

# VISUAL BASIC

## DOSYA İŞLEMLERİ

Visual Basic programlama dili ile yapılan çözümlenmelerde, programa girilen verilerin ve/veya programın çalıştırılması sonucunda elde edilen bilgilerin bir kayıt saklama ortamına (harddisk, disket v.b.) kaydedilmesi, veya benzer şekilde kaydedilmiş bilgilerin program içinde veri olarak kullanılabilmesi için kayıt ortamlarından okunarak belleğe aktarılması gerekebilir.

Bu nedenle hazırlanan programın çalışması sırasında, interaktif olarak kayıt ortamları ile veri/bilgi alışverişi yapılabilmesi için, bu ortamlara yazma/kaydetme ve bu ortamlardan okuma işlemleri yapmak gerekir. Bu işlemlere özetle "dosya işlemleri" adı verilir.

Kayıt yapabilmek amacıyla oluşturulan dosyalar ;

1. **Sıralı erişimli dosya** (Sequential file)
2. **Rastgele erişimli dosya** (Random file)
3. **İkili dosya** (Binary file)

Olmak üzere üç farklı türde yapılandırılabilirler .

**Sıralı erişimli dosyalarda** mevcut kayıtlar, kayıt kütüğü içinde kayıt formatına göre sıra ile yer alırlar ve herhangi bir kayıta erişmek için, dosya başlangıcından itibaren aranan kayıta kadar olan tüm kayıtların mutlaka okunması (erişilerek geçilmesi) zorunludur. Bu özelliği nedeniyle bu tür dosyalar, sınırlı kullanım alanlarına sahiptirler. Genellikle az veri içeren küçük dosyaların işlenmesi amacıyla, veya kayıtlı dosyalarda zaman içinde çok fazla değişiklik gerektirmeyen ama bunun yanı sıra tüm bilgilere çoğunlukla sıra ile yığın halinde gereksinim duyulan liste yapılarında kullanılırlar. Eğer kayıtlı verilerde sık sık değişiklikler ve düzenlemeler gerekiyorsa sıralı erişimli dosyalar yerine rastgele erişimli dosyalar tercih edilir. Sıralı erişimli dosyalarda aynı anda okuma, yazma veya veri ekleme işlemleri birlikte yapılamaz. Her işlem türü için dosyanın ayrı modda açılması gerekir. Sıralı erişimli dosyalarda kayıtlar ASCII (standart metin kodları) kayıt özelliğinde tutulurlar.

**Rastgele erişimli dosyalar**, sıralı erişimli dosyalardan farklı bir iç yapıya sahiptirler. Butür dosyalarda, her bir kayıtın uzunluğu bellidir ve dosya oluşturulması/ işlenmesi sırasında belirtilir. Rastgele erişimli dosyalarda, gerektiğinde okuma, yazma/kaydetme, veri ekleme işlemleri aynı anda yapılabilir. Dosyanın farklı işlemlere yönelik açılması zorunluluğu yoktur. Bu tür dosyalarda, aranan herhangi bir kayıta doğrudan erişmek mümkündür. Bir veriye erişmek için tüm dosyanın baştan sona taranması gerekmez. Yalnızca aranan kayıtın numarasını belirtmek yeterlidir. Rastgele erişimli dosyalarda da kayıtlar ASCII (standart metin kodları) kayıt özelliğinde tutulurlar.

**İkili dosyalar** ise kayıtları binary modda ( **1** ve **0** ikili sayı kodları) tutarlar. Bu tür dosyalara kayıtlar hangi pozisyondan (kaçıncı byte'tan) itibaren yapılacağı belirtilerek, belirli uzunluklarda **BYTE** dizileri olarak kaydedilir ve okunurlar.

# VISUAL BASIC

## SIRALI ERİŞİMLİ DOSYA İŞLEMLERİ

Sıralı erişimli bir dosya üç ayrı modda kullanılabilir:

1. **OUTPUT** modu: Bir dosya bu modda açılmışsa, verilen dosya adı ile yeni bir kayıt dosyası oluşturulur ve oluşturulan bu dosyaya veri kaydı yapılır. Eğer, bu modda bir dosya açılırken, daha önceden aynı dosya adı ile açılmış bir dosya, kayıt ortamında mevcut ise, eski dosya silinerek yerine aynı isimde boş bir dosya oluşturulur ve yeni kayıtlar bu dosyaya yapılır.
2. **INPUT** modu: Bu modda açılmış bir dosya, kayıt ortamında daha önceden oluşturulmuş ve içinde kayıtlar içeren bir dosya olmalıdır. Bu mod ile söz konusu dosya okunarak içindeki bilgiler, kayıt ortamından program içine çalışma ortamına aktarılır.
3. **APPEND** modu: Bu modda açılmış bir dosya, yine kayıt ortamında daha önceden oluşturulmuş bir dosya olmalıdır. Program içinde yeni oluşturulan bilgiler, **append** modunda açılmış dosyaya, en son kayıttın arkasına eklenecek şekilde kaydedilirler.

Her üç modun kullanımında da **OPEN** komutu kullanılır. Bu komutun kullanımı:

<b>OPEN</b> "dosyaadı" <b>FOR</b>	<b>OUTPUT</b>	<b>AS</b> kanalNo	şeklindedir
	<b>INPUT</b>		
	<b>APPEND</b>		

Örneğin: **OPEN "verilerim" FOR OUTPUT AS #1**  
Veya  
**OPEN "TELEFON.DAT" FOR INPUT AS #2**  
Veya  
**OPEN "verilerim" FOR APPEND AS #1** şeklinde yazılabilirler.

Tırnak içinde verilen dosya adı ile yeni bir dosya oluşturulacaktır. Burada gerekirse dosya adı kayıt ortamındaki sürücü adı, klasör ve alt klasör adları ile birlikte de yazılabilir.

