VERİ YAPILARI VE UYGULAMALARI

Ders Kitapları ve Yardımcı Kaynaklar

- Veri Yapıları ve Algoritmalar
  Dr. Rıfat ÇÖLKESEN,
  Papatya yayıncılık
- C & Data Structures
  P. S. Deshpande, O. G. Kakde

- Ayrıca internet üzerinden çok sayıda kaynağa ulaşabilirsiniz.

Dersin Amacı

- En çok kullanılan veri yapılarını ve bu veri yapılarının soyut özelliklerini gösterebilmek.
- Her bir veri yapısına uygun tipik algoritmaları ve bu algoritmaların performanslarını tartışmak.
- Aynı problemi farklı Veri Yapılarıyla çözmemiz durumunda performanslarının karşılaştırılması.

Dersin Gereksinimleri

- Bu dersteki öğrencilerin Nesne tabanlı programlama dillerinden birisini (Java, C++, C#) veya yordamsal programlama dillerinden birisini(C, Pascal) bildiği varsayılmıştır.
- Bilinmesi gereken konular:
  - Temel veri türleri (int, float)
  - Kontrol yapısı (if else yapısı)
  - Döngüler
  - Fonksiyonlar (Methods)
  - Giriş çıkış işlemleri
  - Basit düzeyde diziler ve sınıflar

Ders İşleme Kuralları

- Derse devam zorunludur.
- Ders başlangıç saatlerine özen gösteriniz.
- Uygulama ve ödevler ara sınav ve genel sınav notlarına etki ettirilecektir.
- Uygulamalar ve ödevler **www.sutef.gen.tr** adresi üzerinden zamanında teslim edilecektir. Verilen tarihten sonra siteye yüklenen ödevler kabul edilmeyecektir.

Ders İşleme Kuralları

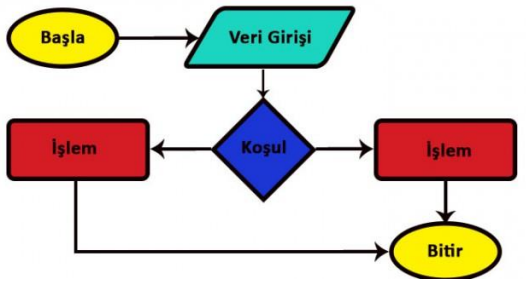
- Ödev ve/veya Projeleri birbirinizden kopya almayınız. Böyle bir durum tespit edildiğinde her iki taraf da 0 almış olur.
- Her öğrencinin sutef.gen.tr adresindeki veri yapıları dersine kayıt anahtarı ile kayıt olması gerekmektedir.
- Sutef.gen.tr üyeliğinizde profil alanına girerek e-mail, cep telefonu vb. bilgilerini **mutlaka güncelleyiniz**.

GİRİŞ ve KAVRAMLAR

- Bu bölümde veri yapıları ve uygulamaları dersine giriş yapacağız.
- Veri yapıları ve modelleri yazılım geliştirme sürecinde önemli bir yer tutar. Yazılımcı belirli bir amaca yönelik bir program tasarlarken, her şeyden önce o anda kullandığı programlama dilinin içerdiği veri yapısını ve veri modellerini kullanır.
- Program için gerekli veri yapısı yoksa onu tercih ettiği veri modeline uygun olarak kendisi tanımlar ve kullanır.

Algoritma

- Algoritma, bir problemin çözümünde izlenecek yol anlamına gelir. Algoritma, bir programlama dilinde (Java, C++, C# gibi) ifade edildiğinde program adını alır.
- Algoritma, belirli bir problemin sonucunu elde etmek için art arda uygulanacak adımları ve koşulları kesin olarak ortaya koyar. Herhangi bir giriş verisine karşılık, çıkış verisi elde edilmesi gereklidir. Bunun dışındaki durumlar algoritma değildir.



Veri

- Veri, algoritmalar tarafından işlenen en temel elemanlardır (sayısal bilgiler, metinsel bilgiler, resimler, sesler ve girdi, çıktı olarak veya ara hesaplamalarda kullanılan diğer bilgiler…).
- Bir algoritmanın etkin, anlaşılır ve doğru olabilmesi için, algoritmanın işleyeceği verilerin düzenlenmesi gerekir.

Veri Nasıl Tutulur?

- ***UInt (İşaretsiz Tam sayı-32 bit):*** ***1111573057***
- ***Raw Data(Ham veri):*** ***01000010010000010100001001000001***
- ***BCD(Kodlama):*** ***0100 0010 0100 0001 0100 0010 0100 0001***
- ***HEX(Onaltılık Taban):*** ***42 41 42 41***
- ***ASCII(Kodlama):*** ***B A B A***

Yazılım ve Program -1

- Yazılım, genel olarak, bir işin program kodları üretilerek yapıldığını belirtir; tanımı, donanım dışında kalan ve kullanıcının/programcının kodlama yaparak istenilen bir işi veya görevin yerine getirilmesi için oluşturduğu programlar/kodlar kümesidir.
- Program ise, kendi içerisinde bir bütün olan ve belirli bir işi/görevi yerine getiren algoritmik bir ifadedir. Bir program yazılımla gerçekleştirileceği gibi donanım tabanlı da tasarlanabilir. Genel olarak, bilgisayar programcıları, işin yazılım tarafıyla ilgilenirken, sistem geliştiriciler hem yazılım hem de donanım tarafıyla ilgilenirler.

