

Quick BASIC

Kursu

Hazırlayan: Mesut AKCAN
Anamur Tek. Lise ve End. Meslek Lisesi
Metal İşleri Bölüm Şefi
makcan@softhome.net
<http://www.mesut.web.tr>
©2000

İÇİNDEKİLER

Bölüm 1 : QuickBasic'e Giriş.....	5
BASIC nedir?.....	5
QuickBasic editörünün kullanımı.....	5
Qbasicde satır numaraları.....	6
PRINT yerine ? (soru işareti).....	6
Qbasicde yardım almak.....	7
Bölüm 2: Qbasicde Veri Tipleri.....	8
Qbasicde Veri Tipleri.....	8
Veri tipleri nasıl kullanılır?.....	8
Data tipi kullanımının bir başka yolu.....	8
Kullanıcı tanımlı Veri Tipleri.....	9
Bölüm 3: CLS , INPUT ve PRINT.....	11
CLS komutu.....	11
Değişkenlere Veri aktarmak.....	11
INPUT Komutu.....	12
PRINT Komutu.....	12
Uygulama.....	13
Bölüm 4: Kontrol blokları.....	14
IF ... THEN ... ELSE.....	14
İç içe IF.	15
Mantıksal karşılaştırmalar.....	15
ELSEIF, SELECT CASE.....	16
Bölüm 5: Döngüler.....	18
GOTO döngüsü.....	18
FOR ... NEXT döngüsü.....	18
DO ... LOOP döngüsü.....	19
WHILE ... WEND döngüsü.....	20
Bölüm 6: Diziler.....	21
Dizi tanımlama. (Hafızada yer açma).....	21
OPTION BASE, LBOUND, UBOUND.....	21
Çok boyutlu değişken tanımlama.....	22
REDIM, ERASE.....	22
Bölüm 7: Goto, Gosub, Sub, Function.....	23
GOTO.....	23
GOSUB.....	23
SUB.....	24
FUNCTION.....	26
STATIC, SHARED, COMMON.....	27
Bölüm 8: Karakterlerle ilgili işlemler.....	29

Quick-Basic Kursu

PRINT.....	29
PRINT USING.....	29
STR\$......	30
VAL.....	30
STRING\$......	31
SPACES.....	31
ASC.....	31
CHR\$......	31
HEX\$, OCT\$......	31
INKEY\$......	31
LCASE\$, UCASE\$......	32
LEFT\$, RIGHT\$, MID\$......	32
LEN.....	32
LTRIM\$, RTRIM\$......	32
Bölüm 9: Rakamlarla ilgili işlemler.....	34
ABS.....	34
INT.....	34
FIX.....	34
RANDOMIZE.....	34
RND.....	34
DÖNÜŞTÜRME FONKSİYONLARI.....	34
MATEMATİKSEL FONKSİYONLAR.....	35
LEN , ASC, VAL.....	36
Bölüm 10: Dosya İşlemleri-1.....	37
OPEN.....	37
Sıralı Erişimli Dosya Oluşturma ve Yazma.....	37
Sıralı Erişimli Dosyadan Okuma.....	39
Satır Satır okuma.....	39
Dosyaya Ekleme Yapma.....	40
UYGULAMA.....	40
Bölüm 11: Dosya İşlemleri-2.....	42
Rasgele Erişimli(RANDOM) Dosya Oluşturma.....	42
Rasgele Erişimli Dosyaya Yazma.....	43
Rasgele Erişimli Dosyadan Okuma.....	44
Bölüm 12: Dosya işlemleri-3.....	46
Binary(ikili) Dosya Oluşturma ve Yazma.....	46
Binary Dosya Okuma.....	46
Okuma ya da yazma konumu.....	47
UYGULAMA.....	47
Dosyayı Silme.....	48
Dosya adını değiştirme.....	48
Konum değiştirme.....	49
Klasördeki Dosyaların Listesi.....	49

Quick-Basic Kursu

Yeni Klasör Oluřturma.....	49
Klasörü Silme.....	49
Dosya Kopyalama.....	49

Bölüm 1 : QuickBasic'e Giriş

Bu kurs sizin bir QuickBasic derleyicisine ve editörüne sahip olduğunuzu varsaymıştır. Burada anlatılacak olanlar QuickBasic v4.5 'a göredir. QuickBasic'in daha kısılmış bir sürümü olan QBASIC v1.1 MS-DOS un son sürümlerinde beraber kurulmaktaydı. Sisteminizde DOS yüklü değilse QBASIC.EXE olmayabilir.