Örneğin; **OPEN "D:\ODEV\HESAPLAR\verilerim" FOR OUTPUT AS #1**

ifadesi, **D** hardiskinde **ODEV** ana klasörü altında yer alan **HESAPLAR** alt klasörü içinde **verilerim** adlı yeni bir kayıt dosyası oluşturur. Benzer şekilde ;

**OPEN "verilerim" FOR INPUT AS #1**

ifadesi de, programın çalışmakta olduğu aktif klasör içindeki **verilerim** adlı dosyayı okumak ve bilgileri program içine aktarmak için açar. Mevcut dosyaya veri eklemek istenildiğinde de benzer şekilde ;

**OPEN "verilerim" FOR APPEND AS #1**

ifadesi de, programın çalışmakta olduğu aktif klasör içindeki **verilerim** adlı dosyayı kayıtlı mevcut en son verinin peşine, programda girilen yeni verileri ekleyerek kaydetmek üzere açar.

Dosya adları OPEN komut satırında yukarıda tanımlandığı şekilde kullanılabilir. Örneğin;

**D\$="c:\belgelerim\odev\hesaplar.dat"**  
**OPEN D\$ FOR OUTPUT AS #1** şeklinde de yazılabilir.

**OPEN** komut dizisinde sonda yer alan **AS #1** ibaresindeki # işareti seçimidir. İfade **AS 1** şeklinde yazılsa da çalışacaktır.

Hangi modda (output/input/append) açılmış olursa olsun, dosya ile ilgili tüm işler (okuma/kayıt) bittiğinde dosya mutlaka kapatılmalıdır. Kapatma komutu;

**CLOSE** şeklinde kullanılır. **OPEN** ve **CLOSE** komut satırları arasında, verilerin kaydedilmesi veya okunması ile ilgili işleri belirleyen komut satırları yer alır.

Açılmış bir sıralı erişimli dosyaya kayıt işlemi **PRINT** veya **WRITE** komutları ile yapılır. Bu komutları yazılış şekilleri;

**PRINT #no,değişkenler**

**WRITE #no,değişkenler**

şeklindedir. Her iki ifadede de #

işaretinden sonra yer alan no ibaresi açılan dosyanın kanal numarasını gösterir (Bu değer 1, 2 veya 3 olabilir. Değişkenler ise, bir değişken olabileceği gibi, virgül veya noktalı virgülle ayrılmış birden çok değişken olabilir. Örneğin **PRINT** komutu;

**PRINT #1, X\$,A,B**

veya

**PRINT #2, A\$**

veya

**PRINT #1, X;Y;C\$**

şeklinde yazılabilir. **PRINT** komutunda birden

çok değişken virgül kullanılarak yazdırıldığında dosya içinde her bir **PRINT** ifadesinde yazdırılan değerler, belirli **TAB** aralığında, aralarında bir ayraç kullanılmadan tek satır olarak kaydedilirler. Noktalı virgül kullanıldığında da yine değerler birbirine yanaşık düzende ve aralarında bir ayraç olmadan kaydedilirler. Benzer şekilde **WRITE** komutu da;

**WRITE #1, X\$,A,B**

veya

**WRITE #2, A\$**

veya

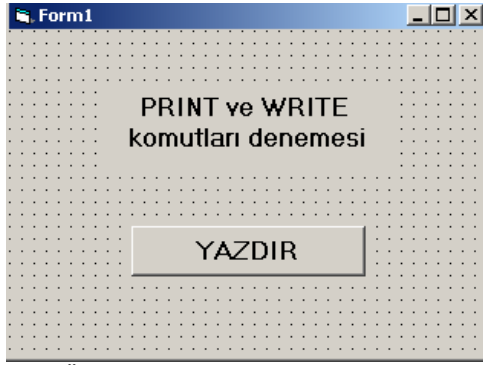
**WRITE #1, X;Y;C\$**

şeklinde yazılabilir. **WRITE** komutunda

birden çok değişken virgül kullanılarak yazdırıldığında dosya içinde her bir **WRITE** ifadesinde yazdırılan değerler, aralarında virgülle ayrılarak tek satır olarak kaydedilirler. Noktalı virgül kullanıldığında da değerler virgül kullanılmış gibi, yine aralarında virgülle ayrılarak tek satır olarak kaydedilirler

**OPEN** komut satırında yer alan **AS #1** ifadesindeki # işareti seçimli olmasına karşın dosyaya kayıt yapılırken kullanılan **PRINT** ve **WRITE** komutlarında seçimli değildir ve mutlaka yukarıda gösterildiği şekilde kullanılmalıdır.

OPEN komutu OUTPUT modunda bir dosya açacak şekilde PRINT ve WRITE komutları ile virgül ve noktalı virgül kullanarak, dosyaya veri yazdırma örneğinde aşağıda örneklenmiştir.

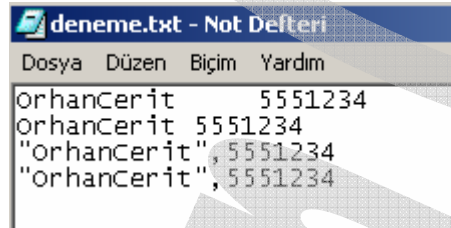


Örnek bir Form tasarım görünümü

```
Project1 - Form1 (Code)
Command1 Click
Private Sub Command1_Click()
Open "d:\deneme.txt" For Output As 1
Print #1, "OrhanCerit", 5551234
Print #1, "OrhanCerit"; 5551234
Write #1, "OrhanCerit", 5551234
Write #1, "OrhanCerit"; 5551234
Close
End Sub
```

PRINT ve WRITE komutları ile dosyaya bilgi yazdırma kod sayfası

Yukarıdaki örnek program çalıştırıldığında D hardiskinde kök dizinde açılan **DENEME.TXT** dosyasına PRINT ve WRITE komutlarında belirtilen değerler yazılmaktadır. Oluşturulan bu dosya herhangi bir editör ile (WORD, WORDPAD, NOTDEFTERİ v.b.) açıldığında aşağıdaki ekran görüntüsü ortaya çıkacaktır.



