlemlerin yapıldığı yer (Beyin)
 - RAM: Ara sonuçların saklandığı yer (Hafızamız ve kağıt)
 - Sabit Disk: Verilerin ve komutların saklandığı yer (Ders kitabı)

	RAM	ROM	İKİNCİL
Okuma	+	+	+
Yazma	+	-	+
Elektrik Kesintisine Dayanıklılık	-	+	+
Hız ve Maliyet	↑	↑	↓
Kapasite	↓	↓↓	↑

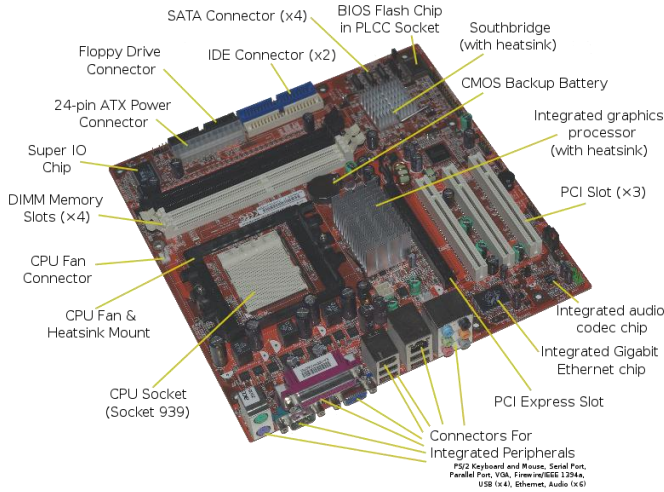
- Kapasite:
 - KB: 1024 Byte (2E10)
 - MB: 1024 KB
 - GB: 1024 MB
 - TB: 1024 GB
 - ...

10



GÜNÜMÜZ BİLGİSAYARLARI

- Kişisel bilgisayar bileşenleri:
 - Anakart temsili şekli üzerinden açıklamalar ve bireysel sorulara yanıtlar.
 - Tavsiye: Bilgisayar dergileri okuyun ve siteleri/forumları takip edin!



13

İŞLETİM SİSTEMLERİ

- İşletim Sistemi: Operating System O/S OS
 - Bilgisayar kaynakların kullanımının yönetimi ve eş güdümünden sorumlu olan program.
 - Programcılar üstteki işlemlerden soyutlayarak işlerini kolaylaştırır.
 - Komut satırı ve Grafik kullanıcı arabirimi (GUI)
 - Windows, MacOS, Unix, Solaris, Linux, Pardus
 - Katmanlar: Kullanıcı – Uygulamalar (Program) – OS – Donanım

14

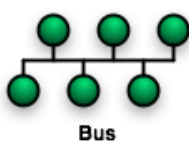
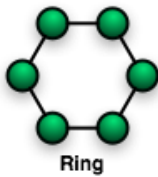
BİLGİSAYAR AĞLARI

- Ağ: İki veya daha fazla aygıtın birbirleri ile bağlı olarak çalışmaları suretiyle oluşan sistem.
- Ağa bağlı aygıtlara 'düğüm' (node) de denir.
- Ağ, kendisine bağlı aygıtlar arasında bir iletişim ortamı sağlar.
- Bilgisayar ağı: Bilgisayarların üzerindeki bilgi ve kaynakların paylaşılabilmesi için oluşturulmuş ağ yapısıdır.
- Internet: Kamusal kullanıma açık bilgisayar ağı
 - Intranet: Kurumsal ağ, Internet teknolojileri kullanılır, ancak Internet'ten yalıtılmıştır (kısmen veya tamamen)
- Bilgisayar ağlarının olası kıldığı uygulamalara örnekler:
 - Bir banka şubesinden yatırılan paranın, diğer şubeden anında çekilebilmesi,
 - Bir çok bilgisayarın ortak bir problemi parçalara bölerek eşzamanlı ve dolayısıyla daha çabuk çözebilmesi,
 - Teknik ekibin zor durumda kalan kullanıcıların bilgisayarlarına kendi bilgisayarlarından erişerek yardım etmeleri,
 - vb.

15

BİLGİSAYAR AĞLARI

- Ağ ilingesi (topolojisi): Düğümlerin birbirine bağlantı şekilleri.
 - Yıldız: Star, Halka: Ring, Anayol: Bus
 - Farklı şekiller de vardır, ancak bu dersin konusu değildir.



- Yıldız: Ortadaki düğüm bilgi akışını yönlendirir.
- Halka: Dönen bir jeton (Japonlar'ın yemek biçimi örneği)
- Anayol: Tüm düğümler hattı dinler

BİLGİSAYAR AĞLARI

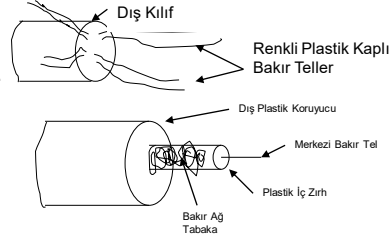
- Kablo sistemleri:

- İkili Kablo: Twisted Pair.

- Telefon kablosunda iki, ağ kablosunda 4 çift tel bulunur.
- Aynı kabloda tek bir haberleşme yapılabilir.

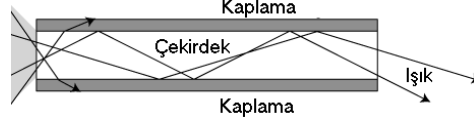
- Tek Eksenli Kablo: Coaxial.

- Daha pahalı ancak daha güvenilir.
- Aynı kabloda 1344 ses kanalı
- Ör: Çanak anten kablosu



- Fiber Optik Kablo: Cam elyafı ve ışık!

- Aynı kabloda 24000 ses kanalı! (Işığın farklı dalga boyları sayesinde)
- Kaplamanın yansıtıcı özelliği vardır. En dışta da plastik kaplama vardır.
- Bükülmeye karşı hassastır.
- Işık elektrondan hızlıdır!



PROGRAMLAMA DİLLERİ

- Girdi – İşlem – Çıktı modeli

- Girdi: Veri, İşlem: Program, Çıktı: Bilgi

- Programlama:

- Bilgisayara ne yapması gerektiğinin anlatılması.
- Temel aşamaları : Sorunu anlama – nasıl çözüleceğini tasarlama – çözümü gerçekleştirme – sinama – hata giderme.
- Bir program için tarifini en derin ayrıntısına kadar içermelidir.
 - Ör: Kantinden su alıp gel yerine : Ayağa kalk – yanındaki arkadaşlarından izin iste – merdivenlerden in – ...

- Programlama dilleri:

- Bilgisayara yapacaklarını anlatmanın belirli bir kurallar dizisine oturtulmuş yolu.
- Programı bir kompozisyona, programlama dilini ise kompozisyonun yazıldığı dile benzetebilirsiniz.
- Dikkat : Pascal dersi değil, Yapısal programlamaya giriş dersi. Araç/amaç.

PROGRAMLAMA DİLLERİNİN DÜZEYLERİ

- Makine dili:
 - Makine dili, işlemcinin konuştuğu dildir.
 - Komutlar tamamen ikili sayılardan oluşur.
 - 0101 0001 0010 0110
 - Size ne ifade etti? Bkz. Assembly dili örneği
 - CPU komutları doğrudan yürütür.
 - Her işlemci, kendine özgü bir komut kümesine sahiptir.
 - Güçlü yönleri:
 - En hızlı çalışan programlar, makine dili ile yazılır.
 - Zayıf yönleri:
 - İkili komut kodlarını ezberleme zorunluluğu.
 - Tüm alt düzey ayrıntıları bilmek zorunluluğundan dolayı aşırı zor, programcıya eldeki soruna odaklanma fırsatı bırakmayabiliyor.

19

PROGRAMLAMA DİLLERİNİN DÜZEYLERİ

- Assembly dili:
 - Komutların ikili düzen karşılıklarını ezberlemek gerekmez.
 - Örnek komut: ADD %r1, %r2, %r6
 - %r = Register = yazmaç = CPU içi bellek; $(6)_{10} = (0110)_2$
 - ADD = 5. komut, $(5)_{10} = (0101)_2$
 - Örnek komut: ADD %r1, %r2, %r6
 - Assembler programı, komutları ikili kodlar haline getirip işletir.
 - Güçlü yönleri:
 - Hızlı çalışma
 - Zayıf yönleri:
 - Tüm alt düzey ayrıntıları bilmek zorunluluğundan dolayı aşırı zor, programcıya eldeki soruna odaklanma fırsatı bırakmayabiliyor.

20

PROGRAMLAMA DİLLERİNİN DÜZEYLERİ

- Üst düzey programlama dilleri:
 - Programcıyı alt düzey ayrıntıları bilmek zorunluluğundan kurtarmak amacıyla geliştirilmiştir.
 - İlk örnekleri: Fortran, Cobol, BASIC
 - Kısaca anlat.
- Programlama yaklaşımları
 - İşlevsel/Functional: Program matematiksel bir hesaplama. Ör: Fortran
 - Sıralı/Structured: GOTO'nun sakıncaları. Ör: Basic, Cobol
 - Yapısal/Procedural: Veri iyi tanımlanmış yapılar halinde düzenlenir, veriler ile bunların üzerlerinde çalışan işlevler birbirinden ayrıdır. Ör: Pascal, C
 - Sıralı <> Procedural, Sıralı < Procedural
 - Nesneye Yönelik: Veri ve üzerinde tanımlanmış eylemler bir bütündür.
- Her yeni yaklaşım, soyutlamayı bir derece yukarı taşımaktadır.
 - Soyut kelimesi sizi korkutmasın, soyutlama daha karmaşık sorunları ele alabilmenize yardımcı olur.
 - Çalıştığınız düzeyin bir altını da bilmeniz yararlıdır.

21

ALGORİTMALARA GİRİŞ

- Algoritma: Bir sorunu çözmek için önerilen adımlar.
 - Bir çok günlük işlem gibi, bilgisayar programlarının da temelinde algoritmalar bulunur.
 - Yemek pişirmek, lastik değiştirmek, kompozisyon yazmak, vb.
- Algoritma adının kaynağı: 9.yy, Fars bilgini El-Harezmi
 - Ondalıkli sayılarda dört işlemin yapılması için gerekli adımları belirtmiştir.
 - İkili sayı sistemini ve sıfır rakamını da bulmuştur.
 - El-harezmi – Algorizm – Algoritma
- İlk algoritma: MÖ 3.yy, Öklid, en büyük ortak böleni bulma algoritması
- Algoritmaların özellikleri:
 - İşlem adımlarının ayrıntı düzeyi: Algoritmayı kullanacak kişinin anlayış düzeyinde olmalıdır.
 - Kesin ve net adımlardan oluşmalıdır.
 - Sonlu sayıda adımdan oluşmalıdır.
 - Verilen geçerli girişlere karşılık olarak istenilen doğru sonuçları üretmelidir.

