

KOD ÖRNEKLERİ (SCHEME)

İçindekiler

SORU 1: object array (nesne dizisi)

SORU 2: Swing Kütüphanesi

SORU 3: İstanbul Ünı. Nesne Yönelik Programlama Vize Çözümleri

SORU 4: Dosyayı Tersten Basan Kod

SORU 5: Nesne Yönelimli Programlama Dersi Quiz Çözümlü

SORU 6: Dosya ve Bağlı Liste Uygulaması

SORU 7: 2 Boyutlu Dizi İçerisine Sarmal Seri Giriş

SORU 8: Nesne Yönelimli olarak Tic Tac Toe Oyunu

SORU 9: Visual Basic ile Gösterici (Pointer) Kullanımı

SORU 10: C ile Programlamaya Giriş Quiz Soruları ve Çözümleri

SORU 11: JAVA ile Zar uygulaması

SORU 12: Ekrana Kare Çizdiren Kod

SORU 13: Matrisin tersinin alınması (Mantrix Inverse)

SORU 14: Dizgi (String)

SORU 1: object array (nesne dizisi)

Bu yazının amacı, nesne yönelimli programlama konusu altında, sıklıkla geçen nesne dizilerini (object array) anlatmaktır. Kısaca bir dizinin elemanlarına nesne atamanın nasıl olacağı anlatılacaktır.

Öncelikle nesne yönelimli programlama dillerinde, kullanılan [nesne atıflarının \(object referrer\)](#) birer [gösterici \(pointer\)](#) olduklarını bilmemiz gerekir.

Örnek olarak bir sınıf tanımı ile başlayalım:

```
class insan{
    int yas;
    String isim;
}
```

Şimdi yukarıdaki sınıftan üretilen bir nesneyi ele alalım:

```
insan ali = new insan();
```

Burada ali ismi ile geçen [değişken \(variable\)](#) bir [nesne atıfıdır \(object referrer\)](#) ve insan sınıfından yeni bir nesne üretilmiş, ve bu nesne gösterilmiştir.

Ayrıca JAVA dilindeki dizi tanımlamasını da hatırlatalım:

```
int x[] = new int[5];
```

Yukarıdaki tanım ile, bir int dizisi tanımlanmıştır. Bu dizinin kendisi de aslında bir nesnedir ve kendisine özgü bazı fonksiyon ve özellikleri bulunur. Ancak konumuz kapsamında 5 elemanlı (0'dan başlayarak 4'e kadar giden (0 ve 4 dahil)) elemanları olduğunu ve int tipinde tanımlandığı için her elemanına birer tam sayı konulabilen (int) bir [dizi \(array\)](#) olduğunu söylememiz yeterlidir.

Ardın hatırladığımız dizi tanımı ile nesne tanımını birleştirebiliriz:

```
insan x[] = new insan[5];
```

Elemanlarının her biri birer nesne atfı olan bir dizi tanımlanmıştır. Buna göre dizinin her elemanına, insan tipinden bir nesne göstericisi konulabilir.

Ancak nesne göstericileri için geçerli olan kurallar burada da geçerlidir. Örneğin yukarıdaki tanımdan hemen sonra aşağıdaki gibi bir kullanım hata verecektir:

```
x[2].yas = 20;
```

Buradaki hatanın sebebi henüz üretilmemiş olan bir nesnenin bir özelliğine erişiliyor olmasıdır. x[2] ile gösterilen dizinin RAM'de henüz kapladığı yer belirtilmemiş ve dolayısıyla bu alana erişime izin verilmemiştir. Çözüm olarak önce new komutu ile hafızada bu nesnenin yaşayacağı alan tanıtılmalıdır.

```
x[2] = new insan();
```

Ancak yukarıdaki şekilde bir tanımlamadan sonra bir önceki adımda hata veren satırı yazıp çalıştırabiliriz.

JAVA dilinde ayrıca for dizilerinin özel bir kullanımı aşağıdaki şekildedir:

```
for(insan i : x)
```

Yukarıdaki kod satırı, x dizisinin eleman sayısı kadar döner. Örneğin dizimiz 10 elemanlı ise, döngü 0'dan 9'a kadar dönecek ve her dönüşünde ilgili eleman değerini i isimli nesne göstericisine atayacaktır.

Yukarıda anlatılan bilgiler ışığında, derste yazdığım bir kodu aşağıda yayınlıyorum:

```
class insan{
    int yas;
    String isim;
    public String toString(){
        return "yas: "+yas+" isim: "+isim;
    }
}
```

Yukarıdaki kodda, bir insan sınıfı tanımlanmıştır. Bu tanıma göre insanın yaşı ve ismi bulunmaktadır. Ayrıca toString ismi verilen özel bir fonksiyon yazılmıştır. Bu fonksiyonun amacı, bu [sınıftan \(class\)](#) üretilen bir [nesnenin \(object\)](#), [dizgiye \(string\)](#) çevrilmesi durumunda otomatik olarak çalışıyor olmasıdır. Aslında bu fonksiyon, her [nesne \(object\)](#) için otomatik olarak tanımlıdır. Bunun sebebi java.lang.Object sınıfında tanımlı bir fonksiyon olmasıdır ve bu fonksiyon her sınıf tanımında [ezilebilir \(override\)](#). Örnek kodumuzda da üzerine yükleme (override) işlemi yapılmış ve fonksiyon ezilmiştir. Böylece insan sınıfından bir nesne, dizgi olarak kullanıldığında fonksiyonda belirtildiği üzere “yas: “ yazısına insanın yaşı eklenecek ardından “isim:” yazısının ardında insanın ismi eklenerek döndürülecektir.

Test amacıyla yazılmış örnek bir kod aşağıda verilmiştir:

```
public static void main(String args[]){
    insan i[];
    i = new insan[5];
    insan ali = new insan();
    ali.e = x;
    ali.yas = 20;
    ali.isim= "ali yilmaz";
    System.out.println(" alinin adresi : "+
ali.e.adres);

    i[0] = ali;
    i[1] = new insan();
    i[1].yas = 30;
    i[1].isim = "veli demir";
    System.out.println("i[0]:" +i[0] );
    i[2] = new insan();
    i[3] = new insan();
    i[4] = new insan();
    i[2].isim = "ahmet yildiz";
    i[3].isim = "veli yilmaz";
    i[4].isim = "cem yildiz";
    for(int j=0;j<i.length;j++){
        i[j].yas =10+ (int)( Math.random() * 40);
    }
    for(insan in : i){
        System.out.println(in.toString());
    }
}
```

Yukarıdaki kodda örnek olarak 5 elemanlı bir insan dizisi oluşturulmuştur. Bu diziye önce ali isimli bir nesne göstericisinin içerisine atanan nesne yerleştirilmiş (dizinin 0. sırasına) ardından dizinin 1. elemanı için yeni bir nesne oluşturulmuştur. Son olarak dizinin 3., 4. ve 5.

elemanları için arka arkaya nesne oluşturulmuş (hafızada yer açılmış) ve isim ataması yapılmıştır. Kodun bundan sonraki kısmında bir döngü ile dizideki bütün insanlara rast gele yaş ataması yapılmıştır (random fonksiyonu) ve ikinci bir döngü ile insanlar ekrana bastırılmıştır. Buradaki bastırma işlemi sırasında insanlar daha önceden tanımlı olan toString fonksiyonu marifetiyle önce [dizgiye \(string\)](#) çevrilmiş ve ardından ekrana bastırılmıştır.

SORU 2: Swing Kütüphanesi

Bu yazının amacı, JAVA dilinde bulunan ve grafiksel kullanıcı ara yüzü geliştirmekte kullanılan SWING kütüphanesine hızlı bir başlangıç yapmaktır.

Öncelikle Swing kütüphanesinin gelişiminden hızlıca bahsedelim. Swing kütüphanesi JAVA diline JDK 1.1 sürümünden sonra eklenmiştir. Daha önce AWT (Active Window Toolkit) ismi verilen bir kütüphane kullanmaktaydık ancak çok daha fazla ve gelişmiş özelliklerle gelen swing kütüphanesi neredeyse tamamen AWT yerine geçti denilebilir. AWT halen kullanılmakta, bunun en büyük sebebi JAVA'nın Visual Studio gibi geliştirme ortamlarından farklı olarak geçmişle uyumlu olmasıdır (back compatibility). Yani, örneğin 1997 yılında yazdığım bir kodu hala derleyip çalıştırabiliyorum.

AWT'nin bulunmasındaki diğer önemli bir faktör ise, Swing kütüphanesinin AWT üzerinde geliştirilmiş olmasıdır. AWT'den bu kadar bahsettikten sonra son olarak aynı projede hem AWT hem de Swing kullanmamanız gerektiğini, aksi halde ağır ve hafif bileşenlerin ekranda hatalı görüleceğini (heavy weighted, light weighted) söyleyerek Swing kütüphanesini anlatmaya başlayalım.

Swing kütüphanesi, javax altında bulunmaktadır ve bütün swing sınıfları (classes) javax.swing paketi ile başlar.

Örneğin, ekrana basit bir düğme koyarak uygulama geliştirmeye başlayalım.

```
1 import javax.swing.JButton;
2 import javax.swing.JFrame;
3 public class swingbuton extends JFrame{
4     public swingbuton(){
5         this.add(new JButton("tikla"));
6     }
7     public static void main(String args[]){
8         swingbuton sb = new swingbuton();
9         sb.pack();
10        sb.setVisible(true);
11
12    }
13 }
```

Yukarıdaki kodun çıktısı, aşağıdaki şekilde bir pencereye yerleştirilmiş bir düğmeden (button) ibarettir:

```
shedai@pardus201 $ javac swingbuton.java
shedai@pardus201 $ java swingbuton
tikla
```

Kodumuzda kullandığımız iki swing bileşeni olan JFrame ve JButton import edildikten sonra sınıfın bir JFrame olacağını, extends JFrame yazıp JFrame sınıfını kalıtımla (inheritance) olarak belirtiyoruz.

Kodumuzda bulunan iki metottan ilki bir inşa fonksiyonu olup (constructor), this.add fonksiyon çağırılması ile JFrame içinde tanımlı olan add fonksiyonuna bir adet JButton parametre verilmiştir.

Klasik olarak swing kütüphanesinde, bir bileşenin diğer bileşenleri içinde barındırma özelliği bulunuyorsa (container) bu durumda add fonksiyonu, parametre olarak aldığı bileşeni, barındırıcı bileşene eklemektedir.

Kodun 5. satırında yapılan bu ekleme işleminde dikkat edilecek bir husus da, JButton bileşenini oluşturan ve new ile çağrılan yapıcı fonksiyonun (constructor) parametre olarak bir dizgi (string) almış olmasıdır. Bu dizgi, düğmenin üzerinde görülecek olan yazıdır.

Son olarak main fonksiyonunda, yeni tanımladığımız bu sınıftan bir nesne (object) oluşturarak bu objenin referansına sb ismini veriyoruz. Kodun 9. ve 10. satırlarında ise sırasıyla sb isimli bu nesneyi (ki artık JFrame özelliği taşıyan ve içerisinde bir buton olan nesnedir) paketliyoruz ve gösteriyoruz. Paketleme işlemini basitçe, bir pencerenin içerisindeki bileşenlerin sığacağı en ufak hali alması olarak düşünebilirsiniz. Gösterme işlemi ise bir bileşenin görünürlülüğünü (visibility) doğru yapmaktan ibarettir (true). Tersisi şekilde görünürlülüğü yanlış (false) yapılarak bir bileşen gizlenebilir. Yine daha önceki sürümlerde bu işlem için sırasıyla show() ve hide() fonksiyonları da kullanılmaktaydı ancak JDK 1.3 sonrasında bu fonksiyonlar kaldırıldı.

Şimdi gelelim bu düğmenin tıklanması halinde eylemi algılamaya.

Yukarıdaki kodu yazıp denediyseniz tıklama eyleminin herhangi bir sonuç doğurmadığını görmüşsünüzdür. Bunun sebebi JAVA dilinde tanımlı olan herhangi bir eylemin sınıf içerisinde yer alamamasıdır.

JAVA dilinde eylemi üreten ve yakalayan nesneler farklı olabilir. Örneğin yukarıdaki örnekte düğmeyi tanımladığımız swingbuton isimli sınıf düğmeye tıklandığında devreye girebileceği gibi tamamen başka bir sınıf da devreye sokulabilir. Biz basit olsun diye ilk örneğimizde yine aynı sınıfta eylemi yakalayacağız. Bunun için öncelikle sınıfa bu özelliğin kazandırılması gerekir. JAVA dilinde bir sınıfın herhangi bir bileşen ile ilgili bir eylem yakalayabilmesi için ActionListener olması gerekir.

```
public class swingbuton extends JFrame implements ActionListener
```

şeklinde sınıfın tanımını değiştirerek bu sorunu çözebiliriz. Ancak, arayüzlerden (interface) hatırlanacağı üzere bir arayüzü (interface) sınıfa kazandırmanın (implements), bağlayıcı bir sonucu olmaktadır.

```
1  import javax.swing.JButton;
2  import javax.swing.JFrame;
3  import java.awt.event.*;
4  public class swingbuton extends JFrame implements ActionListener
5  {
6      JButton dugme = new JButton("tikla");
7      this.add(dugme);
8      dugme.addActionListener(this);
9  }
10 public void actionPerformed(ActionEvent e){
11     System.out.println("butona basıldı");
12 }
13 public static void main(String args[]){
14     swingbuton sb = new swingbuton();
15     sb.pack();
16     sb.setVisible(true);
17 }
18 }
19 }
```

Kodumuzda yeni olan en büyük fark, sınıfta artık ActionListener arayüzünün implement edilmesidir. Bu arayüzün sınıfa uygulanması ile birlikte, sınıfta "actionPerformed" isimli fonksiyonu bulundurma zorunluluğu ortaya çıkar. Nitekim yukarıdaki kodun 10-12. satırları arasında bu fonksiyon kodlanmış ve herhangi bir eylem olması halinde çalışacak olan bu fonksiyonun içerisinde ekrana "butona basıldı" mesajının yazılması sağlanmıştır.

Diğer bir ilave ise artık düğmenin bir eylem dinleyicisi olmasına olan ihtiyaçtır. Düğmemize basıldığında hangi sınıfın bu eylemi yakalacağını kodda belirtmemiz gerekir ki bu durum da 8. satırda kodlanmıştır.

Yukarıdaki kodun ekran çıktısı aşağıdaki şekildedir:

```
shedai@pardus2011 Belgeler $ javac swingbuton.java
shedai@pardus2011 Belgeler $ java swingbuton
butona basıldı
butona basıldı
butona basıldı
```

Birden fazla düğme bulunması halinde düğmeleri ayırt etmek için eylemin özel olarak koşullandırılması gerekir.


```

1  import javax.swing.JButton;
2  import javax.swing.JFrame;
3  import java.awt.*;
4  import java.awt.event.*;
5  public class swingbuton extends JFrame implements ActionListener
6      JButton dugme, dugme2;
7  public swingbuton(){
8      dugme = new JButton("tikla");
9      dugme2 = new JButton("ikinci dugme");
10     this.setLayout(new FlowLayout());
11     this.add(dugme);
12     this.add(dugme2);
13     dugme.addActionListener(this);
14     dugme2.addActionListener(this);
15 }//www.bilgisayarkavramlari.com
16 public void actionPerformed(ActionEvent e){
17     if(e.getSource()==dugme){
18         System.out.println("butona basıldı");
19     }
20     if(e.getSource()==dugme2){
21         System.out.println("ikinci dugmeye basıldı");
22     }
23 }
24 public static void main(String args[]){
25     swingbuton sb = new swingbuton();
26     sb.setVisible(true);
27
28 }
29 }

```

Kodun yeni halinde, düğmelerin tanımlandığı satırı sınıf geçerlilik alanına alarak actionPerformed fonksiyonu içerisinde hangi düğmeye tıklandığını kontrol edebiliyoruz. Bunun için parametre olarak gelen ActionEvent tipindeki e nesnesinin kaynağını okutan .getSource metodunu çağırıyor ve hangi nesneye eşit olduğunu sorguluyoruz.

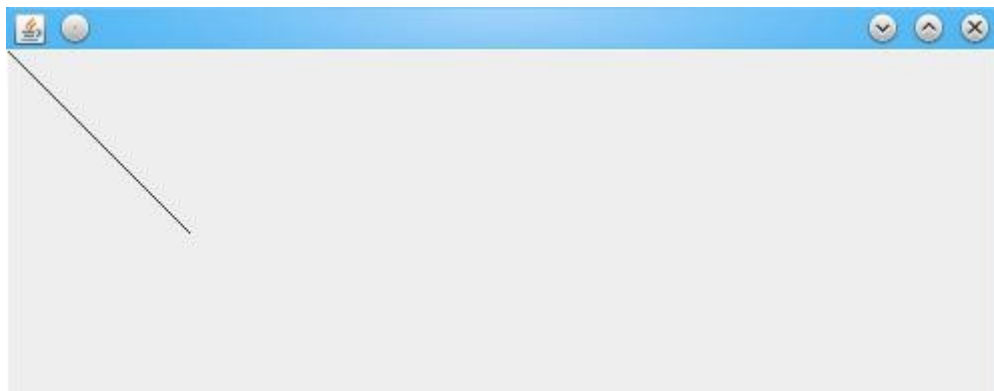
Ayrıca kodda yapılan yeni bir ilave, birden fazla bileşeni aynı anda ekranda gösterebilmek için bir yayılım (Layout) eklenmesidir. Yukarıdaki kodda, 10. satırda bulunan komut, this ile ifade edilen JFrame nesnesinin yayılımının FlowLayout tipinde olacağını belirtmektedir. Bunun anlamı eklenen her bileşen, bir öncekinin sağına ilave edilecek, nihayet ekranda daha fazla sağa eklenecek yer kalmayınca aşağıya geçileceğidir.