Örnek program ile oluşturulan dosyada yer alan kayıtlar

Örnek program kodunda ilk PRINT satırında virgül kullanılarak yazdırılan iki değer aralıklı yazıldığı, ikinci PRINT komutunda noktalı virgülle yazdırılan değerlerin ise birbirine yakın yazıldığı görünmektedir. WRITE komutunda ise her iki yazdırma (, veya ;) yönteminin de aynı sonucu verdiği, değerler arasına ayrıca virgül simgesinin de eklendiği görünmektedir.

Herhangi bir dosyaya yazdırılan veriler INPUT veya LINE INPUT komutları ile geri okunarak program içine değer olarak aktarılabilir. Bu komutların kullanımı;

**INPUT #no, değişkenler**

ve

**LINE INPUT #no, değişken**


şeklindedir. Her iki

ifadede de # işaretinden sonra yer alan no ibaresi açılan dosyanın kanal numarasını gösterir (Bu değer 1, 2 veya 3 olabilir. Değişkenler ise, bir değişken olabileceği gibi, virgül veya noktalı virgülle ayrılmış birden çok değişken olabilir.

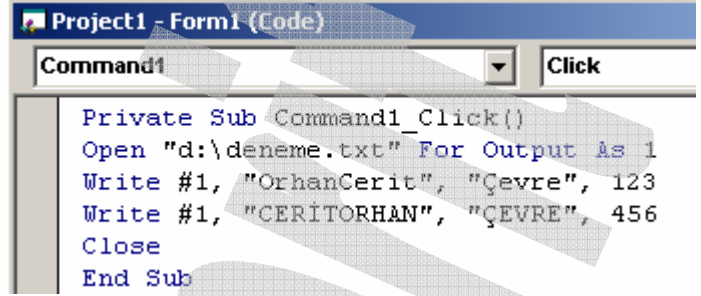
**LINE INPUT** komutunda \$ işareti ile belirlenen karakter dizisi değişkeni (örneğin X\$) kullanılmalıdır. Bu ifade ile, PRINT veya WRITE komutu ile tek satıra yazdırılmış olan tüm değerler, tek bir ifade satırı olarak belirtilen değişkene aktarılır.

**INPUT** komutunda ise birden çok değişken kullanılabilir ve özellikle WRITE komutu ile yazdırılmış bulunan ve aralarında virgül ayraç olan tek satırdaki tüm değerler, belirtilen değişkenlere ayrı ayrı aktarılır.

**WRITE** komutunu kullanarak bir dosyaya kaydedilen bilgileri INPUT ve LINE INPUT komutları ile okuyarak form üzerine yazdırma örneği:



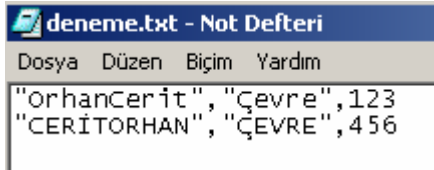
Örnek Form Tasarımı



```
Private Sub Command1_Click()  
Open "d:\deneme.txt" For Output As 1  
Write #1, "OrhanCerit", "Çevre", 123  
Write #1, "CERİTORHAN", "ÇEVRE", 456  
Close  
End Sub
```

Deneme.txt dosyasına veri yazdırma kod örneği

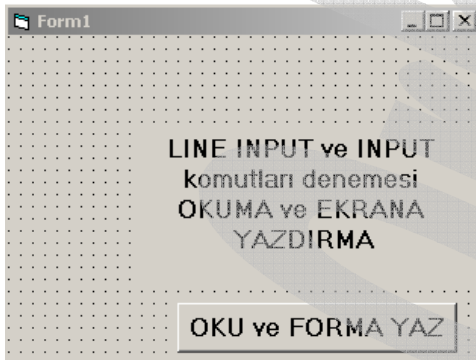
Yukarıda form tasarımı ve algoritma kod örneği verilen program, DENEME.TXT dosyasına iki satır bilgi yazdırmaktadır. Oluşturulan dosyada yer alan bilgiler, bir editör ile bakıldığında gösterdiği yapı aşağıdaki şekilde görüldüğü gibidir.



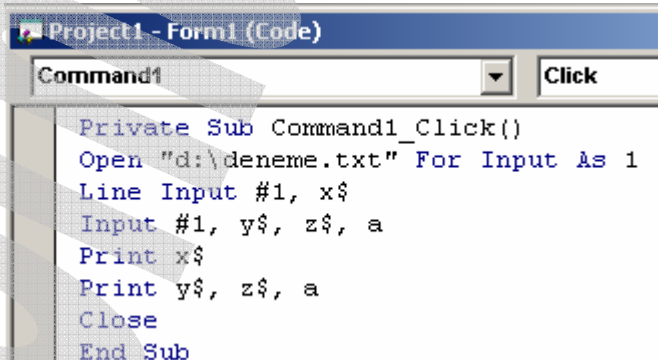
```
deneme.txt - Not Defteri  
Dosya Düzen Biçim Yardım  
"orhanCerit", "Çevre", 123  
"CERİTORHAN", "ÇEVRE", 456
```

Yandaki şekilde, yukarıdaki örnek form ile oluşturulan **DENEME.TXT** dosyasının iç yapısı görülmektedir.

Kayıtlı bu bilgileri INPUT ve LINE INPUT komutları ile okumak üzere yeni bir form tasarımı yapıldığında;



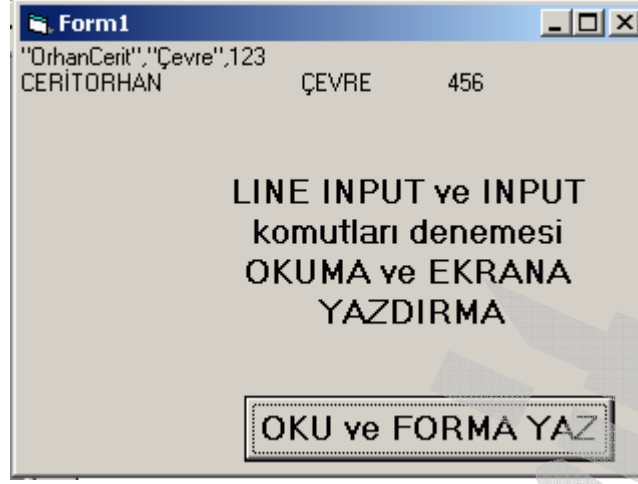
Veri okuma örnek form tasarımı



```
Private Sub Command1_Click()  
Open "d:\deneme.txt" For Input As 1  
Line Input #1, x$  
Input #1, y$, z$, a  
Print x$  
Print y$, z$, a  
Close  
End Sub
```

Veri okuma ve form üzerine yazdırma örnek kodu

Yukarıdaki program çalıştırıldığında ve komut düğmesi tıklandığında aşağıdaki form görünümünü ortaya çıkar.