22

ALGORİTMALARA GİRİŞ

- Algoritmaların özellikleri:
 - İşlem adımlarının ayrıntı düzeyi:
 - Algoritmayı kullanacak kişinin anlayış düzeyinde olmalıdır.
 - Adımlar, adımları yürütecek donanımın anlayabileceği düzeyde olmalıdır.
 - Bilgisayarın anlayabileceği temel işlemler:
 - Matematik ve mantıksal hesaplamalar
 - Karşılaştırmalar
 - Bilgi girişi ve bilgi çıkışı
 - Bir başka adıma geçme
 - Yineleme

23

ALGORİTMALARA GİRİŞ

- Algoritmaların özellikleri (devam):
 - Kesin ve net adımlardan oluşmalıdır.
 - Su ılıyınca diğer kaba aktar yerine su sıcaklığı 36°C'ye çıkınca
 - Ilık kavramı değişik kişilerce farklı yorumlanabilir!
 - Sonlu sayıda adımdan oluşmalıdır.
 - Tamamlanmayan işlem bir işe yaramaz.
 - Verilen geçerli girişlere karşılık olarak istenilen doğru sonuçları üretmelidir.
 - Karekök hesaplama algoritmasına eksi sayı girilemez.
- Değişken:
 - Matematik anlamı: Bir niceliği ifade etmek için kullanılan sembol.
 - Bilgisayar programında: Niceliğin saklanabileceği bir alan.

24

ALGORİTMALARA GİRİŞ

- Değişkenin tipi (type):
 - Farklı tipte veriler, farklı tipte değişkenlerde saklanır.
- Temel veri tipleri:
 - Tamsayı
 - Ondalık sayı
 - Karakter: Tek harf
 - Karakter katarı (String): Birden fazla harf
 - Mantıksal (Boolean): Doğru/yanlış (true/false)
- Kuvvetli tiplmeli (strongly typed) diller:
 - Bu tür dillerde bir değişkenin tipi önceden tanımlanır ve o değişkene sadece o tipten veriler atanabilir.
 - Bir kez tanımlandıktan sonra o değişkenin tipi değiştirilemez.
 - Bölümümüzdeki eğitim planının çoğunda kuvvetli tiplmeli diller yer almaktadır.

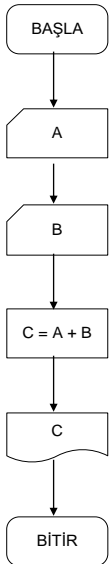
25

ALGORİTMALARA GİRİŞ

- Giriş – İşlem – Çıkış modeli
 - Örnek: İki sayının toplanması
 - Giriş: A ve B değişkenleri (Toplanacak sayılar)
 - İşlem: Toplama
 - Çıkış: A + B
 - Sanal kod (Pseudocode) ile yazılan program:

```
BAŞLA
OKU A
OKU B
C = A + B
YAZ C
BİTİR
```

 - Bu komut ile A ve B değişkenlerinin sakladığı değerler toplanır ve sonuç C değişkenine yazılır.
 - Bu bir atama komutudur, denklem değildir.
 - Sanal kod gerçek bir programlama dilinde değildir, algoritmanın üst düzey bir tarifini verir.
 - Yan tarafta ise programın akış şeması ile temsili görülmektedir.



26

ALGORİTMALARA GİRİŞ

- Akış şemaları:
 - Algoritmanın adımlarının şekillerle gösterilmesi.
 - Programı/Algoritmayı görselleştirmeye yarar.
 - Programların kuralları olduğu gibi, akış şeması şekillerinin de kuralları vardır.
 - Topografik kurallar ileride verilecek, bu aşamada aşağıdaki kutu çeşitlerine ve anlamlarına dikkat ediniz:



- İpuçları
 - Algoritmayı akış şeması ile çizime dökmek, algoritmayı daha iyi anlamanızı sağlar.
 - Temel işlemlere alıştıkça, akış şeması çizme gereksinimi duymayacaksınız. Ama o zamana dek çizin.

27

ALGORİTMALARA GİRİŞ

- Sorun: Bir sekizlinin ondalık karşılığını hesaplayan algoritma
 - Giriş: Sekizlinin basamakları
 - İşlem:
 - Toplam değişkenine sıfır ilk değeri atanır.
 - Kullanıcı en küçük basamaktan başlayarak basamakları girer.
 - Birinci basamağın değeri 1 ile çarpılıp, toplama eklenir.
 - İkinci basamağın değeri 2 ile çarpılıp, toplama eklenir.
 - Üçüncü basamağın değeri 4 ile çarpılıp, toplama eklenir.
 - ...
 - Sekizinci basamağın değeri 128 ile çarpılıp, toplama eklenir.
 - Çıkış: Toplam değişkeninde sayının onluk karşılığı birikmiş olur.

28

ALGORİTMALARA GİRİŞ

- Algoritmayı gerçekleyen sanal kod ve akış şeması:
BAŞLA
TOPLAM = 0
OKU BASAMAK
TOPLAM = TOPLAM + BASAMAK * 1
OKU BASAMAK
TOPLAM = TOPLAM + BASAMAK * 2
OKU BASAMAK
TOPLAM = TOPLAM + BASAMAK * 4
...
OKU BASAMAK
TOPLAM = TOPLAM + BASAMAK * 128
YAZ TOPLAM
BİTİR
- Dikkat: Basamak değişkenine yeni değer atanınca eski değer kaybolur!
 - Ancak bu örnekte eski basamakları saklamaya gerek yok.
- Yapılan tekrar nedeniyle bu çözüm en ideali değildir. İleride tekrarlanan işleri kodlamak için daha uygun yollar öğreneceksiniz.

```
graph TD; A([BAŞLA]) --> B[TOPLAM = 0]; B --> C[/BASAMAK/]; C --> D[TOPLAM = TOPLAM + BASAMAK * 1]; D --> E[/BASAMAK/]; E --> F[TOPLAM = TOPLAM + BASAMAK * 2]; F --> G[...]; G --> H[/BASAMAK/]; H --> I[TOPLAM = TOPLAM + BASAMAK * 128]; I --> J[/TOPLAM/]; J --> K([BİTİR]);
```

29

ALGORİTMALARA GİRİŞ

- İşlem Öncelikleri: Bazı aritmetik işlemlerin diğerlerine göre önceliği bulunur:
 - Matematik derslerinden biliyorsunuz ancak yine de hatırlatalım:
 - Çarpma, bölme ve mod alma işlemleri, toplama ve çıkarma işlemlerinden daha yüksek önceliklidir; yani daha önce yapılırlar.
 - Parantezler ise en yüksek önceliğe sahiptir.
 - $17 \bmod 5 = 2$
 - $X = 2 + 3 * 5$ işleminin sonucu 17 olur, 25 değildir.
 - $X = (2 + 3) * 5$ işleminin sonucu 25 olur, 17 değildir.
 - Tamsayı bölme: div komutu ile.
 - X bir integer ise $x := 19 / 5$; komutu derleme hatası verir.
 - Bu komut $x := 19 \text{ div } 5$; şeklinde yazılmalıdır.

30

ALGORİTMALARA GİRİŞ

- Sorun: İşaretli tamsayıların saklanması
 - Bir sayının pozitif mi, negatif mi olacağını nasıl belirleyebiliriz?
 - İki çözüm: İki'nin tümleyenleri gösterimi ve fazlalık gösterimi
 - İki çözüm de MSB'yi (En yüksek anlamlı ikili) işaret biti olarak kullanmaya dayanır, ancak çözüm sayıyı yazıp işaret bitini 0 veya 1 yapmakla çözülemez.
 - Öyle yapsaydık -0 ve +0 değerleri olurdu ki hem -0 anlamsızdır, hem de onun yerine bir tane daha fazla sayı simgelenebilir.

31

ALGORİTMALARA GİRİŞ

- Sorun: İşaretli tamsayıların saklanması
 - İki'nin tümleyenleri gösterimi (two's complement notation):
 - Üç ikili ile

Üç ikili ile		Dört ikili ile	
011	3	0111	7
010	2	0110	6
001	1	0101	5
000	0	0100	4
111	-1	0011	3
110	-2	0010	2
101	-3	0001	1
100	-4	0000	0
 - Peki, verilen bir işaretli onlu sayıyı bu dönüşüm tablolarına bakmadan nasıl iki tabanına çevireceğiz?
 - Pozitif sayılarda sorun yok, ancak negatif sayılar için bir algoritma gerekmektedir.

32

ALGORİTMALARA GİRİŞ

- Sorun: Onluk düzendeki bir negatif sayıyı ikinin tümleyenleri gösterimine çevirme:
 - Algoritma:
 - Sayıyı pozitif olarak ikili düzende yazın.
 - LSB'den (en düşük anlamlı/en sağdaki ikili) başlayarak 1 kopyalayana kadar basamakları kopyalayın.
 - Bundan sonra basamakları evirerek (DEĞİL işlemi ile) kopyalayın.

0 1 1 0
↓ ↓ ↓ ↓
1 0 1 0

Düz ok : Düz kopyalama
Kesikli ok: Evirerek kopyalama

33

ALGORİTMALARA GİRİŞ

- Sorun: İşaretili tamsayıların saklanması: Artık gösterim (excess notation) ile:
 - Bu kez MSB = 1 iken sayı pozitifdir, aksi halde negatiftir.

Üç ikili artık gösterimi ile

111 3
110 2
101 1
100 0
011 -1
010 -2
001 -3
000 -4

Dört ikili artık gösterimi ile

1111 7
1110 6
1101 5
1100 4
1011 3
1010 2
1001 1
1000 0
0111 -1
0110 -2
0101 -3
0100 -4
0011 -5
0010 -6
0001 -7
0000 -8

- Dört ikili ile olan artık gösterime, bu şekilde gösterilebilen en küçük negatif sayı -8 olduğu için "artık 8 gösterimi" de denir.
 - Benzer şekilde, üç ikili ile olanı "artık 4 gösterimi" adını alır.