Yazılım ve Program -2

- Yazılım bir veya birden çok programın bir araya gelişmesinden oluşan bir program kümesidir; özellikle, büyük ölçekli yazılımlar birçok programın ve dokümanın bir araya gelmesinden oluşur.
- Program kodu, bir işin yapılması için algoritmik ifadeyi gösteren programın herhangi bir programlama diline dayanılarak, o dilin deyimleri, fonksiyonları ve özellikleri ile elde edilmiş program parçasıdır.

Birleştirici (Assembly) dilde:

```
LDA $3F00
ADD A, #5
STA $F0A4
```

C Programlama Dilinde

```
kucuk=A[0];
  for(k=1; k<N; k++)
    if(A[k]<kucuk)
      kucuk=A[k]
```

Makine kodunda program parçası:

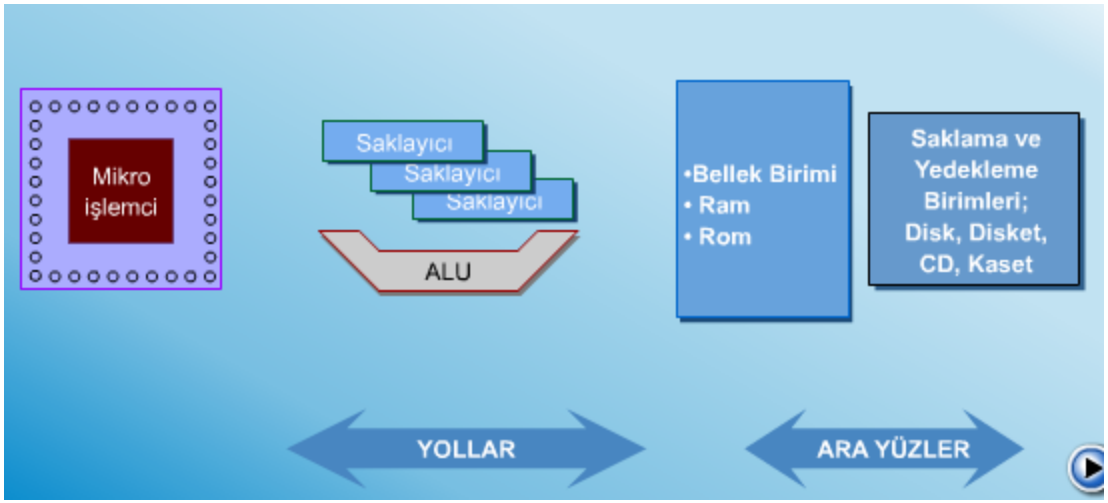
t_“ëø3À"^€>{‰ tÈè˜_Ž_sÒ¿°_‹
_PŠ_è__Xu_€|_:u_Š_èŠ_,Aƒ Ç_sÀ&¢
Ût4¯>_Ó€_v,P¾C†Æ_¥è_è'_‹÷¿__
_YX_Ç@Qè"_‹ÙY;Ãr‹Øƒ Ã_Qèc
(DOS'un command.com adlı dosyasından alınmıştır.)

Donanım ve Bellek -1

- Donanım, genel olarak, pasif veya aktif elektronik elemanlardan oluşan ve bilgisayar sistemini oluşturan işlemci, disk, ana kart, bellek vs. gibi birimlerle bunlar üzerindeki tümdevreler ve aralarındaki bağlantılara verilen adlandırmadır.
- Donanım tabanlı bir çözümde ortada fiziksel bir nesne vardır; yazılım, daha çok sanal bir kavramdır. Yazılım sonucu, ortalıkta fiziksel bir nesne yoktur; yalnızca, yazılıma ait programlar, kodlar vardır ve onlarda yine bir donanım parçası olan bellek veya saklama birimi üzerindedir. Bellek, program kodlarının ve program parametrelerinin üzerlerinde saklandığı donanım birimidir.
- Bilgisayar belleği denildiğinde, doğrudan bilgisayarın iç yollarına ve işlemcinin doğrudan adresleme yaparak erişebildiği bellek birimleri veya işlemci içerisindeki bellek birimi akla gelir.

Donanım ve Bellek -2

- Bellek hiyerarşisi, saklama birimleri dahil işlemci içerisindeki saklayıcıya kadar sayısal verinin tutulabildiği bellek ve saklama birimlerinin erişim zamanı açısından değerlendirilmesini gösteren bir hiyerarşik düzenlemedir. Bellek hiyerarşisi register-işlemci dahili bellek ana bellek-saklama birimleri şeklinde sıralanır.