Swing kütüphanesi kullanılarak mevcut bileşenler, yukarıda gösterildiği şekilde ekrana eklenip kullanılabileceği gibi, bileşenler üzerinde değişiklik yapılması da mümkündür. Örneğin bir swing bileşeni olan Jpanel üzerinde çizim yapmaya çalışalım:

```
1
2 import javax.swing.*;
3 import java.awt.*;
4 public class cizim extends JPanel{
5     //www.bilgisayarkavramlari.com
6     public void paintComponent(Graphics g){
7         g.drawLine(0,0,90,90);
8     }
9     public static void main(String args[]){
10        JFrame jf = new JFrame();
11        jf.add(new cizim());
12
13
14        jf.setSize(500,500);
15        jf.setVisible(true);
16    }
17 }
```

Yukarıdaki kodda bulunan sınıf, Jpanel bileşenini miras almış ve bu bileşenin “paintComponent” isimli metodunun üzerine ezmiştir (override). Neticede Jpanel jf isimli JFrame’e eklenmiş ancak yeni çizim özellikleri ile görüntülenmiştir. Kodun ekran çıktısı aşağıdaki şekildedir:



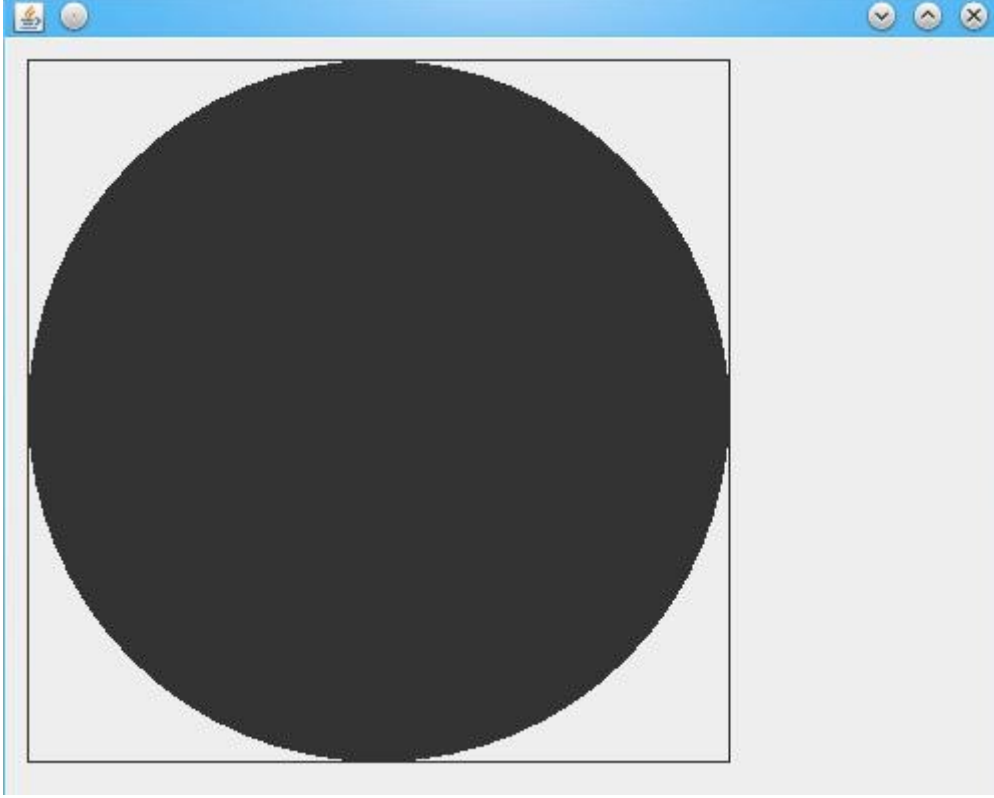
paintComponet fonksiyonu yukarıda gösterildiği şekilde ezilerek (override) Graphics kütüphanesinden istenilen bir nesne, yukarıda gösterildiği şekilde ekrana eklenebilir.

Ayrıca JAVA'nın kuvvetli kütüphanelerinde birisi de Graphics2D kütüphanesidir. 2 boyutlu şekillerin bulunduğu kütüphane olarak tanımlayabileceğimiz bu kütüphane ile görsel pek çok işlem yapılabilir. Bu kütüphaneyi kullanmak için örnek kod aşağıda verilmiştir:

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.geom.*;
4  //www.bilgisayarkavramlari.com
5  public class cizim2d extends JPanel {
6      private Ellipse2D.Double circle =
7          new Ellipse2D.Double(10, 10, 350, 350);
8      private Rectangle2D.Double square =
9          new Rectangle2D.Double(10, 10, 350, 350);
10
11     public void paintComponent(Graphics g) {
12         clear(g);
13         Graphics2D g2d = (Graphics2D)g;
14         g2d.fill(circle);
15         g2d.draw(square);
16     }
17     protected void clear(Graphics g) {
18         super.paintComponent(g);
19     }
20     protected Ellipse2D.Double getCircle() {
21         return(circle);
22     }
23     public static void main(String args[]){
24         JFrame jf = new JFrame();
25         jf.add(new cizim2d());
26         jf.setSize(500,500);
27         jf.setVisible(true);
28     }
29 }
```

Yukarıdaki yeni kodda, paintComponent fonksiyonu altında, Graphics2D sınıfından türetilmiş bir nesne olan g2d nesnesine, şimdiye kadar kullandığımız ve klasik Graphics kütüphanesinin bir nesnesi olan g nesnesini tip inkılabı ile (type casting) atıyoruz. Bu sayede g2d nesnesinin fonksiyonlarını kullanabiliyoruz.

Kodda, dikdörtgen ve daire şeklinde iki nesne ekrana eklenerek gösterilmiştir. Kodun çıktısı aşağıda verilmiştir:



Aynı ekranda hem çizim hem de bileşen (component) kullanan kod (Ahmet beyin sorusu üzerine ekliyorum):

Sorunumuz, aynı ekranda hem çizim yapabilmek, yani Graphics kütüphanesini kullanmak, hem de java'nın bileşenlerinin (örneğin düğme (buton) veya yazı alanı (text areat) gibi) kullanabilmek.

Bunu yapan örnek bir kodu aşağıdaki şekilde kodlayabiliriz:

```

2  import javax.swing.*;
3  import java.awt.*;
4  class cizim extends JPanel{
5      public void paintComponent(Graphics g){
6          super.paintComponent(g);
7          g.drawLine(10,10,80,80);
8      }
9  }
10 public class ekran{
11     //www.bilgisayarkavramlari.com
12     public static void main(String args[]){
13         JFrame x = new JFrame();
14         cizim c = new cizim();
15         x.setLayout(new GridLayout(2,2));
16         x.add(c);
17         x.add(new cizim());
18         x.add(new JButton("deneme"));
19         x.add(new JTextArea("www.bilgisayarkavramlari.com"));
20         x.setSize(400,200);
21         x.setVisible(true);
22     }
23 }

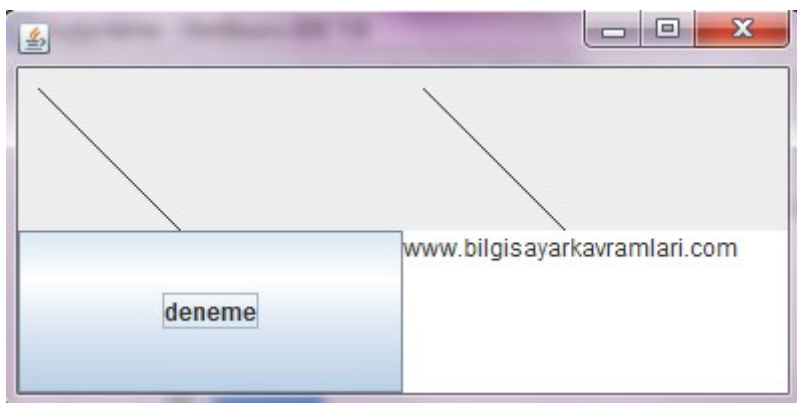
```

Kodda, öncelikle bir JPanel sınıfı oluşturuyoruz. Basitçe JPanel miras alan bir sınıfımız var, ismi cizim ve bu sınıf içerisinde, yukarıda da anlatılan paintComponent metodunun üzerine ezere (method overriding) Graphics kütüphanesini kullanıyoruz (örnek olarak bir çizgi çizdirilmiştir).

Ardından main fonksiyonumuzu içeren bir sınıf kodlanmıştır. Bu sınıf içerisinde x nesne atfı (object referrer) ile gösterilen bir JFrame oluşturulmuş ve bileşen yayılımı (layout) GridLayout olarak ayarlanmıştır. Kabaca ekran 2×2 boyutlarında dört parçaya bölünmüştür. Javada bu yayılıma ekleme yapıldığında sol üst köşeden başlanarak bileşenler eklenir ve ilk satır bitince aşağıya geçilir.

Buna göre sırasıyla eklediğimiz bileşenler, c ismindeki ve cizim sınıfından türetilmiş bir JPanel, satır içinde (inline) olarak türetilmiş bir cizim nesnesi, üzerinde “deneme” yazan bir düğme (buton) ve içinde “www.bilgisayarkavramlari.com” yazılı bir yazı alanı (JTextArea).

Ekranın çıktısı aşağıdaki şekildedir:



Görüldüğü üzere, ekranda çizim içerikli iki panel ve iki adet swing kütüphanesinden bileşen eklenmiştir.

SORU 3: İstanbul Ün. Nesne Yönelik Programlama Vize Çözümleri

■ Bir masa lambasının 4 farklı durumu bulunabilmektedir. Kapalı, az ışık, orta ışık ve çok ışık seviyelerinde yanabilen bu lambayı nesne yönelimli olarak programlayınız. Herhangi bir insanın bu lambanın durumunu değiştirebilmesi ve lambanın mevcut durumunu görebilmesini sağlayınız. (JAVA dilinde sadece gerekli sınıfları yazınız) (10 puan)

Masa lambasında durumu tutan bir değişkene ve bu durumu değiştiren bir metoda ihtiyaç duyulmaktadır. 4 farklı durum olduğu için en uygun değişken int tipidir.

```
1 package javaapplication14;
2 class masaLambasi{
3     private int durum;
4     // kapsülleme gereği private tanımlıdır.
5     public masaLambasi() { }
6     // herhangi bir yapıcı fonksiyon (consturctor) tanımlandığında
7     // bulunması zorunlu default constructor
8     public masaLambasi(int durum) { this.durum = durum; }
9     // dışarıdan başlangıç durumunu belirleyen yapıcı fonksiyon
10    public void yeniDurum(int durum) { this.durum = durum; }
11    // dışarıdan durumu değiştirmeye yarayan fonksiyon
12    public int durumuBildir () { return durum; }
13    // dışarıdan soran kişiye durumu bildirir.
14 }//www.bilgisayarkavramlari.com
15 class insan{
16     masaLambasi m;
17     public insan(){
18         m = new masaLambasi();
19     }
20 }
21 public class test{
22     public static void main(String args[]){
23         insan ali = new insan();
24         ali.m.yeniDurum(2);
25         System.out.println(ali.m.durumuBildir());
26     }
27 }
```

Yukarıdaki kod doğru olmakla birlikte, insan sınıfında yapıcı fonksiyon içerisinde masa lambası tanımlanmaması kabul edilebilir (insanın masa lambası olabilir ama her insanın masa lambası olmak zorunda değildir). Veya durumu döndüren fonksiyon (durumuBildir) değer döndürmek yerine ekrana durumu basabilir, veya String tipinde kapalı, az, orta ışık, çok ışık şeklinde değerler döndürebilir. Benzer şekilde yeniDurum fonksiyonu parametre olarak int yerine String tipinde değer alabilir. Bu açıklamada olduğu üzere makul kabul edilebilecek değişken tipinde ve fonksiyon yapısında değişiklik olabilir.

■ Bir bankanın ATM cihazına gelen müşteriler, giriş (şifre yazarak), para yatırma, para çekme ve bakiye sorgulama işlemlerini yapabilmektedir. Bu sistemi nesne yönelimli olarak

modelleyiniz, ihtiyaç duyuluyorsa ilave sınıfları da tanımlayınız. Her sınıfın metotlarını ve özelliklerini belirtiniz ve modelinizi JAVA dilinde kodlayınız. (30 puan)

```
2   class ATM{
3       insan i;
4       insan [] musteriler =new insan[100];
5       boolean onay = false;
6   public ATM(insan i ){
7       this.i = i;
8   }
9   void paracekmek(int miktar){
10      if(onay)
11          i.bakiye -= miktar;
12  }
13  void paracekmek(int miktar,insan x){
14      if(onay)
15          x.bakiye -= miktar;
16  }
17  void parayatirmek(int miktar){
18      if(onay)
19          i.bakiye += miktar;
20  }
21  void giris(int sifre){
22      for(int j = 0;j< musteriler.length;j++){
23          if(musteriler[j].hno==i.hno)
24              if(musteriler[j].sifre == i.sifre)
25                  onay = true;
26      }
27  }
28  void cikis(){
29      onay = false;
30  }
31  }
```

```
33  class insan{
34      int sifre;
35      int hno;
36      int bakiye;
37  }
38  public class test{
39  public static void main(String args[]){
40      insan ali = new insan();
41      ali.hno = 123;
42      ali.sifre = 321;
43      ATM a =new ATM(ali);
44  }
45  }
```

■ Aşağıda verilen sınıfların özellik ve metotlarının isimlerini yazınız. Bu sınıflardan hangilerinin arasında ilişki olduğunu ve bu ilişkinin çeşidini yazınız.

{çember, daire, nokta, şekil, doğru, kare, yamuk, piramit, küp, silindir }

Yukarıdaki tanımınız ışığında, bir şekil vektörünü parametre olarak alıp bu vektördeki elemanların alanlarını hesaplayan fonksiyonu JAVA dilinde kodlayınız. (Bu sorudaki vektör kelimesi, java.util.Vector sınıfını ifade etmektedir, sorudaki hesaplama fonksiyonu dışında kod yazmanız gerekmemektedir ancak isterseniz sınıfların metotlarını ve özelliklerini kod yazarak da gösterebilirsiniz. Sınıflar arasındaki ilişkilerin isimlerini her durumda yazınız.) (50 puan)

```
2  import java.util.*;
3  interface alaniolmak{
4      float alan();
5  }
6  class nokta {
7      int x,y,z;
8  }
9  class dogru {
10     nokta n;
11
12 }
13 class kenar {
14     nokta n;
15     int boy;
16 }
17 class kare extends sekil implements alaniolmak{
18     nokta n;
19     kenar d;
20     public float alan(){
21         return d.boy*d.boy;
22     }
23 }
24 class cember {
25     nokta n;
26     int yaricap;
27 }
28 class daire extends sekil{
29     nokta n;
30     int yaricap;
31     public float alan(){
32         return (float)Math.PI * yaricap * yaricap;
33     }
34
35 }
```

```

37 class kup{
38     nokta n;
39     dogru d;
40 }
41 abstract class sekil{
42     abstract float alan();
43 }
44 //www.bilgisayarkavramlari.com
45 public class test{
46     public static void main(String args[]){
47         kare k = new kare() ;
48         nokta n = new nokta();
49         n.x = 1;
50         n.y = 2;
51         n.z = 4;
52         kenar d = new kenar();
53         d.boy = 3;
54         kenar d2 = new kenar();
55         d2.boy =4;
56         k.d = d;
57         k.n = n;
58         System.out.println(k.alan());
59         Vector v = new Vector();
60         v.add(k);
61         daire da = new daire();
62         da.n = n;
63         da.yaricap = 3;
64         v.add(da);
65         for(int i = 0;i< v.size();i++){
66             sekil s = (sekil)v.elementAt(i);
67             System.out.println(s.alan());
68         }
69     }
70 }

```

1. Aşağıdaki kod parçasının ekran çıktısını yazınız (10 Puan):

int x = 6; // 110

int y = 3; // 011

// 101 = ^

// 010 = 2

// 111 = |

System.out.println(x^y|2);

if(++y == 3&& x<6)

```
System.out.println(y);
```

```
System.out.println(y);
```

Not: Bu sorunun cevabı hem “Thinking in JAVA” kitabında hem de “JAVA Tutorial” içeriğinde bulunmaktadır.

7

3

Yukarıdaki ilk işlem x^y işlemidir ve ikilik tabanda XOR işlemi olarak çalışır. Bu durumda $6 = (110)_2$ ve $3 = (011)_2$ olarak çevrildikten sonra aralarında özel veya (xor) operatörü çalıştırıldığında, 101 sonucu bulunur. Ardından gelen tek boru (pipe) işlemi ise ikilik tabanda veya (or) işlemidir ve $(101)_2 \mid (010)_2$ şeklinde düşünülmelidir. Bu işlemin sağ tarafı 2 sayısının ikilik tabandaki karşılığıdır. Bu işlemin sonucu da $(111)_2$ olarak bulunur ki bu değer onluk tabanda 7 yapar.

Ardından gelen if satırında önce x değerinin 6'dan küçük olup olmadığı kontrol edilir. Bu kontrol doğru değildir (false) dolayısıyla if satırındaki ikinci kısım çalışmaz. Yani $y++ == 3$ satırı çalışmaz. Bunun sebebi JAVA dilindeki kısa devre (short circuit) uygulamasıdır. Ve işlemi ile birbirine bağlı olan if kontrollerinde sol tarafın false dönmesi halinde sonuç false olacağı için sağ tarafa bakılmaz. Ayrıca bu if satırının altında olan if bloğu da sonuç false olduğu için çalışmaz ve ikinci println fonksiyonu ekrana basılmaz.

Son satırdaki println fonksiyonu ise y'nin orijinal değeri olan 3'ü ekrana basar.

“Bilgisayarların, bilgisayar bilimindeki yeri, teleskopların, astronomi bilimindeki yerinden fazla değildir.” — Dijkstra

SORU 4: Dosyayı Tersten Basan Kod

Öncelikle algoritmamızı inşa edelim.

Ters almak gibi işlemler yapı olarak [özyineli \(recursive\) fonksiyonlara](#) çok uygundur. Genelde [stack \(yığın\)](#) yapısının kullanıldığı [özyineli fonksiyonlar](#) bilgiyi tutma ve ters çevirme (son giren ilk çıkar (LIFO) algoritması) için elverişli olurlar. Bu yüzden biz de bir [özyineli fonksiyon](#) kullanarak dosyamızdan karakter karakter değerleri okuyacak sonra da bunları ekrana basacağız.

Kodu aşağıda veriyorum:

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <conio.h>
4  //www.bilgisayarkavramlari.com
5  void ters(FILE * file) {
6      if(feof(file))
7          return;
8      int okunan;
9      char x;
10     okunan = fscanf(file,"%c",&x) ;
11     ters(file);
12     putchar(x);
13 }
14
15 int main() {
16     int i;
17     FILE *fd;
18     if(!(fd=fopen("girdi.txt","r"))) {
19         printf("Dosya acilamadi\n");
20         exit(1);
21     }
22     ters(fd);
23     fclose(fd);
24     getch();
25     exit(0);
26 }

```

Kodumuzdaki main fonksiyonu sadece dosyayı açmakta ve sonra ters fonksiyonunu çağırılmaktadır. Dosya ters fonksiyonunda işlendikten sonra (ki bu işleme dosyayı tersten basmak olacak) dosya kapatılmakta ve program sonlandırılmaktadır.

Gelelim ters fonksiyonuna. Bu fonksiyon, dosya göstericisini (file pointer) parametre olarak alıp, dosya sonu olup olmadığını kontrol ediyor. Feof fonksiyonu, şayet dosyanın sonuna ulaşırsa, return; satırını çalıştıracak ve dolayısıyla fonksiyondan çıkılacak. Aslında bu durum bizim dosyamızın sonu ve dolayısıyla fonksiyonumuzun sonu ve dolayısıyla [özyineleme yığınınımızın \(recursion stack\)](#) sonu olmaktadır.