34

ALGORİTMALARA GİRİŞ

- Sorun: Onluk düzendeki bir negatif sayıyı artık gösterimine çevirme:
 - Algoritma: İkkinin tamlayanları için olan ile çok benzer
 - Sayıyı pozitif olarak ikili düzende yazın, ancak MSB'yi 1 yapın.
 - LSB'den (en düşük anlamlı/en sağdaki ikili) başlayarak 1 kopyalayana kadar basamakları kopyalayın.
 - Bundan sonra basamakları evirerek (DEĞİL işlemi ile) kopyalayın.

1 1 1 0
↓ ↓ ↓ ↓
0 0 1 0

Düz ok : Düz kopyalama
Kesikli ok: Evirerek kopyalama

35

ALGORİTMALARA GİRİŞ

- İkkinin tümleyenleri gösterimi ile toplama işlemi
 - Onluk tabanda yaptığımız gibi, ancak işaretli sayılarda taşma/overflow hatası olabilir.
 - Çünkü bit uzunluğumuz tüm sistemde sabittir, değişmez.
 - Örneklerde 4-bit ikkinin tümleyeni gösterimi olsun.
 - Bazı işlemler taşmaya rağmen doğru sonuç verebilir.
 - $1 + 2 = 3$ olmalı, sonuç?
 - Bazı işlemler ise taşma nedeniyle yanlış sonuç üretir.
 - $5 + 4 = 9$ olmalı, sonuç?
 - Taşma hatası sabit basamak sayısı kullanıldığında on tabanında da olur elbet.

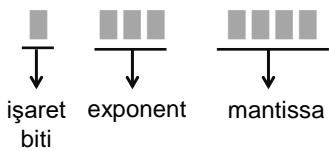
36

ALGORİTMALARA GİRİŞ

- Sorun: İkili düzendeki kesirli sayıların onluk düzene çevrilmesi
 - $(101.101)_2 = (X)_{10}$
 - Nasıl ki noktanın sağı ikinin tam (pozitif) katlarını simgeliyorsa, noktanın solu da ikinin kesirlerini (negatif katlarını) simgeler.
 - 101.101
 - $\rightarrow 1 \times 1/8 \ (2^{-3}) = 1/8$
 - $\rightarrow 0 \times 1/4 \ (2^{-2}) = 0$
 - $\rightarrow 1 \times 1/2 \ (2^{-1}) = 1/2$
 - $\rightarrow 1 \times 1 \ (2^0) = 1$
 - $\rightarrow 0 \times 2 \ (2^1) = 0$
 - $\rightarrow 1 \times 4 \ (2^2) = 4$
 - Kesir toplamı: $5/8$
 - Tamsayı toplamı: 5
- $X = 5\frac{5}{8}$

37

ALGORİTMALARA GİRİŞ

- Sorun: İkili düzende kesirli sayıların gösterilmesi
 - Önceki örnekte küsurat noktasını elle gösterdik, ancak sadece 1 ve 0'ların saklanabildiği sistemde nokta simgelenemez.
 - Noktanın yerini bir şekilde belirleyebilmemiz lazım.
 - Çözüm: Kayar nokta gösterimi (floating-point notation)
- Kayar nokta gösterimi: Sekizlinin hangi basamaklarının ne amaçla kullanılacağı önceden kararlaştırılmıştır.
 - MSB'den başlayarak 1 bit işaret, 3 bit exponent ve 4 bit mantis için kullanılır.
 - 
 - exponent: Küsurat noktasının yerini belirler (Üç ikili artık gösterimi ile)
 - mantissa: Saklanan değer verisini içerir
 - Dikkat: Mantis hem tamsayı hem küsurat için kullanıldığından sadece 4 ikililik alanda gösterebileceğimiz gerçek sayı aralığı kısıtlıdır.

38

ALGORİTMALARA GİRİŞ

- Örnek: Kayar nokta gösterimindeki bir sekizliyi on tabanına çevirelim: 01101011

Önce işaret-tamsayı-küsurat ayırımı yapalım: 0 110 1011

MSB = 0 → Pozitif sayı,

exponent = $(110)_2 = (+2)_{10}$ (3-bit artık)

mantiste önce küsurat noktası en solda: .1011

sonra noktayı exponent kadar basamak sağa ötele 10.11

(exponent negatif olsaydı sola ötelerdik)

- Tamsayı ve küsurat kısımlarını ayrı ayrı onluk tabana çevirelim (daha önceki yalın biçimde)

+10.11

→ 1 x 1/4 (2^{-2}) = 1/4 • Kesir toplamı: 3/4
→ 1 x 1/2 (2^{-1}) = 1/2
→ 0 x 1 (2^0) = 0 • Tamsayı toplamı: 2
→ 1 x 2 (2^1) = 2 • Sonuç: 2 3/4

39

ALGORİTMALARA GİRİŞ

- Örnek: Bir ondalık sayıyı kayar nokta gösterimine çevirelim: 1.125 yani $1\frac{1}{8}$
 - Uygulama kolaylığı için ondalık kısım uygun bir kesir seçildi
 - Adım 1: Mantis bölümü doldurulur:
 - Tamsayı: $(1)_{10} = (1)_2$, Küsurat: $(1/8)_{10} = (001)_2$
 - Mantis = 1001
 - Dikkat: Mantis doldurulurken ilk 1 ile başlanır ve alanın kalan basamakları olursa onlar 0 ile doldurulur. Buna normalizasyon denir ki bu örnekte gerekmemiştir.
 - Adım 2: Eksponent bölümü doldurulur:
 - Noktanın en soldan kaç basamak sağa çekilmesi gerekmektedir? Yanıt: 1
 - $(1)_{10} = (101)_2$, Üç ikili artık gösterimi ile.
 - Adım 3: İşaret bölümü doldurulur:
 - Pozitif sayı kodladığımızdan 0 olur.
 - Sonuç: 01011001

40

ALGORİTMALARA GİRİŞ

- Normalize kayar nokta gösterimi:
 - Mantisteki yer kısıtından bahsedilmişti. Bu nedenle Mantis doldurulurken 4 bitlik alan doldurulmaya değerdeki ilk 1 sayısından başlanır ve alanın kalan basamakları olursa onlar da 0 ile doldurulur.
 - Normalizasyonun olumlu etkileri:
 - Sıfır olmayan tüm pozitif veya negatif sayıların mantisi 1 ile başlar.
 - Aynı sayısal değer birden fazla gösterimi olması engellenir.
 - Örnek: $3/8$ sayısını kayar nokta gösteriminde kodlayalım:
 - $(3/8)_{10} = (011)_2$
 - Mantis: **1100** (**0110** değil, doldurma 0'lar bu ve önceki satırda daha silik).
 - Eksponent: 100 (Nokta 0 basamak sağa çekilecek, 3 ikili artık gösteriminde)
 - İşaret: 0
 - Sonuç: 0 100 1100

41

ALGORİTMALARA GİRİŞ

- Kayar nokta gösteriminde yuvarlama hataları (truncation/round-off errors):
 - Mantisin alamayacağı sayıları kodlamaya çalışırken veya kesirli sayılarla işlem yaparken karşımıza çıkabilir.
 - Örnek 1: $2^{5/8}$ sayısını çevirelim:
 - Tamsayı: $(2)_{10} = (10)_2$, Küsurat: $(5/8)_{10} = (101)_2$, yani mantis = 10.101, halbuki yerimiz 4-bit.
 - Hatayı göz ardı edip devam edersek mantis = 1010 (En düşük anlama sahip basamağın silinmesine göz yumduk)
 - Eksponent $(2)_{10} = (110)_2$, Üç ikili artık gösterimi ile.
 - İşaret ikilisi = 0
 - Sonuç: $(0\ 110\ 1010)_2 = (2^{1/2})_{10}$
 - Kayıp: $2,625 - 2,500 = 0,125$ yani bu işlem için %4,762'lik hata!
 - Çözüm: Daha fazla sekizli kullanılması.
 - Güncel sistemlerde en az 4 sekizli kullanılıyor (32-bit).
 - Hem eksponent hem mantis için artacak yere rağmen hala gerçek sayı işlemlerinde sorunlar çıkabilir.

42

ALGORİTMALARA GİRİŞ

- Kayar nokta gösteriminde yuvarlama hataları (devam):
 - Örnek 2: $2\frac{1}{2} + \frac{1}{8} + \frac{1}{8}$ işlemini yapalım:
 - Mantiste normalizasyon kullanıldığı için doğrudan ikili gösterimlerde toplama yapamayacağız. İki bileşeni toplayıp sonucu ikiliye çevireceğiz.
 - $2\frac{1}{2} + \frac{1}{8} = 2\frac{5}{8}$ halbuki bu sayıda taşma olduğunu gördük, sonuç $(2\frac{1}{2})_{10}$ olmuştu.
 - Yani ilk $\frac{1}{8}$ 'in toplanması uçtu gitti! İkincisi de aynı şekilde uçup gidecek.
 - Bu kez toplamayı sağdan sola yapalım:
 - $\frac{1}{8} + \frac{1}{8} = \frac{1}{4}$, çevirirsek .01 yani 0 011 1000 taşma yok
 - $2\frac{1}{2} + \frac{1}{4} = 2\frac{3}{4}$ çevirirsek 10.11 yani 0 110 1011 taşma yok, sonuç doğru saklanabildi.
 - Sorunun tanımı: Çok büyük bir gerçek sayı ile çok küçük bir gerçek sayı toplandığında küçük sayıda yuvarlama hatası olabiliyor.
 - Çözüm: Önce küçük sayıları kendi aralarında toplayarak ara toplamı büyütmek.
 - Modern derleyiciler ve paket yazılımlar zaten bu şekilde çalışıyorlar.

43

KARAR VERME İŞLEMLERİ İLE AKIŞ DENETİMİ

- Akış: Algoritma adımlarının veya program komutlarının işlenme sırası.
- Şimdiye dek incelediğimiz algoritmalarda adımlar birbirini izleyen sırada yürütüldü (düz sıralama: direct sequencing).
- Oysa algoritmanın her adımının her zaman çalıştırılması istenmeyebilir, değişen koşullara göre farklı işlemlerin yapılması gerekebilir
 - Koşulsal dallanma: conditional branching
 - Karar verme işlemidir.
 - Akış şemalarında altıgen olarak gösterilir.
- Örnek algoritma: Yetişkinliğe karar verme
 - Giriş: Ülkedeki reşitlik yaş sınırı ve kişinin yaşı
 - İşlem:
 - Eğer kişinin yaşı reşitlik sınırına eşit veya büyük ise kişinin reşit olduğunu bildir.
 - Aksi halde kişinin reşit olmadığını bildir.
 - Çıkış: Yapılan bildiri

44

KARAR VERME İŞLEMLERİ İLE AKIŞ DENETİMİ

- Algoritmanın akış şeması:

```
graph TD; A[BAŞLA] --> B[SINIR, YAŞ]; B --> C{YAŞ >= SINIR}; C -- F --> D["Kişi henüz reşit değildir."]; C -- T --> E["Kişi kanunen bir yetişkindir."]; D --> F[BİTİR]; E --> F;
```
- Tek bir BAŞLA ve BİTİR noktasının olmasına dikkat ediniz.
- Değişken adı yeterince açıklayıcı değilse, kullanıcıdan ne istendiğini anlatan bir YAZDIR kutusu ekleyiniz.