Windows 95 CD nizdeki \OTHER\OLDMSDOS\ klasörünüze açıp QBASIC.EXE yi C: \WINDOWS\COMMAND\ klasörünüze kopyalayın. Qbasic'in, QuickBasic den kısılmış olan en önemli özelliği dosyayı derleyip EXE haline getirememesidir.

Win95 CD niz yoksa ftpsearch.lycos.com , www.yahoo.com , www.altavista.com , www.google.com gibi arama sitelerine bağlanıp qbasic.exe yi aratıp bulun ve sisteminize kopyalayın.

NOT: Bu kursta QuickBasic kelimesi yerine kısaca Qbasic kullanılacaktır. Ayrıca, QuickBasic' i nerde bulacağınız hakkında bana soru sormayın.

BASIC nedir?

BASIC kelimesi **B**eginners **A**ll-Purpose **S**ymbolic **I**nstruction **C**ode kelimelerinin baş harflerinden oluşmuş bir kelimedir. Genel amaçlı yeni başlayanlar için bir programlama dilidir. Öğrenmesi kolay, yüksek seviyeli bir dildir. Basic kodları bir derleyici ile çalıştırılabilir(.EXE) ye dönüştürülebilir.

QuickBasic editörünün kullanımı

QuickBasic paketini bilgisayarınıza kurduğunuzda ; dos komut satırında

QB

yazıp ENTER'e bastığınızda karşınıza basic programınızı yazmanız için bir editör programı yükler. Dos un EDIT ine benzer özelliklere sahiptir. Basic programınız yazdıktan sonra F5 tuşu ile çalıştırabilirsiniz. Programın çalışması bitince herhangi bir tuşa basarak, program hala çalışıyorsa CTRL-PAUSE tuşlarına basarak programın çalışmasını durdurup editöre dönebilirsiniz.

Not: QBASIC kullananlar komut satırında QBASIC yazarak çalıştıracaklar.

UYGULAMA:

dos komut satırında qb yazıp enter tuşuna basarak qbasic editörünü çalıştırın. Aşağıdaki resimde olduğu gibi editöre yazın.

```
File Edit View
CLS ' ekranı siler
? "Merhaba Dünya !"
```

Quick-Basic Kursu

Klavyeden F5 tuşuna basarak yazdığımız basic kodunun çalışmasını sağlayın. Bir problemle karşılaşmazsanız ekranındaki yazılar temizlenecek ve ekranın ilk satırında



yazacak. Editör ekranına dönmek için herhangi bir tuşa basın.

Not: ' (ayırma) işaretinden sonra yazılanlar açıklama içindir, yazılmasa da olur.

Qbasicde satır numaraları

Satır numaraları gereksizdir. Ama kullanabilirsiniz de. Kullanırsanız; numaraların birbirini takip etmesi şart değildir. QBasicde satır numaraları yerine okunurluğu kolaylaştırması için ETİKET kullanılır. Etiketleri de sadece GOTO veya GOSUB ile gidilecek satırlara koymak yeterlidir.

```
10 A = A + 1
20 IF A > 20 THEN GOTO 50
30 PRINT A
40 GOTO 10
50 END
```

Yukarıdaki program kodları yerine; aynı işi yapan, satır numarası vermeden, etiket kullanarak yazılmış hali aşağıda.

```
basla:
A = A + 1
IF A > 20 THEN GOTO bitis
PRINT A
GOTO basla
bitis:
END
```

