

INTERNET TABANLI PROGRAMLAMA

Active Server Pages (ASP)

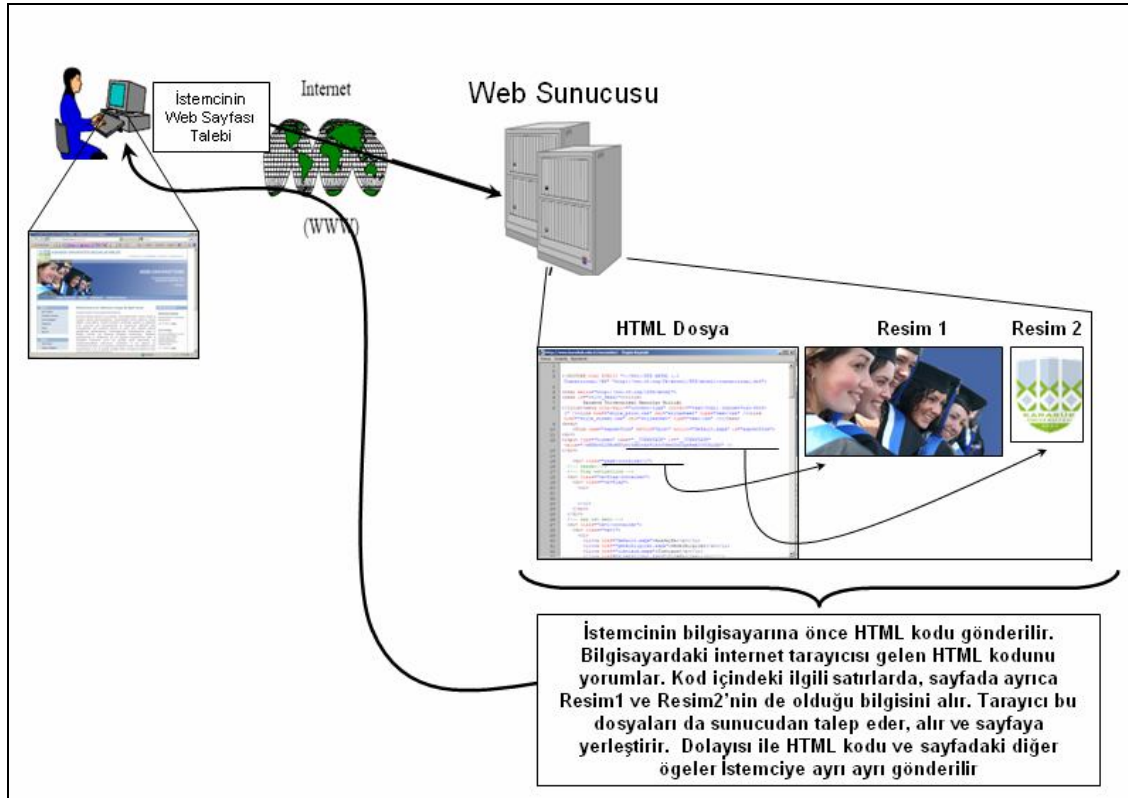
Vbscript ile Programlama Ders Notları

Doç. Dr. İsmail Rakıp Karaş

ASP TEKNOLOJİSİ (ASP Technology)

Giriş

Önceki derslerden de hatırlayacağınız üzere, HTM uzantılı dosyalar (HTML kodları) sunucu tarafından istemciye gönderilmekte ve bu kodlar istemcinin bilgisayarındaki tarayıcı tarafından yorumlanmakta/çalıştırılmakta idi. (*server, client, browser, HTML files, interpret, interpretation, running*)



HTML kodlarının sunucudan istemciye transferi ve istemcinin bilgisayarındaki tarayıcı tarafından yorumlanması/çalıştırılması

Bu yaklaşımda sunucu HTML kodları ile ilgilenmemektedir, sunucu sadece bu kodları istemciye göndermekle yükümlüdür. Oysa bu yaklaşım çoğu zaman yetersiz kalmaktadır. Artık istemci tarafında çalışan statik sayfalar yerine (*static pages*), sunucu tarafında da koşan etkileşimli ve dinamik sayfalara ihtiyaç duyulmaktadır (*interactive and dynamic pages*). Bunun da ötesinde, güvenlik, tarayıcıların yetersizlikleri ve istemci yükünü azaltmak gibi sebeplerle bir takım işlemlerin sunucu tarafında yapılması zorunludur (*security, inability of browsers*). PHP, ASP, ASP.Net gibi teknoloji ve diller bu tür işlemleri gerçekleştiren sunucu tarafı bazı çözümlerdir (*server side solutions*). Bu derste ASP teknolojisi üzerinde durulacaktır.

Şişman ve Zayıf İstemci Yaklaşımı

Yukarıdaki HTM dosyaları örneğinde olduğu gibi, eğer işlemler istemcinin bilgisayarında gerçekleştiriliyorsa bu yaklaşıma Şişman İstemci (Fat Client) yaklaşımı denmektedir (İstemci taraflı yaklaşım/client side approach). Şişman denmesinin sebebi işlem yükünün daha çok istemci tarafında olmasındandır. Bu yaklaşımın tersine, işlem yükü sunucu tarafında olduğunda, bu yaklaşıma ise Zayıf İstemci (Thin Client) yaklaşımı denmektedir (Sunucu taraflı yaklaşım/server side approach). Bu durumda istemcideki işlem yükü az, yani zayıftır. Sunucu ise, sadece verileri gönderilmekle değil, göndermeden önce ilgili kodları yorumlamak, çalıştırmak ve değerlendirmek ile de yükümlüdür.

Zayıf İstemci yaklaşımının ilk dönemlerinden bir karikatür:

1.



Bilgi İşlem Personeli:

- Patron... Zayıf İstemci (Müşteri) Yaklaşımını öğrenmemiz gerekiyor.
- Bize eğitim lazım.

2.



(Kilo Avcıları Derneği Seanslarına Hoş Geldiniz.)

- Çikolatadan uzak duramıyorum
- Kahretsin...

İki Örnek:

Konuyu daha iyi kavramak ve somutlaştırmak açısından aşağıdaki örnekleri inceleyelim. Dikkat edilirse her iki örnekte de HTML kodları içinde tanımadığımız etiketlerle başlayan farklı satırlar vardır (kırmızı satırlar). Bu satırlar birer script kodudur. Soldaki örnekteki satırlar JavaScript, sağdakiler ise VbScript kodudur.

Merhaba.htm

```
<HTML>
<HEAD>
<TITLE>JavaScript ile Tarih</TITLE>
<meta http-equiv="content-type"
content="text/html; charset=ISO-8859-9">
<meta http-equiv="Content-Type"
content="text/html; charset=windows-1254">
</HEAD>
<BODY BGCOLOR=WHITE>
<H1>Merhaba Dünya</H1>
<H2>Bugün:</H2>
<H3>
<SCRIPT LANGUAGE=JAVASCRIPT>
<!--
tarih = new Date();
document.write(tarih);
//-->
</SCRIPT>.
</H3></BODY>
</HTML>
```

Merhaba.asp

```
<HTML>
<HEAD>
<TITLE>ASP İLE İLK SAYFA</TITLE>
<META http-equiv="content-type"
content="text/html; charset=ISO-8859-9">
<META http-equiv="Content-Type"
content="text/html; charset=windows-1254">
</HEAD>
<BODY>
<H1><CENTER>Merhaba Dünya!</H1>
<H2>Bugün:
<%
Response.Write(WeekdayName(WeekDay(Date))
& " " & Date & " " & Time)%>
</CENTER>
</H2>
</BODY>
</HTML>
```

Aralarındaki farkları daha iyi görebilmeniz açısından bu kodları içeren dosyalar bir sunucuya yüklenmiştir. Lütfen aşağıdaki adreslere girerek bu kodların tarayıcınızda nasıl görüntülendiğini inceleyiniz:

<http://www.ismailkaras.com/ASP/merhaba.htm>

<http://www.ismailkaras.com/ASP/merhaba.asp>

Sayfalara girdiğinizde her iki görüntünün de, birbirine benzer şekilde günün tarihini verdiğini göreceksiniz.

Şimdi de, tarayıcınızın “Kaynağı Göster/Show Source Code” özelliğini kullanarak ilgili sayfaların kodlarını inceleyiniz. İlk adrese ait koda baktığınızda yukarıda solda görülen kodların aynen burada da yazdığını göreceksiniz. İkinci adrese ait koda baktığınızda ise yukarıdaki sağdaki örnekten farklı olarak

`Response.Write(WeekdayName(WeekDay(Date)) & " " & Date & " " & Time)%>` yazan yerde günün tarihinin olduğunu göreceksiniz.

Bunun sebebi şudur; JavaScript, istemci tarafı (Şişman istemci) bir dil iken, VbScript tersine **sunucu tarafında** çalışan (Zayıf İstemci) bir dildir. Dolayısı ile ilk sayfanın Kaynak koduna (Source code) baktığınızda kodu aynen görebilmeniz sebebi bundandır. Kodlar sizin (istemci) tarafınızda çalışmakta, bu kodlar bilgisayarınıza geldiği için kaynağını görebilmektesiniz. Sağ tarafta çalışan VbScript kodu ise sunucuda çalışmakta istemciye ise sadece işlemin sonucu gönderilmektedir. Bu yüzden Kaynak kodu içinde VbScript kodu görünmemektedir. Çünkü bu kod

tarayıcınıza gelmemiştir. Tarayıcınıza gelen, sunucuda çalıştırılan VbScript kodunun ürettiği sonuçtur.

Bu örnekler Zayıf-Şişman Kullanıcı yaklaşımını güzel bir şekilde sunmakla birlikte, yukarıda anlatılanlarda başka iki şey daha dikkatinizi çekmiş olmalı.

- Script türü diller (Script Languages) HTML kodları ile birlikte (iç içe/nested) yazılmaktadırlar.
- VbScript kodunun başladığı yerde % işareti göze çarpmaktadır.

ASP ve VbScript Dili (VbScript Language)

Genel yanlışlığın aksine ASP bir dil değildir. ASP (Active Server Pages) bir teknolojinin adıdır. Adı üstünde, sunucuda çalışan aktif/dinamik sayfalardır. Bu sayfalar içinde çalışan kodlar ise VbScript kodlarıdır. Yani ASP teknolojisinin kullandığı dil VbScript'tir.

VbScript bir script dilidir. Yani Visual Basic dilinin web uygulamalarına yönelik olarak basitleştirilmiş bir şeklidir (Simplified VB). Visual Basic dilinin masaüstü programcılıkta sağladığı yetenekleri sergilemekten uzaktır. Bununla birlikte web uygulamalarında etkin bir şekilde kullanılabilecek araç ve yöntemler içerir. Kodlama şekli (syntax) Visual Basic'e oldukça yakındır. Birini bilen diğerine geçişi çok kolay olur.

Yukarıda da dikkat çekildiği üzere VbScript kodları HTML kodları ile birlikte içiçe/gömülü (nested/embedded) olarak kullanılabilir. HTML kodu içinde VbScript kodunun başladığı <% etiketi ile anlaşılır. Dolayısı ile VbScript kodunun başlangıç etiketi <% bitiş etiketi ise %> 'dir. Bu etiketlerin arasındaki kodlar sunucu tarafından yorumlanması gereken veriler olarak algılanır. Sunucu, ASP dosya içindeki htm kodlarının yorumlanmasını istemciye bırakarak aynen gönderir, VbScript kodlarını ise yorumlar (interpret) ve sonuçlarını gönderir (send results).

ASP sayfalarının uzantısı .asp'dir. Sunucuya kaydedilmiş bir dosyanın uzantısı asp ise, sunucu bu dosyanın çalıştırılması ve yorumlanması gereken bir dosya olduğunu algılar ve içine bakarak VbScript kodlarını arar. Fakat dosyanın uzantısı .htm ise sunucu hiçbir şekilde bu dosyayı yorumlamaz, talep edildiğinde tümünü istemciye gönderir. Dolayısı ile bir dosyanın içinde VbScript kodu olsa bile uzantısı .asp olmadığı sürece sunucu bu dosyanın içindeki kodlara bakmayacaktır. Doğrudan istemciye gönderecektir. İstemciye tarayıcı ise VbScript kodlarını anlayamayacağından o satırları yok varsayacak, görselleştiremeyecektir (visualization).

Sonuç olarak aynı dosya içindeki, VbScript kodları sunucu, html kodları tarafından yorumlanır ve birleştirilerek tarayıcıda görselleştirilir.

Örnek:

Yukarıdaki "Merhaba.asp" isimli dosyanın uzantısı (extension) htm şeklinde değiştirilerek "Merhaba_.htm" olarak da aynı sunucuya kaydedilmiştir. Aşağıdaki linke tıklayarak dosyayı tarayıcınıza çağırınız.

http://www.ismailkaras.com/ASP/merhaba_.htm

Öncekinin aksine sayfada tarihin görüntülenmediği (the date is not shown/visualized) dikkatinizi çekmiş olmalı. Şimdi, tarayıcınızın “Kaynağı Göster/Show Source” özelliğini kullanarak kodlarını inceleyiniz. Yukarıdaki durumun tersine

`Response.Write(WeekdayName(WeekDay(Date)) & " " & Date & " " & Time)%>` yazan Vbscript kodunu görebileceksiniz. Dosya htm olduğu için sunucu dosyayı yorumlamadan (without interpretation) bütünüyle istemciye göndermiş, tarayıcınız ise VbScript kodlarını tanımadığından (not recognized) bu satırları yok varsaymıştır (ignored). Sonuç olarak, eğer bir dosya içine VbScript kodu yazıyorsak mutlaka dosyanın uzantısını .asp ye dönüştürmeliyiz, ki bu kodlar sunucu tarafından yorumlansın.

İstemci/kullanıcı .asp uzantılı bir dosyayı çağırdığında (linke tıkladığında/click to link), sunucu öncelikle dosyanın uzantısına bakar. Asp olduğunu görünce içindeki kodları okumaya başlar. Htm kodlarını yorumlamadan aynen istemciye gönderir. <% etiketini gördüğünde VbScript kodunun başladığını algılar ve o noktadan sonra bu kodu yorumlamaya/çalıştırmaya başlar. Ardından kodun çalıştırılması ile elde edilen sonucu istemciye gönderir. Etiket bitip tekrar htm'ye dönüldüğünde yorumu bırakıp yine aynen kodları istemciye gönderir.

Örnekler:

Aşağıdaki örneklerle ait linklere tıklayınız ve “Kaynağı Göster” ile tarayıcınızda açılan kodlar ile aşağıdaki orijinal kodları karşılaştırınız. Farklarını görünüz. İnceleyerek ASP'nin yapısını ve VbScript dilini anlamaya çalışınız ve sunucu tarafında olanları tahmin ediniz. Yukarıda anlatılanları örnekler üzerinde görünüz.

<http://www.ismailkaras.com/asp/gunler.asp>

```
<HTML>
<HEAD>
<TITLE>ASP GÜNLERİ SAYMA</TITLE>
<META http-equiv="content-type" content="text/html; charset=ISO-8859-9">
<META http-equiv="Content-Type" content="text/html; charset=windows-1254">
</HEAD>
<BODY>
<H2>
<CENTER>
<%
Dim Gun
Gun = Array("Trakya", "Ege", "Marmara", "Doğu Anadolu", "Güneydoğu Anadolu", "Akdeniz", "Karadeniz")
For sayac = 0 to 6
    Response.Write Gun(sayac)
    Response.Write "<BR>"
Next
%>
</CENTER>
</H2>
</BODY>
</HTML>
```

<http://www.ismailkaras.com/asp/deneme.asp>

```
<% Option Explicit %>
<HTML>
<%
Dim Degisken(2), Toplam
Degisken(1) = "Mustafa"
Degisken(2) = "Durcan"
Toplam = degisken(1) + Degisken(2)

=Toplam %>
</HTML>
```

<http://www.ismailkaras.com/asp/hosgeldiniz01.asp>

```
<HTML>
<HEAD>
<TITLE>ASP ILE SAATE GÖRE SELAM</TITLE>
<META http-equiv="content-type" content="text/html; charset=ISO-8859-9">
<META http-equiv="Content-Type" content="text/html; charset=windows-1254">
</HEAD>
<BODY>
<H2>
<CENTER>
<%
If Hour(Now) <12 Then
    Response.Write "Günaydın! "
Elseif Hour(Now) >= 18 Then
    Response.Write "İyi akşamlar! "
Else
    Response.Write "Tünaydın! "
End If
Response.Write "<BR>"
Response.Write "Sitemize Hoşgeldiniz"
%>
</CENTER>
</H2>
</BODY>
</HTML>
```

<http://www.ismailkaras.com/asp/hosgeldiniz02.asp>

```
<HTML>
<HEAD>
<TITLE>ASP ILE SAATE GÖRE SELAM</TITLE>
<META http-equiv="content-type" content="text/html; charset=ISO-8859-9">
<META http-equiv="Content-Type" content="text/html; charset=windows-1254">
</HEAD>
```

```
<BODY>
<H2>
<CENTER>
<%
Select Case Hour(Now)
    Case 0,1,2,3,4,5,6,7,8,9,10,11
        Response.Write "Günaydın!"
    Case 12,13,14,15,16,17
        Response.Write "Tünaydın"
    Case Else
        Response.Write "İyi Akşamlar!"
End Select
Response.Write "<BR>"
Response.Write "Sitemize Hoşgeldiniz"
%>
</CENTER>
</H2>
</BODY>
</HTML>
```

ASP Dosyalarını Kendi Bilgisayarınızda Çalıştırmak (Running in own Computer)

Soru: Yazdığım ASP dosyalarını denemek için mutlaka bir sunucuya mı yüklemem gerekiyor (upload to a server)? Kendi bilgisayarımda çalıştıramaz mıyım? Kendi bilgisayarımın hem sunucu, hem de istemci gibi davranmasını sağlayamaz mıyım (behavior either server or client) ?