- Akış şeması kuralları:
 - Ok ucu çizgi sonunda olabilir. Ok köşeli olacak: Yatay – dikey
 - Karar altıgeninin iki ucundan çıkış olmalı. Solda False, sağda True ve F/T harfleri mutlaka gösterilmeli
 - Kolların birleşme noktasının vurgulanmasına dikkat
 - Bir kolda işlem yapılmasa bile o kol çizilmelidir.
- Dikkat:
 - Bir seferde iki değişken okuduk, normalde programda kullanıcıya ne girmesi gerektiğinin mesajı verilir ve değişkenler tek tek istenir.
 - Bilgisayar programlarında değişken adlarında Türkçe karakter ve boşluk kullanılamaz.

45

KARAR VERME İŞLEMLERİ İLE AKIŞ DENETİMİ

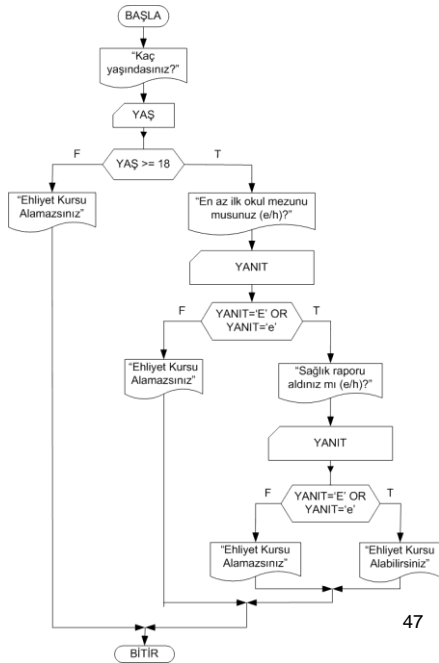
- Daha ayrıntılı koşullar oluşturmak için mantıksal işlemler de kullanılabilir.
- Örnek:
 - VEYA kullanımı
 - Kullanıcıdan ne istediğinin belirtilmesi
 - Alınan yanıtta büyük/küçük harf farklılığının (case sensitivity) değerlendirilmesi
- Tırnak kullanımına dikkat:
 - Tek karakter tek tırnak, karakter katarı çift tırnak ile gösterilir.
 - Bir değişkenin değeri yazılacaksa hiç tırnak kullanılmaz

```
graph TD; A[BAŞLA] --> B["Yaşınız 18 veya üzerinde mi (e/h)?"]; B --> C[YANIT]; C --> D{YANIT='E' OR YANIT='e'}; D -- F --> E["Yetişkin değilsiniz."]; D -- T --> F["Bir yetişkinsiniz."]; E --> G[BİTİR]; F --> G;
```

46

KARAR VERME İŞLEMLERİ İLE AKIŞ DENETİMİ

- Örnek: Bir kişinin ehliyet kursuna yazılıp yazılamayacağına karar verilmesi:
 - Kişi 18 yaşını doldurmuş, en az ilkokulu bitirmiş ve sağlık raporu alabilmişse ehliyet kursuna yazılabilir.
- “Alamazsınız” mesajı için birden fazla kutu, GOTO komutunu önlemek için.
 - Aynı kutuya yazmanız, işlerin ancak bir fonksiyonda toplanması durumunda uygun olur.
 - Böylece istersek hangi nedenle olumsuz yanıt verildiği de açıklanabilir.



47

PASCAL PROGRAMLAMA DİLİNE GİRİŞ

- Pascal programlarının genel yapısı:

```
{ Kıvrık parantez içerisine istenildiği kadar satırlık yorum yazılabilir. }
program ProgramAdi; { Programınıza anlamlı bir isim veriniz. }
var { Değişkenler, kullanılmadan önce tanımlanmalıdırlar. }

    birHarf : char;
    ay, gun : integer;
    oran : real;

const { Sabitler: Bir kez değer aldıktan sonra değiştirilemeyen değişkenler }
    pi = 3.1416; { = ile atama, atanan değere göre değişken tipi kendiliğinden
    belirlenir. Program kodu içerisinde atama ise := ile yapılır. }

begin -----> BAŞLA([BAŞLA])
    { Program kodu }
end. -----> BITIR([BITİR])
```
- Değişkenlere anlamlı adlar atayın! Boşluk ve Türkçe karakter içeremez. camelCasing veya underscore_separator yapılabilir.
- Diğer kısımlarda anlamlı boşluklandırma.
- Pascal dilinde büyük/küçük harf ayrımı yoktur.
- Dev-Pascal kurunca icons dizininin adını icon olarak değiştiriniz.

48

PASCAL DİLİNDE KARAR VERME

- Karar verme ifadesinin kod karşılığı:

```
graph TD
    KOŞUL{KOŞUL} -- F --> İŞLEM_F[İŞLEM_F]
    KOŞUL -- T --> İŞLEM_T[İŞLEM_T]
    İŞLEM_T --> İŞLEM_F;
    İŞLEM_F --> BİRLEŞİM(( ))
    İŞLEM_T --> BİRLEŞİM
    BİRLEŞİM --> İŞLEM_F;
```

```
if koşul then begin
    //koşul doğru
    komut1;
    komut2;
    ...
end //doğru bloğunun sonu
else begin //koşul yanlış
    komut1;
    komut2;
    ...
end; //yanlış bloğunun sonu
```

- Pascal noktalı virgül konusunda çok tutucu! `if koşul then`
`İŞLEM_T;`
- `end` vurgusu: Özellikle önceki örnekteki gibi iç içe (nested) yapıların daha iyi anlaşılmasını sağladığı için önemlidir.
- Artık önceki akış şemamızı Pascal dilinde kodlayabiliriz:

49

PASCAL DİLİNDE KARAR VERME

```
graph TD
    BAŞLA([BAŞLA]) --> Y1[Kaç yaşındasınız?]
    Y1 --> Y2{YAŞ >= 18}
    Y2 -- F --> A1[Ehliyet Kursu Alamazsınız]
    Y2 -- T --> Y3{En az ilköğretim mezunu musunuz (e/h)?}
    Y3 -- F --> A1
    Y3 -- T --> Y4{Sağlık raporu aldınız mı (e/h)?}
    Y4 -- F --> A1
    Y4 -- T --> A2[Ehliyet Kursu Alabilirsiniz]
    A1 --> BİRLEŞİM1(( ))
    A2 --> BİRLEŞİM1
    BİRLEŞİM1 --> BİTİR([BİTİR])
```

```
program Ehliyet;
var
    yas : integer;
    yanıt : string;
begin
    Write('Kaç yaşındasınız? ');
    ReadLn( yas );
    if yas >= 18 then begin
        Write('En az ilköğretim mezunu musunuz (e/h)? ');
        ReadLn(yanıt);
        if (yanıt = 'e') or (yanıt = 'E') then begin
            Write('Sağlık raporu aldınız mı (e/h)? ');
            ReadLn(yanıt);
            if (yanıt = 'e') or (yanıt = 'E') then begin
                WriteLn('Ehliyet kursu alabilirsiniz');
            end
            else begin
                WriteLn('Ehliyet kursu alamazsınız');
            end;
        end
        else begin
            WriteLn('Ehliyet kursu alamazsınız');
        end;
    end
    else begin
        WriteLn('Ehliyet kursu alamazsınız');
    end;
end;
end;
ReadLn;
end.
```

- Koddaki girintilendirmeye dikkat ediniz.
- Son ReadLn komutu program çıktısının çabucak kaybolmasını önleyip sonuçları değerlendirebilmeniz için gereklidir.
- Türkçe karakterler komut satırında düzgün görüntülenememekte.
- Kodlarken her seferinde bir print ile kullanıcıdan ne istediğimizi belirtmeye dikkat edin.

50

ÇEVİRİMLER İLE AKIŞ DENETİMİ

- Şimdiye kadar verdiğimiz komutlar yalnız bir kez işlendi veya koşula bağlı olarak hiç işlenmedi.
- Bir komutu birden fazla kez işletmek istiyorsak:
 - Algoritma: 1'den N'e kadar olan sayıların karelerinin toplamı
 - Giriş: N
 - İşlem:
 1. i = 1, toplam = 0 (i: sayaç değişkeni)
 2. toplam = toplam + i * i
 3. i = i + 1
 4. Eğer i <= n ise 2. işlemten itibaren yinele
 - Çıktı: toplam

$$\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6} = \frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$$

```
graph TD
    Start([BAŞLA]) --> N[/N/]
    N --> Init[i = 1, TOPLAM = 0]
    Init --> LoopBody[TOPLAM = TOPLAM + i * i]
    LoopBody --> Inc[i = i + 1]
    Inc --> Decision{i <= N}
    Decision -- T --> LoopBody
    Decision -- F --> Output[/TOPLAM/]
    Output --> End([BİTİR])
```

51

ÇEVİRİMLER İLE AKIŞ DENETİMİ

- Bir komutu birden fazla kez işletmeye verilen ad döngü veya çevrimdir (loop)
- Döngüler programlamada çok kullanıldıklarından programlama dillerinde hazır işlem olarak gelmekte ve akış şemalarında ayrı bir şekil olarak gösterilmektedir.
- Önceki algoritmada çarparak toplama işlemini yinelemek için bir sayaç değişkenini artırıp değerini denetliyorduk.
- Yineleme komutu bu işlemleri kendisi yapar.
- Gösterim:

i ← ilk	++
Son Değer	
- Son değer dahil
- Artım yerine eksiltme de olabilir.

```
graph TD
    Start([BAŞLA]) --> N[/N/]
    N --> Init[TOPLAM = 0]
    Init --> LoopBox
    subgraph LoopBox [ ]
        direction TB
        i1[i ← 1]
        i2[1]
        N1[N]
    end
    LoopBox --> Decision{i <= N}
    Decision -- T --> LoopBox
    Decision -- F --> Output[/TOPLAM/]
    Output --> End([BİTİR])
```

- Bu ve bir çok örnekte sıfırlamayı for içinde yapmamalı
- Aksi halde toplam her iterasyonda tekrar sıfırlanır ve sonuç hatalı hesaplanır.