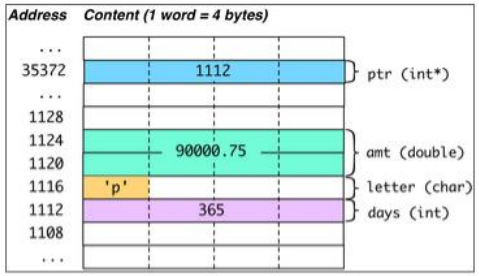
Bellek Yönetimi

1. Let's now consider the code below. Study it before moving on.

```
char letter = 'p';
int days = 365;
double amt = 90000.75;

int *ptr;       //declare pointer to an int
ptr = &days;    //point ptr at days
printf("There are %d days\n", days);
printf("There are %d days\n", *ptr);

(*ptr)--;   //decrement days by 1
printf("There are now %d days\n", days);
printf("There are now %d days\n", *ptr);

//print addresses
printf("Location of days: %p\n", &days);
printf("Location of ptr: %p\n", &ptr);
printf("Value of ptr: %p\n", ptr);
```

Show Line Numbers

2. In this simplified example, we'll assume that the operating system places `days` in bytes **1112** to **1115**, `letter` in byte **1116**, and `amt` in bytes **1120** to **1127**.



3. Here is an example output when this program is executed.

```
There are 365 days
There are 365 days
There are now 364 days
There are now 364 days
Location of days: 0x458
Location of ptr: 0x8A2C
Value of ptr: 0x458
```

Bellek Yönetimi


```
struct MyPointersArray {
  DWORD m_n;
  PVOID m_arr[1];
} object;
...
malloc( sizeof(DWORD) + 5 * sizeof(PVOID) );
...
```

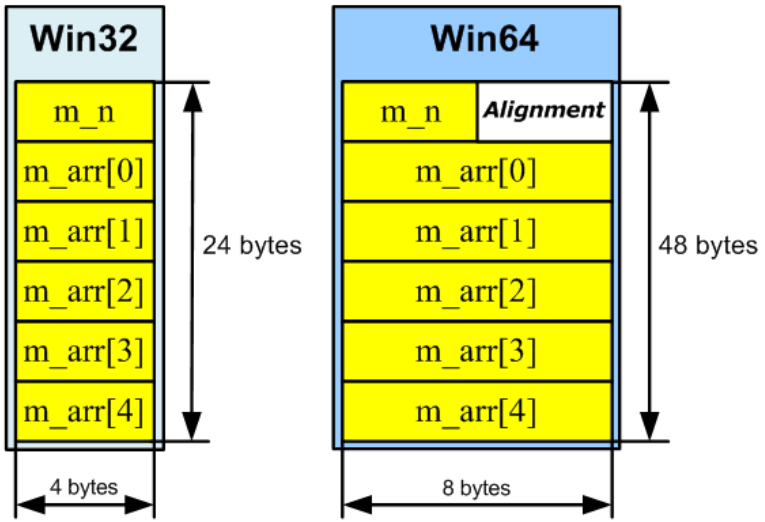


Figure 29 - Data arrangement in memory in 32-bit and 64-bit systems.

The correct calculation of the size in shown in the following:

```
struct MyPointersArray {
  DWORD m_n;
  PVOID m_arr[1];
} object;
...
malloc( FIELD_OFFSET(struct MyPointersArray, m_arr) +
        5 * sizeof(PVOID) );
...
```

Veri Yapısı ve Veri Modeli

- Veri yapısı (Data Structure) verinin veya bilginin bellekte tutulma şeklini veya düzenini gösterir.
- Veri Elemanlarını saklamayı ve veri yığını içinden çekmeyi sağlayan işlemlerin organize edildiği yapılara **Veri Yapıları** denir.
- Tüm programlama dillerinin, genel olarak, tamsayı, kesirli sayı, karakter ve sözcük saklanması için temel veri yapıları vardır. Bir program değişkeni bile basit bir veri yapısı olarak kabul edilebilir.

Veri Yapısı ve Veri Modeli

- Veri modeli (Data Model), verilerin birbirleriyle ilişkisel veya sırasal durumunu gösterir; problemin çözümü için kavramsal bir yaklaşım yöntemidir denilebilir.
- Veriyi Yönetmek amacıyla oluşturulan mantıksal sistem; bilgisayar formunda verinin gösterimini sağlayan **veri modeli**.
- Bilgisayar ortamında uygulanacak tüm matematik ve mühendislik problemleri bir veri modeline yaklaştırılarak veya yeni veri modelleri tanımlaması yapılarak çözülebilmektedir.

Veri Yapılarına Neden İhtiyaç Vardır?

- Bilgisayar yazılımları gün geçtikçe daha karmaşık bir hal almaktadır. Yazılımların programlanması ve yönetimi zorlaşmaktadır.
- Örneğin 8 milyar sayfanın indekslenmesi (Google)
- Veri yapıları, daha etkin ve daha doğru program yazmayı
- sağlar.
- İyi bir yazılım için gereksinimler:
- Temiz bir tasarım
- Kolay bakım ve yönetim
- Güvenilir
- Kolay kullanımlı
- Hızlı algoritmalar

Veri Yapılarına Neden İhtiyaç Vardır?