Fonksiyonun geri kalanında işlem gayet basit bir şekilde fscanf ile dosyadan okumak ve ardından tekrar ters fonksiyonunu çağırarak ve sonra da okunan karakteri ekrana basmak şekline ilerler.

Buradaki kritik nokta, ters fonksiyonunu, ekrana karakter basmadan çağırmasıdır. Şayet kodun 11. ve 12. satırları yer değiştirecek olsaydı, yani önce ekrana basıp sonra ters fonksiyonunu çağırarak olsaydı, bu durumda dosyanın içeriği olduğu gibi ekrana basılacaktır. Biz bunun yerine o andaki okunan karakteri bekletiyoruz ve sonraki karakteri basması ve ters çevirmesi için yeniden fonksiyona veriyoruz. Bu işlem dosya sonuna kadar gidiyor. Nihayet dosya sonuna erişilince [özyineli yığınınımız \(recursion stack\)](#) toparlanmaya başlıyor ve bu toparlanma sırasında ekrana karakterleri basarak ilerliyor.

Yukarıdaki kod, Dev-CPP ile test edilmiş ve çalışmaktadır. Örneğin girdi.txt dosyası olarak aşağıdaki içerikte bir dosyayı verirsek:

```
www
.bilgisayar
kavramlari
.com
sadi
evren
seker
```

Programın çıktısı aşağıdaki şekilde olur:

```
rekes
nerve
idas
moc.
iralmarvak
rayasiglib.
www
```

SORU 5: Nesne Yönelimli Programlama Dersi Quiz Çözümü

Quiz soruları ve çözümleri aşağıdaki şekildedir:

1. Bir oylama için program yazmanız isteniyor. Oylamaya katılan 5 aday bulunuyor ve bu adayların numarası (1'den 5'e kadar bir sayı) oy pusulasına yazılarak oy kullanılıyor. Programınızda 10 adet oyu okuyup ekrana adayların aldıkları oy miktarını basan bir program yazınız. Kullanılan oyun 1-5 arasında olmaması halinde problemi istisna yakalama ile çözünüz.

Çözüm

Oylama programında basitçe her oyun tutulduğu bir oy pusulası ve bir de oy verilen adaylar olacağını düşünebiliriz. Bu durumda oy pusulası ve aday isimli iki adet sınıf tanımlayıp bu sınıfların kodlamasını aşağıdaki şekilde yapabiliriz.

```
1  #include <iostream>
2  using namespace std;
3  class aday{
4  private:
5      int adayno;
6      int oysayisi;
7  public:
8      aday(){
9          oysayisi=0;
10     }
11     int getOysayisi(){
12         return oysayisi;
13     }
14     int getAday(){
15         return adayno;
16     }
17     void setaday(int adayno){
18         this->adayno= adayno;
19     }
20     void oysayisiniArttir(){
21         oysayisi++;
22     }
23 };;
```

Yukarıdaki kodda, aday bilgisi tutuluyor. Programımızda olması gereken bilgi, adayın numarası ve oy sayısıdır. Biz de bu bilgileri tutan iki adet değişken tanımlıyor ve bu

değişkenlerin getter/Setter fonksiyonlarını kodluyoruz. Ayrıca sınıfın [inşa fonksiyonunda \(constructor\)](#), başlangıç olarak oy sayısını 0 yapıyoruz. Son olarak adayların oy sayısını arttıran bir metod yukarıdaki kodun sonunda yer almaktadır. Basit bir arttırma işlemidir.

Oy pusulasını tutan kod ise aşağıdaki şekilde kodlanmıştır:

```
1  #include <iostream>
2  using namespace std;
3  class oypusulasi{
4  private:
5      int oy;
6  public:
7      int getOy(){
8          return oy;
9      }
10     void setOy(int o){
11         oy = o;
12     }
13 };
```

Yukarıdaki kodda, görüldüğü üzere, oy pusulasında sadece kime oy verildiği bilgisi tutulmuştur. Bu bilgiyi [kapsülleme gereği getter ve setter fonksiyonları](#) ile erişilebilir halde sınıfta kodluyoruz.

Son olarak kodumuzu test etmek için bir main fonksiyonu yazalım:

```
3  #include "aday.h"
4  #include "oypusulasi.h"
5  #include "hatalioy.h"
6  //www.bilgisayarkavramlari.com
7  #include <iostream>
8  #include <conio.h>
9  using namespace std;
10
11 int oyoku(){
12     int oy;
13     cin >> oy;
14     if(oy>=5||oy<=1)
15         throw hatalioy();
16     return oy;
17 }
18
19 int main(){
20     aday adaylar[6];
21     oypusulasi oylar[10];
22     for(int i = 0;i<10;i++){
23         try{
24             oylar[i].setOy(oyoku());
25             adaylar[oylar[i].getOy()].oysayisiniArttir();
26         }
27         catch(hatalioy h){
28             cout << "girilen oy hatalidir" <<endl;
29         }
30     }
31     for(int i = 1;i<6;i++){
32         cout << i <<". adayin oy sayisi: " << adaylar[i].getOysayisi() << endl;
33     }
34     getch();
35 }
```

Oy okuma işlemi için “oyoku” isimli fonksiyon yazılmıştır. Bu fonksiyonun özelliği, oyları okurken yanlış bir oy girilmesi durumunda, (oyların 1 ile 5 arasında olacağını hatırlayınız) bir

istisna fırlatmasıdır. Bu istisna, try bloğu içerisinde çağrıldığı yerde yakalanır ve kodun 27nci satırında bulunan catch bloğu ile ekrana bir hata mesajı basılır. Şayet oy başarılı bir şekilde okunabilirse, bu durumda ilgili oy pusulasına (ki toplam 10 adet pusula bulunuyor) oy değeri kaydedilir. Şayet bir istisna burada oluşmamışsa kodun 25. Satırına geçebiliriz. İstisna oluşması durumunda kodun 25. Satırı hiç çalışmadan catch bloğuna gidilir.

Kodumuzun 25. satırında, adayların tutulduğu nesne dizisinin (object array) ilgili elemanın (hangi elemana oy verildiyse) oy miktarını 1 arttıran oysayisiniArttir() fonksiyonu çağrılır.

Son olarak kodun 31-33 satırları arasında, adayların oy durumları ekrana basılır.

Yukarıdaki kodda bulunan ve kendi istisnamız olan hatalioy sınıfını aşağıdaki şekilde kodlayabiliriz:

```
1 #include <iostream>
2
3 class hatalioy{
4 private:
5     int oy;
6 public:
7     hatalioy(){
8     }
9     hatalioy(int oy){
10         this->oy=oy;
11     }
12     int getOy(){
13         return oy;
14     }
15 };
```

1. Bir banka yazılımı için aşağıdaki özelliklerle program yazmanız isteniyor. Sistemde tutulacak veriler:
 1. Hesap sahibinin ismi
 2. Hesap numarası
 3. Bakiye
 4. Hesap tipi (vadeli / vadesiz)

Yukarıdaki hesap bilgileri için aşağıdaki işlemleri yapmanız isteniyor:

1. Hesapların ilk değer olarak bakiyesinin 0 atanması gerekir.
 2. Hesaba para yatırılabilir
 3. Hesaptan para çekilebilir
 4. Hesap bilgisi görüntülenebilir (sahibinin ismi, numarası, tipi ve bakiyesi)
1. 2. Sorudaki programınızı, 10 müşteriye hizmet verebilecek şekilde genişletin.
 2. 2. Sorudaki programınızı, hesap hareketleri görüntülenecek şekilde genişletin. Buna göre bir hesap hareketinin, tarihi, saati, miktarı ve hesap numarası bulunur.

Çözüm:

Yukarıdaki sorunun çözümü için basitçe bir hesap nesnesi oluşturmamız yeterli olacaktır. Yukarıdaki 4. soru için hesap hareketlerini tutan ilave bir sınıfa daha ihtiyacımız olacak. Öncelikle, hesapların tutulduğu bir hesap sınıfını programlayarak başlayalım.

```

7 class hesap{
8 private:
9     int hesapno;
10    float bakiye;
11    string hesapSahibi;
12    int tipi; // 0 vadeli, 1 vadesiz
13    hesaphareketi h[100];
14    int hhsayisi;

```

Soruda tanımlandığı üzere, her hesap için hesapno, bakiye , hesap tipi ve hesap sahibi bilgilerini tutuyoruz. Ayrıca hesap hareketlerinden oluşan ilave bir dizi, her hareketi ayrıca kayıt altına alıyor.

Hesap sınıfımızın constructor fonksiyonunda, hesap bakiyesini 0 olarak atıyoruz.

```

16 hesap(){
17     bakiye=0;
18     hhsayisi = 0;
19 }

```

Ayrıca sınıfımızdaki hesap hareketlerini tutan dizinin kaçınıcı elemanında olduğumuz bilgisi, hhsayisi isimli değişkende durmaktadır. Bu değişken de henüz hesap hareketi olmadığı için 0 olarak atanıyor.

Sırasıyla hesap sınıfındaki metotların kodlamasını inceleyelim:

```

20 void hesapDetay(){
21     cout << "Hesap No:" << hesapno << " bakiye : " << bakiye << " hesapsahibi : " << hesap
22
23     if(tipi == 0)
24         cout << " vadeli";
25     else
26         cout << " vadesiz";
27     cout << endl << "Hesap Hareketleri" << endl << "-----"<<endl;
28     for(int i = 0;i<hhsayisi;i++){
29         cout << h[i].getmiktar() << endl;
30     }
31
32 }

```

Yukarıda, hesap detaylarını ekrana basan fonksiyon verilmiştir. Fonksiyondaki 21nci satırda bulunan cout komutu ile ekrana hesabın temel 3 bilgisi bastırılmıştır. Ardında hesabın tipi yazıya çevrilerek bastırılmıştır. Hesap tipini tutmak için kullanılan int tipi, kullanıcı açısından anlamsız olabilir o yüzden yazıya çevirme işlemi yapılır.

Son olarak 27-30 satırları arasında, hesap hareketlerini bastırmak için, mevcut hesap hareketi sayısını tutan değişken kadar dönen döngü içerisinde her hesap hareketinin miktarı basılmaktadır.

Bu çıktının örneği aşağıda verilmiştir.

```
Hesap No:123 bakiye : 700 hesapsahibi : Ali Demir vadesiz
Hesap Hareketleri
-----
1000
-250
-100
50
```

Yukarıdaki ekran görüntüsünde, örnek olarak bir hesabın detayları basılmış ve bu hesapta yapılan hesap hareketleri altında gösterilmiştir.

Hesap hareketi olarak kabul edilebilecek iki temel işlem para çekmek ve para yatırmak işlemleridir. Bunların kodlamasını aşağıda anlatalım:

```
60 void paracek(float miktar){
61     if(bakiye-miktar<0)
62         throw yetersizBakiye(bakiye-miktar);
63     h[hhsayisi].sethesapno(hesapno);
64     h[hhsayisi].setmiktar(miktar*-1);
65     hhsayisi++;
66     bakiye -=miktar;
67 }
68 void parayatir(float miktar){
69     h[hhsayisi].sethesapno(hesapno);
70     h[hhsayisi].setmiktar(miktar);
71     hhsayisi++;
72     bakiye += miktar;
73 }
```

Para çekme işlemi, belirtilen miktar kadar bakiyeyi azaltmamaktadır. Bu işlem sırasında öncelikle miktarın yeterli olup olmadığı kontrol edilir. Şayet miktar yeterli bulunursa bakiye azaltılır, şayet yeterli bulunmazsa bu durumda bir istisna (exception) fırlatılır. Burada fırlatılan istisna, proje kapsamında bizim tanımladığımız yetersizBakiye istisnasıdır.

Yetersiz bakiye istisnası, basit bir şekilde bakiye bilgisini de içinde saklayan bir sınıf (class) olarak tasarlanabilir:

```
1 class yetersizBakiye{
2     private:
3         float miktar;
4     public:
5         yetersizBakiye(){
6         }
7         yetersizBakiye(float miktar){
8             this->miktar = miktar;
9         }
10        float getMiktar(){
11            return miktar;
12        }
13 };
```

Hesap hareketlerinin içerisine, ayrıca miktar konulurken -1 değeri ile çarpılmaktadır. Bunun anlamı, hesap hareketinin eksi değer olarak tutulmasıdır.

Benzer kodlama, para yatırma fonksiyonunda da kullanılmaktadır. Yatırılan miktar, hesap hareketlerine işlendikten sonra, bakiye değeri parametre olarak verilen miktar kadar arttırılmaktadır.

Hesap hareketlerinin detayı için sınıf kodlamasını aşağıdaki şekilde yazabiliriz:

```
1 #include <iostream>
2 #include <string.h>
3
4 using namespace std;
5
6 class hesaphareketi{
7 private:
8     int hesapno;
9     float miktar;
10    string tarih;
11    int saat;
```

Bu sınıf, basit bir şekilde hesapta yapılan işlemleri tutmak için tasarlanmıştır. İlgili üyelerin getter / setter fonksiyonlarının yazılması yeterlidir.

SORU 6: Dosya ve Bağlı Liste Uygulaması

Gelen bir soru üzerine (Ayşenur Doğan sormuş), iki farklı [dosya okuyup](#), okunan dosyalardaki verileri iki farklı [bağlı listeye](#) koyan kodu yazmaya çalışalım.

Öncelikle bağlı listemizin düğüm yapısını tanımlayalım. Tek yönlü bir bağlı liste işimizi çözecektir. Dolayısıyla sadece next isimli bir [gösterici \(pointer\)](#) koyarak her düğümün (node) bir sonraki düğüme bağlanmasını sağlıyoruz.

```
5 typedef struct dugum{
6     char kelime[100];
7     dugum *next;
8 };
```

Ayrıca her düğümde en fazla 100 karakter uzunluğunda bir [dizgi \(string\)](#) tutuyoruz. Dosyadaki her kelime bu dizginin içerisine yerleştirilecek.

Ardından klasik olarak dosya açma işlemini yapıyoruz:

```

10 int main(){
11     FILE *fp1, *fp2;
12     char dosya1[20],dosya2[20];
13     printf("birinci dosyanin ismini giriniz");
14     scanf("%s",dosya1);
15     if((fp1=fopen(dosya1,"r"))==NULL){
16         puts("dosya1 acilamadi.\n");
17         getch();
18         return 1;
19     }
20     printf("ikinci dosyanin ismini giriniz");
21     scanf("%s",dosya2);
22     if((fp2=fopen(dosya2,"r"))==NULL){
23         puts("dosya2 acilamadi.\n");
24         getch();
25         return 1;
26     }

```

Yukarıdaki kodda görüldüğü üzere, [iki dosya göstericisi \(file pointer\)](#) tanımlanmıştır. Ayrıca dosyaların isimlerinin kullanıcıdan okunması için dosya1 ve dosya2 isimli iki [dizgi \(string\)](#) tanımlanarak kullanıcıdan dosya isimleri okunmuştur.

Okunan dosyaların açılmaması halinde hata mesajı verilerek çıkılmıştır.

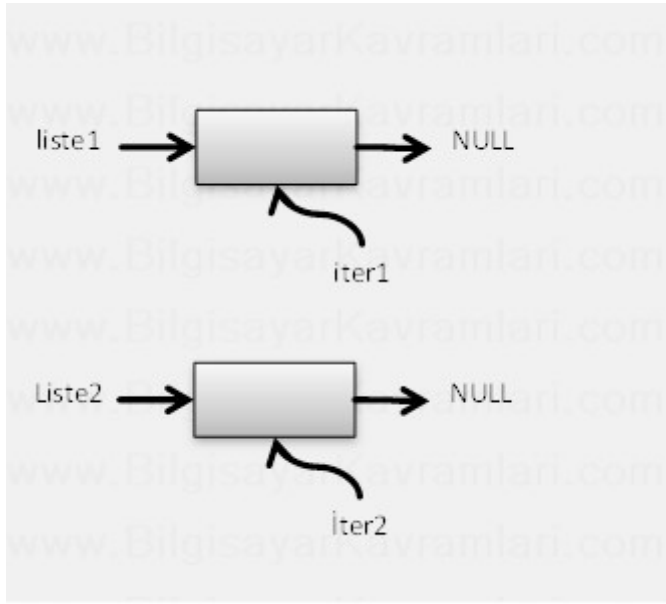
Sıra [bağlı listelerin \(linked list\)](#) tanımlanmasında:

```

dugum *liste1=(dugum*)malloc(sizeof(dugum));
dugum *liste2=(dugum*)malloc(sizeof(dugum));
dugum *iter1 = liste1;
dugum *iter2 = liste2;

```

Yukarıdaki kodda, iki adet bağlı liste, daha önceden tanımladığımız dugum [yapısı\(struct\)](#) uygun olarak birer düğümü gösterecek şekilde hafızaya yerleştiriliyor. Ayrıca bağlı listemiz [tek yönlü olduğu için \(singular\)](#) listenin ilk elemanını kaybedemeyiz. Bu yüzden listenin üzerinde hareket edecek olan bir iter değişkeni tanımlıyoruz (literatürde iterator olarak geçer). Yani listenin son elemanını sürekli olarak iter, ilk elemanını ise liste ile başlayan değişkenler tutuyor. Şimdilik hafızadaki bağlı liste görüntüsü aşağıdaki şekildedir:



Yukarıdaki şekilde görüldüğü üzere, hafızada (RAM) iki adet liste, bu iki listenin içinde de birer adet boş düğüm (node) ve her düğümü gösteren ilave birer iter değişkeni bulunmaktadır.