Elbetteki kendi bilgisayarınızı ASP sunucusu gibi kullanabilirsiniz. Bunun için IIS'in (Internet Information Server) kurulması gerekir. IIS'in kurulumu hakkında internetten bir çok kaynağa ulaşabilirsiniz. Bununla birlikte ASP dosyalarımızı denemek için biz bu derste kurulumu çok daha basit olan bir başka programı kullanacağız:
Babyweb.exe

Programı kullanabilmek için şu adımları uygulayınız:

1. www.ismailkaras.com/babyweb.exe adresinden indireceğiniz (download) Babyweb.exe dosyasını herhangi bir yere kopyalayınız.
2. ASP dosyalarınızı saklamak ve denemek üzere bir klasör açınız (Örn: C:\ASPdeneme).
3. Babyweb.exe'yi çalıştırınız (Run it). Settingse tıklayıp "Web Pages" yazan satıra bu klasörün adresini (address of folder) girin. (Sağındaki üç noktaya tıklayarak da yolu gösterebilirsiniz.)
4. "Tamam" dedikten sonra pencereyi aşağıya alın. Programın saatini yanında çalışmaya devam ettiğini göreceksiniz.
5. Belirlediğiniz klasör artık sizin sunucunuzdaki kök klasörünüz (root folder/localhost) oldu. Çalıştırabilmek için bütün .asp uzantılı dosyalarınızı burada tutmalısınız. Bu klasördeki ASP dosyalarını çalıştırmak için şunları yapmalısınız:

- a. Örneğin bilgisayarınızın adı hakan1 olsun (Bilgisayarınızın adına "bilgisayarım>özellikler" den bakabilirsiniz (Tam bilgisayar adı). Bilgisayar adınız Türkçe karakter ve boşluk içeriyorsa ismini değiştiriniz.)
- b. Çalıştıracacağınız dosya ise merhaba.asp olsun. Bu durumda dosyayı çalıştırmak için tarayıcının adres satırına şöyle yazmalısınız:

<http://hakan1/merhaba.asp>

Ya da daha kolay ve kısa yoldan dosyalarınıza şu şekilde de ulaşabilirsiniz:

<http://localhost/merhaba.asp>

VBSCRIPT ile PROGRAMLAMA

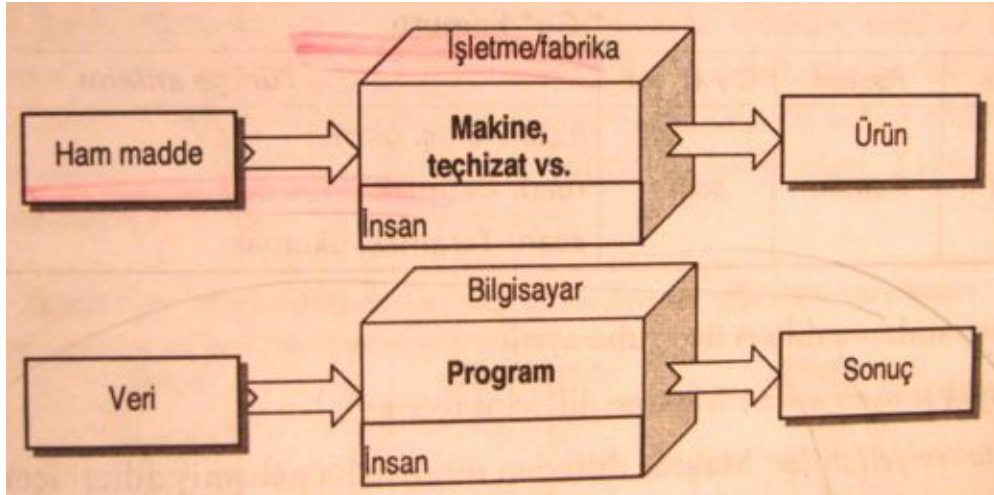
Yukarıdada belirtildiği gibi VbScript, Visual Basic dilini temel alan bir script dilidir. Dolayısı ile tüm diğer dillerde olduğu gibi VbScript 'de de Programlama ilkeleri önemlidir. Bu açıdan öncelikle VB ile Programlamanın temelleri üzerinde durulacaktır.

PROGRAMLAMANIN TEMELLERİ (Fundamentals of Programming)

1. TEMEL KAVRAMLAR (Basics)

Bilgisayar Programı ve Programlama

İnsanla bilgisayar arasındaki iletişim aracı olan **program**; giriş değerlerini kullanarak istenilen çıkış değerlerinin elde edilebilmesi için bilgisayara iletilen komutlar dizisidir. Bilgisayarların işlem yapabilmek ve problemleri çözebilmek için programa ihtiyaç duyarlar. Bu programlar ise bilgisayara insan tarafından bildirilir yada başka bir ifadeyle öğretilir. Bu öğretme işlemine ise **programlama (programming)** denir.



Bilgisyardaki program ile fabrikadaki makinenin çalışma şekli birbirine benzer

Algoritma

Kullanıcı bilgisayara ne öğretirse onun karşılığını alır. Girdiği programlarla, hangi verileri nasıl işleyeceğini ve ne tür sonuçlar üreteceği bilgisayara bildirir. Kullanıcının bunu yapabilmesi için problemin çözüm yolunu öncelikle kendisinin bilmesi gerekir. Dolayısı ile bilgisayara bir program girmeden/öğretmeden önce kullanıcının bunu nasıl yapacağını planlaması önemlidir. Bu planlama adım adım yapılır ve bu işleme algoritma ismi verilir.

Dolayısı ile **algoritma** bilgisayardaki bir işlemin/işlemlerin gerçekleştirilmesinde izlenecek kurallar ve adımlar (set of rules/steps) dizisidir. Algoritma hazırlayan kullanıcı, bilgisayardan yapmasını istediği işlemleri kendi anlayacağı dilde (insan dilinde) adım adım yazar. Algoritma, bir işin hangi etaplardan geçilerek yapılacağını gösteren çalışma planıdır (study plan), Algoritma bir programlama dili değildir.



Algoritma sözcüğü 9. yy başlarında yaşamış, ünlü matematikçi Muhammed bin Musa el-Harezmi'nin, Arapların ona san olarak verdikleri el-Harezmi sözcüğünden batılıların yaptığı bir terimdir.

el-harezmi > algorizma > algoritma

İyi bir algoritma, en kısa zamanda işlem yapacak, en az bellek kullanacak ve bilgisayar kaynaklarını en az tüketecek biçimde tasarlanmalıdır.

Algoritma hazırlarken

- Yapılacak iş iyice irdelenir. Tüm olasılıklar gözden geçirilir. (examined and considered all probabilities)
- En az işlem adımında , en kısa sürede, en doğru ve en hassas (in shortest path, shortest time, to best and most sensitive result) sonuca ulaştıracak çözüm yolu belirlenir.

Algoritma tasarımı yapmak,

- Programı yazmayı kolaylaştırır (make easier).
- Hatalı kodlama oranını azaltır (reduce the rate of bad code).
- Program yazımı için geçen süreyi kısaltır (timesaving).
- İşlem akışını açık bir şekilde gösterdiğinden program kontrolünü kolaylaştırır.
- Sonradan yapılacak düzenlemelerde kolaylık sağlar.

Örnek: İnternette Tiyatro için Bilet Satma Algoritması:

1. Kullanıcının istediği oyun, gün ve yer bilgileri alınır.
2. Veritabanı sorgulanarak, belirtilen günde oynayan oyunun boş yerleri çıkartılır.
3. Boş yer sayısı sıfırsa, o günde belirtilen oyun oynanmamıştır ya da oyundaki bütün yerler satılmıştır.
4. Her iki durumda da bilet kesilemediği için ekranda hata mesajı gösterilir. Gün ve oyun bilgilerini baştan almak için ilk etaba dönülür.
5. Kullanıcıdan oturmak istediği yer bilgisi alınır.
6. İsteddiği yerin dolu olup olmadığı kontrol edilir.
7. Yer dolu ise ekrana hata mesajı gösterilir ve yer bilgisi tekrar alınmak üzere 5. etaba dönülür.
8. Yer boşsa, veritabanında oyunun yer kayıtları güncellenir.
9. İstenilen gün, oyun ve yer bilgilerini içeren bilet yazıcıdan çıkartılır.

Programlama Dilleri

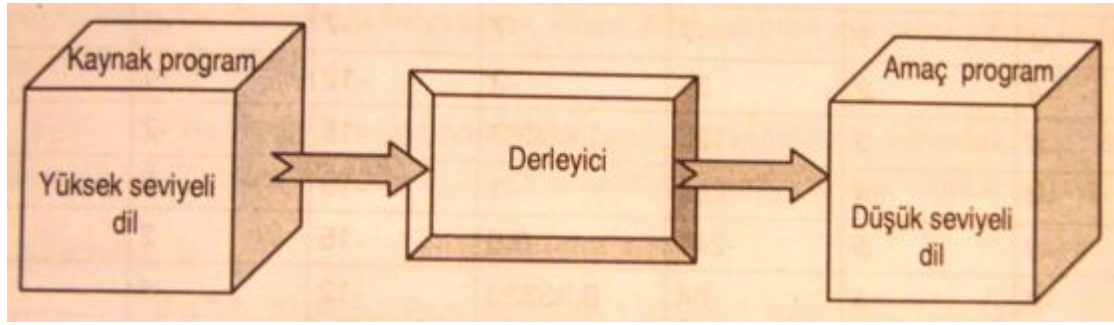
Nasıl ki yer soran birisine, anladığı dildeki (Türkçe, İngilizce vb.) kelimeleri kullanarak yol tarifi yapıyoruz. Bilgisayardan bir şeyler yapmasını istediğimizde de bilgisayarın anladığı dilleri kullanmamız gerekir. Kullanıcının bir programı bilgisayara girerken/öğretirken kullandığı özel kelime, işaret ve sembollerin bütününe programlama dili denir. Programlama dilleri genel olarak üç gruba ayrılır.

1. Düşük seviyeli diller (Low-level languages): Makine dillerini içerir. 1 ve 0'lardan oluşan çok karmaşık bir dil
2. Orta seviyeli diller (Mid-level): Makine dilinden biraz daha gelişmiş bir dildir.
3. Yüksek seviyeli diller (High-level): İnsan konuşma diline yakın diller.

Günümüzde yaygın olarak kullanılan programlama dilleri yüksek seviyeli dillerdir. Ve bunlardan bazıları şunlardır: Pascal, Cobol, Basic, ,C++, C#, Delphi, Visual C, **Visual Basic** vb.

(Biz bu derste basit, anlaşılır ve görsel (simple, understandable, and visual) olması sebebi ile Visual Basic dilini ve bu dilde program yazmayı öğreneceğiz.)

Yüksek seviyeli bir dilde yazılan programın çalışabilmesi için bunun makine diline (yani EXE uzantılı dosyaya) dönüştürülmesi gerekir. Bu dönüştürme işlemine derleme işlemi (compilation process) denir.



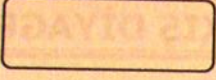
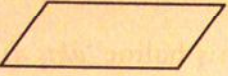
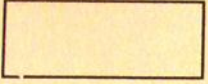
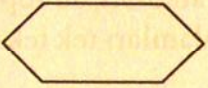
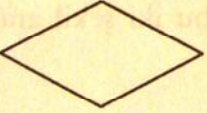
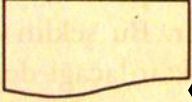
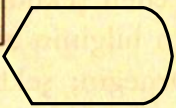

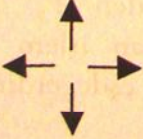
Derleme işlemi

Program Kodlama (Program Coding) Ve Program Kaynak Kodu(Program source Code)

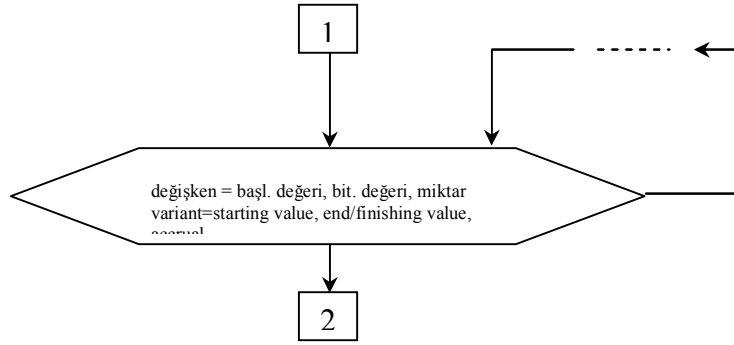
Tasarlanmış olan algoritmanın belirli bir programlama dilinde, o dilin komut ve terminolojisi kullanılarak yazılmasına program kodlama, yada kodlama, yazılan bu metne de program kodu veya program kaynak kodu denir.

2. AKIŞ DİYAGRAMLARI (Flowcharts)

Algoritmaların özel geometrik şekillerle çizilmiş haline akış diyagramı denir. Akış diyagramlarında kullanılan şekiller ve anlamları aşağıdaki gibidir:

Şekil	Anlamı	Açıklama
	Başla / Dur	Akış diyagramlarında algoritmanın başladığı ve bittiği noktaları belirtir. (Start-Begin/Stop-Exit)
	Bilgi/veri girişi	Dışarıdan bir bilgi girişi olacağı zaman bu şekille ifade edilir. Kullanıcı tarafından bir değer girileceğini belirtir. (Input/Data entering)
	İşlem	Program esnasında gerçekleştirilecek işlemler (örneğin matematiksel işlemler) bu şekil ile ifade edilir. (Process)
	Döngü	Programın bu noktada belirli bir döngüye girdiğini belirtir. Döngünün detayları aşağıda verilmiştir. (Loop)
	Karar/Karşılaştırma	Belirli bir karşılaştırma yapılarak bir karar verilecek ve buna bağlı olarak algoritmanın yönü değişecekse bu şekil kullanılır. Bu şekilden ayrılan ok sayısı birden fazladır (Decision)
 veya 	Bilgi/veri yazma	Ekran, yazıcı ve benzeri türden çevre birimlerinde bir çıktı yada görüntü bildirimi yapılacaksa bu şekil kullanılmalıdır. (Document/Output)
	Bağlantı	Akış diyagramlarındaki okların birleşim noktalarına konulur. Zorunlu değildir. (Node)
	İşlem akış yönü	Oklar yukarıda bahsedilen şekillerin arasını birleştirmek ve akışın yönlerini göstermek için kullanılırlar. (Arrows)

Akış Diyagramlarında Döngüler (Loops in Flowcharts)



Akış diyagramlarındaki döngülerin çalışma şekli şöyledir: 1 yolundan (path) gelen algoritma yandaki şekli görünce döngüye girer. Burada belirtilen değişken başlangıç değerinden başlayarak her seferinde farklı değerler alarak büyüyecek (increase) yada küçülecek (decrease) ve sonunda bitiş değerine ulaşacaktır. Bitiş değerine ulaştığında döngü sona erecek ve algoritma 2 yolundan çalışmaya devam edecektir. Aşağıda, başlangıç-bitiş değerlerine ve artış yada azalma miktarlarına göre farklı döngü örnekleri görülmektedir.

<p>A flowchart for a loop where the variable 'i' starts at 1, ends at 20, and increases by 3. The diamond contains 'i = 1, 20, 3'. An arrow points down from the diamond to the next step. A feedback arrow starts from the right side of the diamond, goes right, then up, then left, and finally down into the top of the diamond.</p>	<p>i değişkeni 1'den başlayacak ve 3'er artarak 20'ye kadar devam edecektir. Yani, ilk döngüde $i=1$, ikincide $i=4$, sonrasında ise 7, 10, 13, 16, 19 şeklinde devam edecektir. Ardından döngü bitecek ve akış aşağıya doğru devam edecektir. Bu döngü esnasında i'nin her değeri için noktalı kısımda verilmiş işlemler tekrarlanacaktır.</p>
<p>A flowchart for a loop where the variable 'j' starts at 30, ends at 4, and decreases by 2. The diamond contains 'j = 30, 4, -2'. An arrow points down from the diamond to the next step. A feedback arrow starts from the right side of the diamond, goes right, then up, then left, and finally down into the top of the diamond.</p>	<p>j değişkeni 30'dan başlayacak ve 2'ser eksilerek (decreasing by twos) 4'e kadar (until four) azalacaktır. Yani, ilk döngüde $j=30$, ikincide $j=28$ gibi. Ardından döngü bitecek ve akış aşağıya doğru devam edecektir. Bu döngü esnasında j'nin her değeri için noktalı kısımda verilmiş işlemler (processes) tekrarlanacaktır.</p>
<p>A flowchart for a loop where the variable 'k' starts at 1, ends at 99, and increases by 1. The diamond contains 'k = 1, 99'. An arrow points down from the diamond to the next step. A feedback arrow starts from the right side of the diamond, goes right, then up, then left, and finally down into the top of the diamond.</p>	<p>k değişkeni 1'den başlayacak ve 1'er artarak 99'e kadar devam edecektir. Yani, ilk döngüde $k=1$, ikincide $k=2$ gibi. Ardından döngü bitecek ve akış aşağıya doğru devam edecektir. Bu döngü esnasında k'nin her değeri için noktalı kısımda verilmiş işlemler tekrarlanacaktır. Burada dikkat edilmesi gereken nokta; eğer artış miktarı verilmemişse her zaman için bu değer +1'e eşittir.</p>

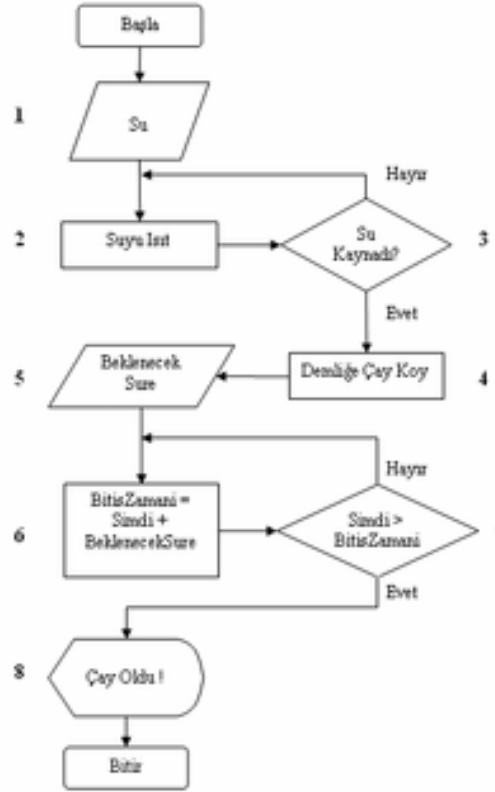
Akış Diyagramları ile Çeşitli Algoritma Örnekleri