Çalıştırılan programın sonuç görünümü

Sonuç ekrana bakıldığında ilk satırda görülen ifade `LINE INPUT #1, X$` komut satırı ile okutulan dosya içindeki ilk kaydın `X$` değişkenine tüm olarak aktarıldığını ve tek bir değer olarak ele alındığını göstermektedir. Bu satır `PRINT X$` komutu ile form üstüne yazdırılmıştır. Bu satırdaki virgül veya tırnak işaretlerinin özel anlamları olmayıp, tüm karakter dizisi içinde onlarda birer karakter olarak yer almaktadırlar. İkinci satırda yer alan ifadeler ise,

**INPUT #1, Y\$,Z\$,A** komutu ile okutulan ve **PRINT Y\$,Z\$,A** komutu ile form üzerine yazdırılan satırdır. Dolayısı ile Dosya içinde **WRITE** komutu ile ve virgülle ayrılarak yazılmış üç değer de ayrı ayrı değerler olarak okunmuş **Y\$, Z\$** ve **A** değişkenlerine aktarılarak form üzerine yazdırılmıştır.

Sıralı erişimli bir dosyada daha önceden kaydedilmiş pek çok bilgi olabilir. Bu bilgiler içinde herhangi birisi aranırken daima ilk bilgiden en son bilgiye kadar tüm kaydın taranması gerekebilir. Eğer dosya içindeki kayıt sayısı belli değilse, okutma işlemi (`LINE INPUT` veya `INPUT`) satırları numaralandırarak ve bu satır numaralarına göre iş akışını sürekli okumaya yönlendirerek ardışıklı okuma sağlanabilir. Bilindiği üzere Visual Basic kodları yazılırken satırlara numara verme zorunluluğu yoktur. Program çalıştırıldığında kod içinde yer alan satırlar, yazılış sırasına göre işleme girerler. Ancak, bu akış sırasında iş farklı satırlara yönlendirilebilir. Bunun için yönlendirilecek satırların adreslenmesi (numaralanması) gerekir. Yönlendirme işlemi;

**GOTO SatırNo** komutu ile yapılır. Bu komutta "SatırNo" ifadesi, daha önceden numara verilmiş ve iş akışı sırasında yönlendirilecek satırı belirleyen ifadedir.

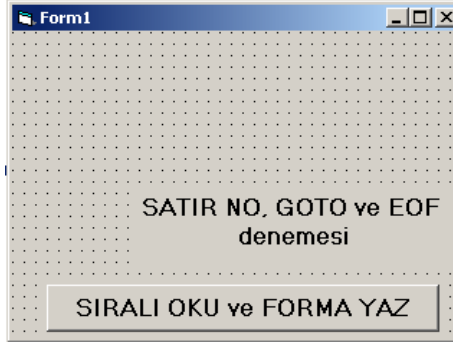
**GOTO** ile yönlendirilen satırlarda, iş akışına göre alt satırlarda yapılan bir işten daha üst satırlarda yapılan bir işe yönlendirme yapıldığında, programın **sonsuz döngü** olarak adlandırılan çevrime girme olasılığı ortaya çıkar. Bu nedenle bu tür yönlendirmelerin koşullu ifadeler ile denetlenmesinde yarar vardır. Ayrıca, özellikle, **GOTO** ile yönlendirme yapılan bir ardışıklı kayıt okuma çevrimi yapılmış ise, tüm kayıtlar sıra ile okunarak dosya sonuna ulaşıldığında, program halen yeni veri okumaya çalışacak ancak dosya sonuna geldiği ve okunacak veri kalmadığı için hata mesajı üretecektir. Bu nedenle, bu tür ardışıklı okuma işlemlerinde okuma sırasında dosya sonuna gelinip gelinmediğinin kontrol edilmesi ve dosya sonuna

ulaşılması ise okuma işleminin kesilmesi gerekmektedir. Bu amaçla, dosya sonu kontrolü **EOF** komutu ile gerçekleştirilir. Bu komutun kullanımı:

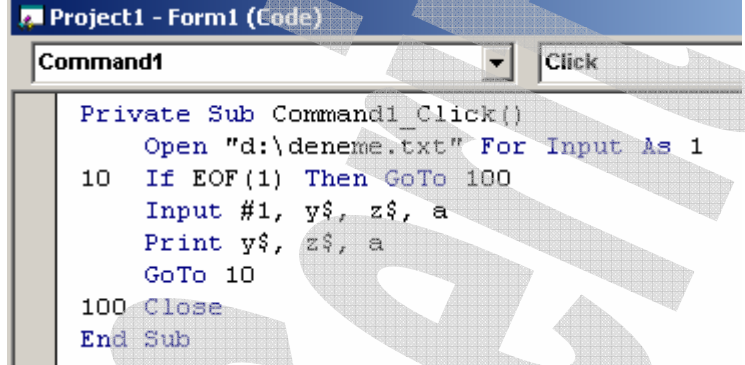
**EOF(KanalNo)** şeklindedir.

Buradaki "**KanalNo**" ifadesi, **OPEN** komutunda, okuma için açılmış dosyada **AS** ibaresinden sonra tanımlanan numaradır.

Örneğin, yukarıda, INPUT ve LINE INPUT komutlarının anlatılması sırasında oluşturulan DENEME.TXT dosyasında mevcut kayıtlı veriler, EOF kontrolü yaparak ardışıklı olarak okunmak istenirse aşağıda örneklediği şekliyle bir form tasarımı ve kod yazımı yapılabilir.