52

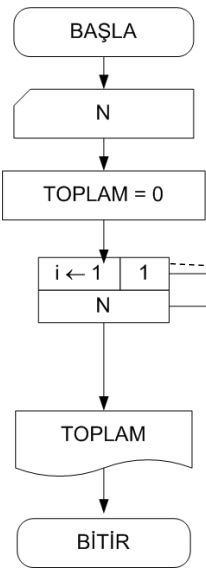
PASCAL DİLİNDE ÇEVİRİMLER

```
for sayaç := ilk değer to son değer do begin
    komutlar;
end;
```

- Sayaç bir değişken olmalıdır.
- İlk değer, son değer değişken veya tamsayı olabilir.
- Sayacın değeri her çevrimin sonunda bir artar.
- Değişkenler daha önce tanımlanmış olmalıdır.
- Son değer bir sabit olabilir.
- Sayacın değerini döngü içerisinde değiştirmeyin.
- İlk değer son değerden büyükse çevrime girilmez.
- Geriye doğru sayım için to yerine downto
 - İlk değer son değerden küçükse çevrime girilmez.
- Birden daha fazla artırım Pascal'da desteklenmiyor!
 - O durumda while kullan (ileride.)

53

PASCAL DİLİNDE ÇEVİRİMLER



- Pascal kodunda atama := ile

```
program KareTopla;
var
    n, toplam, i : integer;
begin
    Write('Üst sınır girin: ');
    ReadLn(n);
    toplam := 0;
    for i := 1 to n do begin
        > toplam := toplam + i * i;
    end;
    WriteLn('Toplam: ', toplam);
    ReadLn;
end.
```

- Toplam yazdırma işlemini for döngüsünde yapmamak lazım. Aksi halde her iterasyonda ara toplam yazdırılacak ve boş yere N adet satır yazılacak.

54

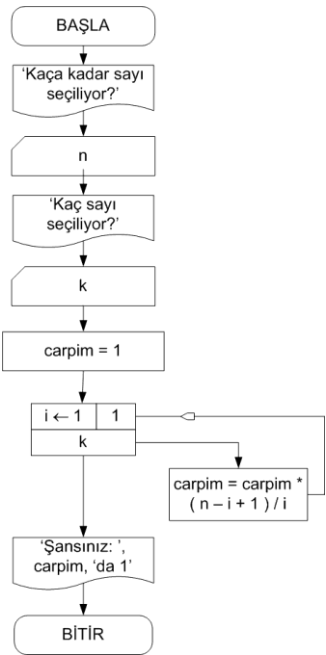
ÖRNEK PROBLEM

- Bir arkadaşımız bizden bir sonraki sayısal lotoda çıkacak numaraları bilen bir program yazmamızı istedi.
- Bir "kahin" program yazamayacağımıza göre, 1..N arası sayıdan K tanesinin çekildiği bir şans oyununda büyük ikramiyeyi tutturma şansının ne olduğunu bildiren genelleştirilmiş bir programla idare edelim.
- İpucu (Sınavlarda ipucu vermek zorunda değiliz!):

In general, if you pick k numbers out of n , there are
$$\frac{n \times (n-1) \times (n-2) \times \dots \times (n-k+1)}{1 \times 2 \times 3 \times \dots \times k}$$
possible outcomes.

55

ÇÖZÜM



```
program Piyango;
var
    n, k, i : integer;
    carpim : real;
begin
    Write('Kaça kadar sayı seçiliyor? ');
    ReadLn(n);
    Write('Kaç sayı seçiliyor? ');
    ReadLn(k);
    carpim := 1;
    for i := 1 to k do begin
        carpim := carpim * (n-i+1)/i;
    end;
    Write('Şansınız ', carpim :12:1, 'da 1'dir. ');
    WriteLn(' Bol Şans!');
    ReadLn;
end.
```

56

ÇÖZÜMSÜZ BASİT PROBLEMLER

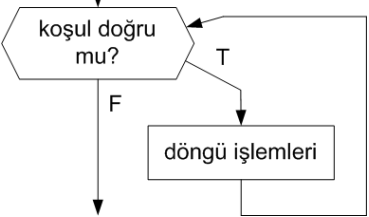
- P1-1: Klavyeden girilen bir tamsayının tek mi çift mi olduğunu bildiren program
- P1-2: Klavyeden girilen a ve b tamsayıları için a'nın b'ye tam bölünüp bölünemediğini bildiren program
- P1-3: Klavyeden girilen n tamsayısı için n! hesaplayan
- P1-4: Klavyeden girilen n ve r tamsayıları için C(n,r) hesaplayan program
- P1-5: Klavyeden katsayıları girilen 2. derece denklemin köklerini hesaplayan program

57

ÇEVİRİMLER İLE AKIŞ DENETİMİ

- while ifadesi ile döngü: Bir komutu belli bir koşul geçerli olduğu sürece yinelemek için.

```
while koşul do begin
    komutlar;
end;
```



- Döngüye girmeden önce döngüye girmeyi kesinleştirmek için koşulu doğrulamak gerekir.
- Koşul değişir değişmez döngü sonlanmaz, sadece koşulun denetlendiği anlarda döngünün sonlanıp sonlanmayacağına karar verilir.
 - Ek if komutları ile döngüde hatalı işlem yapılmaması sağlanabilir.
 - if komutu denetiminde verilecek bir **break** komutu ile döngüden çıkılabilir.
- Akış şemasında döngünün başlangıç ve bitiş oklarının while altıgeninin sırasıyla sağ alt ve üst yan kenarlarına iliştiniz.

58

ÇEVİRİMLER İLE AKIŞ DENETİMİ

- Örnek: Negatif bir sayı girilinceye kadar, girilen sayıların karekökünü hesaplama

```
program KarekokHesap1;  
var  
    sayi, sonuc: real;  
begin  
    Write('Bir sayi girin: ' );  
    ReadLn( sayi );  
    while sayi >= 0 do begin  
        sonuc := Sqrt( sayi );  
        WriteLn('Karekoku: ' :16,sonuc :0:5);  
        Write('Bir sayi girin: ' );  
        ReadLn( sayi );  
    end;  
end.
```

Neden? →

59

ÇEVİRİMLER İLE AKIŞ DENETİMİ

- Bir de 0 girince programdan çıkılmasını sağlayacak şu haline bakınız:

```
program KarekokHesap2;  
var  
    sayi, sonuc: real;  
begin  
    sayi := 42;  
    while sayi <> 0 do begin  
        Write('Bir sayi girin: ' );  
        ReadLn( sayi );  
        if sayi > 0 then begin  
            sonuc := Sqrt( sayi );  
            WriteLn('Karekoku: ' :16,sonuc :0:5);  
        end  
        else if sayi < 0 then begin  
            WriteLn('Negatif sayıların karekoku hesaplanamaz');  
        end;  
    end;  
end.
```

- Sayı isteme komutlarını iki kez vermekten kurtulduk. Ancak koşul değişir değişmez döngü sonlanmadığı için ek if denetimlerine gerek duyuldu.

60

ÇEVİRİMLER İLE AKIŞ DENETİMİ

- While ile for benzetimi; birden farklı artırım ile!

```
program CiftSayilarFor;
var
  ilk, son, sayac : integer;
begin
  Write('Verilen aralıktaki çift ');
  WriteLn('sayıları yazan program. ');
  Write('Başlangıcı girin: ');
  ReadLn(ilk);
  Write('Bitişi girin: ');
  ReadLn(son);
  for sayac := ilk to son do begin
    if sayac mod 2 = 0 then
      Write( sayac, ' ');
    end;
    ReadLn;
  end.
```

- For ile olan kod daha basit gözüküyor ama daha fazla işlem yapıyor!

```
program CiftSayilarWhile;
var
  ilk, son, sayac : integer;
begin
  Write('Verilen aralıktaki çift ');
  WriteLn('sayıları yazan program. ');
  Write('Başlangıcı girin: ');
  ReadLn(ilk);
  Write('Bitişi girin: ');
  ReadLn(son);
  if ilk mod 2 <> 0 then
    ilk := ilk + 1;
  sayac := ilk;
  while sayac <= son do begin
    Write( sayac, ' ');
    sayac := sayac + 2;
  end;
  ReadLn;
end.
```

61

ÇEVİRİM İFADELERİ

- İç içe çevrimler:

- Döngü içerisine if kontrolü koyabileceğimiz gibi, döngü içinde döngü de kullanabiliriz.
- For içinde for, for içinde while, while içinde for, ... her kombinasyon mümkündür.
- Dış çevrim daha seyrek, iç çevrim daha sık döner.
- Toplam tekrar sayısı = dış çevrim sayısı * iç çevrim sayısı.

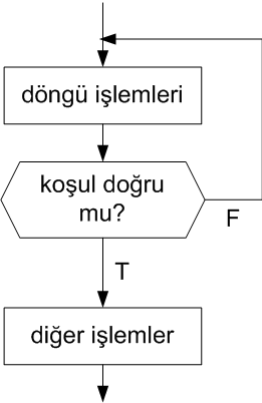
```
program ic_ice_for_deneme;
var
  i, j: Integer;
begin
  Writeln(' (i,j) ');
  for i := 1 to 3 do begin
    for j := 1 to 4 do begin
      Writeln(' (' , i , ' , ' , j , ') ');
    end;
  end;
  ReadLn;
end.
```

(i , j)
(1 , 1)
(1 , 2)
(1 , 3)
(1 , 4)
(2 , 1)
(2 , 2)
(2 , 3)
(2 , 4)
(3 , 1)
(3 , 2)
(3 , 3)
(3 , 4)

62

ÇEVİRİM İFADELERİ

- repeat-until ifadesi ile döngü: Bir komutu belli bir koşul geçerli olana kadar yinelemek için.
repeat
 komutlar;
until *koşul;*
- while'dan farklarına DİKKAT:
 - Koşul **doğru** olunca döngüden çıkılır.
 - Bazı dillerde tersidir
 - (Ör. C, Java: do – while şeklinde).
 - Kontrolün **sonda** yapılması.
 - Bu bize ne kazandırır?
 - İlk değer atama işlemini blok öncesinde yapma zorunluluğu kalkar.
 - SORU: Çevrimi bitiren koşul gerçekleşir gerçekleşmez döngü sonlanır mı?
 - Cevap while ile aynı: Hayır. İşlem bloğu yine birden fazla işlemden oluşur, koşul until kontrolüne gelmeden önce değişebilir.