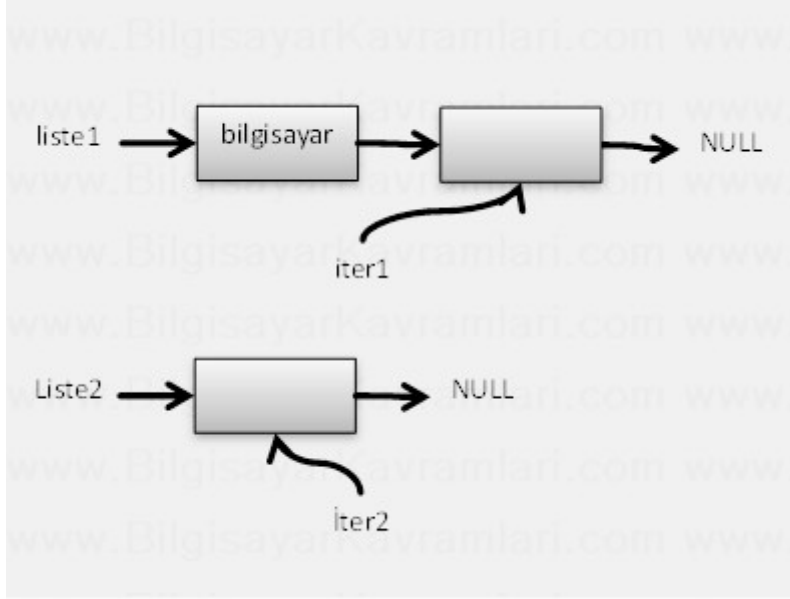
Gelelim dosyadan okuyup listenin içerisine değerleri atan koda:

```
31 while(!feof(fp1)){
32     char gecici[100];
33     fscanf(fp1, "%s", gecici);
34     //printf("okunan: %s", gecici);
35     strcpy(iter1->kelime, gecici);
36     iter1->next = (dugum*)malloc(sizeof(dugum));
37     iter1 = iter1->next;
38 }
39 //perror("gecti");
40 iter1->next = NULL;
41 while(!feof(fp2)){
42     char gecici[100];
43     fscanf(fp2, "%s", gecici);
44     //printf("okunan: %s", gecici);
45     strcpy(iter2->kelime, gecici);
46     iter2->next = (dugum*)malloc(sizeof(dugum));
47     iter2 = iter2->next;
48 }
```

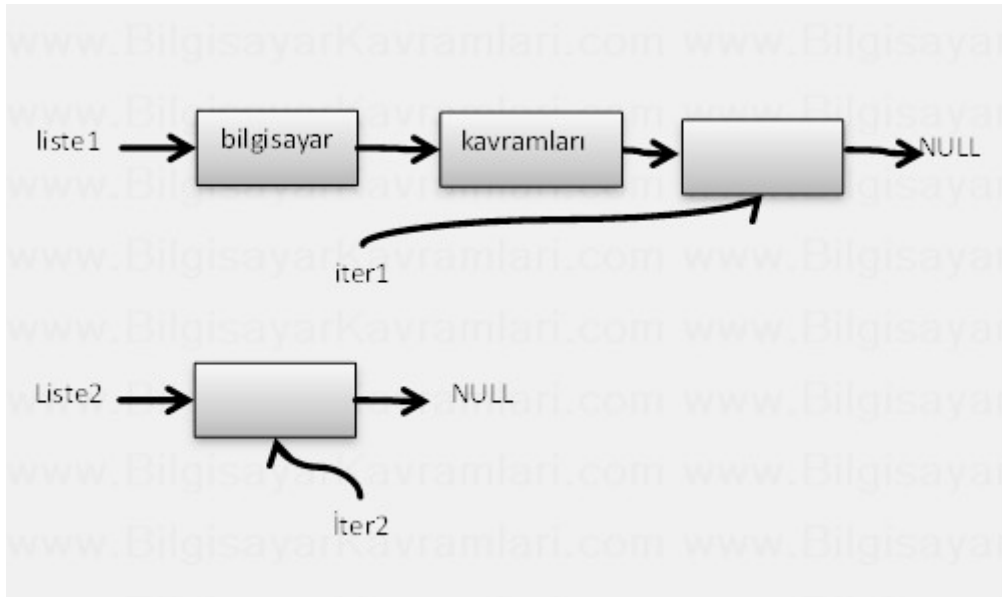
Kodumuz basitçe, dosyanın sonuna gelinip gelinmediğini kontrol eden bir while döngüsü içerisinde çalışıyor. Yani kodun 31 ve 41 numaralı satırlarında yapılan kontrol, ilgili dosya göstericisinin, [dosya sonuna ulaşp ulaşmadığını kontrol etmekte](#) ve ulaşmadığı sürece döngü devam etmektedir.

Her iki döngü için de geçerli olmak üzere, bir geçici isimli [dizgi \(string\)](#) tanımlanarak dosyadan geçici olarak bu dizgiye okuma işlemi yapılmıştır. Ardından [strcpy isimli string.h kütüphanesinde bulunan fonksiyon ile](#), okunan bu dizgi, o anda iter tarafından gösterilen düğüme konulur. İter göstericisinin sonraki düğümünde yer açılır ve iter bu yeni boş düğümü gösterir.

Örneğin dosyadan okunan ilk kelime “bilgisayar” olsun. Bu durumda bağlı liste yapısı aşağıdaki şekilde olacaktır:



Görüldüğü üzere, iter göstericisi tarafından gösterilen kutuya bir değer eklenip, yeni boş bir kutu, next ile gösterilen boş alanda açılmakta ardından iter bu boş kutuyu göstermektedir. Örneğin anlaşılması açısından bir de “kavramları” kelimesini okuduğumuzu düşünelim:



Yukarıda görüldüğü üzere her seferinde, iter tarafından gösterilen düğüme, okunan [dizgi \(string\)](#) yerleştirilmekte ve yeni iter tarafından gösterilen düğümün sonrasına (next) boş bir düğüm konulmakta ve iter bu yeni düğümü gösterilecek şekilde ilerletilmektedir.

Her şey bittikten sonra, bağlı listelere okunan değerleri ekrana bastırıyoruz:

```

50     iter1=liste1;
51     iter2=liste2;
52     printf("\n%s dosyasinin icerigi:\n ----\n",dosya1);
53     while(iter1!=NULL){
54         printf("%s\n",iter1->kelime);
55         iter1=iter1->next;
56     }
57     printf("\n%s dosyasinin icerigi:\n ----\n",dosya2);
58     while(iter2!=NULL){
59         printf("%s \n",iter2->kelime);
60         iter2=iter2->next;
61     }

```

Listelerin başını tutan liste1 ve liste2 isimli değişkenleri [gösterecek](#) şekilde iter değişkenlerini başa çekiyoruz. Buradaki amaç yine listenin ilk elemanını kaybetmeden bastırma işlemini yaptırmaktır. İter değişkenleri, listenin ilk elemanlarını gösterdikten sonra, gösterdikleri düğümlerin değerlerini ekrana basıp sonraki düğüme (next node) iletilemektedir.

Kodun ekran çıktısı aşağıdaki şekildedir:

```

C:\Users\sheda\\Desktop\Untitled1.exe
birinci dosyanin ismini girinizyeni.txt
ikinci dosyanin ismini girinizyeni1.txt
yeni.txt dosyasinin icerigi:
----
sadi
evren
seker
seker
1DZ
yeni1.txt dosyasinin icerigi:
----
bilgisayar
kavramlari
deneme
123
123

```

SORU 7: 2 Boyutlu Dizi İçerisine Sarmal Seri Girişi

Bu yazının amacı, bir C sorusunun çözümünü açıklamaktır. Sorumuz basitçe $n \times n$ boyutlarındaki bir matrisin (soruyu basitleştirmek için n sayısını tek sayı olarak kabul edeceğiz) içerisine aşağıdakine benzer şekilde ardışık sayıları sarmal olarak yerleştirmek.

Örnek 3×3 boyutlarındaki matris:

1	2	3
8	9	4
7	6	5

5×5 boyutlarındaki örnek matris:

1	2	3	4	5
---	---	---	---	---

16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

7×7 boyutlarındaki örnek matris:

1	2	3	4	5	6	7
24	25	26	27	28	29	8
23	40	41	42	43	30	9
22	39	48	49	44	31	10
21	38	47	46	45	32	11
20	37	36	35	34	33	12
19	18	17	16	15	14	13

Yukarıdaki matrislere bakıldığında, sayısal olarak bazı özellikler bulunabilir. Bizim amacımız yukarıdaki problemi, 2 boyutlu probleminden tek boyutlu probleme indirgemektir. Bunun sebebi diziler üzerinde işleme yapan en önemli enstrümanımız olan döngülerin tek boyutlu olarak çalışmasıdır. Burada iç içe yazılan iki döngünün iki boyutlu çalıştığı iddia edilebilir ancak bu durumda bile aslında yapılan iş, tek boyutlu bir taramanın, ikinci döngü ile (dıştaki döngü ile) tekrarlanmasıdır. Yani döngüler, yetenek olarak tek boyutlu, doğrusal hareket sağlar.

Bu tespitten sonra yukarıdaki problemi çözmek için sayısal özellikler bulmaya ve problemi tek boyutlu problem haline getirmeye çalışıyoruz.

İlk tespitimiz, 2 boyutlu bir matris içerisine ardışık sayıları yerleştirmek dolayısıyla matrisin ortasındaki sayının, matris boyutunun karesi olacağını biliyoruz. Örneğin 7×7 boyutlarındaki matris için ortada 49 , veya 5×5 boyutlarındaki matris için ortadaki değerin 25 olması gibi.

Ayrıca ortaya kadar olan köşegenleri bulursak tek boyutlu seriler oluşturmuş oluruz.

Örneğin aşağıdaki renklendirilmiş değerlerden başlayan ve sağa doğru giden tek boyutlu seriler, 1'er 1'er artan değerlerdir:

1	2	3	4	5	6	7
24	25	26	27	28	29	8
23	40	41	42	43	30	9
22	39	48	49	44	31	10
21	38	47	46	45	32	11
20	37	36	35	34	33	12
19	18	17	16	15	14	13

Bu değerlerden başlayarak sağa doğru giden seri aşağıdaki şekilde işaretlenebilir:

1	2	3	4	5	6	7
24	25	26	27	28	29	8

23	40	41	42	43	30	9
22	39	48	49	44	31	10
21	38	47	46	45	32	11
20	37	36	35	34	33	12
19	18	17	16	15	14	13

Amacımız, bu seriyi veren ve yukarıdaki şekilde, kırmızı renkle işaretlenmiş seri başlarını veren formülü bulmak.

Serimizde bulunan sayılar:

1, 25, 41, 49

Bu sayıların arasında bir bağlantı bulmamız gerekiyor. Sayıları tersten yazar ve aralarındaki farkları bulursak:

49, 41, 25, 1 ve aralarındaki farklar : 8 , 16, 24 şeklinde giden seridir. Bu serinin özelliği 8'in katları olmasıdır.

Benzer durum diğer boyutlardaki matrisler için de geçerlidir. Örneğin matris 9×9 boyutlarında olsaydı:

81, 73, 57, 33, 1 ve aralarındaki farklar: 8, 16, 24, 32 şeklinde olacaktı.

Benzer şekilde diğer köşegen değerlerini de bulalım:

1	2	3	4	5	6	7
24	25	26	27	28	29	8
23	40	41	42	43	30	9
22	39	48	49	44	31	10
21	38	47	46	45	32	11
20	37	36	35	34	33	12
19	18	17	16	15	14	13

Mavi renkle işaretli köşegen değerlerini veren formülü inceleyelim :

19, 37, 47, 49 , bu sayıları ters sıralarsak : 49, 47, 37, 19 ve aralarındaki farklar: 2, 10, 18 şeklinde ilerlemektedir. Bu sayılar arasındaki fark da dikkat edilirse 8 'dir.

Benzer şekilde matris boyutu 9×9 olsaydı, 81, 79, 69, 51, 25, ve aralarındaki farklar: 2, 10, 18, 26 şeklinde olacaktı. Bu sayılardan başlayarak ilerleyen tek boyutlu dizilerimiz aşağıdaki şekilde olacaktı:

1	2	3	4	5	6	7
24	25	26	27	28	29	8
23	40	41	42	43	30	9

22	39	48	49	44	31	10
21	38	47	46	45	32	11
20	37	36	35	34	33	12
19	18	17	16	15	14	13

Son serimiz aşağıda işaretlenen seridir:

1	2	3	4	5	6	7
24	25	26	27	28	29	8
23	40	41	42	43	30	9
22	39	48	49	44	31	10
21	38	47	46	45	32	11
20	37	36	35	34	33	12
19	18	17	16	15	14	13

Serimiz: 49, 45, 33, 13 şeklinde ilerlemektedir ve farklar: 4, 12, 20 şeklinde ilerlemektedir.

Şimdi, yukarıda incelediğimiz bu 4 durum için döngüleri yazmaya başlayalım. İlk durum için kodumuz aşağıdaki şekilde olur:

```
int kat = 0;

int fark = 0;

for(int i = n/2 ; i>=0; i--){

    a[i][i] = n*n-fark;

    kat++;

    fark = fark + kat*8;

    for(int j = i+1; j<n-i; j++){

        a[i][j]=a[i][j-1]+1;

    }

}
```

Yukarıdaki kodda, dikkat edilirse, bir fark değişkeni, sürekli olarak $n*n$ değerinden çıkarılıyor. Buradaki değer a dizisinin $[i][i]$ adresine yerleştiriliyor. Bu değer normalde, dizinin köşegen değeridir. Bu döngüdeki i değişkeni $n/2$ değerine kadar ilerlemektedir, dolayısıyla köşegenin yarısına kadar ilerleyen döngü, tam olarak 2 boyutlu dizinin ortasında bitmektedir.

Hesaplanan köşegen değerlerinden başlanarak, satırları dolduran döngü ise içerde yer almaktadır. Burada basitçe, hesaplanan her köşegen değerinden başlayarak satır değerlerini dolduran döngü yazılmıştır.

İkinci döngümüz aşağıdadır:

```
for(int i = n/2 ;i>0;i-){  
    for(int j = i;j<n-i+1;j++){  
        a[j][n-i]=a[j-1][n-i]+1;  
  
    }  
}
```

Bu döngü, bir öncekine göre oldukça basittir çünkü önceki döngüde hesaplanan köşegen değerlerini devam ettirmektedir. Kısacası yeniden köşegen değeri hesaplanmadığı için, mevcut değerler üzerinden işlem yapılmakta ve tek boyutlu hareket tekrarlanmaktadır.

Üçüncü olarak alt satırları yazan döngümüzü kodlayalım:

```
kat = 2;  
fark = 0;  
  
for(int i = n/2 ;i>0;i-){  
    fark = fark +kat;  
    a[n-i][i-1] = n*n-fark;  
  
    kat = kat+8;  
  
    for(int j = i;j<n-i+1;j++){  
        a[n-i][j]=a[n-i][j-1]-1;  
  
    }  
}
```



```
}
```

Hatırlanacağı üzere bu döngü, 2 den başlayarak 8'er 8'er artan döngüydü. Dolayısıyla kat değişkenini bu sefer 2 den başlatıp, ters köşegeni dönen döngüyü yazıyoruz. İçerideki döngümüz, sütun değerlerini yukarıdan aşağıya doğru dolduruyor. [Burada ters köşegen için \$i+j=n\$ formülünü hatırlayalım](#). Dolayısıyla $a[j-i][j]$ değeri ters köşegeni vermektedir.

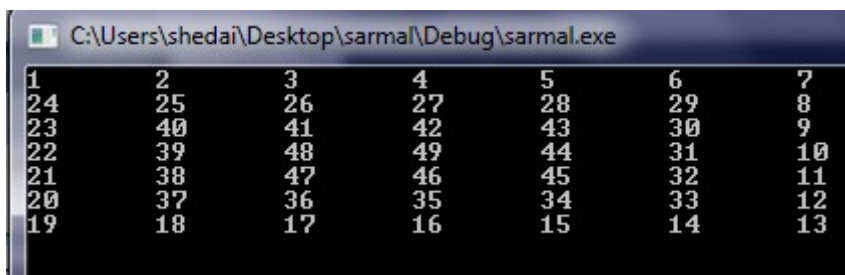
Son olarak ortanın solundaki kolonları dolduran döngülerimizi yazıyoruz:

```
for(int i = 0 ;i<n/2;i++){  
    for(int j = i+2;j<n-i;j++){  
        a[n-j ][i]=a[n-j+1][i]+1;  
    }  
}
```

Bu son döngü yapısında, dış döngümüz, köşegen yarısına kadar ilerlerken, içerideki döngümüz sütunları doldurmaktadır. Bu döngüyü yukarıda yazılan ikinci döngünün (ortanın sağındaki değerleri dolduran döngünün) tersi olarak düşünmek mümkündür. Bu dördüncü döngü için de başlangıç değerleri zaten hesaplanmıştır.

Kodun örnek çıktıları aşağıda verilmiştir:

7×7 boyutları için :



1	2	3	4	5	6	7
24	25	26	27	28	29	8
23	40	41	42	43	30	9
22	39	48	49	44	31	10
21	38	47	46	45	32	11
20	37	36	35	34	33	12
19	18	17	16	15	14	13

9×9 boyutları için:

C:\Users\sheda\\Desktop\sarmal\Debug\sarmal.exe								
1	2	3	4	5	6	7	8	9
32	33	34	35	36	37	38	39	40
31	56	57	58	59	60	61	40	11
30	55	72	73	74	75	62	41	12
29	54	71	80	81	76	63	42	13
28	53	70	79	78	77	64	43	14
27	52	69	68	67	66	65	44	15
26	51	50	49	48	47	46	45	16
25	24	23	22	21	20	19	18	17

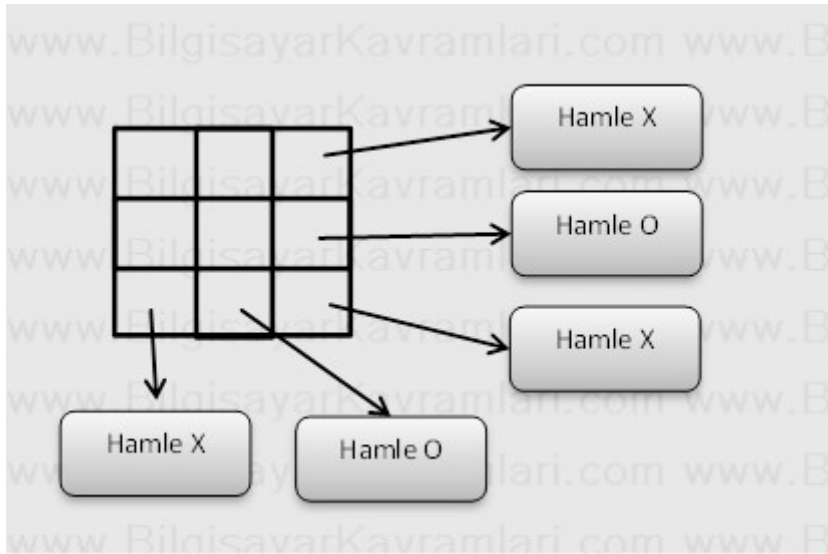
SORU 8: Nesne Yönelimli olarak Tic Tac Toe Oyunu

Bu yazının amacı, tictactoe oyununu nesne yönelimli olarak kodlamak ve bu sırada, aşağıdaki konuları açıklamaktır.

1. Nesnelerden oluşan bir dizi kullanımı (Object Array)
2. [Kapsülleme \(Encapsulation\)](#)
3. [İstisna yakalama \(Exception Handling\)](#)

Kodumuz basitçe, oynanan her hamleyi 3×3 boyutlarında, iki boyutlu bir dizide tutacaktır. Ayrıca oyunun kazanılması durumunda bir istisna oluşturulacak ve bu istisnanın yakalanması ile, oyun sona erecektir.