ÖRNEK 1: Çay demleme işleminin algoritma ve akış diyagramı ile gösterimi

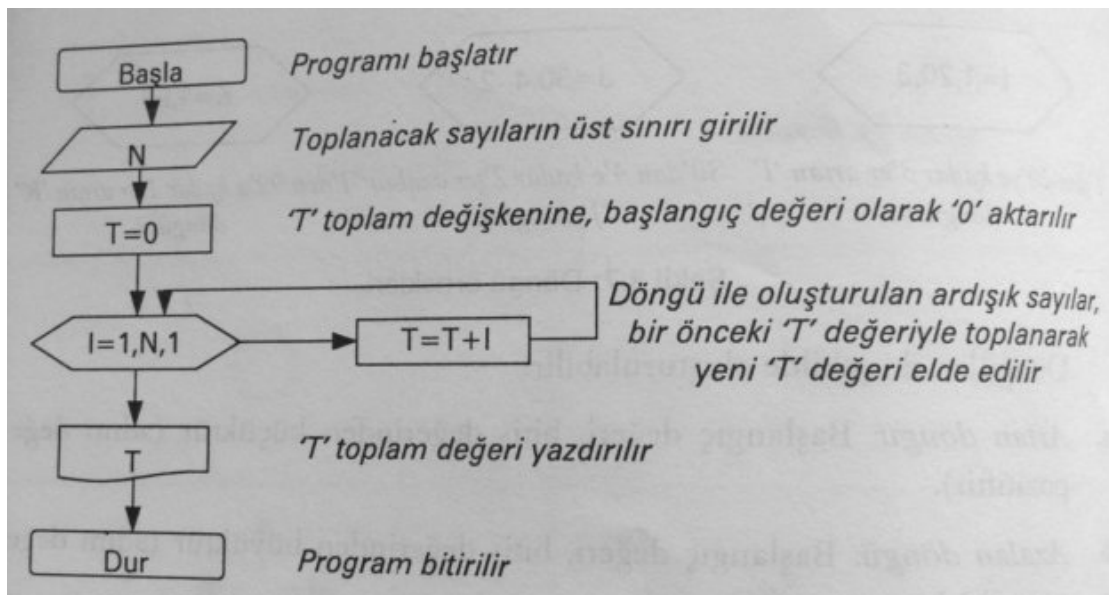
ALGORİTMA:

1. Kullanıcıdan su vermesi beklenir.
2. Suyu ısıtma işlemi yapılır.
3. Suyun kaynayıp kaynamadığı kontrol edilir. Kaynamamışsa 2. adıma dönülür.
4. Çay daha önceden hazır olduğu için, kullanıcıdan beklenmez. Demliğe çay koyma işlemi yapılır.
5. Kullanıcıdan, demleme işleminin ne kadar süreceği bilgisi alınır.
6. Kullanıcıdan alınan demleme süresi ile şimdiki zaman (çayın demlenmeye başladığı zaman) toplanır. Çıkan değer, Bitis Zamani isimli değişkene atılır. Bu değişken demleme işleminin ne zaman biteceği bilgisini tutar.
7. Şimdiki zaman, bitiş zamanından küçükse çayın demlenmesi için ayrılan süre daha dolmamış demektir. Bu süre dolana kadar 7. adım tekrarlanır.
8. Çayın demlendiğini, kullanıcıya ekran üzerinde bildiren bir mesaj çıkartılır.

AKIŞ DIYAGRAMI:



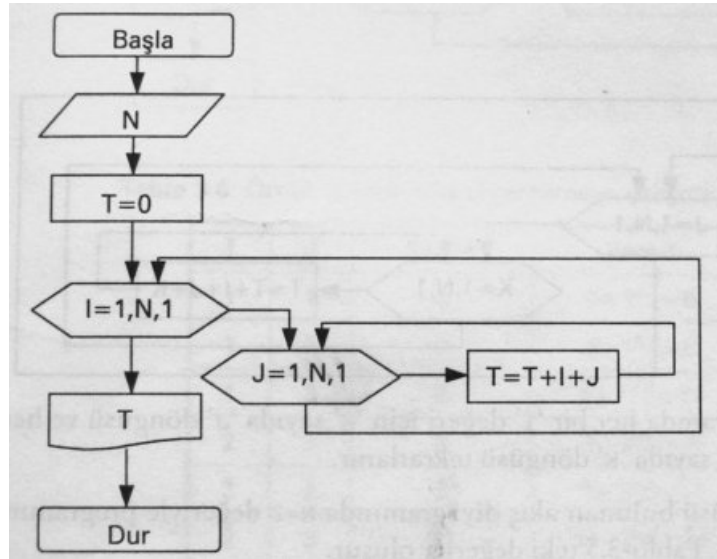
ÖRNEK 2 (DÖNGÜ-Loops): Kullanıcının girdiği N değerine bağlı olarak, 1'den N'e kadar olan sayıların toplamını (sum of numbers one to n) hesaplayan ve ekrana yazdıran programın akış diyagramı:



Eğer N değeri için kullanıcı 5 girmişse i değişkeninin döngünün her çevriminde alacağı değerler aşağıdaki gibi olacaktır:

I	Eski T	Yeni T
1	0	$0+1=1$
2	1	$1+2=3$
3	3	$3+3=6$
4	6	$6+4=10$
5	10	$10+5=15$

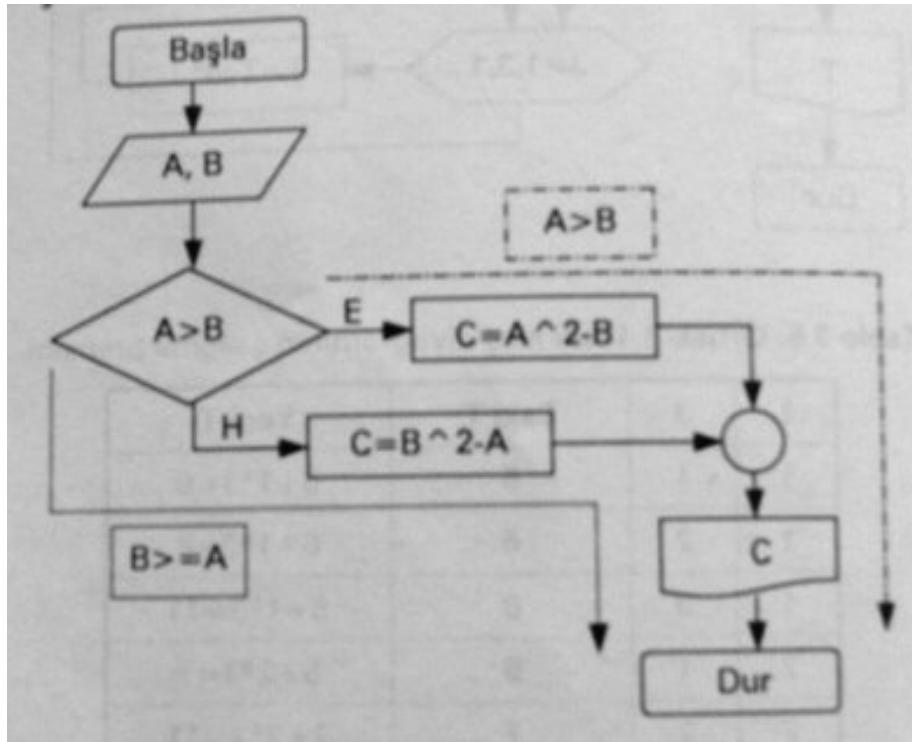
ÖRNEK 3 (İÇ İÇE DÖNGÜ-nested loops): Kullanıcının girdiği N değerine bağlı olarak, içi içe iki döngü (nested loops) ile yapılan işlemlere ait akış diyagramı:



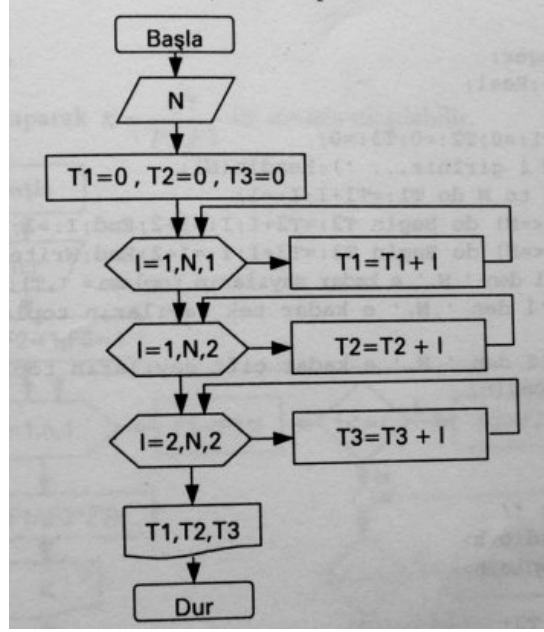
Eğer N değeri için kullanıcı 3 girmişse i ve j değişkenlerinin döngünün her çevriminde alacağı değerler aşağıdaki gibi olacaktır:

I	J	Eski T	Yeni T
1	1	0	$0+1+1=2$
1	2	2	$2+1+2=5$
1	3	5	$5+1+3=9$
2	1	9	$9+2+1=12$
2	2	12	$12+2+2=16$
2	3	16	$16+2+3=21$
3	1	21	$21+3+1=25$
3	2	25	$25+3+2=30$
3	3	30	$30+3+3=36$

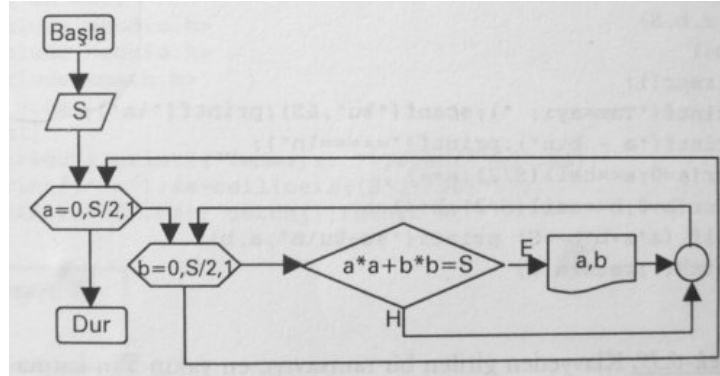
ÖRNEK 4 (KARŞILAŞTIRMA-Decision): Kullanıcının girdiği A ve B değerlerini karşılaştıran ve buna göre $C=A^2-B$ yada $C=B^2-A$ işlemlerini yaptırıp sonucunu ekrana yazdıran programın akış diyagramı.



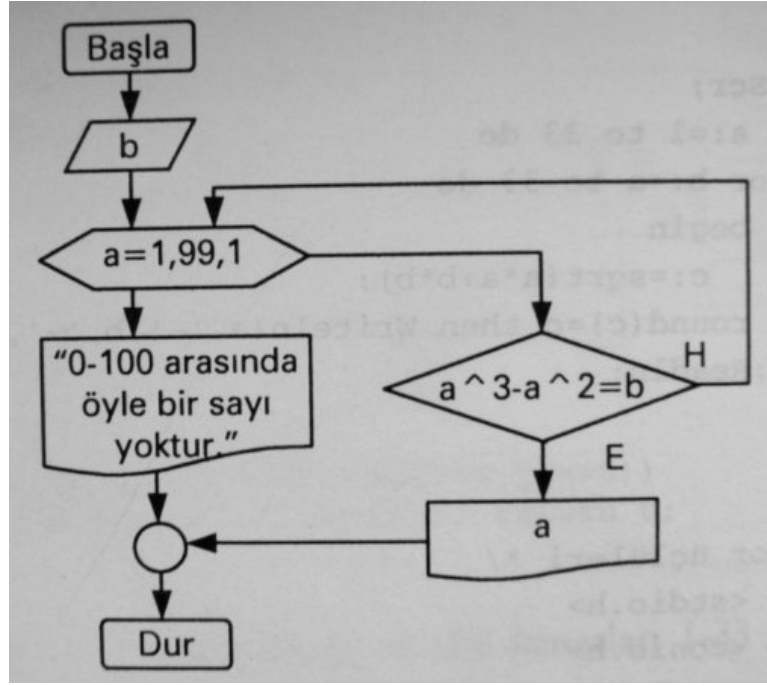
ÖRNEK 5 (DÖNGÜ): Kullanıcının girdiği N değerine bağlı olarak, 1'den N'e kadar olan tam sayıların, tek sayıların ve çift sayıların toplamını hesaplayan ve ekrana yazdıran programın akış diyagramı (Döngülerin başlangıç ve artış değerlerine dikkat)



ÖRNEK 5 (İÇ İÇE DÖNGÜ ve KARŞILAŞTIRMA-nested loops and desicion): Kullanıcının girdiği S tamsayısının iki sayının kareleri toplamı şeklinde yazılıp yazılamayacağını veren programın akış diyagramı:



ÖRNEK 6 (DÖNGÜ ve KARŞILAŞTIRMA-loop and desicion): Kullanıcının girdiği b tamsayısına göre $a^3 - a^2 = b$ şartını sağlayan 0 ile 100 arasındaki sayıları bulan programın akış diyagramı:



3. BİLGİSAYAR PROGRAMLARI İLE GERÇEKLEŞTİRİLEN TEMEL İŞLEMLER

Bilgisayar programları ile gerçekleştirilen işlemler genel olarak üçe ayrılır:

1. Matematiksel (aritmetik) işlemler (Mathematical Processes)
2. Karşılaştırma (karar) işlemleri (Decision Processes)
3. Mantıksal (lojik) işlemler (Logical Processes)

3.1. Matematiksel (aritmetik) işlemler (Mathematical Processes)

Bilgisayarda matematiksel işlemler yaptırılırken mutlaka kendi diline uygun bir ifadeyle yazılmalıdır. Bu ifadeler aşağıdaki gibi olmalıdır.

İşlem	Matematik	Bilgisayar
Toplama	$a + b$	$a + b$
Çıkarma	$a - b$	$a - b$
Çarpma	$a \cdot b$	$a * b$
Bölme	a / b	a / b
Üs alma	a^b	$a \wedge b$

Matematiksel işlemler ve bilgisayar dilindeki karşılıkları

Matematiksel işlem öncelik sıraları (order of priority) çok önemlidir. Bilgisayar işlemleri aşağıdaki tabloda belirtilen sıra ile yapar. Yani, bir matematiksel ifadede, bilgisayar önce parantez içlerini (paranteses), ardından üslü ifadeleri (exponential expression), ardından çarpma (multiplication) ve bölmeyi (division) ve en son olarak da toplama (addition) ve çıkarmayı (subtraction) yapar. İşlemler yazılırken bu sıraya dikkat edilmezse sonuçlar hatalı çıkar.

İşlem öncelik sıraları		
Sıra	İşlem	Bilgisayar dili
1	Parantezler	$((.....))$
2	Üs alma	$a \wedge b$
3	Çarpma ve bölme	$a * b$ ve a / b
4	Toplama ve çıkarma	$a + b$ ve $a - b$

Matematiksel işlem öncelik sıraları

Aynı derecede önceliğe sahip işlemlerde işlem sırası soldan sağdır (Priority order in same level is left to right). Dolayısı ile yazılırken buna dikkat etmek çok önemlidir.

Örneğin A*B/C işleminde bilgisayar önce A ile B'yi çarpacak, ardından çıkanı C'ye bölecektir.

Örnekler:

Matematiksel yazılım	Bilgisayara kodlanması
$a+b-c+2abc-7$	$a+b-c+2*a*b*c-7$
$a+b^2-c^3$	$a+b^2-c^3$
$a-\frac{b}{c}+2ac-\frac{2}{a+b}$	$a-b/c+2*a*c-2/(a+b)$
$\sqrt{a+b}-\frac{2ab}{b^2-4ac}$	$(a+b)^{(1/2)}-2*a*b/(b^2-4*a*c)$
$\frac{a+b-c}{\sqrt{a^2+b^3}}-\frac{2(ab+ac+bc)}{9}$	$(a+b-c)/((a^2+b^3)^{(1/2)})-2*(a*b+a*c+b*c)/9$
$\frac{a^2+b^2}{2ab}$	$(a^2+b^2)/(2*a*b)$
$A+\frac{B.C}{D}-E.F$	$A+B*C/D-E*F$

$a+\sqrt[3]{abc}-\frac{1}{1+\frac{1}{a+\frac{1}{b+\frac{1}{c+\frac{1}{abc}}}}}$	$a+(a*b*c)^{(1/3)}-1/(1+1/(a+1/(b+1/(c+1/(a*b*c))))))$
$\sqrt[3]{\sqrt[4]{a+b-\frac{c}{ab}}}+\frac{1}{\sqrt{1+\frac{1}{\sqrt{1+\frac{1}{abc}}}}}$	$((a-b)^{(1/5))/(a+b-c/(a*b))^{(1/4)})^{(1/3)}+1/((1+1/((1+1/(a*b*c))^{(1/2))))^{(1/2))}$

Matematiksel ifadeler yazılırken parantezlere dikkat edilmezse sonuçlar yanlış çıkacaktır. Aşağıda parantezlerinin yerleri farklı olan aynı ifadelerin sonucu nasıl değiştirdiği görülmektedir.

$$\begin{aligned}
c*d/(a*d)+b+c*d/a &\Rightarrow \frac{cd}{ad}+b+\frac{cd}{a}=\frac{c}{a}+b+\frac{cd}{a} \\
c*d/a*d+b+c*d/a &\Rightarrow c\frac{d}{a}d+b+\frac{cd}{a}=\frac{cd^2}{a}+b+\frac{cd}{a} \\
c*d/a*d+(b+c)*d/a &\Rightarrow c\frac{d}{a}d+\frac{(b+c)d}{a}=\frac{cd^2+(b+c)d}{a}
\end{aligned}$$

3.2. Karşılaştırma (karar) işlemleri (Desicion Processes)

Matematikteki karşılaştırma işaretlerinin bilgisayar dilindeki karşılıkları farklıdır. Bilgisayarda program yazarken bu karşılıklara dikkat etmek çok önemlidir. Bunlar aşağıdaki gibidir.

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

3.3. Mantıksal işlemler (Logical Processes)

Mantıksal işlemleri VE, VEYA ve DEĞİL ifadeleriyle açıklanır. Bunların kullanımı günlük hayattaki kullanımımıza benzer.

Mantıksal işlem	Komut olarak
VE	AND
VEYA	OR
DEĞİL	NOT

Örnek:

$a > 10$ and $b > 10$ ise BİNGO!