63

ÇEVİRİM İFADELERİ

- Karekök hesaplama programının repeat ile yazılması:
program repeatDongusu;
var
 deger, sonuc: real;
begin
 Write('Karekok hesaplama programı. ');
 Write('Programdan cikmak icin sifir girebilirsiniz. ');
 repeat
 Write('Bir sayi girin: ');
 Read(deger);
 if deger > 0 **then begin**
 sonuc := Sqrt(deger);
 WriteLn('Karekoku: ' :16, sonuc :0 :5);
 end
 else if (deger < 0) **then**
 WriteLn('Negatif sayilarin karekoku hesaplanamaz. ');
 until deger = 0; //DİKKAT! while'da deger >= 0 idi.
end.
- Bu örnekte de karekök hesaplamadan önce bir sınama gerekti. Ancak döngüye girişi sağlamak için ilk değer ataması yapmak zorunda kalmadık.

64

ÇÖZÜMSÜZ BASİT PROBLEMLER

- P2-1: Girilen pozitif n tamsayısının, iki tam sayının kareleri toplamı şeklinde yazılıp yazılamayacağını hesaplayan program.
- P2-2: Basit bir piyango çekilişi: Bilgisayar iki basamaklı rastgele bir sayı üretir, Kullanıcı da bir tahmin girer. Kullanıcı sayıyı doğru tahmin ederse 10,000TL, sayının basamaklarını ters sırada tahmin ederse 3,000TL, tek bir basamağı tahmin ederse 1,000TL kazanır. Her bir piyango bileti ise 100TL'dir. Kullanıcı başta ne kadar para ile oyuna başlamak istediğini girsın ve çıkmak isteyene dek oyun sürsün.
- P2-3: Ekrana çarpım tablosu yazdıran program
- P2-4: Bir pozitif tamsayının mükemmel sayı olup olmadığını bildiren program. Bir mükemmel sayı, kendisi hariç tüm bölenlerinin toplamına eşit olan sayıdır. Ör. $6=3+2+1$, $28= 14+7+4+2+1$
- P2-5: Bkz. BBG1-P2.docx
- P2-6: Bkz. BBG1-P2.docx, Fahri Vatansever p282

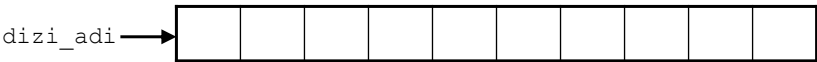
65

DİZİLERE (ARRAYS) GİRİŞ

- Birçok programlama probleminin çözümü için, birbiri ile ilişkili ve aynı tipten verileri ayrı ayrı değişkenlerde tutmak yerine, bunları birlikte saklamak daha uygundur.
- Birlikte saklanan veriler tek bir birleşik "veri yapısı" içerisinde yer alır.
- Bu derste dizi (array) adlı veri yapısını inceleyeceğiz.
- Dizi tanımlama: Değişken tanımlama bloğu içerisinde.

var eleman türü
`dizi_adi : array [1..10] of Integer;`
alt sınır .. üst sınır
eleman sayısı

- Bellekteki durum: Herbiri bir eleman alabilecek, ardışıl konumlanmış hücreler.



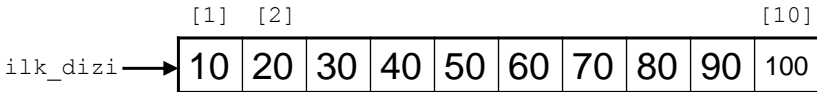
66

DİZİLERE (ARRAYS) GİRİŞ

- Dizi içerisindeki elemanlara ulaşım: Köşeli parantez içerisindeki indeks bilgisi ile.

```
var
    ilk_dizi : array [1..10] of Integer;
begin
    ilk_dizi[1] := 10;
    ilk_dizi[2] := 20;
    ...
```

- Bellekte oluşan durum:



- İndeks değeri olarak tamsayı değişken de kullanılabilir:

```
i := 5;
WriteLn( i, '. kutunun içeriği = ', ilk_dizi[i] );
```

67

DİZİLERE (ARRAYS) GİRİŞ

- Bu öğrendiklerimize göre, az önceki 10 ayrı ayrı atama komutu yerine daha doğru bir çözüm ne olabilir?
- Yanıt: Dizi elemanlarına sıralı erişim.

- Diziyi tanımladıktan sonra tüm elemanları ile ilgili her tür işlemi sıra ile dizinin tüm elemanlarına uygulamak için for döngüsünden yararlanılabilir.

- Örnek: 10 elemanlık diziye 10'un katlarını atama

```
program DiziTanimlari;
var
    sayilar : array [1..10] of Integer;
    i : integer;
begin
    for i := 1 to 10 do begin
        sayilar[i] := i * 10;
        WriteLn( 'Sayı ', i, ' = ', sayilar[i] );
    end;
end.
```

- Döngünün ilk ve son değerlerinin dizinin alt ve üst sınırları ile aynı olmasına dikkat edin, aksi halde çalışma anı hatası alırsınız.

68

DİZİLERE (ARRAYS) GİRİŞ

- Dizilerle ilgili bazı çeşitli konular:
 - Alt sınır 1 olmak zorunda değildir, başka bir tamsayı da olabilir.
 - Dikkat: Bazı dillerde dizi aralığı alt sınırı 1 değil 0'dır ve değiştirilemez. Örnek: C, Java.
 - Üst ve alt sınırlar sabit bir değişken ile belirlenebilir.

```
const
    altSinir = 0;
    ustSinir = 9;

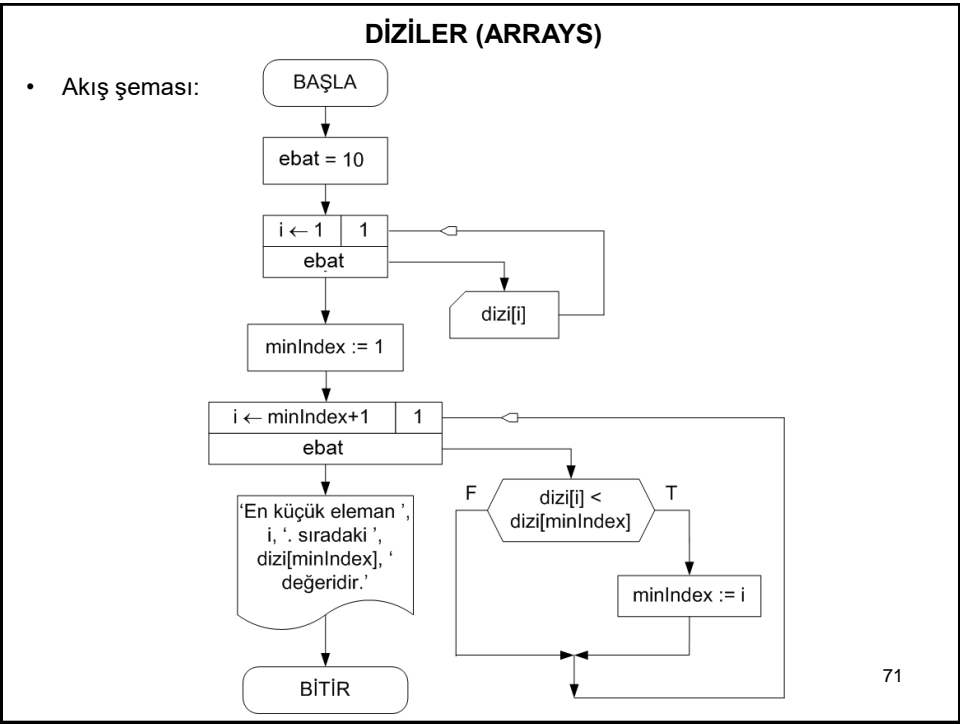
var
    sayilar : array [altSinir..ustSinir] of Real;
```
 - Sabit değişkenin değerini bir tek yerde değiştirmekle, bu değişkenin geçtiği her yere bu değişiklik aktarılmış olur.
 - Pascal dışındaki bazı dillerde dizilerin uzunluğu çalışma anında değiştirilebilir.
 - Ancak bunun için dizinin yeniden tanımlanması gerekir ve bu sırada dizideki mevcut veri kaybolur.
 - Yine de elimizden gelen birşeyler vardır, sizce bu ne olabilir?

69

DİZİLER (ARRAYS)

- Diziler ile ilgili problemler-1: Dizideki en küçük elemanı bulmak.
 - Değerin kaç olduğunu bulmak yetmez, en küçük elemanın dizideki yerini de bulmak gerekir.
 - Sorunu çözmek için elemanın indeksini saklamak yeter.
 - Algoritma:
 - Giriş: En küçük elemanı bulunacak dizi
 - Kabul ve kısıtlar: Dizinin eleman sayısını (ebatını) sabit değişkenle programcı belirler.
 - İşlem:
 - Dizinin ilk elemanını en küçük farzet.
 - Nasıl? minIndex := 1
 - Varsayılan en küçük elemanı sırayla dizinin diğer elemanları ile karşılaştır.
 - Nasıl? for çevriminde i sayaç değişkeni kullanarak
 - Karşılaştırdığın eleman baktığından küçükse, artık o elemanı en küçük farzet.
 - Nasıl? minIndex := i
 - Sonuç: En küçük elemanın indeksi ve dolayısı ile değeri

70



DİZİLER (ARRAYS)

• Program kodu:

```
program EnKucukBul;
const
    ebat = 10;
var
    dizi : array [1..ebat] of Integer;
    i, minIndex : Integer;
begin
    Write( ebat, ' elemanlı bir dizideki ' );
    WriteLn( 'en küçük elemanı bulan program.' );
    for i := 1 to ebat do begin
        Write( i, '. sayıyı giriniz: ' );
        ReadLn( dizi[i] );
    end; { Giriş kısmı bitti, işlem kısmı başlıyor! }
    minIndex := 1;
    for i := minIndex+1 to ebat do begin
        if dizi[i] < dizi[minIndex] then
            minIndex := i;
        end;
    end;
    Write( 'En küçük: ', minIndex, '. elemandır ' );
    WriteLn( 've değeri ', dizi[minIndex], ''tir.' );
    ReadLn;
end.
```

72

DİZİLER (ARRAYS)