Bu problemin çözümü için, aşağıdaki şekilde bir nesne dizisi tasarlıyoruz:



Yukarıdaki şekilde, 3×3 boyutlarında görülen tablo üzerindeki hamleler ayrı birer nesnede tutulmak istenmiştir. Dolayısıyla her hamleyi tutan bir nesne ve bu hamlelerin tanımlandığı bir sınıfa ihtiyacımız bulunuyor.

Bu sınıfı aşağıdaki şekilde kodlayabiliriz.

Hamle.h dosyası

```
class hamle{  
  
private:
```

```

        int sembol; // x:1 veya o:2 değeri

public:

    int getSembol();

    void setSembol(int);

    hamle(int);

    hamle();

};

```

Kodda bulunan sembol değişkeni, basitçe hamlenin X veya O olmasını belirliyor. Şayet oynanan hamle X ise, değer olarak 1, O ise değer olarak 2 veya boş hamle ise, yani henüz tablonun bu alanına oynanmamış ise 0 değerini tutuyor.

Kapsülleme işlemi için bu değişkeni private tanımlıyor ve getter/setter metotlarını oluşturuyoruz. Ayrıca sınıfımızı kullanacak kişiler için bir adet int parametresi alan [yapıcımız \(constructor\)](#) bulunuyor.

Son olarak boş yapıcı (default constructor) tanımlıyoruz ki, tablo içerisine hamleleri boş olarak yerleştirdiğimizde, boş hamleleri belirten 0 değerini sembol değişkenine atayabilelim.

Bu sınıfın metotları aşağıdaki şekilde kodlanabilir:

Hamle.cpp dosyası:

```

#include
"hamle.h"

int hamle::getSembol(){

    return sembol;

}

void hamle::setSembol(int sembol){

    this->sembol =sembol;

}

hamle::hamle(int sembol){

    this->sembol= sembol;

}

hamle::hamle(){

    sembol = 0;
}

```

```
}
```

Her hamleyi tutmaya yarayan yukarıdaki koda ilave olarak bu hamlelerin oynandığı konumları tutan bir tahta sınıfı kodlamamız gerekiyor.

```
class Tahta{  
  
private:  
  
    hamle tahta[3][3];  
  
    int sıra;  
  
public:  
  
    Tahta();  
  
    void oyna(int x,int y, int sıra);  
  
    void oyna(int x,int y);  
  
    void bas(void);  
  
    int kazandi();  
  
};
```

.Yukarıdaki kodda görüldüğü üzere, 3×3 boyutlarında bir tahta, her elemanı hamle sınıfından değerler tutacak şekilde tanımlanmış. Bu tahta dizisine (array) erişmek için oyna metodu bulunuyor. Basitçe, tahta üzerinde bir adrese herhangi bir hamle konulabilmesi için oyna fonksiyonundaki x ve y değerlerinin verilmesi gerekiyor.

Ayrıca oyna metodu [üzerine yükleme \(overloading\)](#) ile iki farklı [şekilde tanımlanmış \(polymorphism\)](#). Bu sayede, kodumuzu kullanan diğer programcılar, isterlerse o andaki sırayı belirleyerek oynayabilecekler (örneğin istedikleri bir adrese, özel olarak X veya Y değeri koyabilecekler) veya sıradaki hamleyi (örneğin sıra X'te ise doğrudan X değeri) verilen adrese konulabilecek.

Yukarıda, tanımı verilen sınıfın kodlaması aşağıdaki şekildedir:

```
Tahta::Tahta(){  
  
    sıra = 1;  
  
}
```

[İnşa metodunun \(constructor method\)](#) içerisinde, ilk sıranın kimde olduğunu belirtmek için sıra değişkenine 1 değeri konulmuştur. Bunun anlamı, şayet programcı özel olarak bir sıra belirtmezse, ilk hamlenin X tarafından yapılacağıdır.

```

void Tahta::oyna(int x,int y,int sıra){

    if(tahta[x][y].getSembol()==0){

        tahta[x][y] = hamle(sıra);

    }

    else

        cout << "bu koordinata sembol yerlestirilmistir"<<endl;

}

```

Tahta üzerinde yapılan hamlelerin koordinatları oyna fonksiyonuna verilmektedir. Yukarıdaki oyna fonksiyonu, aldığı x ve y değerlerine, verilen sıradaki sembolü yerleştirir. Buna göre yeni bir nesne oluşturulup verilen x,y koordinatlarındaki dizi içerisine yerleştirilmektedir.

Aynı fonksiyonu aşağıdaki şekilde üzerine yüklenmesi mümkündür.

```

void Tahta::oyna(int x, int y){

    if(tahta[x][y].getSembol()==0){

        tahta[x][y] = hamle(sıra);

        sıra %=2;

        sıra++;

        if(kazandi()==1)

            throw kazanma(1);

        else
if(kazandi()==2)

            throw kazanma(2);

    }

    else

        cout << "bu koordinata sembol yerlestirilmistir"<<endl;

}

```

Yukarıdaki ikinci haliyle oyna fonksiyonu, sadece x ve y değerleri almaktadır. Bu sayede, sınıf içerisinde bulunan sıraya göre hamle yapılmaktadır. Elbette her hamleden sonra sıranın değiştirilmesi gerekiyor. Bunun için sıra değişkeninin 2ye bölümünden kalan alınıp değeri bir arttırılıyor (sıra değerinin 0 olamayacağını, 0 değerinin oynanmamış alanları temsil ettiğini hatırlayınız).

Yukarıdaki kodda bulunan özel bir durum ise, kazanma koşulunu kontrol etmektedir. Şayet X kazandıysa 1 veya O kazandıysa 2 kontrolü yapıp ilgili istisna (exception) fırlatılmaktadır.

Bu işlem sırasında kendi yazdığımız istisna sınıfını (exception class) kullanıyoruz. Yani yukarıdaki kodda bulunan kazanma sınıfı bizim kodladığımız bir sınıf. Bu sınıfın kodu aşağıda verilecek.

Son olarak Tahta sınıfında, kazandı kontrolü yapılması için aşağıdakine benzer bir kod yazılabilir:

```
int Tahta::kazandi(){  
  
    if((tahta[0][0].getSembol()==1      &&tahta[0][1].getSembol()==1      &&tahta[0]  
[2].getSembol()==1 )  
  
        ||(tahta[1][0].getSembol()==1      &&tahta[1][1].getSembol()==1      &&tahta[1]  
[2].getSembol()==1 )  
  
        ||(tahta[2][0].getSembol()==1      &&tahta[2][1].getSembol()==1      &&tahta[2]  
[2].getSembol()==1 )  
  
        ||(tahta[0][0].getSembol()==1      &&tahta[1][0].getSembol()==1      &&tahta[2]  
[0].getSembol()==1 )  
  
        ||(tahta[0][0].getSembol()==1      &&tahta[1][0].getSembol()==1      &&tahta[2]  
[0].getSembol()==1 )  
  
        ||(tahta[0][1].getSembol()==1      &&tahta[1][1].getSembol()==1      &&tahta[2]  
[1].getSembol()==1 )  
  
        ||(tahta[0][2].getSembol()==1      &&tahta[1][2].getSembol()==1      &&tahta[2]  
[2].getSembol()==1 )  
  
        ||(tahta[0][0].getSembol()==1      &&tahta[1][1].getSembol()==1      &&tahta[2]  
[2].getSembol()==1 )  
  
        ||(tahta[0][2].getSembol()==1      &&tahta[1][1].getSembol()==1      &&tahta[2]  
[0].getSembol()==1 ))  
  
        return 1;  
  
    else  
if((tahta[0][0].getSembol()==2      &&tahta[0][1].getSembol()==2      &&tahta[0]  
[2].getSembol()==2 )  
  
        ||(tahta[1][0].getSembol()==2      &&tahta[1][1].getSembol()==2      &&tahta[1]  
[2].getSembol()==2 )
```

```

        ||(tahta[2][0].getSembol()==2    &&tahta[2][1].getSembol()==2    &&tahta[2]
[2].getSembol()==2 )

        ||(tahta[0][0].getSembol()==2    &&tahta[1][0].getSembol()==2    &&tahta[2]
[0].getSembol()==2 )

        ||(tahta[0][0].getSembol()==2    &&tahta[1][0].getSembol()==2    &&tahta[2]
[0].getSembol()==2 )

        ||(tahta[0][1].getSembol()==2    &&tahta[1][1].getSembol()==2    &&tahta[2]
[1].getSembol()==2 )

        ||(tahta[0][2].getSembol()==2    &&tahta[1][2].getSembol()==2    &&tahta[2]
[2].getSembol()==2 )

        ||(tahta[0][0].getSembol()==2    &&tahta[1][1].getSembol()==2    &&tahta[2]
[2].getSembol()==2 )

        ||(tahta[0][2].getSembol()==2    &&tahta[1][1].getSembol()==2    &&tahta[2]
[0].getSembol()==2 ))

        return 2;

    else

        return 0;

}

```

Yukarıdaki bu kod, 16 ayrı kazanma koşulunu kontrol etmektedir. Kazanan kişinin 1 veya 2 olmasına göre de sonucu döndürmektedir.

Yukarıdaki ikinci oyna metodunda bulunan istisna sınıfı ise aşağıdaki şekilde kodlanabilir:

```

class kazanma{

private:

    int kim;

public:

    kazanma(int x){

        kim = x;

    }

    int getKim(){

        return kim;

    }

}

```



```
};
```

Son olarak, yukarıdaki kodlarımızı çalıştıracak olan main fonksiyonunu aşağıdaki şekilde kodlayalım:

```
#include  
"hamle.h"
```

```
#include  
"tahta.h"
```

```
#include  
<iostream>
```

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

```
using  
namespace std;
```

```
int main(){
```

```
    Tahta t = Tahta();
```

```
    try{
```

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

```
            int x,y;
```

```
            cin>>x;
```

```
            cin>>y;
```

```
            t.oyna(x,y);
```

```
            t.bas();
```

```
        }
```

```
    }catch(kazanma k){
```

```
        t.bas();
```

```
        if(k.getKim()==1)
```

```
        cout << "kazanan: X" << endl;

    else

        cout << "kazanan: 0" << endl;

    }

    getch();

}
```

Yukarıdaki kodda, 9 hamlenin yapıldığı bir for döngüsü bulunuyor. Bu döngü içerisinde, kullanıcıdan x ve y değerleri okunuluyor ve ardından bu okunan değerlere Tahta sınıfından tanımlanan t değişkeni ile oynanıyor. Burada kullanılan oyna fonksiyonu, sırayı almamaktadır. Dolayısıyla Tahta sınıfındaki tanımlı olan sırayı kullanmakta ve bu sıra otomatik olarak her hamleden sonra değiştirilmektedir.

Ayrıca bu döngü içerisindeki oynama işlemi sırasında herhangi bir taraf kazanacak olursa, kazanan kişi için bir istisna fırlatılacak ve bu fırlatılan istisna, yukarıdaki kodda bulunan catch bloğu ile yakalanacaktır. Burada yakalanan istisna, bizim tanımladığımız kazanma sınıfından olduğu için, sınıfın içerisinde tutulan kimin kazandığı bilgisini alıp buna göre uygun kazanma mesajını ekrana basabiliyoruz.

SORU 9: Visual Basic ile Gösterici (Pointer) Kullanımı

Sitede gelen bir soru üzerine bu yazıyı yazmaya karar verdim. Bilgisayar dilleri (makine işlemeli diller, machine processing languages) tasnif edilirken, visual basic gibi görsel tasarıma dayalı diller üst seviye dil (high level language) olarak kabul edilirler. Hatta hiç kod yazmadan program üretilmesine izin verdiği için visual basic'i bir dilden çok görsel geliştirme ortamı olarak kabul eden (basic'i dil ancak visual basic'i bir geliştirme ortamı) tasnifler söz konusudur.

[Göstericiler \(pointers\)](#) ise tam tersine düşük seviyeli dillerde daha çok ihtiyaç duyulan ve programın geliştirildiği bilgisayarın hafızasına (ram) doğrudan erişim yapan yapılardır. Dolayısıyla tanım itibarıyla visual basic ve [gösterici \(pointer\)](#) kavramları birbiri ile uyumsuzlar. Ancak .Net teknoloji ile birlikte nesne yönelimli programlama (object oriented programming) yapısı kazanan adeta kabuk değiştiren visual basic ile gösterici kullanımı ihtiyacı artmıştır.

Nesne yönelimli dillerin çoğunda (örn. JAVA) doğrudan gösterici kullanımı bulunmaz. Bunun yerine [nesne atfı ismi verilen \(object referrer\)](#) yapılar kullanılır ve bu yapılar aslında birer göstericidir.

Visual basic de nesne yönelimli programlama ile birlikte bu nesne atıflarını içermiş ve artık gösterici yapısını nesne yönelimli olarak içermiştir ancak biz bu yazı kapsamında, nesne

yönelimli programların bize sunduğu bu imkanı bir kenara bırakarak, saf visual basic dilinde göstericileri nasıl kullanabileceğimizi irdeleyeceğiz.

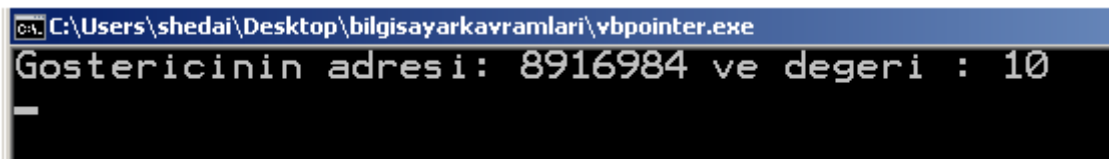
Konuyu aşağıdaki örnek üzerinden anlamaya çalışalım.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <conio.h>
4  //www.bilgisayarkavramlari.com
5  int main()
6  {
7      int *p;
8      p=(int *)malloc(sizeof(int));
9      *p=10;
10     printf("Gostericinin adresi: %d ve degeri : %d \n",p,*p);
11     free(p);
12     getch();
13     return 0;
14 }
```

Yukarıda, C dilinde yazılmış basit bir gösterici kodu bulunmaktadır. Kodumuzun 7. Satırında p isimli bir tam sayı göstericisi (integer pointer) tanımlanmış, 8 . satırda bu göstericinin gösterdiği yerde 1 tam sayı konulacak bir yer hafızada ayrılmış (memory allocation) ve 9. Satırda bu göstericinin gösterdiği adrese 10 değeri konulmuştur.

Kodun 10. Satırında göstericinin adresi (hafızada nereyi gösterdiği) ve gösterdiği yerdeki değerler ekrana basılmış ve 11. Satırda gösterici için ayrılan hafıza bölmesi serbest bırakılmıştır (free).

Yukarıdaki bu kodun çalışan hali aşağıda verilmiştir:



```
C:\Users\shedai\Desktop\bilgisayarkavramlari\vbpointer.exe
Gostericinin adresi: 8916984 ve degeri : 10
```

Şimdi yukarıdaki bu basit C dili ile yazılmış gösterici kodunu Visual Basic dili ile kodlamaya çalışalım.

```

18 Private Sub Form1_Load(ByVal sender As System.Object,
19     Dim p As Long
20     Dim hHeap As Long
21     hHeap = GetProcessHeap()
22     p = HeapAlloc(hHeap, 0, 2)
23     If p <> 0 Then
24         Dim i As Integer
25         i = 10
26         CopyMemoryWrite(p, i, 2)
27         Dim j As Integer
28         CopyMemoryRead(j, p, 2)
29         MsgBox("Gostericinin adresi: " & CStr(p) & _
30             vbCrLf & " ve degeri: " & CStr(j))
31         HeapFree(GetProcessHeap(), 0, p)
32     End If
33 End Sub

```

Yukarıdaki kodu inceleyecek olursak, yapı olarak bir form yükleme fonksiyonuna yazılmış kodumuz, p ve hHeap isimli iki adet Long tipinde değişken tanımlayarak başlıyor. Buradaki p değişkenini bir gösterici olarak kullanacağız. Dikkat edilmesi gereken husus, p değişkeninde gösterici mantığında olduğu üzere bir hafıza bölmesinin adresinin tutulmasıdır. Yani C dilinde olan p göstericisinde nasıl hafızanın bir adresi tutuluyorsa, visual basic örneğinde de hafıza adresini tutmaya çalışacağız.

Öncelikle çalışan programımızın yani [işlemin \(process\)](#) çalıştığı alanı tutan hHeap isimli değişkenimizden HeapAlloc fonksiyonu ile (C dilindeki malloc benzeri) hafızada bir alan ayırması yapıyor.

Bu ayırma sırasında 0'dan başlanarak 2 boyutunda ayırma işlemi yapılmıştır. 2 sayısı, visual basic dilindeki bir tamsayının (integer) hafızada kapladığı alan boyutudur.

Sonuçta kodun 22. Satırı itibariyle, p isminde bir değişken ve bu değişkenin içinde, hafızada bize ayrılmış 2 byte uzunluğunda bir alanın adresi bulunmaktadır.

Ardından 23. Satırda böyle bir adres ayırımı başarılı olduysa, artık bu adres alanına bir integer değer koymayı deneyebiliriz.

24. satırda i isimli bir tam sayı değişkeni tanımlanıyor, 25. Satırda bu değişkene sayısal olarak 10 değeri konuluyor ve sonunda 26. Satırda, p göstericimizin gösterdiği adrese, i değişkeninin içindeki değeri konuluyor. Bu sırada CopyMemoryWrite fonksiyonu kullanılmıştır. Bu fonksiyon kernel32 dinamik kütüphanesi içinde bulunan [RtlMoveMemory](#) fonksiyonudur. Dolayısıyla bu fonksiyonu kullanmak için aşağıdakine benzer bir şekilde kodun başında tanım yapılmalıdır:

```

Private Declare Sub CopyMemoryWrite Lib "kernel32" Alias _
    "RtlMoveMemory" (ByVal Destination As Long, _
    ByVal Source As Long, ByVal Length As Long)

```

Ardından j isimli bir deęişken tanımlanarak, p göstericimizin gösterdiği hafıza alanına daha önceden (kodun 26. Satırında) yazdığımız deęeri okumaya çalışıyoruz. Bunun içinde yine kernel32.dll kütüphanesinden tanımladığımız fonksiyonumuzu kullanıyoruz.

Sonuç bir mesaj kutusu içerisinde gösteriliyor ve nihayetinde 31. Satırda bu ayrılan hafıza alanı serbest bırakılmaktadır.

SORU 10: C ile Programlamaya Giriş Quiz Soruları ve Çözümleri

Soru 1) Bir dosyaya isminizi yazdırınız.

Çözüm 1) dosyaya ismimizi yazdıran kodu aşağıdaki şekilde yazabiliriz:

```
1  #include <stdio.h>
2  #include <conio.h>
3
4  int main(){
5      FILE *fp;
6      fp = fopen("dosya.txt", "w");
7      fprintf(fp, "Sadi Evren SEKER");
8      fclose(fp);
9  }
```

Yukarıdaki kodda dikkat edilecek bir husus, dosyanın “r” deęil “w” şekliyle açılması ve dosyanın mutlaka kapatılmasıdır. Bilindięi üzere kapatılmayan dosyalara yazıldığından kesin olarak emin olamayız ve dosya kapatılmadıysa içerisine bilgi yazılmamış olabilir.

Yukarıdaki kod çalıştırıldıktan sonra aynı dizin içerisinde “dosya.txt” isimli bir dosya oluşturulur ve bu dosyanın wordpad benzeri bir programla açılması durumunda içinde ismimizin yazdığını görebiliriz.



Soru 2) Klavyeden okunan bir sayının asal çarpanlarını ekrana basan kodu yazınız.

Çözüm 2)

```

1  #include <stdio.h>
2  #include <conio.h>
3  //www.bilgisayarkavramlari.com
4  int main(){
5      int sayi;
6      printf("bir sayi giriniz");
7      scanf("%d",&sayi);
8      for(int i = 2;sayi>1;i++){
9          while(sayi%i==0){
10             printf("%d ",i);
11             sayi = sayi / i;
12         }
13     }
14     getch();
15 }

```

Yukarıdaki kodda görüldüğü üzere önce bir sayı okunmuş (kodun 7. Satırı) ardından bir döngü içerisinde, 2'den başlayarak bütün sayılara bu girilen sayıyı bölme denenmiştir. For döngüsünün (8. Satır) koşulu, sayının 1'den büyük olmasıdır. Dolayısıyla sayı 1 olunca yani bütün asal sayılara bölümü tamamlanınca çalışma duracaktır.

Yukarıdaki kodda bulunan iç döngü (9. Satırdaki while döngüsü) ise bir asal sayının birden fazla kere bölme ihtimaline karşı yazılmıştır. Örneğin 60 girdisi için 2 asal sayısı, 2 kere bölebilir. Bu durumda bir sonraki asal sayı ile deneme yapılmadan önce, şu anda bölmeyi denediğimiz asal sayı ile bölünmeyinceye kadar deneme yapıyoruz.

Kodun 11. Satırında ise, denediğimiz sayının, girilen sayıyı tam bölmesi durumunda bölerek sayıyı küçültüyor ve başarılı olarak bölebildiğimiz bu sayıyı bir çarpan olarak ekrana yazıyoruz.

Kodun örnek çalışması aşağıdaki şekildedir.