Sıralı okuma örnek form tasarımı

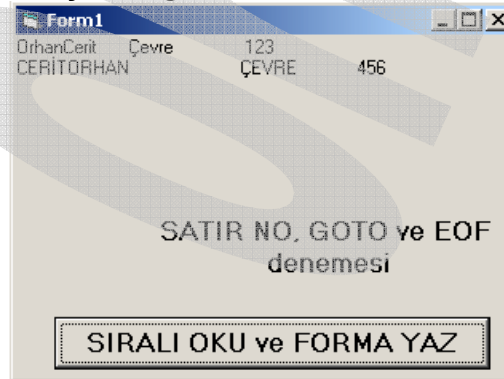


```
Private Sub Command1_Click()  
    Open "d:\deneme.txt" For Input As 1  
    10 If EOF(1) Then GoTo 100  
    Input #1, y$, z$, a  
    Print y$, z$, a  
    GoTo 10  
    100 Close  
End Sub
```

Sıralı okuma örnek kod yazımı. GOTO, EOF ve SatırNo uygulaması

Bu örnekte, komut düğmesi tıklandığında deneme.txt dosyası okunmak üzere açılacaktır. 10 ve 100 numaralar verilmiş satırlar adreslenmiş ve duruma göre yönlendirilecek satırlardır. Program çalışması sırasında, dosya açıldıktan sonra, dosya sonu olup olmadığı 10 numaralı satırda EOF(1) ifadesi ile kontrol edilmektedir. Dosya sonu değilse, INPUT satırı yürütülmekte ve peşine PRINT komutu ile değerler form üzerine yazdırılmaktadır.

Print Satırı altında yer alan GOTO 10 ifadesi iş akışını sorgusuz olarak yukarıdaki 10 numaralı satıra yönlendirmekte ve bu arada tekrar dosya sonu kontrolü yapılmaktadır. Eğer halen okunacak veri varsa okuma gerçeklemede ve bu ardışıklı işlem dosya sonuna kadar sürmektedir. Eğer son veri de okunmuş ve okunacak veri kalmamış ise, iş akışı 10 numaralı satıra geri geldiğinde, program dosya sonu uyarısı ile iş akışını 100 numara ile belirtilen CLOSE satırına yönlendirerek dosyayı kapatmakta ve işi bitirmektedir. Bu program çalıştırılarak komut düğmesi tıklandığında aşağıdaki sonuç form görüntüsü elde edilir.



Sıralı/ardışıklı okuma/yazdırma örneği sonuç görünümü

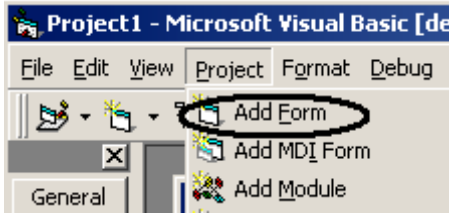
# VISUAL BASIC

## RASTGELE (RANDOM) ERİŞİMLİ DOSYA İŞLEMLERİ

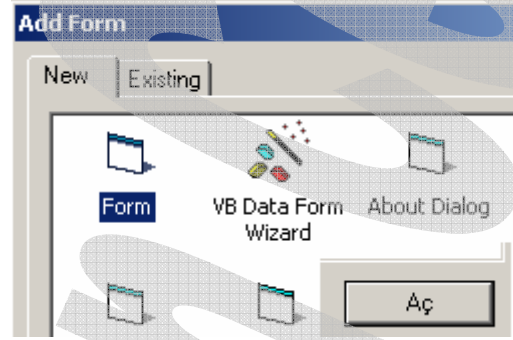
**Rastgele Erişimli Dosyalar**, sıralı erişimli dosyalardan farklı olarak, mevcut kayıtlar arasında ulaşılmak istenen herhangi bir bilgiye doğrudan ulaşılabilen dosyalar olup, bu tür dosyalarda, bir kayıta ulaşabilmek için, sıralı erişimli dosyalarda olduğu gibi, o kayıta kadar olan tüm kayıtların sırayla elden geçirilmesi zorunluluğu yoktur. Herhangi bir kayıta, kayıt numarası vasıtasıyla doğrudan erişilebilir, okunabilir, yazılabilir, kaydedilebilir, silinebilir. Rastgeleli erişimli dosyalarda her kayıt bir kayıt numarasına ve belirli bir kayıt uzunluğuna sahiptir,

Rastgele erişimli dosyaları, bir **telefon rehberi** örneğinde uygulayalım. Örneğimizde, Ad, soyad ve telefon numarası gibi bilgileri bir form üzerinden girelim, kayıt ve sonuç listeleme işlemlerini de ayrı bir form üzerinde gösterelim. Bu amaçla, öncelikle mevcut üzerinde çalıştığımız forma ek olarak ikinci form eklememiz gerekmektedir.

Form eklemek için Visual Basic Menu satırında **Project** seçeneği altında yer alan **Add Form** seçeneği seçilmelidir. Açılan diyalog penceresinden de **Form** seçili olarak **Aç** seçeneği seçilir

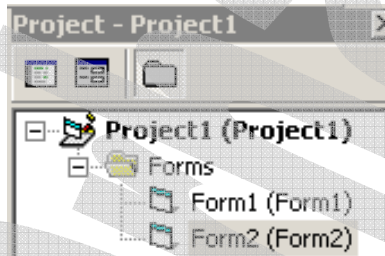


Menüde Yeni form ekleme seçeneği



Yeni Form eklenmesi

Bu seçimler sonucu, Projemizde kullanabileceğimiz iki ayrı form oluşur.



Proje bileşenlerinin görünümü

Birden çok Form kullanılarak çalışıldığında, program akışı sırasında, gerektiğinde bu formların görünmesi veya görünmemesi, programcının tercihine bağlıdır. Formların görünmesi/görünmemesi **Formun Show/Hide** özellikleri ile belirlenir.