- Örnek
- Her biri satır başına ortalama 10 kelimeden ve yine
- ortalama 20 satırdan oluşan 3000 metin koleksiyonu
- olduğunu düşünelim.
- →600,000 kelime
- Bu metinler içinde “dünya” kelimesi ile eşleşecek
- bütün kelimeleri bulmak isteyelim
- Doğru eşleştirme için yapılacak karşılaştırmanın 1 sn.
- sürdüğünü varsayalım.
- Ne yapılmalıdır?

Veri Yapılarına Neden İhtiyaç Vardır?

- Çözüm. 1:
- Sıralı eşleştirme: 1 sn. x 600,000 kelime= 166 saat

- Çözüm. 2:
- İkili Arama (Binary searching):
- - kelimeler sıralanır
- - sadece tek yarıda arama yapılır
- toplam adım sayısı $\log_2 N = \log_2 600000$ yaklaşık 20 adım
- (çevrim) 20 sn.

- 20 saniye veya 166 saat!

Veri Yapılarının Sınıflandırılması

- Veri yapıları,
  - Temel Veri Yapıları
  - Tanımlamalı (Bileşik) Veri Yapıları
- Temel veri yapıları, daha çok programlama dilleri tarafından doğrudan değişken veya sabit bildirimi yapılarak kullanılır.
- Tanımlamalı veri yapıları, kendisinden önceki tanımlamalı veya temel veri yapıları üzerine kurulurlar;
- yani, önceden geçerli olan veri yapıları kullanılarak sonradan tanımlanırlar.

Temel Veri Yapıları

- Tüm programlama dillerinin, genel olarak, karakter, tamsayı, kesirli sayı ve sözcük (karakter katarı) saklanması için temel veri yapıları vardır. Veri yapısı, aslında, ham olarak 1 ve 0'lardan oluşan verinin yorumlanmasını belirleyen biçimleme (formating) düzenidir. Örneğin, 62 sayısının ikili tabandaki karşılığı, 111110 olarak bellekte saklanır.
- Temel veri yapıları aşağıdaki gibi sınıflanabilir:

- **Karakterler**
  - ASCII — Her karakter 8 bit (2^8 = 256 farklı karakter)
  - Unicode — Her karakter 16 bit (2^{16} = 65536 farklı karakter)
- **Tamsayılar**
  - 8 bit — short, short int, ShortInt, byte
  - 16 bit — integer, int, integer16, Int16
  - 32 bit — long, long int, LongInt, integer32, Int32
- **Ondalıklı (Gerçel) Sayılar**
  - 16 bit — half (IEEE 754-2008)
  - 32 bit — single, float (C)
  - 64 bit — double, real (Pascal)
  - 128 bit — quad

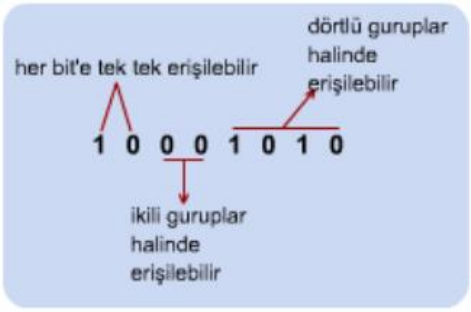
Tanımlamalı Veri Yapıları

- Tanımlamalı veri yapısı, temel veya daha önceden tanımlanmış veri yapılarının kullanılıp yeni veri yapıları oluşturulmasıdır. Üç değişik şekilde yapılabilir:
  - Topluluk (Struct) Oluşturma: Birden çok veri yapısının bir araya getirilip yeni bir veri yapısı ortaya çıkarmaktır. (Java dilinde sınıflar)
  - Ortaklık (Union) Oluşturma: Birden çok değişkenin aynı bellek alanını kullanmasını sağlayan veri yapısı tanımlamasıdır. Ortaklıkta en fazla yer işgal eden veri yapısı hangisi ise, ortaklık içerisindeki tüm değişkenler orayı paylaşır.
  - Bit Düzeyinde Erişim: Verinin her bir bit'i üzerinde diğerlerinden bağımsız olarak işlem yapılması olanağı sunar.
- Her birinin kullanım amacı farklı farklı olup uygulamaya göre bir tanesi veya hepsi bir arada kullanılabilir. Genel olarak, en çok kullanılanı topluluk oluşturmadır; böylece birden fazla veri yapısı bir araya getirilip/paketlenip yeni bir veri yapısı/türü ortaya çıkarılır.