```

C:\Users\shedai\Desktop\bilgisayarkavramlari\quiz\Untitled2.exe
bir sayi giriniz120
2 2 2 3 5 _

```

Soru 3) Bir dizgideki kelime sayısını ekrana bastıran kod yazınız.

Çözüm 3)

```

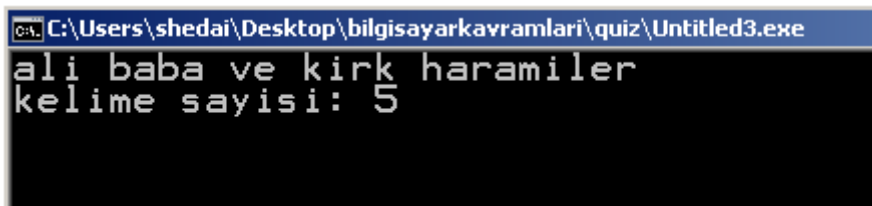
1  #include <stdio.h>
2  #include <conio.h>
3  //www.bilgisayarkavramlari.com
4  int main(){
5      char dizgi[100];
6      gets(dizgi);
7      int kelimesayisi=1;
8      for(int i = 0;dizgi[i]!='\0';i++){
9          if(dizgi[i]==' '){
10             kelimesayisi++;
11         }
12     }
13     printf("kelime sayisi: %d",kelimesayisi);
14     getch();
15 }
16

```

Soruda, girilen bir [dizgideki \(String\)](#) kelime sayısının saydırılması istendiği için, dizginin scanf ile okunması mümkün değildir. Bilindiği üzere scanf tek bir kelime okur ve bu durumda işe yaramaz. Birden fazla kelimenin boşlukları ile okunması istenen durumda gets fonksiyonu kullanılabilir. Yukarıdaki kodun 6. Satırında da bu şekilde dizgi okunmuştur. Ardından dizgide bulunan boşluklar dizgi sonuna kadar saydırılmıştır. Dizgi sonu kontrolü 8. Satırda bulunan for döngüsünde yapılırken, dizgi içerisinde o anda bakılan karakterin boşluk karakteri olup olmadığı 9. Satırdaki if kontrolü ile yapılmıştır. Şayet boşluk karakteri ise, kelimesayısı değişkeni 1 arttırılmıştır.

Basitçe hesaplanacağı üzere, bir dizgideki boşluk sayısı, kelime sayısından 1 eksik olur. Bu problemin çözümü için kelimesayısı değişkeni 0 yerine 1'den başlatılmıştır.

Kodun çalışan hali aşağıda verilmiştir.



```

C:\Users\shedai\Desktop\bilgisayarkavramlari\quiz\Untitled3.exe
ali baba ve kirk haramiler
kelime sayisi: 5

```

Soru 4) Bir dizgideki en uzun kelimeyi bularak ekrana bastıran kodu yazınız

Çözüm 4)

```

1  #include <stdio.h>
2  #include <conio.h>
3  //www.bilgisayarkavramlari.com
4  int main(){
5      char dizgi[100];
6      gets(dizgi);
7      char *kelimebasi;
8      kelimebasi = &dizgi[0];
9      int kelimeboyu = 0;
10     char *yenikelime ;
11     yenikelime = &dizgi[0];
12     int yeniboy = 0;
13     for(int i = 0; dizgi[i] != '\0'; i++){
14         yeniboy ++;
15         if(dizgi[i] == ' '){
16             if(yeniboy > kelimeboyu){
17                 kelimeboyu = yeniboy;
18                 kelimebasi = yenikelime;
19             }
20             yeniboy = 0;
21             dizgi[i] = '\0';
22             i++;
23             yeniboy ++;
24             yenikelime = &dizgi[i];
25         }
26     }
27     if(yeniboy > kelimeboyu){
28         kelimeboyu = yeniboy;
29         kelimebasi = yenikelime;
30     }
31     printf("en uzun kelime : %s", kelimebasi);
32     getch();
33 }

```

Yukarıdaki kod, 3. Sorudaki koda benzer şekilde, dizgi sonuna kadar bütün karakterleri kontrol etmekte ve boşluk karakterlerinde kodun 15. Satırında bulunan if kontrolüne girmektedir. Bu kodda farklı olan bir özellik, iki adet [karakter göstericisi \(character pointer\)](#) ile dizgi üzerinde işaretleme yapılmasıdır.

Bu göstericilerden kelimebasi isimli gösterici, o ana kadar bulunan en uzun kelimeyi göstermekte, yenikelime göstericisi ise, üzerinden geçilen son kelimeyi göstermektedir.

Kodun çalışmasını aşağıdaki şekilde anlayabiliriz:



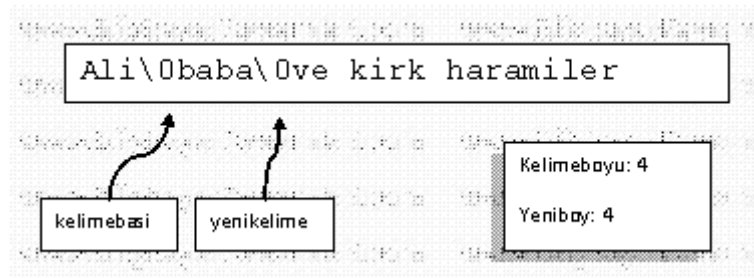
Yukarıda gösterildiği üzere, ilk başta iki gösterici de dizginin ilk karakterini göstermektedir. Ayrıca kelimeboyu ve yeniboy değişkenlerinin değeri 0'dır.

Ardından kodumuz döngü içerisinde karakter karakter ilerlemekte ve ilk boşluğu gördüğü anda aşağıdaki şekilde yeni kelime değişkeni hareket ettirilmektedir.

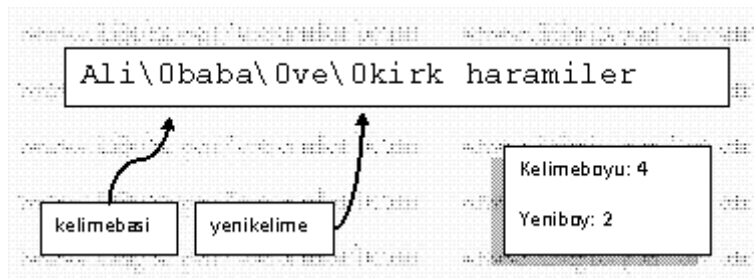


İlk boşluğa gösterici hareket ettirildikten sonra boşluğa kadar olan harf sayısı sayılmakta ve yeniboy 3 olmaktadır.

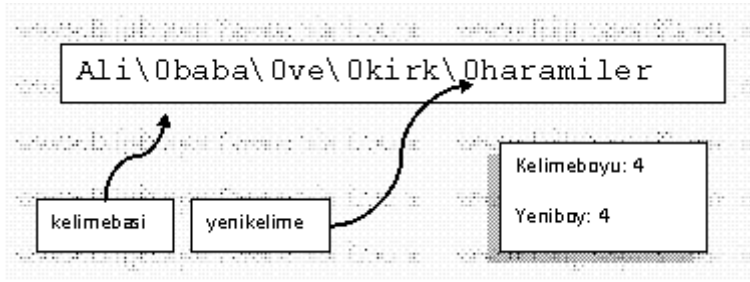
Tekrar bir boşluk karakterine kadar göstericimiz hareket eder:



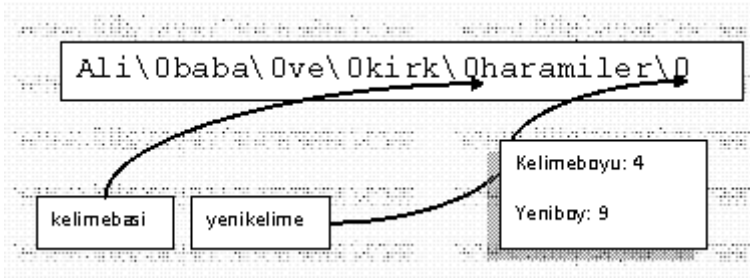
Bulduğumuz yeni kelimenin boyu 4 olduğu ve şimdiye kadar bulduğumuz en uzun kelimedenden daha uzun olduğu için kelimebaşı göstericisini ilerletiyor ve yukarıdaki şekilde bulunan yeni kelimenin ilk karakterine atıyoruz, ayrıca yenikelime göstericisi, mevcut konuma ilerletiliyor ve bir sonraki kelimenin aranmasında kullanılıyor.



Yeni bulunan kelime boyu, o ana kadar bulunan kelime boyundan daha küçük olduğu için kelimebaşı değişkenimiz hala o ana kadar bulunmuş en uzun kelimeyi göstermeye devam ediyor.



Bir önceki örnekte olduğu gibi, bulunan yeni kelimenin boyutu, ilk kelimedenden büyük olmadığı için değişkenimiz ilerlemiyor.



Son olarak bulduğumuz yeni kelimenin boyu, şimdiye kadar bulunan kelimelerden uzun olduğu için, göstericiyi ilerletiyor ve kelimebasi olara bu kelimenin ilk harfini gösteriyoruz. Sonuçta dizgi sonuna kadar giderken, anlık olarak bir kelimenin boyutu değişkende tutuluyor, bir kelimenin boyutunun bu değişkende tutulan boyuttan daha uzun olması durumunda bu kelimeyi gösteren bir gösterici işaretleniyor ayrıca bu yeni kelimenin boyutu, elimizde o ana kadar bulunan en uzun kelime boyutu olarak tutuluyor.

Kodun çıktısı aşağıdaki şekildedir:

```
C:\Users\shedai\Desktop\bilgisayarkavramlari\quiz\Untitled4.exe
ali baba ve kirk haramiler
en uzun kelime : haramiler
```

Soru 5) Kullanıcıdan bir sayı okuyarak, okunan bu sayı boyutlarında bir kare matrisi, her hücreesine, en yakın kenara olan uzaklığı ile doldurunuz.

Çözüm 5)

Soruda her hücreye, en yakın kenara olan uzaklık yazılması istenmiş, bu durumda örneğin 5×5 boyutunda bir matris için aşağıdaki şekilde veri girilmesi gerekir:

1	1	1	1	1
1	2	2	2	1
1	2	3	2	1
1	2	2	2	1
1	1	1	1	1

Görüldüğü üzere kenar değerlerine 1 , bir içerdeki hücrelere 2 ve en ortadaki hücreye 3 yerleştirilmesi istenmiş.

Bu dorunun çözümünde sadece döngüler kullanılarak bir döngünün kenara olan uzaklığı hesaplanıp ekrana yazdırılabilir. Bu yöntemin çözümü aşağıdaki şekildedir:

```
1  #include <stdio.h>
2  #include <conio.h>
3  //www.bilgisayarkavramlari.com
4  int min(int a,int b, int c,int d){
5      if (a<=b&&a<=c&&a<=d)
6          return a;
7      if(b<=a&&b<=c&&b<=d)
8          return b;
9      if(c<=a&&c<=b&&c<=d)
10         return c;
11     if(d<=a&&d<=b&&d<=c)
12         return d;
13 }
14 int main(){
15     printf("matrisin boyutunu giriniz:");
16     int boyut ;
17     scanf("%d",&boyut);
18     for(int i = 1;i<=boyut;i++){
19         for(int j = 1 ; j <=boyut;j++){
20             int a= boyut -i+1;
21             int b= boyut -j+1;
22             printf("%d ",min(i,j,a,b));
23         }
24         printf("\n");
25     }
26     getch();
27 }
28 }
```

Yukarıdaki kodda görüldüğü üzere, her hücre için, hücrenin kenara olan 4 farklı uzaklığı hesaplanmıştır. Bu değerlerden iki tanesi, koordinatları olan i ve j değerleridir ki bu değerler, sol kenara ve üst kenara olan uzaklık olarak düşünülebilir.

Ayrıca sağ kenar ve alt kenara olan uzaklıkların hesaplanması kodun 20. Ve 21. Satırlarında bulunan a ve b değişkenleri ile yapılmıştır. Matrisin boyutundan koordinat çıkarılmış ve 1 ilave edilmiştir.

Sonuçta elimizde 4 farklı kenara olan 4 farklı uzaklık bulunmaktadır. Bizden istenen bu uzaklıklardan en küçüğünü ekrana basmamız. Bu durumda kodun 4-13 satırları arasında bulunan ve verilen 4 sayıdan en küçüğünü döndüren fonksiyonu çağırarak bu 4 uzaklıktan en küçüğünü bulabiliriz.

Kodun çalışan hali aşağıdaki şekildedir:

```
C:\Users\shedai\Desktop\bilgisayarkavramlari\quiz\Untitled11.exe
matrisin boyutunu giriniz:8
1 1 1 1 1 1 1 1
1 2 2 2 2 2 2 1
1 2 3 3 3 3 2 1
1 2 3 4 4 3 2 1
1 2 3 4 4 3 2 1
1 2 3 3 3 3 2 1
1 2 2 2 2 2 2 1
1 1 1 1 1 1 1 1
```

SORU 11: JAVA ile Zar uygulaması

Gelen bir soru üzerine aşağıdaki yazıyı yazıyorum. Soru şu şekilde:

“Konsol programında rastgele zar atan ve çıkan zarı ekrana bastıran kodu yazınız. “

Konsol ekranında zar çizdirmek için öncelikle bir tasarım yapmamız gerekiyor. Konsol ekranında sadece [ascii karakterlerini](#) basabileceğimizi düşünürsek, zarları aşağıdaki şekilde [ascii ekranda](#) göstermemiz mümkün olabilir.

```
1
2 *
3
4 *
5
6 *
7 *
8 *
9 *
10 * *
11
12 * *
13 * *
14 *
15 * *
16 * *
17 * *
18 * *
```

Yukarıdaki her 3 satırda bir zar resmi olduğunu düşünelim. Örneğin zar ile 4 atıldığında 10 -12 satırlar arasındaki karakterler ekrana basılacak olsun.

Şimdi java ile rast gele sayı üretme işi ile ilgilenebiliriz. Bunun için java dilinde bulunan Math.random() fonksiyonunu kullanmamız gerekiyor.

Öncelikle zarları yukarıdaki şekillerde basacak bir program yerine zarın sonucunu doğrudan sayı olarak ekrana basan bir kod yazmaya çalışalım:

İlk denememizde Math.random() fonksiyonunu çalıştırmaya çalışıyor ve aşağıdaki hatalı kodu yazıyoruz:

```

1 // www.bilgisayarkavramlari.com
2 public class zar{
3     public static void main(String args[]){
4         int z = Math.random();
5     }
6 }

```

Yukarıdaki kodu çalıştırdığımızda doğal olarak aşağıdaki hatayı alıyoruz:

```

C:\Users\shedai\Desktop\bilgisayarkavramlari>javac zar.java
zar.java:3: possible loss of precision
found   : double
required: int
        int z = Math.random();
                        ^
1 error
C:\Users\shedai\Desktop\bilgisayarkavramlari>

```

Görüldüğü üzere bize Math.random() fonksiyonunun bir double değeri döndürdüğü ve bir int değerinin içerisine konulamayacağı hatası geliyor. Gerçekten de ilgili java API'si okunduğunda Maht.random() fonksiyonunun 0 ile 1 arasında bir sayı döndürdüğü görülebilir.

Zar değerimiz 1 ile 6 arasında olacağına göre (0 yok 1den başlayacağız), rastgele sayımızı 5 ile çarpıp 1 ekliyoruz.

Bu durum ilk kez rast gele sayılar ile uğraşırken karışık görülebilir ancak şöyle anlatmaya çalışalım

$0 < a < 1$

Aralığında olan a sayısı (ki bu değer aslında Math.random() fonksiyonunun orijinal olarak ürettiği değerdir)

a * 5 yapıldığında:

$0 < a < 5$

Aralığında olur. a+1 yapıldığında da

$1 < a < 6$

Aralığında olmuş olur.

Elbette bu değer aralığına getirdikten sonra da Math.random() fonksiyonu hâlâ double değer üretmektedir. Çözüm olarak [tip inkılabı yapmak \(type casting\)](#) ve tipini int değerine çevirmek gerekir. Öyleyse ilgili satır aşağıdaki şekle çevrilmelidir:

```
int z = (int) Math.random() * 5 + 1
```

Kodun çalışan hali ise aşağıda verilmiştir:

```

1 // www.bilgisayarkavramlari.com
2 public class zar{
3     public static void main(String args[]){
4         int z = (int) (Math.random()*5 +1);
5         System.out.println(z);
6     }
7 }

```

Kodda dikkat edilirse 4. Satırda `Math.random()*5+1` ibaresi parantez içerisine alınmıştır. Bunun sebebi başta bulunan ve tip inkılabı yapan `(int)` komutunun önce `Math.random` değerini `int` yapmasıdır. Dolayısıyla sonuçlar her zaman için 1 çıkacaktır (`Math.random` her zaman 0 ile 1 arasında değer üretecek, `int` çevriminden 0 olacak 5 ile çarpılıp 1 eklenince de hep 1 sonucu çıkacaktır)

Kodun çalışan hali aşağıdaki şekildedir:

```

C:\Users\shedai\Desktop\bilgisayarkavramlari>javac zar.java
C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
4
C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
4
C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
3
C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
1
C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
5

```

Son olarak çizimi yaptıran kısma geçebiliriz. Çıkan sonuca göre ekrana farklı bir değer bastırmamız gerekiyor.

Kodu aşağıdaki şekilde yazabiliriz:

```

1 // www.bilgisayarkavramlari.com
2 public class zar{
3     public static void main(String args[]){
4         int z = (int) (Math.random()*5 +1);
5         if(z==1) System.out.println("\n * \n \n");
6         if(z==2) System.out.println("* \n \n * \n");
7         if(z==3) System.out.println("* \n * \n * \n");
8         if(z==4) System.out.println("** \n \n * \n");
9         if(z==5) System.out.println("** \n * \n * \n");
10        if(z==6) System.out.println("** \n * \n * \n");
11    }
12 }

```

Kodun çalışan hali aşağıda verilmiştir:

```

C:\Users\shedai\Desktop\bilgisayarkavramlari>javac zar.java
C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
* *
* *

C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
*
*

C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
* *
* *

C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
* *
* *

C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
* *
* *
* *

C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
*
*

```

Yukarıdaki kod, hedefimize ulaşmamızı sağlamasına karşılık kodda ufak bir değişiklik yapmamız mümkündür. Kodda 9×9 bir matris üzerinde * karakteri ile kodlama yaptığımıza dikkat etmişsinizdir. Bu değerlerin basıldığı durumları gösterecek olursak:

>=2	Hiçbir zaman	>=4
=6	Zar tek sayı ise	=6
>=4	Hiçbir zaman	>=2

Görüldüğü üzere zar değerine bağlı olarak bazı karelerde * sembolü olmakta ve bazılarında olmamaktadır. Dolayısıyla kodu aşağıdaki şekilde yazmak da mümkün olabilir:

```

1 // www.bilgisayarkavramlari.com
2 public class zar{
3     public static void main(String args[]){
4         int z = (int) (Math.random()*5 +1);
5         if(z>=2) System.out.print(" *");
6         else     System.out.print(" ");
7         System.out.print(" ");
8         if(z>=4) System.out.println(" *");
9         else     System.out.println(" ");
10        if(z==6) System.out.print(" *");
11        else     System.out.print(" ");
12        if(z%2==1) System.out.print(" *");
13        else     System.out.print(" ");
14        if(z==6) System.out.println(" *");
15        else     System.out.println(" ");
16        if(z>=4) System.out.print(" *");
17        else     System.out.print(" ");
18        System.out.print(" ");
19        if(z>=2) System.out.println(" *");
20        else     System.out.println(" ");
21    }
22 }

```

Kodun çalışması sonucu aşağıda verilmiştir:

```

C:\Users\shedai\Desktop\bilgisayarkavramlari>javac zar.java
C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
*
*
C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
*
C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
*
C:\Users\shedai\Desktop\bilgisayarkavramlari>java zar
* *
*
* *

```

SORU 12: Ekranı Kare Çizdiren Kod

Soru: Kullanıcıdan bir sayı alarak ekrana verilen sayı boyutlarında *'lardan oluşan içi dolu bir kare kodu yazınız.

Çözen : Şadi Evren ŞEKER

Yukarıdaki soruyu 4 ayrı dil için (C,C++,JAVA ve C#) ayrı ayrı çözeceğim. Böylelikle okuyucu bu diller arasında mukayese yapabilecek ve bildiği bir dilden diğer dillere kolayca geçebilecektir.

Çözüm:

Soru, yapısı itibariyle 2 boyutlu bir şekil gerektirmektedir. 2 boyutlu bir şekil için 2 adet [iç içe döngü \(nested loop\)](#) kullanılması yeterlidir. Buradaki amaç, döngülerden birisinin yatay diğerinin düşey ekseninde çalışmasıdır. Sorunun a şıkkı için JAVA dilinde çözüm aşağıdaki şekilde yazılabilir:

```
import java.util.Scanner;
public
class kare{
public
static
void main(String args[]){
    Scanner in = new Scanner(System.in);
    int boyut;
    System.out.println("Karenin boyutlarını giriniz: ");
    boyut = in.nextInt();
    for(int i = 0;i<boyut;i++){ //her satir icin
        for(int j = 0;j<boyut;j++){ // boyut kadar yildiz bas
            System.out.print("*");
        }
        System.out.println(); // alt satira gec
    }
}
}
```

Yukarıdaki kodun çalışan hali aşağıdaki şekildedir:

```
İİİİKarenin                    boyutlarını                    giriniz:
¼¼¼¼İİİİ5
İİİİ*****
İİİİ*****
İİİİ*****
İİİİ*****
İİİİ*****
```

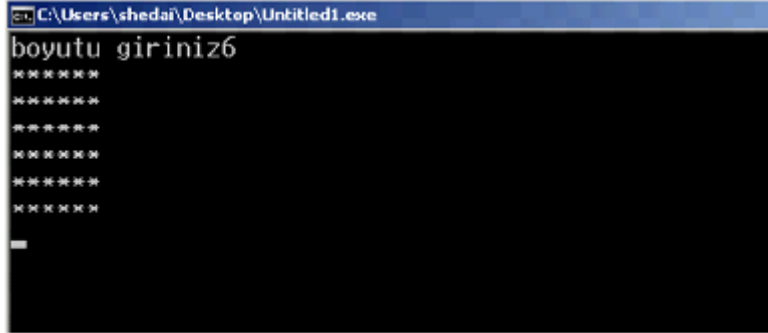
Benzer uygulamayı C dilinde aşağıdaki şekilde yazabiliriz:

```
#include <stdio.h>
#include <conio.h>
int main () {
    printf("boyutu giriniz");
    int boyut;
    scanf("%d",&boyut);
    for(int i = 0;i<boyut;i++){ //her satir icin
        for(int j = 0;j<boyut;j++){ // boyut kadar yildiz bas
            printf("*");
        }
        printf("\n"); // alt satira gec
    }
}
```

```
    getch();  
    return 0;  
}
```

Yukarıdaki kodun örnek ekran çıktısı da aşağıda verimiştir:

www.BilgisayarKavamlari.com



Yukarıdaki soruyu C++ dilinde çözecek olursak:

```
#include <iostream.h>  
int main () {  
    cout << "boyutu giriniz" ;  
    int boyut;  
    cin >> boyut;  
    for(int i = 0; i<boyut; i++){ //her satir icin  
        for(int j = 0; j<boyut; j++){ // boyut kadar yildiz bas  
            cout << "*" ;  
        }  
        cout << endl; // alt satira gec  
    }  
    system("PAUSE");  
    return 0;  
}
```

Yukarıdaki kod Dev-CPP ile kodlanmıştır ve ekran çıktısı aşağıdaki şekildedir:



Son olarak aynı soruyu C# dilinde yazacak olursak:

```
using System;  
  
using System.Collections.Generic;  
  
using System.Linq;  
  
using System.Text;
```

```
namespace ConsoleApplication1

{

    class
    Program

    {

        static
        void Main(string[] args)

        {

            Console.WriteLine("boyutu giriniz");

            int boyut;

            boyut = Console.Read();

            boyut = boyut - 48;

            // ascii int çevir (0-9 arasi için)

            for (int i = 0; i < boyut; i++)

            { //her satir için

                for (int j = 0; j < boyut; j++)

                { // boyut kadar yildiz bas

                    Console.Write("*");

                }

                Console.WriteLine(); // alt satira gec

            }

        }

    }

}
```



SORU 13: Matrisin tersinin alınması (Mantrix Inverse)

Bir matrisin tersini almak çarpma işlemine göre tersini almak anlamındadır. Örneğin A ve B matrislerinin çarpımından C matrisi çıkıyorsa; C matrisi ile A matrisinin tersi B matrisini vermelidir. Bunu bir nevi bölme olarak düşünmek de mümkündür.

2 boyutlu bir matrisin tersini bulmak için öncelikle determinantı hesaplanır ardından aşağıda gösterildiği üzere elemanları yer değiştirilir:

$$A \equiv \begin{bmatrix} a & b \\ c & d \end{bmatrix},$$
$$A^{-1} = \frac{1}{|A|} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$
$$= \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}.$$

Yukarda, ilk satırda verilen A matrisinin [determinantı](#) hesaplanmış (|A|) ve gösterilen yerdeğiştirmiş matris ile skalar (scalar) çarpım yapılmıştır.

Sonuçta elde edilen matris, orjinal A matrisinin tersidir.

Matris 3×3 boyutlarında olsaydı bu hesaplama aşağıdaki şekilde yapılmalıydı:

$$A \equiv \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix},$$
$$A^{-1} = \frac{1}{|A|} \begin{bmatrix} \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} & \begin{vmatrix} a_{13} & a_{12} \\ a_{33} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix} \\ \begin{vmatrix} a_{23} & a_{21} \\ a_{33} & a_{31} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix} & \begin{vmatrix} a_{13} & a_{11} \\ a_{23} & a_{21} \end{vmatrix} \\ \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} & \begin{vmatrix} a_{12} & a_{11} \\ a_{32} & a_{31} \end{vmatrix} & \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \end{bmatrix}.$$

Görüldüğü üzere matrisin boyutunun değişmesi halinde ters alma işlemi de değişmektedir. Yukarıdaki mantık ile örneğin 4×4'lük bir matris'in tersi alınırken önce 3×3 alt matrislerinin yazılması ardından her bir alt matrisin determinantının yukarıdakine benzer şekilde yerleştirilmesi gerekir.

4×4'lük matrisin tersinin alınması (Utku Bey'in isteği üzerine yazıyorum)

matris tersi alınırken, matrisin boyutunun önemi olunmaksızın determinant alınarak işlem yapılır. Örneğin matris boyutu nxn ise bu matriste (n-1)x(n-1) boyutlarındaki alt [matrislerin determinantlarının](#) yer değiştirmiş halleri hesaplanır ve 1/|A| değeri ile skalar çarpım yapılır.

Örneğin matrisimiz aşağıdaki şekilde 4×4 boyutlarında bir matris olsun

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

Yukarıdaki bu matrisin tersini almak için öncelikle determinantını hesaplayalım:

$$\begin{aligned} \det A &= a_{11}a_{22}a_{33}a_{44} + a_{11}a_{23}a_{34}a_{42} + a_{11}a_{24}a_{32}a_{43} \\ &+ a_{12}a_{21}a_{34}a_{43} + a_{12}a_{23}a_{31}a_{44} + a_{12}a_{24}a_{33}a_{41} \\ &+ a_{13}a_{21}a_{32}a_{44} + a_{13}a_{22}a_{34}a_{41} + a_{13}a_{24}a_{31}a_{42} \\ &+ a_{14}a_{21}a_{33}a_{42} + a_{14}a_{22}a_{31}a_{43} + a_{14}a_{23}a_{32}a_{41} \\ &- a_{11}a_{22}a_{34}a_{43} - a_{11}a_{23}a_{32}a_{44} - a_{11}a_{24}a_{33}a_{42} \\ &- a_{12}a_{21}a_{33}a_{44} - a_{12}a_{23}a_{34}a_{41} - a_{12}a_{24}a_{31}a_{43} \\ &- a_{13}a_{21}a_{34}a_{42} - a_{13}a_{22}a_{31}a_{44} - a_{13}a_{24}a_{32}a_{41} \\ &- a_{14}a_{21}a_{32}a_{43} - a_{14}a_{22}a_{33}a_{41} - a_{14}a_{23}a_{31}a_{42} \\ &\neq 0 \end{aligned}$$

Ters alma işlemi için [determinant](#) sıfırdan farklı olmalıdır. Ardından matrisin tersini alalım:

$$\mathbf{A}^{-1} = \frac{1}{\det \mathbf{A}} \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix}$$

Buradaki kapalı şekilde yazılan b değerleri ters alma işleminden sonra yapılan yer değiştirmeyi göstermektedir. Yani örneğin b11 değeri, 1. satır ve 1. sütundaki değerleri kapanmış halde matrisin determinantının alınması ile bulunur.

Benzer şekilde b23 değeri için, orjinal matriste bulunan 2. satır ve 3. sütundaki sayılar dışında kalan sayılardan (ki bu sayılar 3×3 boyutlarında bir matris oluşturur) elde edilen determinant değeridir.

Buradaki b değerleri, matrisin verilen satır ve sütun için kofaktörüdür (cofactor).

Bu değerleri açık şekilde yazacak olursak:

$$\begin{aligned} b_{11} &= a_{22}a_{33}a_{44} + a_{23}a_{34}a_{42} + a_{24}a_{32}a_{43} - a_{22}a_{34}a_{43} - a_{23}a_{32}a_{44} - a_{24}a_{33}a_{42} \\ b_{12} &= a_{12}a_{34}a_{43} + a_{13}a_{32}a_{44} + a_{14}a_{33}a_{42} - a_{12}a_{33}a_{44} - a_{13}a_{34}a_{42} - a_{14}a_{32}a_{43} \\ b_{13} &= a_{12}a_{23}a_{44} + a_{13}a_{24}a_{42} + a_{14}a_{22}a_{43} - a_{12}a_{24}a_{43} - a_{13}a_{22}a_{44} - a_{14}a_{23}a_{42} \\ b_{14} &= a_{12}a_{24}a_{33} + a_{13}a_{22}a_{34} + a_{14}a_{23}a_{32} - a_{12}a_{23}a_{34} - a_{13}a_{24}a_{32} - a_{14}a_{22}a_{33} \\ b_{21} &= a_{21}a_{34}a_{43} + a_{23}a_{31}a_{44} + a_{24}a_{33}a_{41} - a_{21}a_{33}a_{44} - a_{23}a_{34}a_{41} - a_{24}a_{31}a_{43} \\ b_{22} &= a_{11}a_{33}a_{44} + a_{13}a_{34}a_{41} + a_{14}a_{31}a_{43} - a_{11}a_{34}a_{43} - a_{13}a_{31}a_{44} - a_{14}a_{33}a_{41} \\ b_{23} &= a_{11}a_{24}a_{43} + a_{13}a_{21}a_{44} + a_{14}a_{23}a_{41} - a_{11}a_{23}a_{44} - a_{13}a_{24}a_{41} - a_{14}a_{21}a_{43} \\ b_{24} &= a_{11}a_{23}a_{34} + a_{13}a_{24}a_{31} + a_{14}a_{21}a_{33} - a_{11}a_{24}a_{33} - a_{13}a_{21}a_{34} - a_{14}a_{23}a_{31} \\ b_{31} &= a_{21}a_{32}a_{44} + a_{22}a_{34}a_{41} + a_{24}a_{31}a_{42} - a_{21}a_{34}a_{42} - a_{22}a_{31}a_{44} - a_{24}a_{32}a_{41} \\ b_{32} &= a_{11}a_{34}a_{42} + a_{12}a_{31}a_{44} + a_{14}a_{32}a_{41} - a_{11}a_{32}a_{44} - a_{12}a_{34}a_{41} - a_{14}a_{31}a_{42} \\ b_{33} &= a_{11}a_{22}a_{44} + a_{12}a_{24}a_{41} + a_{14}a_{21}a_{42} - a_{11}a_{24}a_{42} - a_{12}a_{21}a_{44} - a_{14}a_{22}a_{41} \\ b_{34} &= a_{11}a_{24}a_{32} + a_{12}a_{21}a_{34} + a_{14}a_{22}a_{31} - a_{11}a_{22}a_{34} - a_{12}a_{24}a_{31} - a_{14}a_{21}a_{32} \\ b_{41} &= a_{21}a_{33}a_{42} + a_{22}a_{31}a_{43} + a_{23}a_{32}a_{41} - a_{21}a_{32}a_{43} - a_{22}a_{33}a_{41} - a_{23}a_{31}a_{42} \\ b_{42} &= a_{11}a_{32}a_{43} + a_{12}a_{33}a_{41} + a_{13}a_{31}a_{42} - a_{11}a_{33}a_{42} - a_{12}a_{31}a_{43} - a_{13}a_{32}a_{41} \\ b_{43} &= a_{11}a_{23}a_{42} + a_{12}a_{21}a_{43} + a_{13}a_{22}a_{41} - a_{11}a_{22}a_{43} - a_{12}a_{23}a_{41} - a_{13}a_{21}a_{42} \\ b_{44} &= a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33} - a_{13}a_{22}a_{31} \end{aligned}$$

şeklinde sayıları sıralayabiliriz.

Yukarıdaki ters alma işlemi kısaca olay iki parçadan oluşuyor, birinci aşamada determinant almanız gerekiyor (nxn boyutundaki bir matris determinantı bilgisayar kullanarak en kolay leibniz yöntemi ile alınır)

ikinci adımda b terimlerini hesaplamanız gerekiyor. Bu terimlerin hesabı için hangi koordinat hesaplanacaksa, o koordinatın satır ve sütunu kapatılıp geri kalan elemanlar bir sarmal şeklinde çarpılır. Bu durumu linkteki örnekten açıkça görebilirsiniz. Örneğin b21 için 2. satır ve 1. sütun kapatılıp kalan elemanlar bir sarmal şeklinde çarpılıp toplanmıştır. Burada mod işlemi ile sarmallık sağlanabilir.

Gauss Jordan Yöntemi

Gelen bir soru üzerine, Gauss Jordan yöntemini adım adım anlatan yeni bir bölüm ekliyorum. Öncelikle yukarıdaki kodda, Gauss Jordan metodunu kullanıyoruz. Bu metotta amaç bir matrisin tersini almak için tersi olan matris ile çarpımını, birim matrise dönüştürmektir.

$$A \times A^{-1} = I$$

işlemine göre şayet A matrisinin tersi isteniyorsa

$A \times B = I$, şeklinde yazdığımız B matrisi, A matrisinin tersidir ve

$I = A \times B$, şeklinde yazılabilir.

Buradaki B matrisini elde ederken aslında birim matris üzerinde işlem yapılarak A matrisinin tersi alınmış olur.

Örneğin aşağıdaki matrisi ele alalım:

101

021

111

Bu matrisin tersini alırken öncelikle yanına bir çizgi çekip birim matrisi yazıyoruz:

101|100

021|010

111|001

Buradaki amacımız sol ve sağ tarafta aynı işlemleri yaparak sol tarafta birim matrisi elde etmektir. Böylelikle sağ tarafta matrisin tersi oluşacaktır. Bu satırlar üzerinde yapılan işlemlere yalın satır işlemleri (elementary row operations) ismi verilir. Bu işlemler aşağıdaki şekilde sıralanabilir:

- bir satırın bir sabit ile çarpılması
- Bir satırın diğer bir satır ile yer değiştirmesi
- Bir satırın diğer bir satırdan çıkarılması

Elbette yukarıdaki bu işlemler, çizginin iki tarafına da uygulanacaktır. Buradaki işlemler aslında bir matriste bir denklemin tutulması durumunda denklemler üzerine yapılan işlemlere

benzetilebilir. Örneğin 3 bilinmeyenli 3 denklem ele alınırsa bir denklemin diğer denklemin çözümünde kullanılırken yapılan işlemler gibidir.

Şimdi matrisin tersini alma işlemi ile devam edelim. Amacımız ilk sütunu birim matrise benzetmek. İlk sütunda 3. satırda bulunan 1 tek farklı sayıdır. Dolayısıyla 3. satırın ilk terimini 0 yapabilmek için ilk satırı, 3. satırdan çıkarabiliriz.

$$101|100$$

$$021|010$$

$$010|-101$$

yukarıdaki işlem sonucunda görüldüğü üzere çizginin solundaki ilk satır 3. satırdan ve çizginin sağındaki ilk satır yine çizginin sağındaki 3. satırdan çıkarılmıştır.

çizginin solunda elde edilen 3. satır aslında birim matrisin ikinci satırıdır. O halde 3. satır ile 2. satırı yer değiştirirsek matrisin sol tarafından birim matrise daha çok yaklaşmış olunur.

$$101|100$$

$$010|-101$$

$$021|010$$

Yine 3. satırı birim matrise benzetmek için bu sefer 2. satırın 2 mislini, 3. satırdan çıkarmak yeterlidir.

$$101|100$$

$$010|-101$$

$$001|21-2$$

Yukarıda görüldüğü üzere çizginin iki yanında da ikinci satır 2 ile çarpılmış ve 3. satırdan çıkarılmıştır.

Sırada ilk satırı birim matrise benzetmek var. Bu işlem için ilk satırdan 3. satırı çıkarmak yeterlidir:

$$100|-1-12$$

$$010|-101$$

$$001|21-2$$

Görüldüğü üzere çizginin sol tarafından birim matri elde edildi. Bu durumda matrisin sağ tarafında elde ettiğimiz matris, ilk matrisin tersidir.

$$1-12$$

-101

21-2

sonucunu bulmuş oluruz.

Programlama

Matrisin tersinin alınması işleminin bilgisayarlar marifetiyle yapılması için algoritmik bir yaklaşıma ihtiyaç vardır. Bu bağlamda matrisin boyu ve içeriğinden bağımsız olarak matrisin tersini almak için aşağıdaki yaklaşımı kullanabiliriz.

Bir matrisin tersi, matrisin kendisi ile çarpıldığına birim matrisi veren matristir. [Birim matris \(identity matrix\)](#) ise diyagonu 1 ve diğer elemanları 0 olan matristir.

Öyleyse bir matrisi gerekli işlemleri yaparak birim matrise dönüştürürsek ve bu işlemler sırasında her elemana yapılan değişimi tutarsak sonuçta elde ettiğimiz bu değişim matrisi, orjinal matrisimizin tersi olacaktır.

Aşağıda bu işlemleri adım adım yapan C kodunu yazıp açıklamaya çalışalım.

İlk adımda matris boyutunu ve içeriğini dolduralım.