$a > 10$ or $b > 10$ ise YESS!

$a > 10$ not ise HEYY!

Buna göre a ve b'nin aşağıdaki değerleri için alınacak sonuçlar:

$a=12, b=15$ için BİNGO! YESS!

$a=8, b=15$ için YESS! HEYY!

$a=13, b=8$ için YESS!

4. VISUAL BASIC SCRIPT (VbScript)

Visual Basic programlama dili, kolay anlaşılır ve konuşma diline yakın komutları olan bir dildir.

Temel düzeyde program kodları yazabilmek için, öncelikle döngü, karşılaştırma, ekrana yazdırma ve değişken tanımlama işlemlerinin VbScript ile nasıl yapıldığından ve fonksiyonlardan bahsedilecek, ardından VbScript’de kullanılan nesneler detaylandırılacaktır.

4.1. Response.Write():

VbScript’te istemcinin ekranına yazdırmak üzere veriyi döndüren metoddur. İstemciye “Dön ve Yazdır” gibi bir anlama gelir. VB’deki Print komutu gibi düşünülebilir. Akış diyagramlarındaki “bilgi-veri yazma” işleminin karşılığıdır. (Response, send to the browser and write to the client screen)

Kullanımı:

A = "Karabük" Response.Write(A)	Tarayıcı ekranına Karabük yazar. Çünkü bir üst satırda A değişkenine Karabük değeri atanmış.
Response.Write("Karabük")	Tarayıcı ekranına Karabük yazar. Çünkü bizzat yazacağı şeyi tanımlıyoruz. (Tırnaklara dikkat!)
Response.Write("Karabük" & "Safranbolu")	Tarayıcı ekranına KarabükSafranbolu yazar. & işareti yan yana yaz demektir. (String ifadeler için & yerine, + da kullanılabilir.)
A ="Karabük" Response.Write(A & "Safranbolu")	Tarayıcı ekranına KarabükSafranbolu yazar. Çünkü A değişkendir ve bir üst satırda belirlenmiştir. Safranbolu ise tırnak içinde yazılmıştır, dolayısı ile bizzat kendisi yazılacaktır.

Response.write metodunun kullanarak VbScript ile HTML kodu yazmanız yani bir tür hile yapmanız da mümkündür:

Örnek:

Response.Write("<body bgcolor= '#000000'>")

Yazdığınızda sunucu <body bgcolor='#000000'> ifadesini istemciye gönderecektir. Bu ifade tarayıcı tarafından yorumlanabilir bir HTML ifadesi olduğu için tarayıcı bunu yorumlayacak ve sayfa rengini değiştirecektir. VbScript etiketi ile birlikte kullanılmak şartı ile Response.write yerine kısaca eşittir işareti de kullanılabilir:

Örn1: <%=("Karabük")%> Şununla aynı sonucu döndürür: Response.Write("Karabük")

Örn2: <%=A%> Şununla aynı sonucu döndürür: Response.Write(A)

Dikkat: Response.Write yerine eşittir işareti konulacaksa mutlaka yukarıdaki gibi VbScript etiketleri arasında kullanılmalıdır. Daha önce etiket açılmış olsa bile kapatılmalı ve eşittir için tekrar açılmalıdır.

4.2. FOR-NEXT Döngüsü (For-Next Loop)

Akış diyagramlarındaki döngü işlemlerinin VbScript’de kodlanması bu komutlarla yapılır.

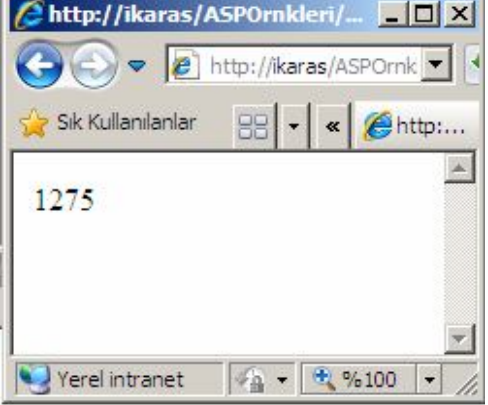
Kullanımı:

FOR değişken=başlangıç değeri TO bitiş değeri STEP miktar

.....
NEXT

şeklindedir. Buradaki NEXT ifadesi döngünün geriye dönme noktasını belirtir. Yani FOR ile başlayan döngü NEXT’e kadar gider ve ilk çevrimini tamamlar. NEXT’i görünce tekrar FOR’a döner ve ikinci çevrime başlar. Bu şekilde döngü bitene kadar devam eder.

Örnek: 1’den 50’ye kadar olan sayıların toplamını hesaplayan ve istemciye döndüren bir VbScript program kodu:

<pre><% Option Explicit Dim t, i t = 0 For i = 1 To 50 t = t + i Next Response.Write(t) %></pre>		<p>1’den 50’ye kadar olan rakamlar toplanıyor ve sonuç istemciye (1275) gönderiliyor. STEP yazılmadığı için döngü birer birer artacaktır. Burada a değeri 50 olduğu için program FOR ile NEXT arasında 50 çevrim yapacaktır.</p>
---------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Örnek dosyalar: fornex1.asp, fornex2.asp

<http://www.ismailkaras.com/asp/fornex1.asp>

<http://www.ismailkaras.com/asp/fornex2.asp>

Do while / Do until Şartlı Döngüsü (Do while / Do until Loop)

Yapısı ForNext’ten daha farklıdır. Döngüden çıkış şarta bağlıdır. Döngü iki şekilde devam eder: 1-Şart devam ettiği sürece (while), 2- şart sona erdiği sürece (until). Şart döngünün başında olabileceği gibi sonunda da olabilir.

Örnekler:

<pre>a=1 Do while a<100 a=a+1 b=b+a Loop</pre>	<pre>a=1 Do a=a+1 b=b+a Loop while a<100</pre>
<pre>a=1 Do until a=100 a=a+1 b=b+a Loop</pre>	<pre>a=1 Do a=a+1 b=b+a Loop until a=100</pre>

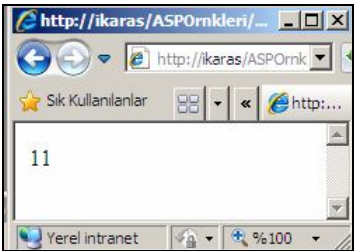
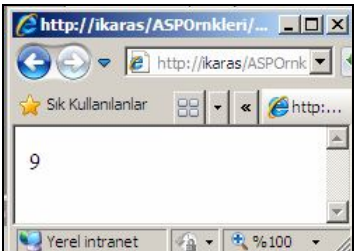
4.3. IF-THEN Karşılaştırma Komutları (IF-THEN Statement)

Akış diyagramlarındaki Karar-Karşılaştırma ifadelerinin karşılığıdır. Burada belirtilen kriterlere bağlı olarak programın akışı değişir.

Kullanımı:

IF şart THEN1 END IF	Şart sağlandığında noktalı kısımda belirtilen işlemler yapılacaktır.
IF şart THEN1 ELSE2 END IF	Şart sağlandığında 1 numaralı kısımda belirtilen işlemler, sağlanmadığında ise 2 numaralı kısımdakiler yapılacaktır.
IF şart1 THEN1 ELSEIF şart2 THEN2 ELSEIF şart3 THEN3 ELSE4 END IF	Şart1 sağlandığında 1 numaralı kısımda belirtilen işlemler yapılır. Sağlanmadığı durumda yeni şartlar eklenebilir (şart2, şart3,...). Sağlanmayan her yeni şart için bir sonraki şarta bakılır. Hiç birisi de sağlanmıyorsa 4 numaralı kısımdakiler yapılacaktır.

Örnek:

<pre><% Option Explicit Dim a, b, c a=5 b=6 If a < 10 Then c = a + b Else c = a - b End If Response.Write(C) %></pre> 	<pre><% Option Explicit Dim a, b, c a=15 b=6 If a < 10 Then c = a + b Else c = a - b End If Response.Write(C) %></pre> 	a değeri 10'dan küçük olursa a ile b'yi toplanacak, büyükse çıkarılacak ve sonuç istemcinin tarayıcısına gönderilecektir.
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------

Örn:

```
<%  
Option Explicit  
Dim a, b, c  
a=7  
b=6
```

```
If a < 10 Then
  c = a + b
elseif a>10 and a<13 then
  c = a * b
elseif a>=13 and a<17 then
  c = a / b
Else
  c = a - b
End If
Response.Write( C )
%>
```

Eğer şart tek bir satırda ifade edilebilecek cinstense, “end if” kullanmadan da yazılabilir: IF şart THEN işlem

Örn:

```
<% If a < 10 Then c = a + b %>
```

Konuyla ilgili örnek dosyalar: If1.asp, if2.asp

<http://www.ismailkaras.com/asp/If1.asp>

<http://www.ismailkaras.com/asp/if2.asp>

SELECT-CASE Şartı (Select-Case Statement)

Duruma bağlı olarak yönlendirme yapmak için kullanılır.

ÖRN:

```
Select Case Hour(Now)
  Case 0,1,2,3,4,5,6,7,8,9,10,11
    Response.Write "Günaydın!"
  Case 12,13,14,15,16,17
    Response.Write "Tünaydın"
  Case Else
    Response.Write "İyi Akşamlar!"
End Select
```

4.4. DEĞİŞKENLER (Variants)

Akış diyagramları ve program örneklerinde de gördüğünüz üzere kodlama yaparken sık sık değişkenlerden faydalanıyoruz. Değişkenler girilen yada üretilen değerleri saklamamızı sağlayan geçici olarak içi doldurulan elemanlardır. Matematikteki x, y gibi terimlere benzetilebilir. Değişkenler harf (letter), rakam (number) ve bazı işaretlerden (sign) oluşabilirler fakat değişkenleri isimlendirirken bazı kurallara uymak gerekir. Değişkenler kesinlikle bir harfle başlamalıdır, rakam yada bir işaretle başlayamazlar. Tanımlanan değişkenlerin harflerinde boşluk (space) yada Türkçe karakterler kullanılmamalıdır. Aşağıda doğru ve yanlış olarak tanımlanmış değişken isimleri görülmektedir.

A, R, c, s, x, pi, ali, isim, ISIM, no, ogrenci_no, a1, F4, memleketi	DOĞRU
1A, 6t, _rt, uyruğu, İSİM, adı soyadı, +,	YANLIŞ

Yukarıdaki örneklerde de göreceğiniz üzere VBScript'te değişkenler Dim komutu ile tanımlanmaktadır. Eğer VbScript kodunun başında **<% Option Explicit %>** ifadesi kullanılmışsa bütün değişkenler tek tek kullanıcı tarafından tanımlanmalıdır. Bu ifade kullanılmamışsa, tanımlama yapılmasa bile sunucu bu değişkenleri kabul edecektir. Fakat sunucunun yanlış yorumlamasından kaçınmak için **<% Option Explicit %>** ifadesini kullanmamız ve sonrasında tüm değişkenleri kendimizin tanımlaması daha uygun bir yöntemdir.

DİZİLER (Dizi Şeklindeki Değişkenler) (Arrays)

Değişkenler tek bir değer tutabileceği gibi, dizi şeklinde tanımlandığında birden fazla değeri de tutabilirler. VBScript'deki diziler matematikteki matrislerin (matrix) karşılığıdır. Aynen matrislerde olduğu gibi nxm boyutunda diziler oluşturulabilir ve bu boyuttaki eleman sayısınca o diziye eleman atanabilir.

ÖRNEK 1 (Bu örnekte 4x3 boyutundaki bir diziye 0 ile 1 arasında rastgele sayıların atanması ve bu sayıların istemcinin ekranına yazdırılması görünmektedir. Rastgele sayı üretme fonksiyonu için Bkz Fonksiyonlar bölümü):

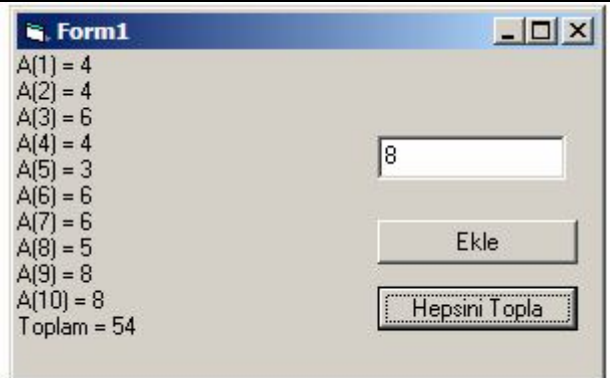
<http://www.ismailkaras.com/asp/dizi.asp>

```
<%  
Dim A(3, 2)  
For i = 0 To 3  
    For t = 0 To 2  
        randomize  
        A(i, t) = rnd()  
        Response.write ( "A(" & i & "," & t & ") = " & A(i, t))  
        Response.write ("<br>")  
    Next  
Next  
>%
```

Aşağıdaki örnekler ise VB 6.0 kodları olsa da dikkatli incelediğinizde dizilere veri girişi ve veri okuma şeklini kavrayacaksınız.

ÖRNEK 2:

```
Dim t As Integer  
Dim a(10) As Integer  
  
Private Sub Command1_Click()  
    t = t + 1  
    a(t) = Text1.Text  
    Print "A(" & t & ") = " & a(t)  
End Sub  
  
Private Sub Command2_Click()  
    For p = 1 To t  
        tp = tp + a(p)  
    Next p  
    Print "Toplam = " & tp  
End Sub  
  
Private Sub Form_Load()  
    t = 0  
End Sub
```



Sol en üstte de tanımlandığı üzere a dizisi 10x1'lik bir dizidir. Ekle butonuna her başta text kutusunun içine yazılan değer diziye eklenmekte ve formun üzerine yazılmaktadır. Hepsini Topla butonuna basıldığında ise dizinin tüm elemanları birbiri ile toplanmakta ve en sona yazılmaktadır.

ÖRNEK 3:

```
Dim A(3, 2) As Integer
```

```
Private Sub Form_Load()
```

```
For i = 1 To 3
```

```
For t = 1 To 2
```

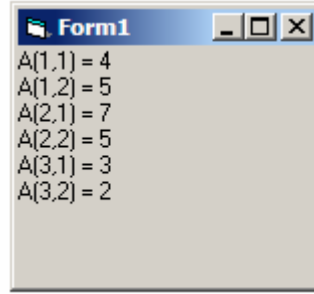
```
    A(i, t) = InputBox("A(" & i & "," & t & ")  
    değeri giriniz...")
```

```
    Print "A(" & i & "," & t & ") = " & A(i, t)
```

```
Next t
```

```
Next i
```

```
End Sub
```



Form1

A(1,1) = 4
A(1,2) = 5
A(2,1) = 7
A(2,2) = 5
A(3,1) = 3
A(3,2) = 2

Sol en üstte de tanımlandığı üzere A dizisi 3x2'lik bir dizidir. Yandaki program kodu ile önce InputBox fonksiyonu kullanılarak bu diziye eleman girişi yapılmakta ardından, elemanlar formun üzerine üstteki gibi sıralanmaktadır.

(Dizilerle ilgili bir başka örnek için bakınız: gunler.asp)

4.5. ASCII KOD TABLOSU (ASCII Code Table)

Bilgisayarda kullanılan temel karakterlerin (harf, rakam, simge ve işaretler) her birinin bir kodu vardır. Bu kod aşağıda listelendiği gibidir. Programlama esnasında bu kodlardan sık sık faydalanılır. Buna göre H harfinin ASCII kodu 72, h harfinin kodu 104, 5 rakamının 53 vb. dir.

0		32		64	Q	96	ˆ	128	Ç	160	á	192	L	224	α
1	☐	33	!	65	À	97	a	129	ü	161	í	193	⊥	225	β
2	☐	34	"	66	B	98	b	130	é	162	ó	194	⊤	226	Γ
3	♥	35	#	67	C	99	c	131	â	163	ú	195	⊥	227	π
4	♦	36	\$	68	D	100	d	132	ä	164	ñ	196	—	228	Σ
5	♠	37	%	69	E	101	e	133	à	165	ñ	197	†	229	σ
6	♣	38	&	70	F	102	f	134	ã	166	±	198	‡	230	μ
7	•	39	'	71	G	103	g	135	ç	167	°	199		231	τ
8	◼	40	(72	H	104	h	136	ê	168	¿	200	u	232	ϑ
9	◊	41)	73	I	105	i	137	ë	169	⌈	201	⌈	233	θ
10	◼	42	*	74	J	106	j	138	è	170	⌋	202	u	234	Ω
11	♂	43	+	75	K	107	k	139	ï	171	½	203	⌈	235	δ
12	♀	44	,	76	L	108	l	140	î	172	¼	204		236	ω
13	ℓ	45	_	77	M	109	m	141	ì	173	ı	205	=	237	∅
14	ℓ	46	.	78	N	110	n	142	Ë	174	«	206		238	€
15	*	47	/	79	O	111	o	143	â	175	»	207	±	239	Π
16	▶	48	0	80	P	112	p	144	É	176	⋮	208	u	240	≡
17	◀	49	1	81	Q	113	q	145	æ	177	⋮	209	⌈	241	±
18	±	50	2	82	R	114	r	146	ff	178	⋮	210	π	242	≥
19	!!	51	3	83	S	115	s	147	ô	179		211	u	243	≤
20	¶	52	4	84	T	116	t	148	ö	180	⌋	212	⌋	244	∫
21	§	53	5	85	U	117	u	149	ò	181	⌋	213	⌋	245	J
22	—	54	6	86	V	118	v	150	û	182		214	π	246	÷
23	±	55	7	87	W	119	w	151	ù	183	π	215		247	≈
24	↑	56	8	88	X	120	x	152	ÿ	184	⌋	216	⌋	248	°
25	↓	57	9	89	Y	121	y	153	ÿ	185		217	J	249	·
26	→	58	:	90	Z	122	z	154	ÿ	186		218	⌈	250	·
27	←	59	;	91	[123	{	155	ç	187		219	⋮	251	J
28	↖	60	<	92	\	124		156	£	188		220	⋮	252	n
29	↗	61	=	93]	125	}	157	¥	189		221	⋮	253	z
30	▲	62	>	94	^	126	~	158	℔	190	J	222	⋮	254	■
31	▼	63	?	95	_	127	△	159	f	191	⌋	223	⋮	255	

ASCII Kod Tablosu

4.6. VBSCRIPT'DE FONKSİYONLAR (Functions)

VbScript'te fonksiyonların kullanımı matematikteki fonksiyonlara benzer. Bir değişken yada değeri belli bir işlemde geçirirler ve yeni bir sonuç üretirler. Yazım şekilleri genelde aynıdır; önce fonksiyon yazılır, ardından parantez içinde değer/değişken ve kriterler yazılır. Örn: SQR(A) veya SQR(36). Yüzlerce farklı fonksiyon mevcuttur. Biz burada bazı temel fonksiyonları ve kullanımlarını göreceğiz.