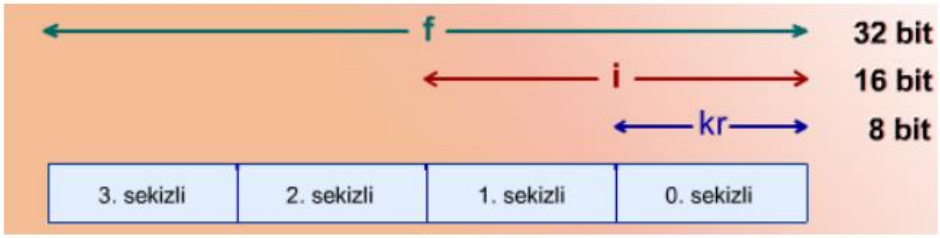
Tanımlamalı Veri Yapıları

- C dilinde tanımlamalı veri yapılarına örnek aşağıda verilmiştir.

```
struct kimlik {
   char ad[15];
   char soyad[20];
   int yas;
   char adres[50];
  };
```




```
union paylas {
   int i;
   flat f;
   char kr;
  };_
```



Veri Yapıları: Detaylı Tanım

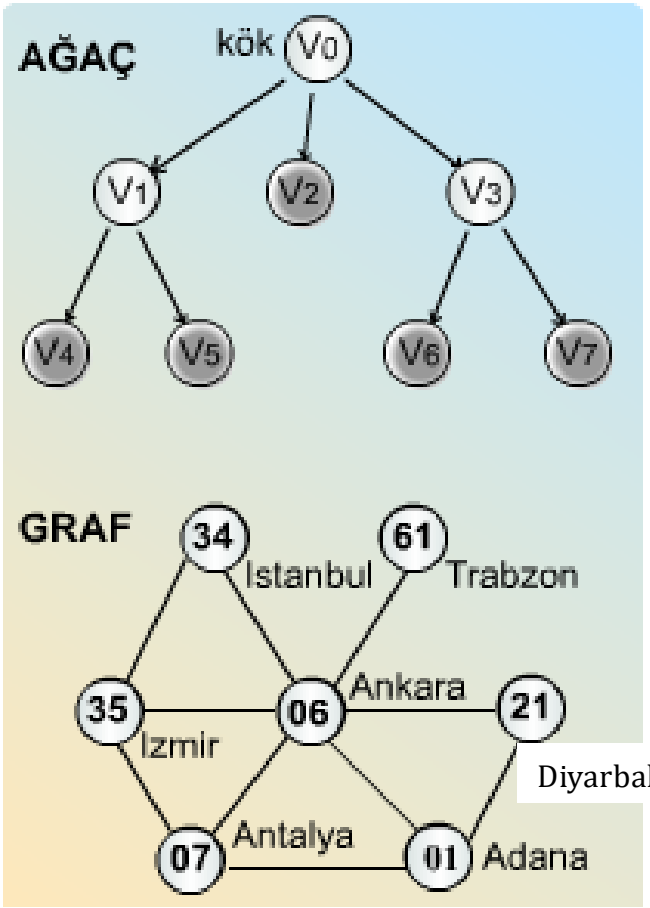
- Veri Yapıları temel veri tiplerinin(int, float, char gibi) gruplanarak nesneleri tanımlamaya yarayacak paketler oluşturmamızı sağlar.
  - Örneğin: matris satır ve sütunlardan oluşan ve bu satır ve sütunlara yerleştirilen dizi elemanlarından oluşan bir pakettir. Matrisi kullanabilmek için matrisin satır ve sütunlardan ve satır ve sütunları oluşturan dizilerden müteşekkil olduğu bilgisinin bilinmesi gerekmektedir.
- C veri yapılarını tanımlamak ve üzerinde işlemler yapabilmek amacıyla **struct** adında bir veri yapısı tanımlamıştır. C++ bu yapıyı **class** olarak geliştirmiştir.

Özet

- Veri yapısı (Data Structure) verinin veya bilginin bellekte tutulma şeklini veya düzenini gösterir.
- Tüm programlama dillerinin, genel olarak, tamsayı, kesirli sayı, karakter ve sözcük saklanması için temel veri yapıları vardır.
- Programcı bu veri yapılarını, bunların bellekte nasıl saklandığı konusuyla ilgilenmeksizin bolca kullanır.
- Tasarlanacak program, temel veri yapısı dışında farklı veri yapılarına ihtiyaç duyuyorsa, bunlar kullanılan programlama dilinin elverdiği ölçüde tanımlanmalıdır. Bu amaçla, yani temel veri yapıları dışında yeni veri yapıları tanımlanması için programlama dillerinde bir çok özellik vardır.
- Örneğin, C programlama dilinde struct ve union deyimleri kullanılarak yeni yeni veri yapıları tanımlanmaktadır.

Özet

- Veri modeli (data model), verilerin birbirleriyle ilişkisel veya sırasal durumunu gösterir; problemin çözümü için kavramsal bir yaklaşım yöntemidir denilebilir. Bilgisayar ortamında uygulanacak tüm matematik ve mühendislik problemleri bir veri modeline yaklaştırılarak veya yeni veri modelleri tanımlaması yapılarak çözülebilmektedir. Örneğin, yanda yaygın olarak kullanılan iki veri modeli, ağaç ve graf veri modeli gösterilmiştir.