**Form1.Hide**

**Form2.Show**

ifadeleri, 1. formun gizleneceğini, 2. formun ise görünür hale getirileceğini belirler.

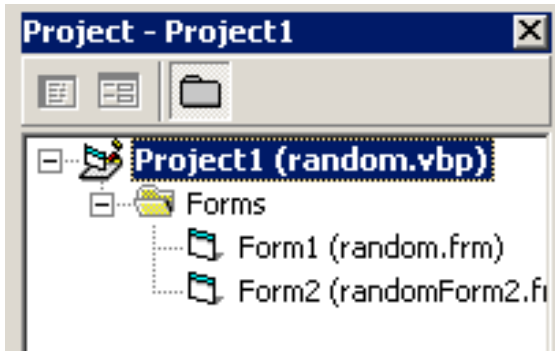


Yukarıda anlatıldığı şekilde oluşturulmuş formlarımızı, program gereksinimlerine uygun olarak aşağıdaki şekilde tasarlayalım

Telefon Rehberi programı Ana kontrol ve veri giriş form tasarımı

Telefon Rehberi programı sonuç listeleme formu

Bu durumda, Formlarımızı yeniden adlandırdığımızda ve veri girişi yaptığımızda formlarımız, aşağıdaki şekilde görüneceklerdir.



Telefon Rehberi programı projesi bileşenleri

Telefon Rehberi programı veri girişi (Form 1)

Söz konusu program çalıştırıldığında sonuç ekranının görünmesi ile ilgili kodlar ve form2 görünümü aşağıdaki gibi olacaktır

Telefon Rehberi programı sonuç listeleme formu

```
Project1 - Form2 (Code)
Command1 Click
Private Sub Command1_Click()
    Form1.Show
    Form2.Hide
End Sub
```

Telefon Rehberi programı Form1/Form2 geçiş algoritması

Rastgele erişimli dosyalarda, her bir kaydın bir numarası ve kayıt uzunluğu olduğu ifade edilmişti. Bu nedenle kayıtlar ile ilgili olarak, hangi verilerin girileceği, girilecek her bir veri için, gerekli olan olası en büyük kayıt uzunluğu, önceden belirlenmelidir. Ayrıca, girilecek verilerin türü de (Tamsayı, Reel sayı, karakter dizisi, tarih v.b.) önceden belirlenmelidir. Veriler ile ilgili bu düzenlemeler, Program algoritmasının başlangıcında (**General**) bölümünde ve **Declarations** kısmında tanımlanırlar.

Bizim örneğimiz için girilecek verileri, kullanıcı tanımlı değişkenler olarak, 15 karakter uzunluğunda **ad** girişi, 15 karakter uzunluğunda **soyad** girişi ve 10 karakter uzunluğunda telefon numarası (**num9** girişi olarak tanımlayalım ve tüm bu değişkenleri **telefon** veri/değişken grubu olarak **deklare** edelim/tanımlayalım.

Bu tanımlama/deklare işlemi için, veri/değişken tipi;

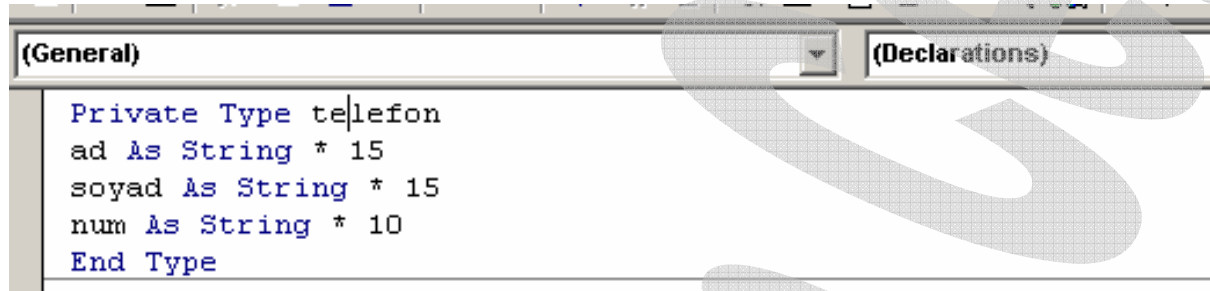
**Private Type** *Değişken Grubunun Adı*

.....

.....

**End Type**

Komut dizisi içine tanımlanır. Aşağıda gösterilen değişken tanımlama örneği, örnek uygulamamız için, ad, soyad ve telefon numarası değişkenlerini **telefon** değişkenine gruplandırmaktadır.



```
(General) (Declarations)
Private Type telefon
ad As String * 15
soyad As String * 15
num As String * 10
End Type
```

Telefon Rehberi programı, kullanıcı tanımlı değişken gruplandırılması/tanımlanması

Rastgele erişimli (random) dosyalara kayıt yapabilmek veya bu dosyalardan kayıt okuyabilmek için, söz konusu dosyanın **RANDOM** özeliğinde açılması gereklidir. Bilindiği gibi sıralı erişimli dosyalarda, yapılacak işin özelliğine göre (okuma, yazma, ekleme gibi) dosyayı farklı özelliklerde (output, input, append gibi) açmak gerekiyordu. Ancak rastgele erişimli dosyalar, tüm okuma ve yazma işlemleri için **random** özeliğinde açılır. Dosya açma işlemi için gerekli komut yazım tarzı (syntax) :

**Open "Dosyaadı" for random as #veriyoluno** şeklindedir. Ayrıca seçimli olarak bu komut dizisine, dosyada yer alan kayıtlar için her bir kaydın uzunluğu da belirtilerek eklenebilir. Bu durumda, komut dizisi;

**Open "Dosyaadı" for random as #veriyoluno len = uzunluk** şeklini alır. Örneğin;

**Open "numaralarım" for random as #1 len = 70** şeklinde bir tanımlama, aktif klasör (içinde çalışılan) içindeki **numaralarım** adlı dosyayı, her bir kayıt için kayıt uzunluğu **70** karakter olacak şekilde açar.