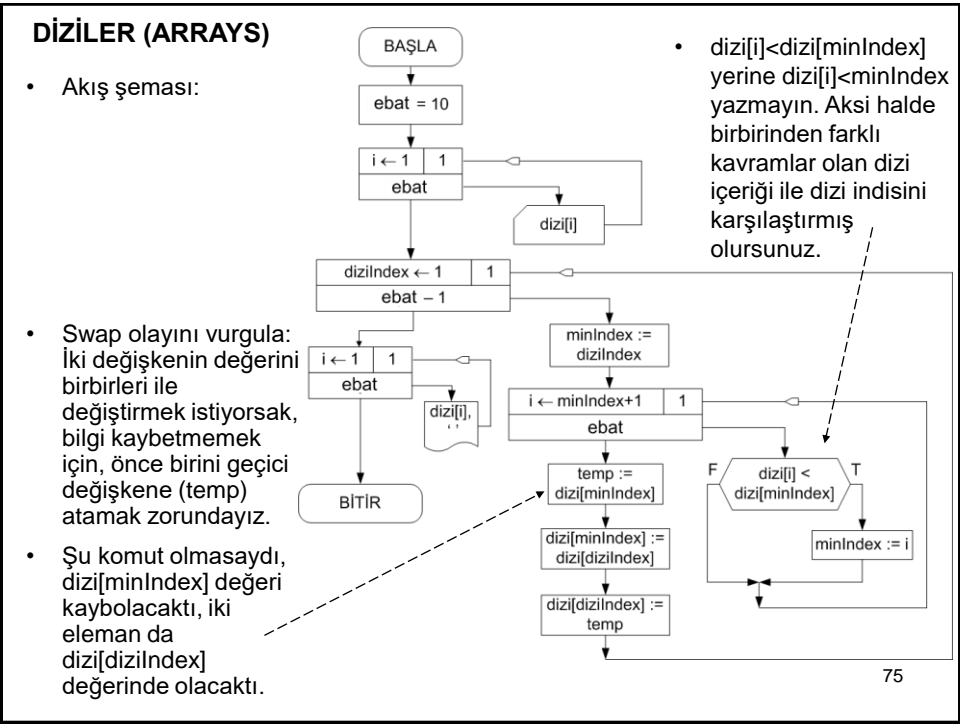
- Dizideki en küçük elemanı bulduk, peki ya en büyüğü nasıl bulunabilir?
 - Benzer şekilde, yalnız karşılaştırmada küçüktür yerine büyüktür karşılaştırması yapılacak
 - Algoritma:
 - Giriş: Dizideki eleman sayısı ve elemanların değerleri
 - İşlem:
 - Adımları size bırakıyorum.
 - Dizinin büyüklüğünü kullanıcının belirlemesini sağlamayı da size bırakıyorum.
 - İpucu: Eleman sayısını kullanıcı girebilir ama siz de dizi için bir üst limit belirleyebilirsiniz.
 - Sonuç: En büyük elemanın indeksi

73

DİZİLER (ARRAYS)

- Dizi ile ilgili problemler:
 - Diziyi sıralamak: Artan veya azalan sırada.
 - Algoritma: Selection Sort
 - Dizideki en küçük elemanı önceki şekilde bul.
 - Bulduğun en küçük eleman ile dizinin başındaki elemanın yerini değiştir: İlk eleman yerine oturdu.
 - Dizinin 2. en küçük elemanını bul. Dizinin başından başlama, çünkü orada 1. en küçük eleman var. Dolayısıyla dizinin 2. elemanından başla.
 - Bulduğun 2. en küçük eleman ile dizinin başındaki elemanın yerini değiştir: 2. eleman yerine oturdu.
 - Bu şekilde tüm elemanları yerine oturt.
 - İşlem:
 - 1. elemanı 2., 3., ... elemanlarla karşılaştır.
 - 2. elemanı 3., 4., ... elemanlarla karşılaştır.
 - ... =>Yukarıdan aşağıya 1,2, ... soldan sağa sütun başı +1'den başlayan bir seri var. Her seri bir döngü: İç içe iki döngü.

74



DİZİLER (ARRAYS)

- Program Kodu:

```
program SiraSelectSort;
const
  ebat = 10;
type
  SayiDizisiTipi = array [1..ebat] of Integer;
var
  dizi : SayiDizisiTipi;
  i, minIndex, diziIndex, temp : Integer;
begin
  Write( ebat, ' elemanlı bir diziyi' );
  WriteLn( ' sıralayan program.' );
  for i := 1 to ebat do begin
    Write( i, ' . sayıyı giriniz: ' );
    ReadLn( dizi[i] );
  end;
  for diziIndex := 1 to ebat-1 do begin
    minIndex := diziIndex;
    for i := minIndex+1 to ebat do begin
      if dizi[i] < dizi[minIndex] then
        minIndex := i;
      end;
      temp := dizi[minIndex];
      dizi[minIndex] := dizi[diziIndex];
      dizi[diziIndex] := temp;
    end;
    Write( 'Sıralı dizi: ' );
    for i := 1 to ebat do begin
      Write( dizi[i], ' ' );
    end;
    ReadLn;
  end;
end.
```

- hemen yazma: swap sorununu tartıştıktan sonra ekle.
- Dizi tanımlamanın bir başka yolu: Önce type bloğunda dizi türü tanımlanır, sonra var bloğunda o tipten dizi değişkenleri tanımlanır
- böyle yazsaydık $dizi[minIndex]$ değeri kaybolacaktı, iki eleman da $dizi[diziIndex]$ değerinde olacaktı.

$dizi[minIndex] := dizi[diziIndex];$
 $dizi[diziIndex] := dizi[minIndex];$

76

KARAKTER KATARLARI İLE ÇALIŞMA

- String değişkeni :

program Tanımlama;

var

 isim : String;

begin

 isim := 'Yunus Emre Selçuk';

end.

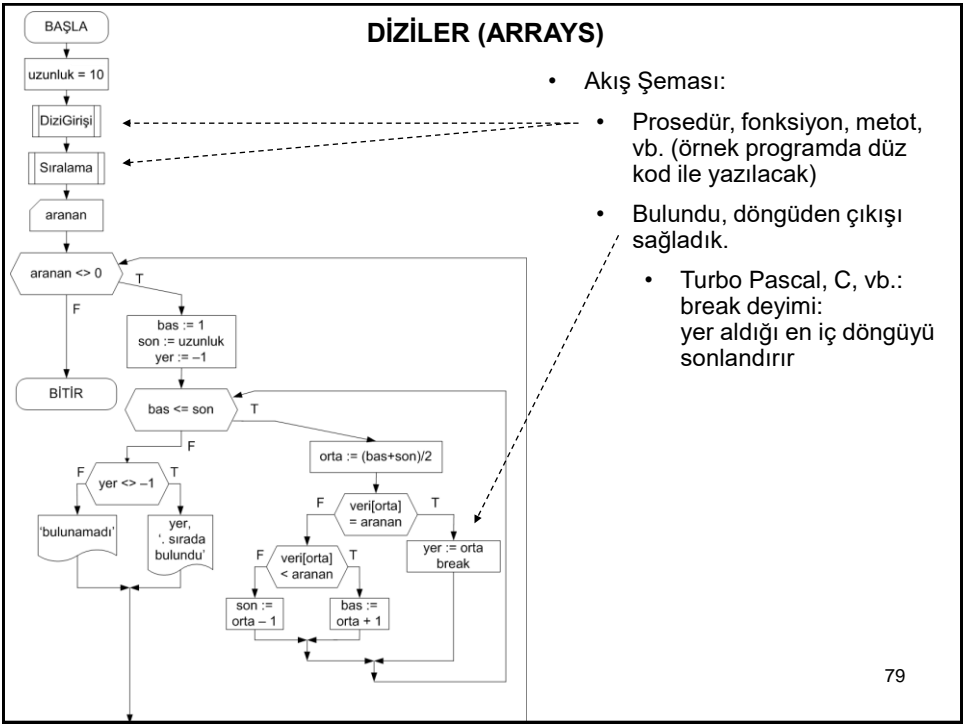
- String değişkeni standart Pascal'da yoktur, ancak Free Pascal derleyicisinde vardır.
- Bu nedenlerle String işlemlerine değinilmeyecektir.
- Ancak String türünden diziler kullanılabilir.

77

DİZİLER (ARRAYS)

- Dizi ile ilgili problemler: Sıralı dizide bir eleman aramak
 - Algoritma: İkili Arama (Binary Search)
 - Giriş: Aranacak veri ve verinin içerisinde aranacağı artan sıralı dizi.
 - İşlem:
 1. Aranacak değeri dizinin ortasındaki elemanla karşılaştır.
 - bas: dizinin başını, son: dizinin sonunu gösteren değişken.
 - Başlangıçta bas := 1, son := uzunluk, orta := (bas + son) div 2
 2. Eşitlerse, bulunmuştur.
 3. Dizinin ortasındaki eleman aranan elemandan küçükse, dizinin sadece sol tarafında ara (bas = orta + 1).
 4. Dizinin ortasındaki eleman aranan elemandan büyükse, dizinin sadece sağ tarafında ara (son = orta - 1).
 5. bas > son ise dizide arayacak yer kalmamış demektir.
 - Sonuç: Aranacak eleman dizide mevcutsa elemanın indeksi
 - Aranacak eleman dizide yoksa -1 sonucu verilsin
 - Gerçekleme ayrıntıları: Dizi bir kez girilsin, sonra kullanıcı 0 girene dek girdiği her değer dizide aransın.

78



DİZİLER (ARRAYS)

- Örnek akış:
 - Dizi: 13 19 27 28 45 57 58 76 83 99
 - Aranan: 27**
bas: 1 son: 10 orta: 5 veri[orta]: 45
bas: 1 son: 4 orta: 2 veri[orta]: 19
bas: 3 son: 4 orta: 3 veri[orta]: 27
Bulunan yer: 3
 - Aranan: 88**
bas: 1 son: 10 orta: 5 veri[orta]: 45
bas: 6 son: 10 orta: 8 veri[orta]: 76
bas: 9 son: 10 orta: 9 veri[orta]: 83
bas: 10 son: 10 orta: 10 veri[orta]: 99
Aranan bulunamadı.
- Algoritmanın karmaşıklığı: $O(\ln N)$
 - $N = 2^k$ eleman için en fazla k adımda aranan elemana ulaşılır.