Bazı temel fonksiyonlar:

ASC Fonksiyonu

Verilen karakter dizisinin ilk karakterinin ASCII kodunu verir.

Örnek kullanımı 1: ASC(B) :B değişkeninin içerdiği değerin ilk harfinin ASCII karşılığını üretir.

Örnek kullanımı 2: ASC("Karabük") :K harfinin ASCII karşılığını üretir.

CHR Fonksiyonu

Verilen bir ASCII kodunun karakter cinsinden karşılığını verir.

Örnek Kullanımı: CHR(112) :112 kodlu karakteri, yani p harfini üretir.

SPACE Fonksiyonu

Verilen sayıda boşluk elde etmek için kullanılır.

Örnek Kullanımı: Space(8) : 8 karakter boşluk üretir.

ARRAY Fonksiyonu

Parantez içinde verilen değerlere bağlı olarak bir dizi oluşturur. Dizin eleman sayısını ve elemanlarını parantez içindeki değerler belirler.

ÖRN:

```
<%  
yonler=Array("kuzey","guney","dogu","bati")  
for i=0 to 3  
response.write yonler(i) & "<br>"  
next  
%>
```

Sonuç:

```
kuzey  
guney  
dogu  
bati
```

STRING Fonksiyonu

Belirli bir ASCII kodlu karakterden belli sayıda üretmek için kullanılır.

Örnek Kullanımı 1: STRING(6, 146) : 146 kodlu karakterden 6 adet üretir.

Örnek Kullanımı 2: STRING(6, "R") : 6 adet R harfi üretir.

ÖRN:

```
<%  
response.write string(5, "!") 'sayfaya "!!!!!" yazar  
%>
```

RIGHT Fonksiyonu

Verilen bir değişken yada değer içinde sağdan belirtilen kadar karakteri ayırır.

Örnek Kullanımı 1: Right(A, 5) : A değişkenine kayıtlı değerin sağından 5 karakter alır

Örnek Kullanımı 2: Right("Ankara", 4) : kara harflerini elde eder.

LEFT Fonksiyonu

Verilen bir değişken yada değer içinde soldan belirtilen kadar karakteri ayırır.

Örnek Kullanımı 1: Left(A, 5) : A değişkenine kayıtlı değerin solundan 5 karakter alır

Örnek Kullanımı 2: Left("Ankara", 4) : Anka harflerini elde eder.

ÖRN:

```
<%
str="Kuze eylen toprağım sunun anınla yare su"
response.write left(str,10) 'sayfaya "Kuze eylen" yazar
%>

<%
str="Kuze eylen toprağım sunun anınla yare su"
response.write right(str,7) 'sayfaya "yare su" yazar
%>
```

MID Fonksiyonu

Verilen bir değişken yada değer içinde m'nci bir karakterden itibaren n adet karakteri ayırır.

Örnek Kullanımı 1: Mid(A, m, n)

Örnek Kullanımı 2: Mid(A, 5, 3) : A değişkenine kayıtlı değer 5'inci karakterinden itibaren 3 karakteri ayırır.

Örnek Kullanımı 3: Mid("Ankara", 3, 2) : "ka" harflerini elde eder.

ÖRN:

```
<%
str="Kuze eylen toprağım sunun anınla yare su"
response.write mid(str,6,5) 'sayfaya "eylen" yazar
%>
```

LEN Fonksiyonu

Verilen bir değişken yada değer karakter cinsinden uzunluğunu verir.

Örnek Kullanımı 1: Len(A) : A değişkeninin karakter cinsinden uzunluğunu verir.

Örnek Kullanımı 2: Len("Ankara") : 6 değerini verir. Çünkü Ankara altı karakterdir.

ÖRN:

```
<%
str="Karabük Üniversitesi Mühendislik Fakültesi"
response.write len(str) 'sayfaya "42" yazar.
%>
```

LCASE Fonksiyonu

Verilen bir değişken yada karakter dizisini küçük harfe çevirir.

Örnek Kullanımı 1: Lcase(F) : F değişkenine kayıtlı değeri küçük harfe çevirir.

Örnek Kullanımı 2: Lcase("ANKARA") : ankara kelimesine dönüştürür

UCASE Fonksiyonu

Verilen bir değişken yada karakter dizisini büyük harfe çevirir.

Örnek Kullanımı 1: UCase(A) : A değişkenine kayıtlı değeri büyük harfe çevirir.

Örnek Kullanımı 2: UCase("ankara") : ANKARA kelimesine dönüştürür

Not: Türkçe karakterler için özel bir fonksiyon oluşturmak gerekir.

ÖRN:

```
<%
response.write lcase("AsP") 'sayfaya "asp" yazar
response.write ucase("asP") 'sayfaya "ASP" yazar
response.write lcase("KİTAP") 'sayfaya "kİtap" yazar
response.write ucase("tırnak") 'sayfaya "TıRNAK" yazar

%>
```

TRIM Fonksiyonu

Verilen bir değişken yada karakter dizisinin iki yanındaki boşlukları kırpar.

Örnek Kullanımı : Trim(" ankara ")

LTRIM Fonksiyonu

Verilen bir değişken yada karakter dizisinin solundaki boşlukları kırpar.

RTRIM Fonksiyonu

Verilen bir değişken yada karakter dizisinin sağındaki boşlukları kırpar.

ÖRN:

```
<%
str=" Fuzûli "
response.write Ltrim(str) ' sayfaya "Fuzûli " yazar
response.write Rtrim(str) ' sayfaya " Fuzûli" yazar
response.write trim(str) ' sayfaya "Fuzûli" yazar

%>
```

REPLACE Fonksiyonu :

Bir string içinden belirlenen bir ifadeyi, belirlenen başka bir ifade ile değiştirir.

Kullanımı :

```
replace(str,StrAra,StrDegistir,index,degisiklik_sayısı,karsilastirma_kodu)
```

str: İçinde değişiklik yapacağımız değişken

StrAra : Değiştirmek üzere arayacağımız ifade

StrDegistir : Aranan ifade bulunduğunda yerine konulacak değer

index : Değiştirmeye kaçınıcı karakterden itibaren başlayacağız ? Varsayılan değer 1. 1 den yüksek bir değer vererseniz değiştirme sonucu verdiğiniz değerden önceki karakterleri almıyacaktır !

degisiklik_sayısı : -1 yazarsanız bulduğu tüm ifadeleri değiştirir. Varsayılan -1

karsilastirma_kodu : 0 vererseniz büyük-küçük harf ayrımı yapar, 1 vererseniz yapmaz; varsayılan değer 0

```
<%
str="Karabük Üniversitesi Edebiyat Fakültesi"

response.write replace(str,"Edebiyat","Fen",1,-1,1)
'Sayfaya " Karabük Üniversitesi Fen Fakültesi" ifadesini yazar

response.write replace(str,"edebiyat","Mühendislik",1,-1,0)
```

```
'Sayfaya "Karabük Üniversitesi Edebiyat Fakültesi" ifadesini yazar  
(karşılaştırma koduna dikkat !)  
%>
```

INSTR Fonksiyonu :

Bir metin (String) içinde verdiğimiz ifadeyi arar, bulursa başlangıç sayısını bize değer olarak geri verir. Bulamazsa dönen değer 0 (sıfır) olur.

Kullanımı :

```
instr(index, str, StrAra, karsilastirma_kodu)
```

index : Aramaya başlanacak karakter sayısını belirler. varsayılan değer 1. Dikkat etmeniz gereken nokta verdiğiniz değerden önceki karakterlere bakmaz.

str : Arama yapacağımız değişken

StrAra : string içinde Aranacak değer

karsilastirma_kodu : Küçük büyük harf ayrımı. varsayılan değer 0

```
<%  
str="Karabük Üniversitesi Edebiyat Fakültesi"  
response.write instr(1,str,"Üniver",1) ' 9 sayısı döner  
  
str="Karabük Üniversitesi Edebiyat Fakültesi"  
response.write instr(1,str,"üniver",0) ' 0 sayısı döner (karşılaştırm  
koduna dikkat)  
  
%>
```

SPLIT fonksiyonu :

Verdiğimiz değişken içindeki ayraçları kullanarak bize tek boyutlu bir dizi değişken oluşturur (Array)

Kullanımı :

```
split(str, ayraç, sayi, karsilastirma_kodu)
```

```
<%  
str="Karabük Üniversitesi Mühendislik Fakültesi Bilgisayar Mühendisliği  
Bölümü"  
  
StrDizi=split(str,"i",-1,1)  
  
for t=0 to ubound(strdizi)  
  
response.write "strDizi(" & t & ") = " & strDizi(t) & "<br>"  
  
next  
  
response.write ubound(strdizi)  
  
%>
```

LBOUND ve UBOUND fonksiyonları:

Bir dizinin en küçük ve en büyük indis değerlerini verir. En küçük indis değeri doğal olarak 0'dır.

Kullanımı:

LBound(dizi_ismi [,boyutu])

UBound(dizi_ismi [,boyutu])

Parametreleri:

dizi_ismi gerekli.

boyutu Opsiyonel. Dizinin hangi boyutunun alt yada üst değerinin istendiğini bildirir.

Örnekler:

Örnek 1:

```
<%  
days=Array("Sun","Mon","Tue","Wed","Thu","Fri","Sat")  
document.write(LBound(days) & "<br />")  
document.write(UBound(days) & "<br />")  
%>
```

Sonuç:

0

6

Örnek 2:

```
<%  
Dim food(2,3)  
food(0,0)="Apple"  
food(0,1)="Banana"  
food(0,2)="Orange"  
food(0,3)="Lemon"  
food(1,0)="Pizza"  
food(1,1)="Hamburger"  
food(1,2)="Spaghetti"  
food(1,3)="Meatloaf"  
food(2,0)="Cake"  
food(2,1)="Cookie"  
food(2,2)="Icecream"  
food(2,3)="Chocolate"  
response.write(LBound(food,1) & "<br />")  
response.write(UBound(food,1) & "<br />")  
response.write(LBound(food,2) & "<br />")  
response.write(UBound(food,2) & "<br />")  
%>
```

Sonuç:

0

2

0

3

CCur() Fonksiyonu

CCur(Sayı)

CCur fonksiyonu girilen herhangi bir sayı değerini Currency yani para birimine dönüştürmeye yarar. Yapılabilecek sayı aralığı ise :

-922,337,203,685,477.5808 to 922,337,203,685,477.5807

Örnek Kod:

```
<% orneksayi=(12345) %>
```

```
<% =CCur(orneksayi) %>
```

Çıktı:

12345

Bu fonksiyon 4 desimal değere yuvarlama yapar.

Örnek Kod:

```
<% orneksayi=(55555.123456) %>
```

```
<% =CCur(orneksayi) %>
```

Çıktı:

55555.1235

Bazı Matematiksel Fonksiyonlar:

SIN, COS ve TAN Fonksiyonları

Verilen bir değişken yada değerın sinüs, cosinüs yada tanjantını alırlar. Dikkat edilmesi gereken nokta açı değeri radyan cinsinden olmalıdır. (180 derece = 3,1415.. radyan)

Örnek Kullanımlar : $\cos(C)$, $\sin(3.14)$

ATN Fonksiyonu

Verilen bir değişken yada değerin arctanjantını alır.

LOG Fonksiyonu

Verilen bir değişken yada değerin logartimasını alır.

Örnek Kullanımlar : $\log(C)$, $\log(54)$

SQR Fonksiyonu

Verilen bir değişken yada değerin karekökünü alır.

Örnek Kullanımlar : sqr(C) , sqr (36)

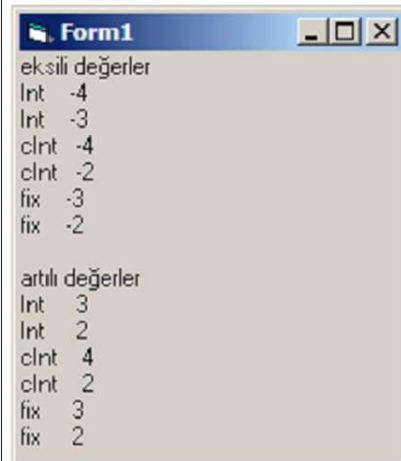
CINT, INT ve FIX Fonksiyonları

Verilen rakamsal bir değişken yada değerin tamsayı kısmını verir yada bir üst veya bir alt rakama yuvarlar. Seçilen fonksiyona bağlı olarak bu durum değişir.

ÖRNEK 4:

```
Private Sub Form_Click()
```

```
Print "eksili değerler"
Print "Int "; Int(-3.65)
Print "Int "; Int(-2.35)
Print "cInt "; CInt(-3.65)
Print "cInt "; CInt(-2.35)
Print "fix "; Fix(-3.65)
Print "fix "; Fix(-2.35)
Print: Print "artılı değerler"
Print "Int "; Int(3.65)
Print "Int "; Int(2.35)
Print "cInt "; CInt(3.65)
Print "cInt "; CInt(2.35)
Print "fix "; Fix(3.65)
Print "fix "; Fix(2.35)
```



INT, CINT ve FIX fonksiyonlarının nega
vada varının üstünde veya altında olu

RND

Rastgele sayı üretmek için kullanılır.

Örn:

```
<%
```

```
dim a
randomize
a=rnd()
response.write a
```

```
%>
```

Zaman ve Tarih Fonksiyonları:

Visual Basic'in hemen hemen bütün zaman-tarih fonksiyonları VBScript'te de kullanılır.

Date: Bugün tarihini verir. (25.03.2000 gibi.)

Time: O andaki saati verir. (22:24:40 gibi.)

Now: O andaki tarih ve saati birlikte verir. (25.03.2000 22:24:40 gibi.)

VBScript'in buna ek olarak Weekday (haftanın günü), WeekdayName (günün adı) ve Monthname (ayın adı) fonksiyonları da vardır. Bu fonksiyonlar değerlerini Date fonksiyonuna göre alırlar. Örneğin,

```
<%= WeekdayName(Weekday(Date))%>
```

komutu bize bugün Cumartesi ise "Cumartesi" değerini verir.

```
<%= MonthName(Month(Date))%>
```

komutu bize bu ay Mart ise "Mart" değerini verir. VBScript'in bunlara ek olarak Day (gün), Month (ay) ve Year (yıl) fonksiyonları da değerlerini Date fonksiyonundan alarak, size bir rakam verirler. Eğer tarih 25 Mart 2000 ise:

```
<%= Day(Date)%>
```

```
<%= Month(Date)%>
```

```
<%= Year(Date)%>
```

değerini verir.

VBScript, bu değerleri doğrudan işletim sisteminden alır. Dolayısıyla, işletim sisteminin bölgesel ayarları Türkiye için yapılmışsa, gün adları Türkçe olarak dönecektir. Ayrıca, tarih ve saat biçimleri de bölgesel ayarlara bağlı olarak, ay önde, gün arkada veya tersi, saat de 12 saat veya 24 saat esasına göre döner. ASP programlarınızı kişisel Web Server'da denerken kendi bilgisayarınızın tarih ve saatini; gerçek Internet'te çalıştırırken Server'ın tarih ve saatini alırsınız. Sayfalarınızda ay ve gün adlarını Türkçe görüntülemek için, önce server'ın bölgesel ayarlarını denemeniz ve eğer isimler Türkçe gelmiyorsa, bunları çeviren sub'lar veya fonksiyonlar yazmanız gerekebilir. (Yukarıda günün tarihini veren kodumuzda, sisteminizin bölge ayarları nasıl olursa olsun ay Türkçe olarak karşımıza çıkacaktır.)

ÖRNEK:

```
<%= Date%><br>
```

```
<%= Day(date)%><br>
```

```
<%= Month(Date)%><br>
```

```
<%= Year(Date)%><br>
```

```
<%=Weekday(Date)%><br>
```

```
<%=Weekday("12.06.1967")%><br>
```

```
<%= WeekdayName(Weekday("12.06.1967"))%><br>
```

```
<%=Month(Date)%><br>
```

```
<%= MonthName(Month(Date))%>
```

Dinamik Dizi Fonksiyonu (Dynamic Array)

Dizi değişkenler, özellikle Web ziyaretçilerimizden gelecek bilgilerin kaydedilmesinde; veritabanından çekeceğimiz verilerin kullanılabilir hale getirilmesinde yararlı bir araçtır. Dolayısıyla ASP sayfalarınızda sık sık çok boyutlu dizi değişkenlerden yararlanacaksınız. Bunun için gerekli araçları kısaca ve topluca ele almamız yerinde olur. Bir dizi değişken oluştururken, değişkenin eleman sayısını belirtmezsek, VBScript, kendi kendine "Anlaşılan bu diziyi dinamik yapmamı istiyorlar!" der. Daha sonra elemanlarının değerleri sonradan belirtilebilecek ve eleman sayısı sonradan artırılabilen bir dinamik dizi değişken oluşturur. Örnek:

```
Dim Ogrenciler()
```

Bu komutla, Ogrenciler dizi deęiřkeni oluřturulur. Ancak, eleman sayısı belirtilmedięi iin dizi dinamiktir. Daha sonra bu dizinin eleman sayısını belirleyebilirsiniz. Bunu,

ReDim Ogrenciler(15)

gibi bir komutla yapabiliriz. řimdi aklınıza řu soru gelebilir: “Peki neden Ogrenciler dizisini bařtan eleman sayısını belirterek tanımlamıyoruz?” Gzel soru! Cevabı řu olabilir mi? “Dizi deęiřkenimizin eleman sayısını henz bilmiyoruz. Programın akıřı iinde bu sayı, bařka bir fonksiyonun, sub’ın veya kullanıcı girdisinin sonucu olarak belirlenebilir.” Fakat hemen belirtmek gereken bir nokta var: ReDim komutu, mevcut bir dizi deęiřkenin iindeki herřeyi siler! Mevcut dizinin elemanlarını ve onların deęerlerini korumak istiyorsak:

ReDim Preserve Ogrenciler(20)

yazmamız gerekir. Buradaki Preserve (koru) komutu, VBScript’e mevcut dizi iindeki elemanları korumasını ve eleman sayısını 20’ye ıkartmasını bildirir. Buna neden gerek olabilir? Ziyaretinin tercihleri deęiřebilir. rneęin, bir elektronik alıřveriř sitesinde ziyaretiniz yeni řeyler alabilir. Daha nceki alıřveriřlerine iliřkin verileri tuttuęunuz dizi deęiřkenin eleman sayısını, daha nceki bilgileri silmeden arttırmanız gerekir. VBScript’in dizi deęiřkenlerini tm aynı adı tařıyan bir liste olarak dřnebilirsiniz; sadece deęiřken adının yanında dizinin kaıncı elemanı olduęunu belirten sayı bulunur:

Ogrenciler(1): Necip

Ogrenciler(2): Serap

Ogrenciler(3): Neslihan

Fakat VBScript ok boyutlu dizi deęiřken de oluřturabilir. İki boyutlu dizi deęiřkeni tablo gibi dřnn. Dizinin elemanları aynı adı tařıyan deęiřkenler fakat bu kez sadece tek sayı deęil sıra ve stn numaraları ile belirleniyorlar:

Ogrenciler(1,1): Necip

Ogrenciler(1,2): Serap

Ogrenciler(1,3): Neslihan

Ogrenciler(2,1): Selim

Ogrenciler(2,2): Murat

Ogrenciler(2,3): Merve

Ogrenciler(3,1): Elif

Ogrenciler(3,2): Hande

Ogrenciler(3,3): Leyla

řimdi, burada  sıralı,  stnlu bir tablo getirebilirsiniz gznzn nne. Bu dizi-deęiřkeni řu komutla oluřturabiliriz:

Dim Ogrenciler(3,3)

Byle bir deęiřkende szgelimi birinci sıra (numarası 1,x olanlar) alıřkanları, ikinci sıradakiler (2,x’ler) daha az alıřkanları belirtebilir. VBScript, , drt ve hatta beř boyutlu dizi deęiřken oluřturur. Bunu nerede kullanacaęınızı siz kararlařtırabilirsiniz.