Rastgele erişimli (random) dosyalarda, okuma ve yazma işlemleri, kayıt numarası esas alınarak **GET** ve **PUT** komutları ile yapılır.

Biz yukarıda tanımladığımız Form tasarımına göre üç ayrı metin kutusunda girilen ad için 15 karakter, soyad için 15 karakter ve telefon numarası için 10 karakter uzunluğunda, (veri grubu toplam 40 karakter uzunluğunda) olan verilerimiz için 1 numaralı komut düğmesi olan **KAYIT EKLE** düğmesi tıklanıldığında kayıt dosyasının açılarak, girilen verilerin numarasına göre (örneğin 5 numaralı kayıt olarak) dosyaya yazdırılmasını isteyelim. Bu durumda gerekli kod;

```
Private Sub Command1_Click()  
Dim tel As telefon  
Open "d:\rehber.txt" For Random As #1 Len = 40  
tel.ad = Text1.Text  
tel.soyad = Text2.Text  
tel.num = Text3.Text  
Put #1, 5, tel  
Close  
End Sub
```

tel veri grubu 5 numaralı kayıt olarak dosyaya kaydediliyor

Bu kod yazılımında, daha önce programın **general / declarations** bölümünde tanımlanan telefon grup değişkeni, **Dim** komutu kullanılarak **tel** adında bir başka grup değişkenine aktarılmaktadır. Bu durumda;

**Tel.ad** tel grubuna ait ad değişkenini  
**Tel.soyad** tel grubuna ait soyad değişkenini  
**Tel.num** tel grubuna ait telefon numarası değişkenini temsil etmektedir.

Ve tüm bu kayıt bilgileri;

**Put #1, 5, tel** ifadesi ile **D** harddiskinde **rehber.txt** adıyla kayıtlı bulunan dosyaya **5** numaralı kayıt olarak yazdırılmaktadır. Bu örnekteki **5 numaralı kayıt** rastgele bir değerdir. Eğer kayıt ortamında 5 numarada başka bir kayıt varsa bu silinerek yerine yeni yazdırılan bilgiler geçecektir. Eğer dosyaya bilgi yazdırma işlemi mevcut kayıtlara ek yeni kayıtlar şeklinde yapılacaksa, (dosyaya ek yapılacaksa), yapılacak kaydın mevcut kayıtların en son numaralısından bir sonraya, yani, **toplamkayitsayısı + 1** numaraya kaydedilmesi gerekir. Bu işlemin yapılabilmesi için en azından bir kayıt grubunun toplam karakter uzunluğu ile kayıt dosyasının toplam uzunluğunun bilinmesi, sonuncu kaydın kayıt numarasının bulunması için yeterlidir.

Bir kayıt dosyasının toplam uzunluğu **LOF** ( Length Of File = Dosya Uzunluğu) fonksiyonu ile sorgulanabilir. Bu fonksiyon kullanıldığında sorgulandığı dosyanın kaç karakterden oluştuğunu gösteren ve sayısal anlamı olan bir değer (sayı) üretir. Bu fonksiyonun genel kullanımı (syntax) ;

**X = LOF(veri\_yolu\_no)** şeklindedir. Buradaki **veri\_yolu\_no** ifadesi, **open** komutu ile açılan dosyanın **#** işareti ile belirlenen numarasını gösterir.

Şimdi **LOF** fonksiyonunu, telefon rehberi uygulaması örneğimize uyarlamak istersek, yukarıda girilen kaydı rastgele **5** numaralı kayıt olarak kaydetmiştik. Bunun yerine bu kaydı en son kaydın arkasına yazdırmak istediğimizi varsayalım. Bu durumda yukarıdaki algoritmamız;

```
Private Sub Command1_Click()  
Dim tel As telefon  
Open "d:\rehber.txt" For Random As #1 Len = 40  
i = (LOF(1) / 40) + 1  
tel.ad = Text1.Text  
tel.soyad = Text2.Text
```

Her bir kayıt 40 Karakter olduğuna göre LOF fonksiyonu ile ölçülen dosya uzunluğu 40'a bölünerek toplam kayıt sayısı bulunur. Bu sayının 1 fazlası yeni eklenecek kaydın numarası olur.

```
tel.num = Text3.Text
Put #1, i, tel
Close
End Sub
```

Yukarıda LOF fonksiyonu ile hesaplanan kayıt numarası (i), kayıta kayıt numarası olarak kullanılmıştır.

Şeklini alır. Böylece (i) değişkeni ile tanımlanan kayıt numarası, dosyaya yapılacak yeni kayıt için kayıt numarasını oluşturur.

Bu telefon rehberi oluşturma örneğimizde, girilmiş olan kayıtların tamamının, oluşturulan 2. Form üzerinde listelenmesi işlemi 1. Form üzerinde **KAYIT LİSTELE** (Command2) komut düğmesi ile yapılmaktadır. Bu durumda, bu düğme tıklandığında, Form2'nin görünür hale gelmesi ve kayıtların burada listelenebilmesi için gerekli algoritma aşağıda verilmiştir.

```
Private Sub Command2_Click()
Dim tel As telefon
Form2.Show
Open "d:\rehber.txt" For Random As #1 Len = 40
n = LOF(1) / 40
For i = 1 To n
Get #1, i, tel
Form2.Print i, tel.ad, tel.soyad, tel.num
Next i
Close
End Sub
```

Form2 görünür hale getiriliyor

Toplam kayıt sayısı bulunuyor

Kayıtlar okunuyor

Okunan kayıtlar Form2 üzerine yazdırılıyor

Bu kod yazılımda yer alan;

**Form2.Print i, tel.ad, tel.soyad, tel.num** ifadesindeki **Form2.Print** komutu, print ile yapılan Form üzerine doğrudan yazdırma işlemini, bu kod Form1 üzerinde yer aldığı için, Form2'ye yönlendirmektedir.