PseudoCode

- Kaba-kod (PseudoCode): Yarı programlama dili kuralı & yarı konuşma dili
- İki sayıyı toplayıp sonucu gösteren algoritma

```
adım 1 – START

adım 2 – declare three integers a, b & c

adım 3 – define values of a & b

adım 4 – add values of a & b

adım 5 – store output of step 4 to c

adım 6 – print c

adım 7 – STOP
```

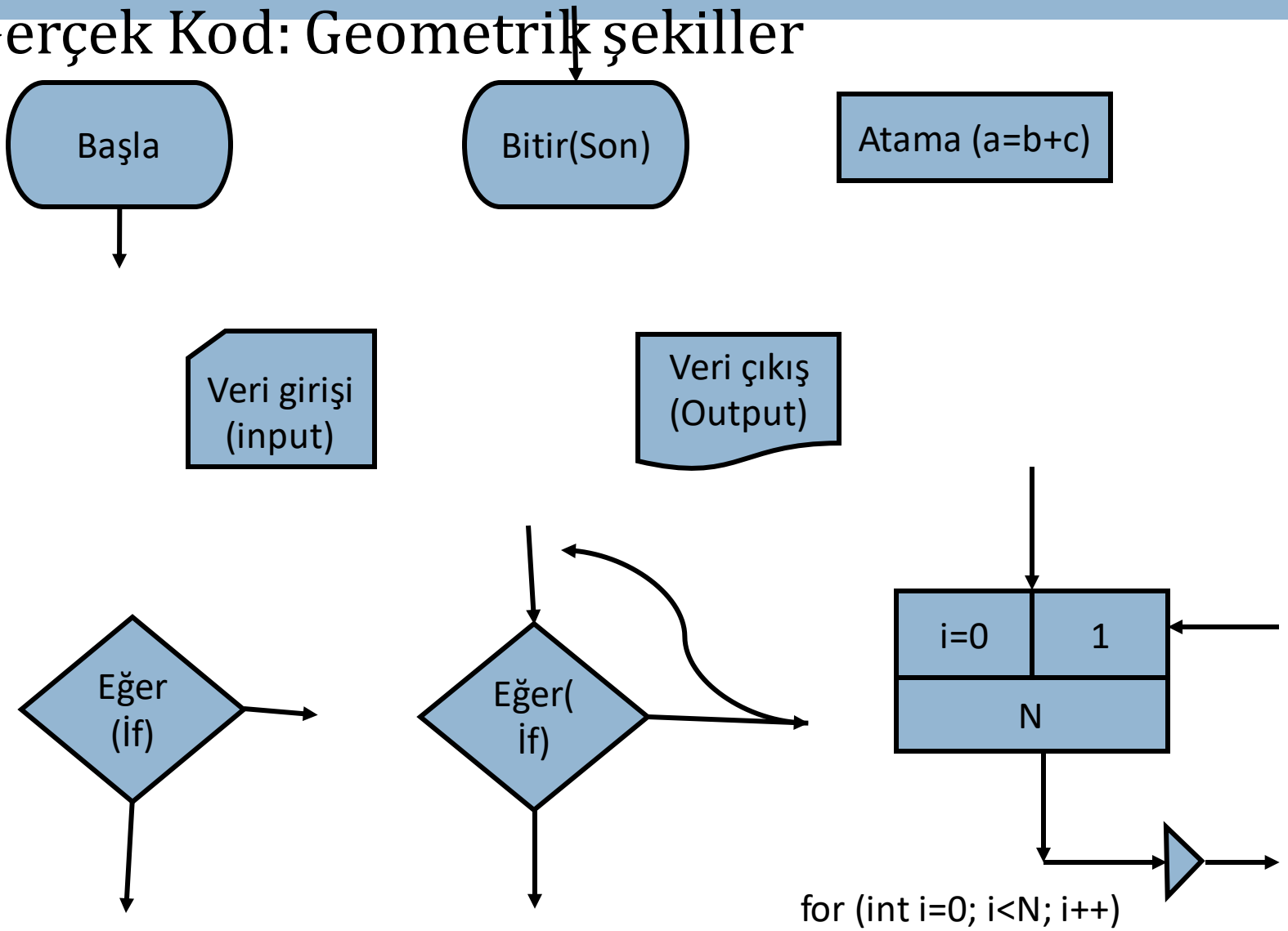
Gerçek Kod

- Gerçek Kod: Programlama Dili & Veri yapısı

```c
#include <stdio.h>

int main()
{
    int i;
    int a[10];
    printf("Enter student's scores: \n");
    for(i = 0; i < 10; i++) {
        scanf("%d", &a[i]);
    }
    printf("Your student's scores are: \n\n");
    for(i = 0; i < 10; i++) {
        printf("%d\n", a[i]);
    }
    return 0;
}
```