Bir dizi değişkenin herhangi bir elemanın değerini, programın herhangi bir aşamasında değiştirebilirsiniz:

Ogrenciler(3,2) = “Caner”

komutu, Hande’nin adını siler ve yerine Caner’in adını yazar.

Dizi değişkenlerimizin eleman sayısını bilmek isteyebiliriz. Kimi zaman dizi değişkenlerimizin eleman sayısı biz belirlemeyiz. Bu bilgi bir formdan gelebilir; bir veritabanından alınabilir. Ancak, mesela, bir döngü için bu değişkenin kaç elemanı olduğunu bilmek gerekir. Elimizde 35 elemanı olan Ogrenciler dizi-değişkeni varsa, bu sayıyı

ElemanSayisi = UBound(Ogrenciler)

komutu ile ElemanSayisi değişkenine yazdırırız. ElemanSayisi’nin değeri bu durumda 35 olacaktır.

Konuyla ilgili örnekler:

dinamikdizi_.htm >> dinamikdizi_.asp

Test Fonksiyonları

VBScript’te kullandığımız bazı değişkenlerin o andaki durumu, programımızın akışını kontrolde kullanacağımız bilgiyi sağlayabilir. Sözgelimi bir değişkenin değeri boş ise, ziyaretçimizin formu tam olarak doldurmadığını düşünebiliriz. VBScript, bize değişkenlerin durumunu sinamamız için bazı özel fonksiyonlar sağlar. Bu özel fonksiyonlardan dönen değer True (doğru) veya False (yanlış) olur; doğru sonucun değeri –1, yanlış sonucun değeri ise 0’dır:

IsArray	Bir değişkenin dizi değişken (Array) olup olmadığını sinar.
IsDate	Bir değişkenin değerinin tarihe (Date) çevrilip çevrilemeyeceğini sinar.
IsEmpty	Bir değişkenin tanımlanıp değer atanmış olup olmadığını sinar.
IsNull	Bir değişkenin geçerli bir değer tutup tutmadığını sinar.
IsNumeric	Bir değişkenin sayı olarak işleme tâbi tutup tutulamayacağını sinar.
IsObject	Bir ifadenin geçerli bir ActiveX veya OLE nesnesine referansta bulunup bulunmadığını sinar.
TypeName	Bir değişkenin türünü belirtir.
VarType	Bir değişkenin türünü belirten sayıyı verir.

Fonksiyonlarla ilgili örnekler:

<p>ÖRNEK 1:</p> <pre><% Dim a Dim d a = "ankara anafartalar caddesi" d = Len(a) For p = 1 To d If Mid(a, p, 1) = "a" Then Response.write (p) End If Next p >%</pre>	<p>1 4 6 8 10 12 15 17</p> <p>Kod çalıştırıldığında “ankara anafartalar caddesi” cümlesindeki a harflerinin kaçınıcı sırada olduğu listelenecektir. Bu örnekte MID ve LEN fonksiyonlarını, değişken isimlendirmeyi, FOR- NEXT döngüsünü, IF- THEN şart ifadelerini, Response.write komutunu görüyorsunuz.</p>
<p>ÖRNEK 2:</p> <pre><% a = " Karabük " b = "Safranbolu" Response.write (b & a & b & "
") Response.write (b & Trim(a) & b & "
") Response.write (b & LTrim(a) & b & "
") Response.write (b & RTrim(a) & b & "
") >%</pre>	<p>Karabük Safranbolu KarabükSafranbolu Karabük Safranbolu Karabük Safranbolu</p> <p>TRIM,LTRIM ve RTRIM fonksiyonlarının kullanımını görüyorsunuz.Karabük kelimesi önce sağındaki ve solundaki boşluklar kırılmadan yazılıyor. Ardından bir alt satırda iki yanındaki boşluklar kırılıyor ve Safranbolu bitişiyor. Ardından sol ve sağından kırılarak yazılıyor.</p>

<p>ÖRNEK 3: http://www.ismailkaras.com/asp/int_cint_fix.asp</p> <pre><% Response.write ("eksili değerler" & ""
"") Response.write ("Int " & Int(-3.65) & ""
"") Response.write ("Int " & Int(-2.35) & ""
"") Response.write ("clnt " & CInt(-3.65) & ""
"") Response.write ("clnt " & CInt(-2.35) & ""
"") Response.write ("fix " & Fix(-3.65) & ""
"") Response.write ("fix " & Fix(-2.35) & ""
"") Response.write ("artılı değerler" & ""
"") Response.write ("Int " & Int(3.65) & ""
"") Response.write ("Int " & Int(2.35) & ""
"") Response.write ("clnt " & CInt(3.65) & ""
"") Response.write ("clnt " & CInt(2.35) & ""
"") Response.write ("fix " & Fix(3.65) & ""
"") Response.write ("fix " & Fix(2.35) & ""
"") %></pre>	<p>Eksilideğerler Int -4 Int -3 clnt -4 clnt -2 fix -3 fix -2</p> <p>artılı değerler Int 3 Int 2 clnt 4 clnt 2 fix 3 fix 2</p> <p>INT, CINT ve FIX fonksiyonlarının negatif, pozitif yada yarının üstünde veya altında oluşuna göre değişimi.</p>
<p>ÖRNEK 4:</p> <pre><% a = "FEN EDEBİYAT FAKÜLTESİ" Response.write (UCase(Left(a, 1))); LCase(Right(a, Len(a) - 1))) %></pre>	<p>Fen edebiyat fakültesi</p> <p>Bu örnekte ise UCASE, LCASE, RIGHT, LEFT ve LEN fonksiyonlarının kullanımları görünmektedir.</p>

ÖRNEK:

Fonksiyonlarla ilgili diğer örnek dosyalar. Bu dosyaları deneyip, kodları inceleyiniz:

Garfield.asp, Tesadüfi.asp, Dogumgunu.asp, yazi-tura.asp, ornek_string_fonk.asp, ornek_chr_fonk.asp, ornek_split_fonk.asp, ornek_split_fonk_.asp, ornek_tarih_saat_fonk.asp, ornek_test_fonks.asp, hosgeldiniz01.asp, hosgeldiniz02.asp, dogumgunu01.asp, dogumgunu02.asp, Trk_krkt_bul_buyut.asp, split_ubound.asp, int_cint_fix.asp

ÖRNEK (saat.asp):

ABD'deki sunucu saatini Türkiye Saatine (TR 10 saat önde) ve AM/PM sistemini 24'lü sisteme çevirme işlemi. Örneği inceleyiniz her türlü alternatifi düşününüz.

```

Amerikada şu an saat: <%=time()%>
<%
'Su anki USA saatini yerel saate cevirme:
sat = split(time(),".",-1,1)
if right(time(),2)="PM" and sat(0)<>12 then
    sat(0) = sat(0) + 12
end if
sat(0) = sat(0) + 10
if sat(0)>=24 then
    sat(0)=sat(0)-24
end if
saat_tr = sat(0) & ":" & sat(1) & ":" & left(sat(2),2)
%>
<br>
Türkiyede Şu anki saat: <%=saat_tr%>

```

PROSEDÜR VE KULLANICI TANIMLI FONKSİYONLAR (Procedure and User Defined Functions)

Prosedürler (sub) ve kullanıcı tanımlı fonksiyonlar (function) birbirine benzerler. İkisi de birer alt program gibi çalışırlar. Kod yazarken, programın başka bir yere gidip oradaki işlemleri tamamlayıp geri dönmesini istersek prosedürü kullanırız. Kullanıcı tanımlı fonksiyonların prosedürden farkı; alt programa giderken bir yada birden fazla veri ile gönderilmesi, o verinin işlenip geri dönmesidir. Bu yolla VBScript'in hazır fonksiyonlarına benzer yeni fonksiyonlar üretmeniz mümkündür.

ÖRNEK (procedure.asp):

```

<% dim f,t, c, p, r
Hesap
f = t + c + p + r
response.write f

Sub Hesap
t = 3
c = (Sin(t) + 5) * Log(t)
p = Tan(c)
r = Cos(p)
end sub %>

```

(Prosedürlerle ilgili detaylı örnek ilerleyen sayfalardaki `giris_yap_.asp` isimli örnek içinde incelenebilir.)

ÖRNEK (function.asp):

```

<% ' Bu örnekte VBScript'in hazır fonksiyonu olan Sqr() fonksiyonunu kullanmak
yerine kendi karekök fonksiyonumuzu üretiyoruz:

response.write karekok(49)

```

```
Function Karekok(x)
Karekok = x ^ (1 / 2)
end function %>
```

ÖRNEK:

Kullanıcı Tanımlı Fonksiyonla Mail Kontrolü:

Ziyaretçilerimizin bilgilerini girerken verdikleri mail adreslerinin gerçek mail adresi olup olmadığını belki kontrol edemeyiz fakat en azından verdikleri mail adresinin mantıken doğru olup olmadığını kontrol edebiliriz. Önce olayın mantığını bizim kurmamız gerekiyor tabii ki.! Gerçek bir mail adresinde sadece bir tane "@" karakteri ve en az bir tane "." (nokta) olur, yasaklı karakterler olmaz ve en önemlisi bir mail adresi minimum "6" karakter olabilir. İşte Fonksiyonumuz ...

```
<%
Function MailKontrol(StrMail)
'fonksiyonumuza gelen değer StrMail

MailKontrol = False
'Functionumuza başlangıçta "False" verelim

Yasak = array("/", "\", "(", ")", "[", "]", "{", "}", "*", "?")
'Yasak karakterleri belirledik ilerde kullanacağız
'Sizde eklemeler yapabilirsiniz isterseniz

if len(StrMail) < 6 then
MailKontrol = False
'toplam karakter sayısı 6 dan az baştan kaybetti ...
exit function 'devam etmeye gerek yok; çıkalım
end if

At = 0 '@
Nokta = 0

For i = 1 to len(StrMail)
'karakter sayısı kadar dönüyoruz
karakter = mid(strMail,i,1)
'birer birer karakterleri alıyoruz

if karakter = "@" then
'geçerli karakter "@" karakterine eşitse
At = At + 1
'sayısını bir arttırıyoruz
end if

if karakter = "." then
'belki de nokta dır
Nokta = Nokta + 1
end if

'döngü içinde döngü kuruyoruz
'geçerli karakteri yasaklı karakterlerle karşılaştırıyoruz !
```

```

for j = Lbound(Yasak) to Ubound(Yasak)
'array değişkenin index inden bitişine kadar dönüyoruz
if karakter = yasak(j) then
'elimizdeki karakteri döngü içinde yasaklı karakterlerle karşılaştırıyoruz
MailKontrol = False
'Functionumuzun değerini False yaptık
exit function
'Devam etmemize gerek yok; çıkıyoruz
end if
next
'iç döngüyü bitiriyoruz

next
'bu next dış döngü için karıştırmayalım

'işte kontroller başlıyor

if At = 1 and Nokta>=1 then
'@ karakteri sadece 1 tane ve nokta sayısı en az 1 ise
MailKontrol = True
'bize verilen bilgi doğru
else
'değilse yanlış demektir
MailKontrol = False
end if

end function
%>

<%
gelen_adres = request.form("gelen_adres")
'formdan gelen değişkeni alıyoruz
if MailKontrol(gelen_adres) then
'fonksiyonumuza yolluyoruz
'o da gereken cevabı veriyor karşı tarafa
response.write "Teşekkürler Kaydınız Yapıldı"
else
response.write "Sanırım Mail Adresinizde bir sorun var"
end if
%>

```

(Benzer bir başka örnek için bakınız: email.asp)

FORMLAR (Forms)

(Bu bölümü HTML Sunum-6 dökümanındaki Formlar kısmı ile birlikte okuyunuz.)

Web sayfalarında hepimiz veri girişi için yazı kutusu (textbox), düğme (buton) gibi nesneleri kullanıyoruz. Örneğin bir siteye login olurken iki farklı textbox'a kullanıcı adımızı ve şifremizi girip gönder butonuna tıklıyoruz. Textbox, buton, listbox, checkbox gibi nesneler formların birer parçasıdır. Form nesneleri içeren bir web sayfasının her hangi bir script kodu içermesi zorunlu değildir. Form sayfaları salt HTML ile tasarlanabilir. Fakat formun gönderdiği bilgiler mutlaka sunucu taraflı bir program tarafından değerlendirilmeli,

yorumlanmalıdır. Dolayısı ile her ne kadar formlar konusu HTML'nin bir parçası gibi gözüксе de, biz konuyu burada, ASP başlığı altında işleyeceğiz.

Bütün HTML etiketleri gibi formlar da bir etiketle başlar ve biter. Başlangıç etiketi <FORM>, bitiş etiketi ise </FORM> şeklindedir. Form ile ilgili tüm işlemler ve tüm nesneler bu iki etiket arasına yazılır.

Formun Bölümleri

HTML formunun üç bölümü vardır. Bunlar, Web tasarımcısının formdan beklediği eylemin (Action) ne olduğunu gösteren ve ziyaretçinin tarayıcısına hitabeden bölümü; ziyaretçinin doldurması gereken boşluklar veya tercih etmesi gereken seçenekler; ve ziyaretçiye bu formun eylem komutunu harekete geçirme veya vaz geçme imkanı veren komut düğmeleri.

Action ve Method

Web alanınızda bir form oluşturmak için kullanacağınız <FORM> etiketi, kullanıcının tarayıcı programına bu formdaki bilgileri ne yapması gerektiğine ilişkin talimatı da içerir. Bunun için FORM etiketinin içinde, tarayıcıya ACTION yüklemiyse bu formun doldurularak gönderilmesi halinde içindeki bilgilerin nerede, hangi adreste, hangi programa teslim edileceğini söylersiniz. HTTP protokolü Web Server ile ziyaretçinin bilgisayarları arasında iki tür iletişime imkan verdiği için bu bölümde tarayıcıya hangi yöntemi seçmesi gerektiğini de METHOD yüklemiyse bildirmeniz gerekir.

Dolayısıyla Form etiketinin yazılış kuralı şöyle olacaktır:

```
<FORM ACTION="url" METHOD=POST>
```

Burada url harfleri yerine bu form ile gelecek bilgiyi işleyecek programın adresi bulunacaktır. Örneğin: "gonder.asp"

Doldurulacak Boşluklar ve İşaretlemeler

<FORM>...</FORM> etiketinin arasını ya kullanıcının dolduracağı boşluklar, ya da tercih yapmasına imkan veren listeler ve düğmelerle doldurmanız gerek. Bunu sağlayan başlıca kontrol elemanlarınız INPUT , SELECT ve TEXTAREA etiketidir. Şimdi bunları sırasıyla inceleyelim:

INPUT METODU

INPUT etiketi ile ziyaretçiye, forma klavyesinden veya fare ile işaretlemek suretiyle bilgi girmesi imkanı veririz. Bu etiketi kullanmanın genel biçimi şöyledir:

```
<INPUT TYPE="..." NAME="..." VALUE="..." SIZE="..." MAXLENGTH="..."  
SCR="..." CHECKED"...">
```

Şimdi bu etiketin kullanım ilkelerini kullanıcının yapabileceği işlere göre ayırarak inceleyelim:

TEXTBOX

- **Kullanıcının klavyesi ile bir metin girmesi için:**

TYPE=TEXT NAME="..." VALUE="..." SIZE="..." MAXLENGTH="..."

"Size" hanesi bu kutunun kullanıcının ekranında gösterileceği genişliği karakter olarak belirler; "Maxlength" hanesi ise kullanıcının girebileceği metnin uzunluğunu karakter olarak belirler. Bu haneyi koymaz ve bir değer vermezseniz, tarayıcı azami metin uzunluğunu 21 karakter olarak varsayar. Bu kutu ekranda gösterildiğinde içinde bir yazı olsun istiyorsanız, bunu "Value=..." hanesine tırnak içinde yazın. Form sunucuya ulaşır bir ASP sayfası tarafından yorumlandığında tanınması için "Name=..." hanesinin içine yazmak suretiyle isim vermeniz gerekir.

PASSWORD TEXTBOX

- **Kullanıcının parola girmesi için:**

TYPE=PASSWORD NAME="..." VALUE="..." SIZE="..." MAXLENGTH="..."