Telefon rehberi örneğimizde **Command3** düğmesi ile temsil edilen **isme göre arama** düğmesi için yazılması gereken kod diziliminde de yukarıda ifade edilen mantık geçerlidir. Ancak, Aramanın yapılabilmesi için aranacak ismin tamamının veya en azından bir kısmının girilmesi, girilen bu adın (veya adın bir kısmının), kayıtlardaki adlarla karşılaştırılarak uygun eşleşmelerde mevcut kaydın 2. form üzerine yazdırılması gerekir. Aranacak ismin (veya ismin bir kısmını içeren metnin, 1. Metin kutusuna girilerek 3. komut düğmesinin tıklandığını varsayalım. Bu durumda girilen metnin, okunan metinle karşılaştırılabilmesi için, karakter dizileri fonksiyonlarına gereksinimimiz vardır. En temel karakter dizisi fonksiyonları ve kullanım şekilleri;

**Left\$(karakter\_dizisi, uzunluk\_değeri)**

**Right\$(karakter\_dizisi, uzunluk\_değeri)**

**Mid\$(karakter\_dizisi, başlangıç\_değeri, uzunluk\_değeri)**

**Len(karakter\_dizisi)**

Fonksiyonlardır. Bu fonksiyonları örneğimizden önce basit bir karakter dizisinde uygulayalım;

**Örneğin;**

**X\$ = "Orhan Cerit Enformatik Bölümü"** ifadesini bu fonksiyonlara uygulayacak olursak;

**Left\$** fonksyonu, parantez içinde tanımlanan karakter dizisinin, soldan itibaren 1. harfi dahil yine parantez içinde tanımlanan uzunluk değeri kadar olan kısmını seçer

**A\$ = Left\$(X\$,3)** eşitliğinin sonucu olarak **A\$**'in değeri “**Orh**” olur.

**Right\$** fonksyonu, parantez içinde tanımlanan karakter dizisinin, sağdan itibaren en sonuncu harfi dahil başa doğru, yine parantez içinde tanımlanan uzunluk değeri kadar olan kısmını seçer

**A\$ = Right\$(X\$,3)** eşitliğinin sonucu olarak **A\$**'in değeri “**ümü**” olur.

**Mid\$** fonksyonu, parantez içinde tanımlanan karakter dizisinin, parantez içinde numarası verilen başlangıç harfinden itibaren sağa doğru, yine parantez içinde tanımlanan uzunluk değeri kadar olan kısmını seçer

**A\$ = Mid\$(X\$,5)** eşitliğinin sonucu olarak **A\$**'in değeri “**han C**” olur. (Dizi oluşturulurken kullanılmışlarsa, kelimeler arası boşluklar da birer karakter olarak sayılırlar)

**Len(Dizi)** fonksyonu da parantez içinde ifade edilen dizi değişkenin içeriğinin toplam kaç karakterden oluştuğunu (dizinin karakter sayısını=boyunu) sayısal bir değer olarak verir.

Bu açıklamaları göz önünde tutarak, telefon rehberi uygulamamızın 3. komut düğmesi için karşılaştırma algoritmamızı aşağıdaki gibi yazabiliriz.

```
Private Sub Command3_Click()
```

```
Dim tel As telefon
```

```
Form2.Show
```

```
Open "d:\rehber.txt" For Random As #1 Len = 40
```

```
n = LOF(1) / 40
```

```
ara = Text1.Text
```

```
For i = 1 To n
```

```
Get #1, i, tel
```

```
If Left$(tel.ad, Len(ara)) = ara Then Form2.Print i, tel.ad, tel.soyad, tel.num
```

```
Next i
```

```
Close
```

```
End Sub
```

Form2 açılıyor

LOF fonksiyonu ile Veri sayısı bulunuyor

Sorgulananak metin tanımlanıyor

Sorgulananak metin, kayıttan okunan adın soldan itibaren, sorgu metninin uzunluğu kadar olan kısmı ile karşılaştırılıyor, eşleşme sağlandığında Form2 üzerine yazdırılıyor

Uygulamamızdaki **4. komut** düğmesi için ise işin bitirilebilmesi amacıyla **End** yazılması yeterlidir. Sonuç olarak, Rasgele Erişimli dosya olarak kaydedilen bir telefon rehberi dosyası için okuma, yazma ve listeleme işlemleri için yazılan kodlar bütün olarak aşağıdaki gibi olacaktır;

**ÖRNEK KOD (Form1 için):**

```
Private Type telefon
```

```
ad As String * 15
```

```
soyad As String * 15
```

```
num As String * 10
```

```
End Type
```

Devam ediyor



**Private Sub Command1\_Click()**

```
Dim tel As telefon
Open "d:\rehber.txt" For Random As #1 Len = 40
i = (LOF(1) / 40) + 1
tel.ad = Text1.Text
tel.soyad = Text2.Text
tel.num = Text3.Text
Put #1, i, tel
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Close
End Sub
```

---

**Private Sub Command2\_Click()**

```
Dim tel As telefon
Form2.Show
Open "d:\rehber.txt" For Random As #1 Len = 40
n = LOF(1) / 40
For i = 1 To n
Get #1, i, tel
Form2.Print i, tel.ad, tel.soyad, tel.num
Next i
Close
End Sub
```

---

**Private Sub Command3\_Click()**

```
Dim tel As telefon
Form2.Show
Open "d:\rehber.txt" For Random As #1 Len = 40
n = LOF(1) / 40
ara = Text1.Text
For i = 1 To n
Get #1, i, tel
If Left$(tel.ad, Len(ara)) = ara Then Form2.Print i, tel.ad, tel.soyad, tel.num
Next i
Close
End Sub
```

---

**Private Sub Command4\_Click()**

```
End
End Sub
```

Programımız **Form1** ve **Form2** olmak üzere 2 formdan oluşmaktadır. Listeleme amacıyla **Form2** görünür hale getirildiğinde, bu form üzerindeki caption özelliği **←-GERİ** olarak tanımlanmış komut düğmesi tıklandığında Form2'nin tekrar görünmez olması amacıyla bu komut düğmesi içinde Form2 kod sayfasında ilgili komut düğmesi Click özelliğine **Form2.Hide** yazılması gerekir.