Akış Şeması

- Gerçek Kod: Geometrik şekiller



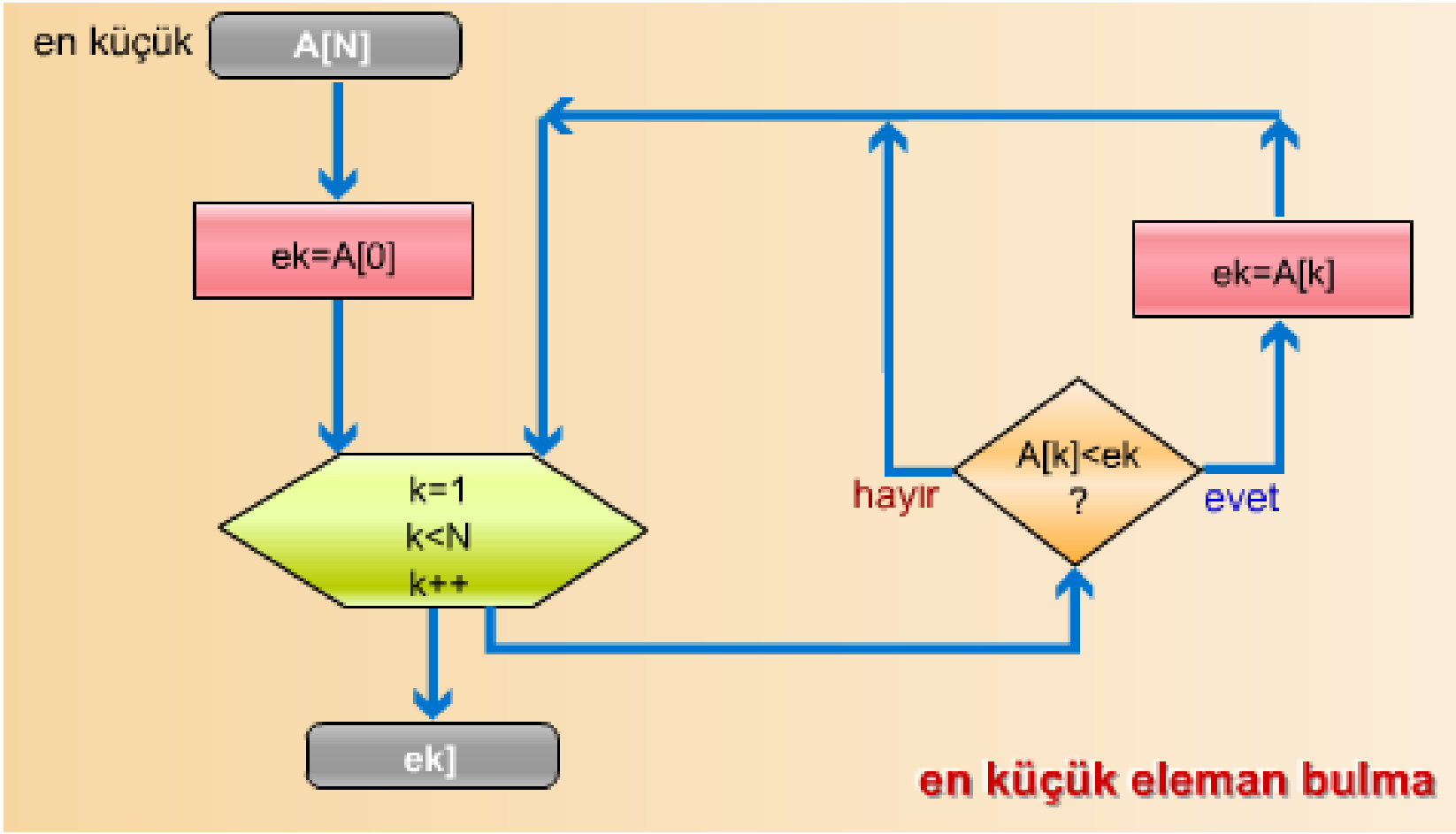
Algoritma

- Algoritma, belirli bir işi veya görevi var olan veya sonradan tanımlanan veri modeline dayandırılarak adım adım ortaya koymak ve bunu bilgisayar ortamında herhangi bir programlama diliyle kodlamaktır.
- Bir program, tasarlanması ne kadar güç görünse de, gerekli veri modeli ve yapısı belirlenmişse ve algoritmik ifadesi ortaya koyulmuşsa, kolayca kodlanabilir; böylelikle programı tasarlamak ve geliştirmek oldukça kolaylaşır.

Algoritma-2

- Program tasarlanırken, kullanılacak veri yapısı ve algoritma doğrudan uygulamaya bağlıdır. Bazı uygulamalar için programlama dilinin hâlihazırda sahip olduğu veri modelleri yeterliyken, bazı uygulamalar için programcı, var olan veri modellerini de kullanarak yeni yeni veri modelleri tanımlar ve programın algoritmik ifadesini bunlara dayanarak geliştirir.
- Algoritma, belirli bir problemin sonucunu elde etmek için art arda uygulanacak adımları ve koşulları kesin olarak ortaya koyar. Bu adımlar, ilgili koşullar altında adım adım izlendiğinde bir sonuca ulaşılır. Bilgisayar uygulamasında, bir yazılım geliştirirken birçok algoritmaya ihtiyaç duyulur. Örneğin,
  - arama algoritması
  - sıralama algoritması
  - matris veya vektörel işlem algoritması
  - graf algoritması
  - bir matematiksel modelin çözülmesi algoritması

Algoritma-3


en küçük
A[N]
ek=A[0]
k=1
k<N
k++
ek]
A[k]<ek
?
hayır
evet
ek=A[k]
en küçük eleman bulma