Metin girme kutusu ile aynı özelliklere sahiptir; fakat bu kutunun içine kullanıcının gireceği bilgiler ekranda gösterilmez yerine yıldız simgesi gösterilir.

HIDDEN

- **Sayfadan gizli bir veri göndermek için:**

type=hidden name=login value=true

Ekrandan veri girişi yapılmaz, geri planda gizli olarak gönderme yapılır. Örneğin kullanıcı ilgili kutulara ismini vb. girer butona bastığında girdiği bilgilerle beraber sistem saati de gönderilebilir. Bu durumda saat hidden olarak gönderilir. Örn;

<input type=hidden name="saat" value=<%=time()%> >

CHECKBOX

- **Kullanıcının bir kutuya işaret koyması:**

TYPE=CHECKBOX NAME="..." VALUE="..." [CHECKED]

Ziyaretçi, bu komutla oluşturacağınız işaret kutusunun içine fare ile tıklamak veya klavyede aralık tuşuna basmak suretiyle bir çarpı işareti girer veya otomatik olarak konan işareti kaldırabilir. Bu kutuda işaret varsa, tarayıcı "Value=" hanesine yazacağınız bilgiler ve kutunun adını bir çift olarak

Sunucuya gönderir. Kutuda işaret yoksa kutunun adı ve değeri Server'a gönderilmez. Bu kutuyu oluştururken mutlaka Name hanesine tırnak içinde bir isim girmeniz gerekir; yoksa gelecek bilgi hiç bir işinize yaramayabilir. Kutunun otomatik olarak işaretlenmesini istiyorsanız, CHECKED kelimesine yer verin; istemiyorsanız, bu kelimeyi yazmayın. Bu kutuyu oluşturmadan önce veya sonra bu kutunun ne işe yaradığını yazın.

OPTIONBOX

- **Kullanıcının yuvarlak bir boşluğun içine siyah bir nokta koyması**

(Radyo düğmesi):

TYPE=RADIO NAME="..." VALUE="..." [CHECKED]

İşaretlenecek yerin kare kutu değil de bir daire olması dışında bu unsurun bütün özellikleri ve ilkeleri CHECKBOX gibidir.

INPUT etiketi ile forma grafik veya gizli metin koymak da mümkündür. HTML ile gelen ekranda kullanıcının fare simgesi ile tıklayabileceği düğme oluşturan BUTTON etiketi yerine, örneğin Gir ve Sil gibi kullanıcının bilgileri Server'a göndermesini veya doldurduğu bilgileri tümüyle silmesini sağlayan işlemler de bu etiketle yapılabilir.

SELECT (LISTBOX)

Bu etiketi kullanarak, formda bir kutu ve yanında bir aşağı ok oluşturabilirsiniz; kullanıcı aşağı oku tıklamak suretiyle açacağı listeden bir unsuru seçerek, kutunun içine yazılmasını sağlayabilir. Arzu ederseniz, bu unsurlardan birisi otomatik olarak seçilmiş olarak da gösterilebilir. Bu etiketin kullanım şekli şöyledir:

<SELECT NAME="...." SIZE="..." [MULTIPLE]>.....</SELECT>

Size hanesine 1, 2, veya 3 vs.. yazarak, ekrandaki kutunun kaç seçenek göstereceğini belirleyebilirsiniz. Bu hane konulmazsa, otomatik 1 seçenek varsayılır. Bu kutuda gösterilecek seçenekler, <SELECT....>..</SELECT> etiketlerinin arasına <OPTION> etiketiyle yazılır. (<OPTION> etiketi kapatılmaz.) Herhangi bir seçeneğin otomatik olarak seçilmesi için önündeki <OPTION> etiketinin içine SELECTED kelimesi konulur. Örnek:
<select name="Temas" size="1"><option selected>Lütfen bir tercih yapınız<option value="Telefon">Telefon<option value="EPosta">EPosta<option value="Gel">Şahsi Görüşme</select></p></div>

TEXTAREA

Textarea, sunucuya inputbox'a göre çok daha uzun metin gönderme imkanı sağlar. bu etiketin kullanım şekli şöyledir:

<TEXTAREA NAME="..." rows=.. cols=..>Kutunun içine otomatik yazılması istenen metin buraya yazılır </TEXTAREA>

"Name=..." yüklemi ile metin kutusuna Server'a gelecek metnin işlenmesi ve kullanılması için gerekli ad verilebilir. "rows=" ve "cols=" yüklemelerinin karşısına verilecek rakamlarla bu kutunun formda kaç satır yüksekliğinde ve kaç harf genişliğinde bir yer alacağı belirtilir. Bu iki ölçünün kutuya girilecek metnin uzunluğu ile ilgisi yoktur.

Gönder ve Sil Butonları

Formun mantıksal olarak sonuna, kullanıcının dolduğu bilgileri ve yaptığı tercihleri formu sunan Internet alanına göndermesini sağlayan bir Gönder butonu konması gerekir. Ve tercihe bağlı olarak forma yazdığı bilgileri ve yaptığı tercihleri değiştirmek isteyenlerin tümüyle silebilmesi için bir Sil düğmesi konabilir.

Bunun için INPUT etiketi şöyle kullanılır:

```
<INPUT TYPE=SUBMIT NAME=Gonder VALUE="Gönder">
<INPUT TYPE=RESET NAME=Sil VALUE="Sil">
```

ÖRNEK:

OrnekForm.htm dosyasını (Babyweb ile) çalıştırarak FormOrnek.asp dosyasındaki sonuçları inceleyiniz.

REQUEST

Yukarıda gibi bir form ile gönderilen veriler Request.form metodu ile elde edilir. Örn:

Gönderme: <input type="text" name="kullanici" size="20">

Alma: <%Request.form("kullanici")%>

Not: Request.form sadece Request şeklinde de kullanılabilir. Örn: Request("kullanici")

Eğer bilgiler form ile değil bir link ile gönderiliyorsa Request.querystring metodu ile elde edilir. Örn:

Gönderme: İlk Sayfa

Alma: <%Request.QueryString("Sayfa")%>

ÖRNEK (Request.Form):

gonderdim.htm > aldim.asp isimli dosyaları deneyip, inceleyiniz.

ÖRNEK (Request.QueryString):

1. gonderdim_qs.htm > aldim_qs.asp
2. linkleverigonder.htm > FormOrnek.asp
3. QString klasörü içindeki cikansec.asp
isimli dosyaları deneyip, inceleyiniz (STOK.mdb ile aynı klasörde olmalı).

Bu dosyada link ile statik ve dinamik veri gönderme, sayfalama, sayfalar arasında ilerleme gibi konuları görebilirsiniz. Sütun başlıkları statik olarak, sayfa gezinme linkleri ise dinamik olarak veri göndermektedir. Bir sayfada gösterilecek kayıt sayısına bağlı olarak sayfa sayısının belirlenmesi hesabına özellikle dikkat ediniz. Kayıt sayısının tam değer oluşunu göz önüne alınız (Örn; 3000). ks kayıt sayısı ise, sayfa sayısı:

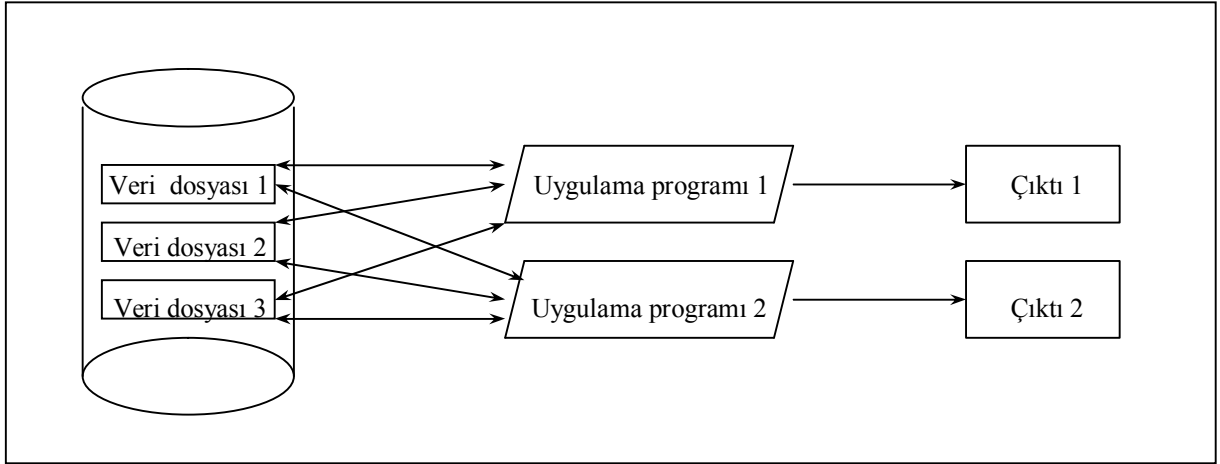
$$\text{sonsayfa} = (\text{int}((\text{ks}-1)/500)) + 1$$

VERİ SAKLAMA YÖNTEMLERİ (Data Storing Methods)

Verilerin bilgisayar ortamında saklanması, günümüzde de kimi zaman kullanılmakta olan dosyalama yönteminin geçmişi, bilgisayarlar eskidir. Bugün, çok daha üstün şartlarda bu işlemi yerine getiren “veri tabanı” adı verilen sistemler geliştirilmiş olmasına rağmen, basit yapısı ve doğrudan ulaşılması gibi sebepler yüzünden, bazı küçük uygulamalarda dosyalama yöntemi hala tercih edilebilmektedir. Bu iki yöntemin farklılıkları ve avantajları nelerdir?

1.1. Klasik Yöntem: Dosyalama (File Operations)

Dosyalama işleminde veriler bir yada birden fazla dosyalar halinde, direk olarak kayıt ortamında saklanmakta, uygulama programları vasıtası ile üzerlerinde işlem (kayıt, sorgu, düzeltme, silme) yapılmaktadır. Verilerin dosyalama yöntemi ile organizasyonunda her bir uygulama programı veri dosyalarına doğrudan erişmektedir. Uygulama programları hazırlanırken, verinin kayıt şeklinden, yerine kadar kayıt ortamındaki her türlü faaliyetin düşünülmesi, kontrol altında tutulması ve ona göre tasarlanması gerekmektedir. Söz konusu programlar verilerin dosyalara nasıl depolanacağını bilmek zorundadırlar. Yani, uygulama programları veri dosyasına erişim için gerekli tüm komutları içermelidir. Bu durum çok sayıda tekrara sebep olmaktadır. Veri dosyalarında herhangi bir değişiklik yapıldığında, erişimi sağlayan komutlar da, her bir uygulama programında ayrı ayrı düzeltilmelidir.[1] Veri dosyaları, bir ağ ortamında, farklı kullanıcılar tarafından, farklı uygulama programlarınca paylaşıldığında bir başka büyük problem ortaya çıkar; güvenlik. Verilerin denetimi ve emniyeti tam olarak sağlanamamakta yada çok uğraşmayı gerektirmektedir. Hangi kullanıcının veriye erişimi mümkün, hangisi değiştirmeye yetkili... bunların çok iyi düşünülüp planlanması ve uygulama programlarında tek tek belirtilmesi gerekir.



Şekil 1.1. Dosyalama yöntemine göre veri ve uygulama programı ilişkisi.

ÖRNEK:

Dosyalama yöntemi için şu örnekleri inceleyiniz:

konuk01.htm, konuk02.htm, konuk_isle.asp : text dosyasına veri girme

konuk_oku.asp : text dosyasındaki veriyi okuma

1.2. Veri Tabanı Kavramı (Database)

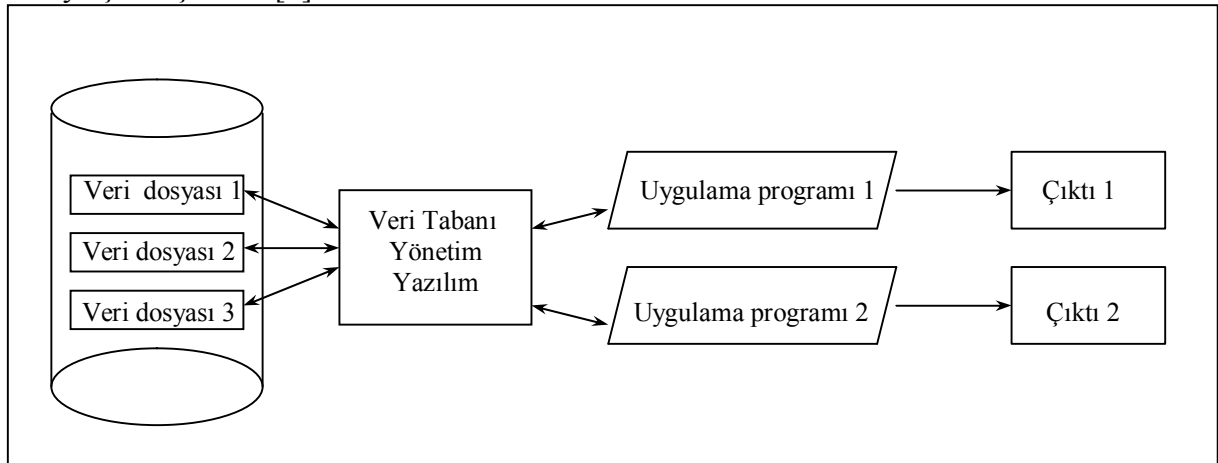
“Veri tabanı birbirinden bağımsız bir çok uygulamada ortaklaşa kullanmak amacıyla verilerin, gereksiz yinelenmelerden arınmış, doğruluğu, tutarlılığı, gizliliği ve güvenliği sağlanmış olarak özel tekniklerle depolanmasını, güncellenmesini ve erişilmesini, genellikle

kullanıcının kolayca öğrenebileceği özel diller aracılığıyla sağlayan bir yazılım sistemidir. (Fischer,1993)” [2]

Veri tabanı kavramı, bilgi işlem dünyasında uzun tecrübe ve aşamalardan sonra ulaşılmış bir kavramdır ve klasik dosya yönetimine bir alternatif olarak, geniş kapasiteli, hızlı, büyük veri yığınlarını taşıyıp saklayabilen donanımlar ile bunlara uygun, kapsamlı, ağ ortamının isteklerine cevap veren, yazılımların geliştirilmesinin sonucu ortaya çıkmıştır. Klasik bir dosyalama sisteminde en önemli özellik uygulamaya bağımlı olmaktır; yani bir dosya hangi yazılım tarafından oluşturulmuşsa o yazılıma bağımlı olarak dosyaya erişilebilir; oysa veri tabanı yönetiminde prensip olarak veri veri-uygulama bağımsızlığı vardır; yani bir kez oluşturulmuş verilere teorik olarak her tür programlama dili yada uygulama programı ile erişme imkanı vardır.[*] [3]

Veri tabanı sistemi veri tabanı ve bunu yöneten özel bir yazılımdan oluşur. Bu özel yazılım veri tabanı yönetim yazılımı/sistemi (VTYS) adını alır. Access, Dbase, Oracle, Paradox gibi yazılımlar bu tür yazılımlardır. Veri tabanı birbiri ile ilişkili veriler topluluğudur ve sadece verileri değil, onlar arasındaki ilişkileri de saklar. Günümüzde kullanılan ilişkisel modele dayalı modern veri tabanlarının yapıları birbirlerine benzerler.[3] Veriler satır (kayıt) ve sütunlardan (alan) oluşan tablolara kaydedilmekte ve ortak yada birbiri ile ilgili verileri içeren farklı tablolar ilişkilendirilebilmektedir. “Veri tabanı yönetim yazılımı, kayıt, silme, düzeltme, sorgulama, indeksleme, çok kullanıcı okuma, güncelleştirme, paylaşma vb. gibi işlemleri gerçekleştirir, organize eder ve veri erişim yollarını, yetkileri ve veri bütünlüğünü denetler. Bunların dışında temel programla ilişkili, kullanıcı arayüzleri, formlar, menüler, raporlar, sorgular, makrolar vb. veri tabanı yönetim yazılımının sağladığı hizmetlerdendir. Sağladıkları bu hizmetlerle beraber günümüzün veri tabanı yönetim yazılımları bir çok açıdan uygulama programlarının görevlerini yerine getirebilmekte, makrolar sayesinde özel amaçlı yazılımlar ilave edilebilmektedir. (Healey,1993)” [2]

Veri tabanı yönetim yazılımının önemli faydalarından biri de veri bağımsızlığı sağlamasıdır. Uygulama programları verilerin nereye, nasıl kaydedildiği ile meşgul olmamakta, sadece ilgili isteği iletmekte, veri tabanı yazılımı, istenen işlemi geri planda, üstelik dosyalama işlemlerinden kat be kat hızlı bir şekilde gerçekleştirmektedir. Bu yazılım, verilerle uygulama programları arasında hem vasıta, hem de denetleyici görevini üstlenmektedir. Veri tabanında yada uygulama programlarında meydana gelen herhangi bir değişiklik bir diğerini etkilememekte, bununla birlikte, veri tabanı yönetim yazılımı verilerin doğru olarak sağlanacağını garanti etmektedir. Böylece, uygulama programları ve veri tabanını korumak için ayrı bir emeğe gerek kalmamaktadır. Veri tabanı sistemi tarafından sağlanan hizmetler, aynı zamanda yeni uygulama programlarının gelişimini de kolaylaştırmışlardır. [1]



Şekil 1.2. Veri dosyaları, veri tabanı yönetim sistemi ve uygulamalar arasındaki ilişki.