80

DİZİLER (ARRAYS)

• Program Kodu:

```
program ikiliArama;
const
    uzunluk = 10;
var
    veri : Array[1..uzunluk] of integer;
    i, j, minIndex, temp : integer;
    bas, son, orta, aranan, yer : integer;
begin
    for i := 1 to uzunluk do begin
        Write( i, ' . elemanı girin: ' );
        ReadLn( veri[i] );
    end;
    Write( 'Sıralanmamış dizi: ' );
    for i := 1 to uzunluk do begin
        Write( veri[i], ' ' );
    end;
    WriteLn;
    for i := 1 to uzunluk-1 do begin
        minIndex := i;
        for j := i+1 to uzunluk do begin
            if veri[j] < veri[minIndex] then
                minIndex := j;
            end;
        end;
        temp := veri[minIndex];
        veri[minIndex] := veri[i];
        veri[i] := temp;
    end;
    Write( 'Sıralanmış dizi: ' );
    for i := 1 to uzunluk do begin
        Write( veri[i], ' ' );
    end;
    WriteLn;
```

81

DİZİLER (ARRAYS)

• Program Kodu (devam):

```
Write( 'Dizide aramak istediğiniz elemanı girin ' );
Write( '(Çıkış için 0): ' );
ReadLn( aranan );
while aranan <> 0 do begin
    bas := 1;
    son := uzunluk;
    yer := -1;
    while bas <= son do begin
        orta := ( bas + son ) div 2;
        if veri[orta] = aranan then begin
            yer := orta;
            break; { yer aldığı en iç döngüyü sonlandırır }
        end
        else if veri[orta] < aranan then
            bas := orta + 1
        else
            son := orta - 1;
        end;
    end;
    if yer <> -1 then begin
        Write( 'Aradığınız eleman dizide ', yer );
        WriteLn( ' . sırada bulunmaktadır.' );
    end
    else
        WriteLn( 'Aradığınız eleman dizide bulunmamaktadır.' );
    end;
    Write( 'Dizide aramak istediğiniz elemanı girin ' );
    Write( '(Çıkış için 0): ' );
    ReadLn( aranan );
end;
end.
```

82

ÇÖZÜMSÜZ PROBLEMLER

- P3-1: Klavyeden girilen N elemanlı A dizisinin elemanlarını klavyeden girilen bir k katsayısı ile çarpıp B dizisini oluşturan ve ekrana yazdıran program.
 - $A = (a_1, a_2, \dots, a_N) \Rightarrow B = k * A = (b_1, b_2, \dots, b_N) = (k*a_1, k*a_2, \dots, k*a_N)$
- P3-2: N elemanlı bir diziyi ters çevirerek bir başka diziye atayan program.
 - Tabi önceki örnekteki gibi bu örnekte ve sonrakilerde de N, $a_1 \dots a_N$ klavyeden girilecek, sonuç dizisi ekrana yazdırılacak.
- P3-3: A ve B dizilerinin skaler çarpımını hesaplayan program.
 - $A*B = a_1*b_1 + a_2*b_2 + \dots + a_N*b_N$ yani sonuç bir tamsayı, dizi değil.
- P3-4: Klavyeden girilen N elemanlı A dizisinin bir küme olup olmadığını belirleyen program.
 - Not: Kümede aynı eleman birden fazla kez tekrarlanamaz. Aksi halde buna küme değil, liste denir. EK: Listeyi kümeye çeviren program.
- P3-5: Klavyeden girilen 0-99 arası sayının yazılışını bildiren program.
 - Örneğin kullanıcı 27 girerse ekrana "yirmi yedi" yazılacak.
- P3-6: Klavyeden girilen sekizlinin ondalık karşılığını hesaplayan program.
 - Bu örneği artık dizilerle güzel güzel yazabilirsiniz!

83

İKİ BOYUTLU DİZİLER

- İki boyutlu diziler:
 - Şimdiye dek gördüğümüz diziler doğrusal bir yapıydı = tek boyutluydu.
 - Bir değişkeni 0 boyutlu bir varlık, yani nokta gibi düşünebilirsiniz. Dizileri ise yanyana dizilmiş noktalardan oluşan bir çizgi gibi, yani bir boyutlu olarak düşünebilirsiniz.
 - Doğal olarak düşüncemiz 2 ve 3 boyutlu dizilere genişleyecektir.
 - 2 boyutlu dizi: Bir satranç tahtası gibi, satırlar ve sütunlardan oluşmuş, dizilerin dizisi.
 - 2 boyutlu dizi tanımlama: [satır,sütun], dikkat: önce satır, sonra sütun.

var

dizi_adi : **array** [1..3,1..10] **of** Integer;

dizi_adi

sütunlar

satırlar

dizi_adi[2,5]

84

İKİ BOYUTLU DİZİLER

- Dizinin bir satırı (sütunu) başka, diğeri başka türden veri içeremez.
 - Bu durumda gerekli sayıda bir boyutlu dizi kullanılır.
- 2 boyutlu diziler 2 boyutlu koordinat sistemlerini modellemek için kullanılabilir.
 - Dikkat: (x,y) matematiksel gösterimi [satır,sütun] gösterimine uymuyor,
 - [y,x] haline dönüyor!
- SOR: 2 boyutlu dizilerin diğ. kullanım alanlarına örnek
 - Matrisler
 - Resim dosyaları
 - Çapraz bulmacalar
 - Excel tabloları
- SOR: 2 boyutlu dizi elemanlarına sıralı erişim nasıl olur?
 - Normal, yani bir boyutlu diziye nasıl eriştik? Bir for ile.
 - 2B → 2 for döngüsü. Ama nasıl iki for? İç içe.
 - Satır satır veya sütunlar halinde ulaşabiliriz.

85

İKİ BOYUTLU DİZİLER

- Satırlar halinde erişim:

```
const
    sutunSayisi = 10;
    satirSayisi = 3;
var
    matris : array [1..satirSayisi,1..sutunSayisi] of Integer;
    satirIndex, sutunIndex : Integer;
begin
    WriteLn( 'Tamsayılardan oluşan bir matrisin girilmesi' );
    for satirIndex := 1 to satirSayisi do begin
        for sutunIndex := 1 to sutunSayisi do begin
            Write( satirIndex, ' . satır ' );
            Write( sutunIndex, ' . sütun ' );
            Write( 'elemanını girin: ' );
            ReadLn( matris[satirIndex,sutunIndex] );
        end;
    end;
end;
```

- SOR: Sütunlar halinde erişim

86

DİZİLER (ARRAYS)

• Örnek kod: Matris çarpımı $c_{i,j} = \sum_{r=1}^n a_{i,r}b_{r,j} = a_{i,1}b_{1,j} + a_{i,2}b_{2,j} + \dots + a_{i,n}b_{n,j}$.

```

program MatrisCarpimi;
const
    SatirSayisi = 3;
    SutunSayisi = 3;
type
    Matris = array [1..satirSayisi,1..sutunSayisi] of Integer;
var
    matris1, matris2, matris3 : Matris;
    satirIndex, sutunIndex, k : Integer;
begin
    Write( SatirSayisi, ' satir ve ', SutunSayisi );
    WriteLn( ' sütunluk iki matrisi çarpan program.' );

    WriteLn( '*** 1. matrisin girilmesi ***' );
    for satirIndex := 1 to SatirSayisi do begin
        for sutunIndex := 1 to SutunSayisi do begin
            Write( satirIndex, '. satir ' );
            Write( sutunIndex, '. sutun ' );
            Write( 'elemanını girin: ' );
            ReadLn( matris1[satirIndex,sutunIndex] );
        end;
    end;
end;

```

87

```

    WriteLn( '*** 2. matrisin girilmesi ***' );
    for satirIndex := 1 to SatirSayisi do begin
        for sutunIndex := 1 to SutunSayisi do begin
            Write( satirIndex, '. satir ' );
            Write( sutunIndex, '. sutun ' );
            Write( 'elemanını girin: ' );
            ReadLn( matris2[satirIndex,sutunIndex] );
        end;
    end;
    for satirIndex := 1 to SatirSayisi do begin
        for sutunIndex := 1 to SutunSayisi do begin
            matris3[satirIndex,sutunIndex] := 0;
            for k := 1 to SatirSayisi do begin
                matris3[satirIndex,sutunIndex] :=
                    matris1[satirIndex,k] *
                    matris2[k,sutunIndex];
            end;
        end;
    end;
    WriteLn( '*** SONUÇ MATRİSİ ***' );
    for satirIndex := 1 to SatirSayisi do begin
        for sutunIndex := 1 to SutunSayisi do begin
            Write( matris3[satirIndex,sutunIndex] :5 );
        end;
        WriteLn;
    end;
end.

```

88

- Algoritmanın büyüklüğü: $O(N^3)$

ÇÖZÜMSÜZ PROBLEMLER

- P4-1: İki kare matrisin toplamını bulan program
- P4-2: Kare matrisin ana köşegeninin toplamını bulan program
 - Ana köşegeni oluşturan elemanlar: a_{11} , a_{22} , a_{33} , ... a_{NN} 'dir.
- P4-3: Kare matrisin transpozisini alan program
- P4-4: Kare matrisin üst yarı toplamını bulan program
 - Ana köşegenin oluşturan elemanlar da dahil olmak üzere, ana köşegenin üzerindeki elemanların toplamının hesaplanması istenmektedir.
- P4-5: Bir kare matrisin simetrik olup olmadığını bulan program
 - Bir matrisin ana köşegene göre karşılıklı tüm elemanları birbirine eşitse, o matris simetrik.

89

EK 1. PASCAL'DA BAZI HAZIR FONKSİYONLAR

- Matematiksel:
 - $\text{Exp}(x)$ e üzeri x
 - $\text{Ln}(x)$ x'in doğal logaritması
 - $\text{Sin}(x)$ Sinüs fonk. (dev-pascal'da hatalı)
 - $\text{Cos}(x)$ Cosinüs fonk. (dev-pascal'da hatalı)
 - $\text{Arctan}(x)$ Arctanjant fonk. (dev-pascal'da hatalı)
 - $\text{Sqrt}(x)$ Karekök alma
 - $\text{Abs}(x)$ Mutlak değer alma
 - $\text{Round}(x)$ Ondalıklı sayıyı en yakın tamsayıya yuvarlama
- Karakter ve karakter katarı işlemleri:
 - $\text{Length}(s)$ String değişkenin uzunluğunu verir
 - $\text{Chr}(x)$ x tamsayısının ASCII düzeninde karakter karşılığını verir
 - $\text{Ord}(c)$ c karakterinin ASCII düzeninde tamsayı karşılığını verir
 - $\text{Val}(s,i)$ s String değerini tamsayıya çevirip i değişkenine atar
 - $\text{Str}(i, s)$ i tamsayısını String'e çevirip s değişkenine atar.

90