```
1  #include<stdio.h>
2  #include<conio.h>
3  int main()
4  {
5      float a[4][4]={1,2,3,4,7,11,9,0,9,8,7,6,1,12,3,14};
6      float b[4][4],c[4][4];
7      for(int i=0;i<4;i++){
8          for(int j=0;j<4;j++){
9              printf("    %f ",a[i][j]);
10             }
11             printf("\n");
12         }
```

Kodun ilk kısmında görüldüğü üzere 4×4 boyutlarında bir matris tanımlanmış ve bu matrisin içeriği ekrana bastırılmıştır. Yukarıdaki tanım itibarıyla matrisimizin içeriği aşağıdaki şekildedir:

1.000000	2.000000	3.000000	4.000000
7.000000	11.000000	9.000000	0.000000
9.000000	8.000000	7.000000	6.000000
1.000000	12.000000	3.000000	14.000000

Şimdi bu matrisin tersini alma işlemi sırasında kullanacağımız birim matrisi döngüler ile oluşturabiliriz:

```

13  for(int i=0;i<4;i++) {
14      for(int j=0;j<4;j++) {
15          if(i==j)
16              b[i][j]=1;
17          else
18              b[i][j]=0;
19      }
20  }

```

Yukarıdaki kod, köşegeninde (diagon) 1 olan ve diğer elemanları 0 olan bir matris inşa etmektedir. Bu işlem için matrisin satır ve sütun koordinatlarını tutan i ve j döngü değişkenlerinin eşit olması durumu 1, diğer durumlar 0 olarak döngülerde kodlanmıştır.

Şu anda, tersi alınması istenen matris a dizisinde, birim matris ise b dizisinde tutulmaktadır. Yapılması gereken, a matrisini b matrisine dönüştürmektir.

```

21  float d,k;
22  for(int i=0;i<4;i++) {
23      d=a[i][i];
24      for(int j=0;j<4;j++) {
25          a[i][j]=a[i][j]/d;
26          b[i][j]=b[i][j]/d;
27      }
28      for( int x=0;x<4;x++) {
29          if(x!=i) {
30              k=a[x][i];
31              for(int j=0;j<4;j++) {
32                  a[x][j]=a[x][j]-(a[i][j]*k);
33                  b[x][j]=b[x][j]-(b[i][j]*k);
34              }
35          }
36      }
37  }

```

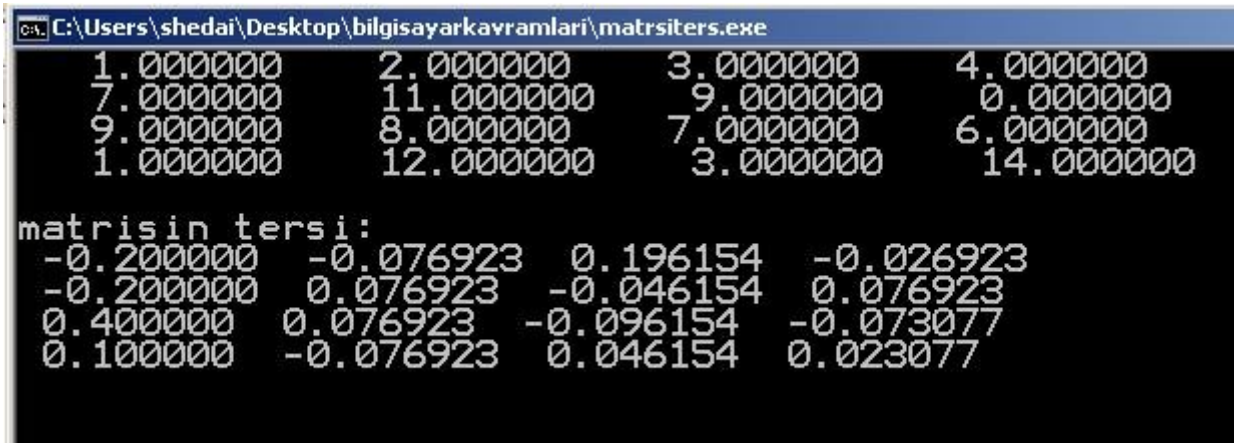
Yukarıda görüldüğü üzere kodun 24-27. satırları arasında matrisin diyagonu olan değerleri 1 yapacak değerler bulunmaktadır. Bir sayının kendisine bölümü 1'dir. Dolayısıyla matrisin tersi olan sonuç matrisimizdeki, ki şu an itibariyle birim matristir, diyagon değerleri, orjinal matrisin diyagon değerlerine bölünmüştür.

Ardından diyagon olmayan değerler için dönene ve 28. satırda başlayan ikinci döngü ile diyagonda olmayan elemanlardan, diyagondaki elemanla çarpımları çıkarılmaktadır. (kodun 32. satırındaki k değişkeni o satırdaki diyagon değerini vermektedir. Bu değer orjinal matristeki değer ile çarpılıp yine aynı satır ve sütundaki değerden çıkarılmaktadır)

Bu işlem hem orjinal matris hem de sonuç matrisinde bire bir yapılıır ve sonuçta sonuç matrisimiz b değişkeni içinde oluşur. Bu değeri bastırmak için aşağıdaki şekilde bir döngü yazılması yeterlidir.

```
38 printf("\nmatrisin tersi: \n");
39 for(int i=0;i<4;i++)
40 {
41     for(int j=0;j<4;j++)
42         printf(" %f ",b[i][j]);
43     printf("\n");
44 }
45 getch();
46 return 0;
47 }
```

Kodumuzun çalışan hali aşağıda verilmiştir:



```
C:\Users\shedai\Desktop\bilgisayarkavramlari\matrsiters.exe
1.000000 2.000000 3.000000 4.000000
7.000000 11.000000 9.000000 0.000000
9.000000 8.000000 7.000000 6.000000
1.000000 12.000000 3.000000 14.000000

matrisin tersi:
-0.200000 -0.076923 0.196154 -0.026923
-0.200000 0.076923 -0.046154 0.076923
0.400000 0.076923 -0.096154 -0.073077
0.100000 -0.076923 0.046154 0.023077
```

SORU 14: Dizgi (String)

Bir dilde bulunan ve o dilin tanımlı olan alfabeti içerisindeki sembollerin çeşitli sayılarda ve çeşitli sırada dizilmesi ile elde edilen yazılardır.

Örneğin bir dildeki alfabe aşağıdaki şekilde tanımlı olsun:

$$\Sigma_1 = \{0,1\}$$

Buna göre dilimizde sadece “0” ve “1” sembolleri tanımlı demektir. Bu dilde örneğin $w_1=0$ veya $w_2=10101011010$ gibi bir dizgi elde etmek mümkündür.

Bir dizginin belirli bir kısmını içeren dizgiye ise alt dizgi adı verilir. Örneğin $w_3=1011$ dizgisi w_2 dizgisinin bir altdizgisidir.

Ayrıca iki dizginin arka arkaya eklenmesine de üleştirme(concatenation) denilir.Örneğin w_1 ile w_3 dizgilerinin üleştirilmiş hali $w_4=01011$ olur.

Dizgiler ile ilgili diğer yazılar:

- [Dizgi Parçalayıcısı \(String Tokenizer\)](#)
- [Dizgi Karşılaştırma \(String Comparison\)](#)
- [Veri Tabanı Dizgi işlemleri \(Database String manipulations\)](#)
- [En uzun ortak küme \(longest common subsequence\) problemi](#)
- [Dizgi Eş şekilliliği \(String Isomorphism\)](#)
- [C dilinde dizgi kopyalama \(strcpy\)](#)
- [Alt Dizgi \(Substring\)](#)
- [Dizgi Hizalama \(String Alignment\) Problemi](#)

Dizgi Karşılaştırma/Arama/Mesafe algoritmaları:

- [Boyer Moore Dizgi Karşılaştırması \(Boyer Moore String Comparison Algorithm\)](#)
- [Knuth Moris Prat Dizgi Arama Algoritması \(Knuth Morris Prat Algorithm\)](#)
- [Dizgi Eş şekilliliği \(String Isomorphism\)](#)
- [Needleman Wunsch Yaslama Algoritması \(String Alignment\)](#)
- [Smith Waterman Yaslama Algoritması \(String Alignment\)](#)
- [Hunt Macllor Yaslama Algoritması \(String Alignment\)](#)
- [Horspool Algoritması \(Dizgi arama algoritması\)](#)
- [Levenstein Mesafesi \(Levenshtein Distance\)](#)
- [Hamming Mesafesi \(Hamming Distance\)](#)
- [Diff komutu](#)
- [Jaccard İndeksi, mesafesi ve katsayısı \(Jaccard Index\)](#)
- [Dice Sorensen Benzerliği \(Dice Sorensen Similarity\)](#)

Dizgi Parçalama (Parsing) algoritmaları:

- [LL\(1\) parçalama algoritması](#)
- [SLR\(1\) parçalama algoritması](#)
- [Earley Parçalama Algoritması](#)

C ile dizgi okuma

C dilinde klavyeden dizgi (String) okumak için kullanılan en basit fonksiyon scanf fonksiyonudur. Bu fonksiyonu basit bir uygulamada aşağıdaki şekilde kullanabiliriz:

```
1  #include <stdio.h>
2  #include <conio.h>
3  //www.bilgisayarkavramlari.com
4  int main(){
5      char isim[100];
6      printf("isminizi giriniz:");
7      scanf("%s",isim);
8      printf("\nisminiz: %s",isim);
9      getch();
10 }
```

Yukarıdaki kodda, 5. satırda tanımlanan karakter dizisi (string) içerisine 7. satırda %s parametresi ile scanf fonksiyonu kullanılarak bir dizgi okunmuştur. Bu dizginin içeriği kodun 8. satırında ekrana basılmıştır.

Örneğin yukarıdaki kod aşağıdaki şekilde çalıştırılabilir:

```
C:\Users\shedai\Desktop\bilgisayarkavramlari\strscanf.exe
isminizi giriniz:ali
isminiz: ali_
```

Görüldüğü üzere, kullanıcı isim olarak “ali” girmiş ve ekranda, girdiği bu dizgiyi görmüştür. Ancak aynı kodu çalıştırarak aşağıdaki şekilde bir dizgi girilirse problem yaşanır:

```
C:\Users\shedai\Desktop\bilgisayarkavramlari\strscanf.exe
isminizi giriniz:ali baba ve kirk haramiler
isminiz: ali
```

Yukarıdaki girdide “ali baba ve kirk haramiler” şeklinde 5 kelimedenden oluşan bir dizgi girilirken, bu dizginin sadece ilk kelimesi scanf tarafından okunmuştur. Aslında burada bir hata yoktur çünkü scanf fonksiyonu, boşluk karakteri veya satır sonu gibi karakterlere kadar olan dizgileri okur. Yukarıdaki gibi birden fazla kelimedenden oluşan dizgiler okunmak istendiğinde, aşağıdaki kodda da gösterilen gets fonksiyonu kullanılabilir:

```
1  #include <stdio.h>
2  #include <conio.h>
3  //www.bilgisayarkavramlari.com
4  int main(){
5      char isim[100];
6      printf("isminizi giriniz:");
7      gets(isim);
8      printf("\nisminiz: %s",isim);
9      getch();
10 }
```

Yukarıdaki kodda bir önceki koda göre sadece 7. satırda bulunan scanf fonksiyonu, gets fonksiyonu ile değiştirilmiştir. Kodumuzun yeni halini çalıştırdığımızda, aşağıdaki şekilde birden fazla kelime okuyabildiğimizi görürüz:

```
C:\Users\shedai\Desktop\bilgisayarkavramlari\strscanf.exe
isminizi giriniz:ali baba ve kirk haramiler
isminiz: ali baba ve kirk haramiler
```

Dizgilerin Eşitliği

İki farklı değişkende bulunan dizginin eşit olup olmadığı, programlama dillerinde bulunan klasik operatörler ile yapılamaz.

Örneğin aşağıdaki kodlama C dili açısından doğru olsa bile mantıksal olarak hatalıdır (logic error):

```
char a[100] = "bilgisayarkavramlari.com";
```

```
char b[100] = "bilgisayarkavramlari.com";
```

```
if(a==b)
```

Yukarıdaki kodun, son satırında bulunan eşitlik kontrolü, C dili açısından ne derleme (compile) ne de çalışma (run-time) hatası döndürmez. Ancak buradaki karşılaştırma aslında iki dizinin (array) hafızada (RAM) aynı yeri gösterip göstermediğini sorgulamaktadır.

Yukarıdaki kod, her zaman için yanlış (false) döner ve if kontrolüne hiçbir zaman girilemez. Bunun yerine dizgilerin içeriklerinin karakter karakter kontrol edilmesi ve dizgilerin boyutlarının eşit olup olmadığının sorgulanması gerekir.. Bizim iki dizinin eşitliğinden anladığımız genelde budur. Bu kontrole literatürde derin karşılaştırma (deep compare) ismi verilmektedir.. Klasik olarak yapılan `a==b` kontrolü ise sığ karşılaştırma (shallow compare) olarak geçmektedir.

Derin karşılaştırma için kendimiz bir fonksiyon yazabileceğimiz gibi, C dilinde var olan `string.h` kütüphanesindeki `strcmp` fonksiyonunu kullanabiliriz. Bu fonksiyon iki dizgiyi (string) sözlük sıralamasına göre karşılaştırır (lexiconically) ve şayet eşitse 0 değerini döndürür.

C dilinde 0 değeri mantıksal olarak yanlış (false) olduğu için eşit olmaları durumunda bir if bloğunun çalışmasını istiyorsak, aşağıdaki şekilde yazabiliriz:

```
if(!strcmp(a,b))
```

yukarıdaki kontrol, a ve b dizgilerinin eşitliği durumunda geçer.

Aynı kontrol JAVA veya C# gibi dillerde, String sınıfının bulunması sayesinde, ilave bir kütüphane ve fonksiyona gerek kalmadan çözülebilir.

Örneğin JAVA dili için:

```
String a = "www.bilgisayarkavramlari.com";
```

```
String b = "www.bilgisayarkavramlari.com";
```

```
if(a.equals(b))
```

kontrolü yapılması yeterlidir. Java dilinde, bulunan ve dizgileri karşılaştırmak için kullanılan `equals` fonksiyonu C# dilinde ilk harf büyük olarak `Equals` şeklinde yazılarak çalıştırılabilir (burada bir kere daha JAVA'yı taklit ederken, java kodlarının çalışmaması için özel gayret sarf eden microsoft geliştiricilerini selamlıyoruz.)

Dizgilerin birbirine eklenmesi (concatenate, üleştirme) için C ve C++ gibi dillerde `strcat` fonksiyonu kullanılabilir.

Örneğin:

```
char a[100]="www.";
```

```
char b[100]="bilgisayarkavramlari.com";
```

```
strcat(a,b);
```

```
printf("%s",a);
```

şeklindeki bir kod, ekrana "www.bilgisayarkavramlari.com" sonucunu basacaktır.

Aynı üleştirme işlemi, JAVA veya C# için basit bir toplama (+) işlemi ile yapılabilir.

```
String a="www.";
```

```
String b="bilgisayarkavramlari.com";
```

```
System.out.println(a+b);
```

yukarıdaki kod, ekrana "www.bilgisayarkavramlari.com" sonucunu basarken aynı kodu C# dilinde sadece son satırını Console.write(a+b) olarak değiştirerek deneyebilirsiniz.