1.2.1. Veri Tabanının Avantajları

Veri tabanının, dosyalama yöntemine göre avantajları şu şekilde özetlenebilir;

1. Bir yada daha fazla kullanıcıya hizmet veren veri tabanı yönetim yazılımı, tek merkezden, veri standartlarının belirlenmesini ve istenen standartta sunulmasını, güvenlik şartlarının yerine getirilmesini, uyumsuzlukların giderilmesini ve veri tabanının bütünlüğünü sağlar. Bu merkezi kontrol sayesinde kullanıcı yetkileri belirlenir ve sürekli olarak denetlenir.
2. Veri tabanı yönetim yazılımı sayesinde, tek bir veri tabanı, aynı anda farklı kullanıcılar tarafından, farklı uygulamalarda kullanılabilir, güvenli ve hızlı bir şekilde paylaşılabilir
3. Uygulama programları, verilerin depolandığı fiziksel ortamdan bağımsız çalışırlar. Veriye erişim veri tabanı yönetim yazılımı aracılığıyla gerçekleştirildiğinden uygulama programlarının veri yapısını bilmesine gerek yoktur.
4. Yeni uygulama programları ve veri tabanı uygulamaları, veri tabanı yönetim yazılımı tarafından sağlanan hizmetler sayesinde kolayca sisteme entegre olabilirler.
5. Dosyalama işleminde, her bir uygulama için farklı veri dosyası yada dosyaları kullanılır ve bu da önemli ölçüde veri fazlalığına sebep olur. Bazı verilerin birden fazla kopyasını saklamak için geçerli sebepler olabilir. Ancak, gereğinden fazla veri tekrarı, emek, zaman ve maliyet kaybıdır. Etkin bir veri tabanı yönetim yazılımı verilerin düzgün bir şekilde depolanmasının yanı sıra, kopyalarının da güncellenmesini sağlar.
6. Sağladığı, menüler, sorgular, raporlar ve arayüzler sayesinde veri tabanı yönetim yazılımının kullanımı kolaydır. Verilerde meydana gelen silinme yada istenmeyen değişiklikler karşısında yedekleme ve işlemlerin izlenmesi gibi fonksiyonlarla geri dönüş yapılabilir.

1.2.2. Veri Tabanının Dezavantajları

Veri tabanı sistemlerinin, faydalarının yanında bazı dezavantajları da olabilir. Bunlar şöyle sıralanabilir:

1. Veri tabanı sistemlerinin yazılım ve donanım, maliyetleri yüksek olabilmektedir. Fakat uygun ve verimli uygulama programları sayesinde uzun vadede elde edilen faydalar bunu fazlasıyla karşılayacaktır.
2. Veri tabanı sistemleri, dosyalamaya nazaran daha karışıktır. Teoride, karmaşık sistemler, veriler üzerinde yapılan işlemlerden olumsuz etkilenmektedirler.
3. Uygulama programı çalışırken veri transferleri esnasında teorik olarak, büyük ölçüde, verinin kaybolması veya bozulması riski vardır. Ancak veri tabanı yönetim yazılımı tarafından yedekleme ve düzeltme prosedürleri, çoğunlukla sağlanmakta ve bu risk en aza inmektedir.

VBScript ile Veritabanına veri girişi

“Veritabanim.mdb” isimli veritabanına ait “BenimTablom1” isimli tablodaki, “IsimSoyad”, “Yasi”, ve “KayitTarihi” alanlarına yeni kayıt girişi için örnek kod:

```
'VT baglantisinin yapılması:
Set Baglantim = CreateObject("ADODB.Connection")
'VT'nin açılması:
Baglantim.Open ("DRIVER={Microsoft Access Driver (*.mdb)};DBQ=" &
Server.MapPath("Veritabanim.mdb"))
'Tablo nesnesinin oluşturulması:
Set Tablom = server. CreateObject("ADODB.Recordset")
'Tablonun açılması:
Tablom.Open "BenimTablom1", Baglantim, 1, 3

'Tabloya veri eklemeye başlangıç:
Tablom.AddNew
'Tablodaki alanlara veri aktarma
Tablom("IsimSoyad") = request("AdiSoyadi")
Tablom("Yasi") = request("Yas")
Tablom("KayitTarihi") = request("KayitTr")
'aktarma işlemi birince tablonun güncellenmesi:
Tablom.Update

'tablonun kapatılması:
Tablom.close
set Tablom= Nothing
'baglantinin kesilmesi:
Baglantim.close
set Baglantim= Nothing
```

ÖRNEK (Örnekler VTUyg klasörü içinde):

VeriGirisi.asp ve VeriGirisiOK.asp isimli dosyaları deneyip, inceleyiniz (Veritabanim.mdb ile aynı klasörde olmalı.)

VBScript ile Veritabanından Veri Okuma

“Veritabanim.mdb” isimli veritabanına ait, “BenimTablom1” isimli tablodadaki “IsimSoyad” alanında belirli bir kaydı aramak için örnek kod:

```
<%
Set oConn = Server.CreateObject("ADODB.Connection")
oConn.Open("DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("Veritabanim.mdb"))
ssql="select * from Benimtablom1 ORDER BY IsimSoyad;"
Set oRS = oConn.Execute(ssql)
```

Do While NOT oRS.EOF

```
if oRS("IsimSoyad") = Request.form("AdSoyad") then
%>
    <%=oRS("IsimSoyad")%> <br>
    <%=oRS("Yasi")%><br>
    <%=oRS("KayitTarihi")%><br>
<%
end if
    oRS.MoveNext
Loop

oConn.Close
Set oRS = Nothing
Set oConn = Nothing %>
```

ÖRNEK:

arama_TextBox.asp > BulGetir.asp,
arama_tablosuz.asp > BulGetirtablosuz.asp,
arama_Combolu.asp > BulGetir.asp
isimli dosyaları deneyip, inceleyiniz (Veritabanim.mdb ile birlikte)

VBScript ile Veritabanındaki Kayıtların Tamamını Listeleme

“Veritabanim.mdb” isimli veritabanına ait, “BenimTablom1” isimli tablodadaki tüm kayıtları listelemek için örnek kod:

```
<%
Set oConn = Server.CreateObject("ADODB.Connection")
oConn.Open("DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &
Server.MapPath("Veritabanim.mdb"))
ssql="select * from BenimTablom1; "
Set oRS = oConn.Execute(ssql)

Do While NOT oRS.EOF
%>
<%=oRS("IsimSoyad")%> <%=oRS("Yasi")%> <%=oRS("KayitTarihi")%>
<br>
<%
    oRS.MoveNext
Loop

oConn.Close
Set oRS = Nothing
Set oConn = Nothing
%>
```

ÖRNEK:

VeriListeleme_tablolu.asp

VeriListeleme_tablosuz.asp
isimli dosyaları deneyip, inceleyiniz (Veritabanim.mdb ile birlikte)

VBScript ile Veritabanındaki bir kaydı Güncelleme

“Veritabanim.mdb” isimli veritabanına ait, “BenimTablom1” isimli tablodadaki belirli bir kaydı düzeltmek için örnek kod:

```
<%  
dsn = "DBQ=" & Server.MapPath("veritabanim.mdb") & ";Driver={Microsoft Access  
Driver (*.mdb)};"  
Set conn=Server.CreateObject("ADODB.Connection")  
conn.Open dsn  
  
SQL = "Update BenimTablom1 Set Yasi = '" & Request.Form("Yas") & "', KayitTarihi  
= '" & Request.Form("KayitTr") & "' Where IsimSoyad = '" &  
request.form("AdiSoyadi") & "'"
```

```
Set RS = conn.Execute(SQL)  
conn.Close  
Set conn = Nothing  
%>
```

ÖRNEK:

Arama_Duzeltme.asp > VeriDuzeltme.asp > VeriDuzeltmeOK.asp
ListedenDuzeltme.asp > VeriDuzeltme.asp > VeriDuzeltmeOK.asp
isimli dosyaları deneyip, inceleyiniz (Veritabanim.mdb ile birlikte)

VBScript ile Veritabanındaki bir kaydı Silme

“Veritabanim.mdb” isimli veritabanına ait, “BenimTablom1” isimli tablodadaki belirli bir kaydı silmek için örnek kod:

```
<%  
Set oConn = Server.CreateObject("ADODB.Connection")  
oConn.Open("DRIVER={Microsoft Access Driver (*.mdb)}; DBQ=" &  
Server.MapPath("veritabanim.mdb"))  
  
set kayit_sil = Server.CreateObject("ADODB.RecordSet")  
  
SQL = "delete * from BenimTablom1 Where IsimSoyad = '" &  
request.form("AdiSoyadi") & "'"
```

```
kayit_sil.Open sql, oConn, 1, 2  
  
set kayit_sil=nothing  
oConn.CLOSE
```

```
SET oConn = NOTHING  
%>
```

ÖRNEK:

```
Arama_Silme.asp > VeriSilme.asp > VeriSilmeOK.asp  
ListedenSilme.asp > ListedenSilmeOK.asp
```

isimli dosyaları deneyip, inceleyiniz (Veritabanım.mdb ile birlikte)

APPLICATION (Uygulama) ve SESSION(Oturum) NESNESİ

ASP açısından, bir site "uygulama programı" (**Application**) sayılır. Her ziyaretçi de bir "oturum" (**Session**) sayılır. Bir takım ASP ve HTML sayfalarından oluşan bildiğimiz Site'ye **application**, her hangi bir ziyarete de **session** denmesinin sebebi nedir? Bunu her iki nesnenin işlevleri ile açıklayabiliriz.

Application nesnesi, sitenin tümüyle ilgili bilgileri (değişkenleri, nesneleri ve metodları) tutar; **Session** nesnesi ziyaretçinin sitemize girmesinden itibaren izini sürer.

Kimi zaman isteriz ki, bir değişkenin değeri bütün sayfalarda aynı olsun; ziyaretçinin sayfa değiştirmesi ile değişkenin değeri değişmesin. Bunu ASP'de yapmak çok kolaydır. ASP'de bu zorluğu yenebilmek için değişkenlerimizi **Session** nesnesi için oluşturabiliriz; ve bu değer ziyaretçinin oturumu boyunca devam eder; bütün ASP sayfalarındaki bütün Fonksiyonlar tarafından bilinebilir.

Örneğin:

```
Session ("Tupras") = 44500
```

bütün **Session** için geçerli bir Tupras değişkeni oluşturur ve ona "44500" değerini atar. Kimi zaman, değişkenin çok daha geniş kapsamlı olmasını, yani ömrünün **Session** ile değil bütün **Application** boyunca belirli olmasını isteyebiliriz. O zaman bu değişkeni **Application** düzeyinde tanımlayabiliriz:

```
Application ("Tupras") = 44500
```

Bu durumda Tupras değişkeni bütün ziyaretçiler için aynı değere sahip olacaktır.

Session nesnesinin oluşabilmesi için, ziyaretçiye mutlaka bir **Cookie** göndererek, sitemizde bir işaret vermemiz gerekir. Oysa, ziyaretçiye sitemize bağlandığı anda bir **Session** kimliği verirsek ve her yeni sayfa talebinde bu kimliği kontrol edersek, kimin hangi oturumunu sürdürdüğünü biliriz. ASP-uyumlu bir Web Server, ziyaretçi yeni bir tercih yapmadığı takdirde her **Session** nesnesini 20 dakika açık tutar; sonra siler. Bu süreyi **Session** nesnesinin **Timeout** özelliği yoluyla değiştirebilirsiniz. **Session** belirleyen **Cookie** ASP-uyumlu Web Server tarafından otomatik olarak gönderilir ve takip edilir; tasarımcı olarak bizim bu konuda bir şey yapmamız gerekmez.

Bir Web programınıza aynı anda kaç kişi ulaşırsa (yani sayfalarınızı kaç kişi talep ederse), o kadar **Session** nesnesi oluşur; fakat siteniz bir adet olduğuna göre bir adet **Application** nesnesi vardır. Bu nesnenin bütün **Session**'lar için sitemizin ihtiyaçlarına uygun ve aynı uygulama kurallarına sahip olmasını sağlayan bir dosya vardır: **Global.asa**. Bu dosya PWS veya IIS kurulurken oluşturulur (ana dizinde oluşturulmalıdır). Bu dosyada, çoğu zaman, sitemize ilk ziyaretçinin gelmesiyle oluşan **Application_OnStart** ve son ziyaretçinin çıkmasıyla oluşan **Application_OnEnd** ile herhangi bir ziyaretçinin bir sayfaya erişmesiyle oluşan **Session_OnStart** ve ziyaretçinin sitemizden çıkması ile oluşan **Session_OnEnd** olayları halinde ne yapılacağı yazılıdır. Bu dosyanın içeriği standart bir ASP dosyasına benzetmekle birlikte adındaki uzatmanın **.asp** değil de **.asa** olmasının sebebi, dosyanın **Active Server Application** dosyası olmasıdır. ASP-uyumlu bir Web Server programı sitemize ulaşan ilk ziyaretçiyi gördüğü anda **Global.asa** dosyasını çalıştırır.

Application ve **Session** nesnelerin kendi başlarına en çok kullanıldığı yer, sitemize gelen ziyaretçilerin sayısını (sitemizin aldığı Hit sayısını) tutmasını sağlamaktır. Bu genellikle **Global.asa** pogramına bir sayaç yerleştirilerek yapılır.

ÖRNEKLER (Session klasörü içinde):

LOGIN1 Örneği:

1. Veri tabanındaki kullanıcı isimleri ve şifrelerine bakarak oturum açma ve kontrol işlemleri için;

Giris.htm>login.asp > index.asp isimli dosyaları deneyip, kodlarını inceleyiniz (Hepsi de BenimVT.mdb ile aynı klasörde olmalı).

LOGIN2 Örneği:

1. Oturum açma ve kontrol işlemleri için;

giris_yap_.asp > menu.asp isimli dosyaları deneyip, kodlarını inceleyiniz.

(Username değişkeninin Session("uname") nesnesi ile menu.asp dosyasına nasıl taşındığına dikkat ediniz. Menu.asp içindeki linkler bu değişkenin dolu olup olmamasına bakılarak gösterilmektedir. Oturum sona erdiğinde artık bu değişken bir değer tutmayacağından linkler kullanıcıya gösterilmeyecek ve doğrudan oturum açma sayfasına yönlenecektir.

2. Veri tabanındaki kullanıcı isimleri ve şifrelerine bakarak oturum açma ve kontrol işlemleri için;

giris_yap_.asp > menu.asp isimli dosyaları deneyip, kodlarını inceleyiniz (Bolum.mdb ile aynı klasörde olmalı).

3. Oturum süresini belirleme işlemi için;

global_time.asa isimli dosyanın kodlarını inceleyiniz. Dosyada oturum süresi 1 dk. olarak ayarlanmıştır. Örneği çalışır halde görmek için;

http://www.ismailkaras.com/giris_yap_.asp adresine girerek oturum açtıktan sonra (kullanıcı adı: 466 şifre: bilgisayar), menü sayfasında bir dakikadan fazla bekleyiniz ve sayfayı yenileyiniz. Oturum süresi dolduğu için (uname değişkeni nothing olduğu için) menüler gösterilmeyecek ve doğrudan oturum açma sayfasına yönlenecektir. (Deneme yaparken global_time.asa isimli dosyanın ismini global.asa olarak değiştiriniz. Global.asa kök dizinde olmalıdır, alt dizinlerde değil)

4. Siteye kaç kişinin bağlı olduğunu görmek için;

global_sayac.asa ve ziyaretci.asp isimli dosyaları deneyip, kodlarını inceleyiniz (Deneme yaparken global_sayac.asa isimli dosyanın ismini global.asa olarak değiştiriniz. Global.asa kök dizinde olmalıdır, alt dizinlerde değil)

RESPONSE NESNESİ

Buffer;

ASP kullanım mantığı olarak response objesini gördüğü anda o anki işlemin sonucunu direk kullanıcıya gönderir. Sen eğer sayfanın daha sonraki aşamalarında yapılacak bir işlemin sonucunda sayfanın görüntüsüyle ilgi bir karar verecekseniz bu değeri true yaparak sayfanın bitimde ancak kullanıcıya göndermesini sağlayabilirsiniz. Değiştirilmedikçe buffer özelliğinin değeri false dir. Genel kullanımı ise

```
<%response.buffer=[true/False]%>
```

True degerini alırsa sayfayı hafızada saklar ve bitiminde gönderir.

False degerini alırsa response'yi direk kullanıcıya gönderir.

Expires;

Kullanıcı browserlarında bazı sayfalar sonraki kullanımlarında çabuk açılması için saklanır. Expires degeriyle bu sayfanın kullanıcı browser'ındaki cache'de nekadardır zaman (dakika cinsinden) saklanacağını belirleyebilirsiniz. Eğer deger sıfır yada eksi değeri girilirse sayfa cache'de hiç saklanmaz. Kullanımı;

```
<% Response.expires=[sayı] %>
```

Write;

Sayfaya yazılması gereken herşeyi response.write ile yazdırabilirsiniz. Bu metod ASP'nin en çok kullanılan metodudur.

```
<% response.write [yazılacak bilgi] %>
```

Örnek: <% response.write "Bilgisayar Mühendisliği" %>

Redirect;

Doğrudan bir sayfaya gitmeyi sağlar;

Örnek: <% response.redirect "menu.asp" %>

Sayfa otomatik olarak menu.asp sayfasına yönlenecektir.

INCLUDE FILE

Bu yöntem herhangi bir dosyayı/içeriği birden fazla web sayfasında otomatik olarak göstermek için kullanılır. Kullanımı:

```
<!-- #include file="altmenu.htm" -->
```

HATA (ERR) NESNESİ

Hangi dille olursa olsun program yazarken hata yapmak kaçınılmaz bir kuraldır. Dolayısıyla kullandığınız programlama dili hatalarınızı kolayca yakalamanıza imkan vermelidir. ASP programlarınızda yazım yanlışlığı, olmayan değişkene gönderme gibi Script hatası olmaması gerekir. Bu tür hatalar, program Web'e gönderilmeden mutlaka ayıklanmalıdır. Fakat programcı olarak öngöremeyeceğiniz, ve çoğu Web ziyaretçisinden veya ziyaretçinin bilgisayarından kaynaklanan hata durumları olabilir. VBScript, şu standart komutla beklenmedik hata durumlarında programın yoluna devam etmesini sağlayabilir:

```
<% On Error Resume Next %>
```

Bu komutla VBScript'e, hata halinde bir sonraki satırdan yoluna devam edecektir. Fakat oluşan hata, programın daha sonra vermesi beklenen sonucu vermesini önleyebilir. VBScript, Err (Hata) Nesnesi'nin bir çok özelliğinden özellikle hata sayısı (Number), tanımı (Description) ve kaynak (Source) özellikleri ile size hatanın ne olduğunu ve nereden kaynaklandığını söyleyebilir. Bu özellikleri kullanarak, programlarınızda, en azından geliştirme aşamasında, örneğin,

```
If Err.Number = xx Then
```

şeklinde bir ifade ile hatanın türüne göre programın kazasız yürümesini sağlayabilirsiniz. Burada xx yerine 108 ayrı hata numarası yapabilirsiniz. Hata numaraları, Microsoft'un VBScript sitesinden edinilebilir.