Program Çalışma Hızı ve Bellek Gereksinimi

- Programın çalışma hızı karmaşıklıkla ifade edilir; bu kavram zaman birimiyle ifade edilmeyip doğrudan işlem adedi veya döngü sayısıyla ifade edilir.
- Çünkü, programın çalışma hızında zaman miktarı programın üzerinde koştuğu donanıma çok bağlıdır; dolayısıyla algoritmaları birbiriyle karşılaştırmak için zaman miktarını kullanmak gerçekçi olmayıp yanılgılara neden olmaktadır.
- Bunun yerine, ilgili algoritmanın bilgisayar donanımından bağımsız olarak kaç adet işlem veya döngüyle gerçekleştirilebileceği hesaplanır.
- Algoritma karmaşıklığı iki açıdan ele alınır. Biri zaman karmaşıklığı, diğeri alan (veya bellek) karmaşıklığıdır. Zaman karmaşıklığı (time complexity) algoritmanın sonuca ulaşması için gerekli zaman hakkında bilgi verir. Alan karmaşıklığı (space complexity) ise algoritmanın ihtiyaç duyacağı bellek miktarı hakkında bilgi verir.
-

Program Çalışma Hızı ve Bellek Gereksinimi

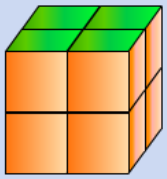
- Algoritma karmaşıklığı iki şekilde ifade edilebilir. Biri doğrudan parametrelere bağlı olarak tam matematiksel ifadeyle; ikincisiyse, parametrelerin karmaşıklığı nasıl etkilediğini mertebe şeklinde gösterir; Mertebesi göstermek için O (büyük o), Ω gibi birçok asimtotik notasyon kullanılır; genel olarak O notasyonunu kullanmak yaygındır. Örneğin iki matrisin çarpılması için gerekli zaman karmaşıklığı $O(n^{2,81})$ olarak hesaplanmışsa bu algoritma zaman karmaşıklığı $O(n^3)$'den daha iyidir.
- Bir program iki açıdan belleğe ihtiyaç duyar; biri kodun tutulması diğeri de kodun çalışması için üzerinde işlem yapacağı verilerin tutulması için.

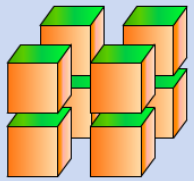
Veritabanı ve SQL

- Veritabanı, bilgilerin belirli bir disiplin altında saklanması ve gerektiğinde hızlı bir şekilde aranıp bulunması/sorgulanması için geliştirilmiş bir saklama/sorgulama sistemidir. SQL (Structured Query Language) ise bir sorgulama dilidir; veritabanı ile etkileşimin kotarılması, veritabanı veri yapısının tanımlanması, veritabanına veri yazma/sorgulama işlemlerinin yapılmasını sağlar.

Böl ve Yönet (Divide-and-Conquer) Yaklaşımı

- Böl ve yönet yaklaşımı bilgisayar biliminde önemli bir yere sahiptir; büyükçe denilebilecek problemler, özellikleri aynı kalmak koşuluyla, daha küçük parçalara ayrılıp küçük problemler haline getirilir ve kolayca çözülür. Örneğin elemanları rastgele yerleştirilmiş elemanlı bir dizi, bölüne bölüne iki elemanlı hale getirilirse sıralanması kolayca yapılabilir: önce ikiye bölünür, dolayısıyla elemanlı iki parça elde edilir; parçalar yeteri kadar küçülmemişse yeniden iki bölünür ve büyüklükte dört parça edilir; yeteri kadar küçülmemişse bölme işlemi tekrarlanarak yeteri kadar küçük hale getirilir.
- Birçok problem doğası gereği bölünmeye çok yatkındır; dolayısıyla böylesi problemlerin çözümü böl ve yönet yaklaşımına çok uygun düşer. Örneğin sıralama/arama algoritmalarında, ağaç veri modelinde, bazı matris işlemlerinde böl ve yönet algoritması çözümün görülmesini/sağlanmasını kolaylaştırır.





Kıyaslama (Benchmarking)

- Kıyaslama, aynı işi yapan iki programın veya yazılımın evrensel anlamda örnek veriler üzerinde çalıştırılarak başarımlarını karşılaştırmaktır; kıyaslama donanım için de yapılabilir.
- Örneğin iki tane bilgisayara, önceden tanımlanmış işler belirli sayıda yaptırılır ve bir başarım sayısı elde edilir; bu iki sayı kıyaslanarak daha büyük olanın daha başarımlı olduğu belirlenir.

Kaynaklar

- Veri Yapıları ve Algoritmalar, Dr. Rıfat ÇÖLKESEN, Papatya yayıncılık
- Ahmet Yesevi Üniversitesi Ders Notları
- http://www.kaanaslan.com/resource/article/display_article.php?id=94
- C & Data Structures, P. S. Deshpande, O. G. Kakde
- C/C++ ile Veri Yapıları ve Çözümlü Uygulamalar, Prof. Dr. Nejat Yumuşak, Muhammed Fatih Adak