Bir satırda birden fazla komut da kullanılabilir. İki komutu ayırmak için : (2 nokta üst üste) kullanılır

```
CLS
PRINT "QBASIC"
PRINT "MERHABA DÜNYA !"
```

Üstteki ile alttaki kodlar aynı işi yapar. Farkı yoktur.

```
CLS : PRINT "QBASIC" : PRINT "MERHABA DÜNYA !"
```

PRINT yerine ? (soru işareti)

```
? "Merhaba Dünya !"
PRINT "Merhaba Dünya !"
```

Yukarıdaki iki satır aynı işi yapar. Yazarken ? işareti PRINT komutuna dönüşür. Bundan sonra eminim ki ? işaretini PRINT e tercih edeceksiniz.

Qbasicde yardım almak

QBasic in menülerinde HELP var. Fareyle tıkladığınızda bir yardım ekranı karşınıza gelecek. Ayrıca komutları yazdığımız bölümdeki kelimelerden faydalanarak da yardım alabiliriz. Örneğin OPEN yazın imleç(yanıp sönen gösterge) kelime üzerindeyken klavyeden F1 tuşuna bastığınızda OPEN komutu hakkında yardım ekranı gelir karşınıza. Burada Kısa açıklama, Detaylı açıklama, Örnek programı görebilirsiniz. Tabii burada yardım bilgileri Türkçe değil, İngilizce.

Bölüm 2: Qbasicde Veri Tipleri

Qbasicde INTEGER, SINGLE, DOUBLE gibi veri tipleri vardır. Veriler belleğe saklanmadan önce bellekte ne kadar yer kaplayacağını belirtmek için kullanılır. Değişkenin içeriği ancak bu şekilde bellekte düzenli bir şekilde yerleşir. Örnek

A% = 126

A değişkeninin yanındaki % işareti değişkenin bir INTEGER veri tipine sahip olduğunu gösterir. Ona aktarılan bilgi bellekte 2 karakterlik yer kaplar.

Değişkenlerin veri tipini tanımlamak şart değildir. O zaman tanımlanmamış değişkenler basic tarafından SINGLE olarak kabul edilir.

Qbasicde Veri Tipleri

Qbasic diğer dillerdeki gibi tüm veri tiplerini kullanmanıza imkan vermez, fakat genellikle diğerlerine ihtiyaç duymayacaksınız. Qbasic ile kullanabileceğiniz veri tipleri INTEGER, LONG, SINGLE, DOUBLE ve STRING dir. Daha detaylı bilgi almak için menüden **HELP/Contents** 'i ve çıkan ekranda **Data Types** i tıklayın.

Veri tipleri nasıl kullanılır?

```
A% = 253
Y! = 3.141
Z& = 254144
```

A nın INTEGER olduğunu öğrenmiştik ama Y! ve Z&
! işareti SINGLE verilere sahip değişkenleri tanımlamak için, & işareti LONG verilere sahip değişkenler içindir.

ARAŞTIR: Şimdi siz DOUBLE ve STRING değişkenler için hangi işaretler kullanılması gerektiğini HELP den bulmaya çalışın.

Data tipi kullanımının bir başka yolu

Değişkenlerin data tiplerini tanımlamak için birkaç alternatif yöntem vardır. Bunlar:

```
DIM A AS INTEGER
DIM B AS SINGLE
DIM C AS LONG
```


Quick-Basic Kursu

Bu stil en güzel görüneni ve iyi programcılarının kullandığı stildir. Tanımları düzgün yaptıktan sonra programın ileriki aşamalarında yalnızca değişken adını kullanmanız yeterlidir.

```
DEFINT A
DEFSNG B,K,N
DEFLNG C-D
```

Bu da başka bir stil Değişkenin baş harfine göre tanımlanmamış olan tüm değişkenler DEF in yanındaki değişken türünde olur. İyi bir stil sayılmaz.

```
A% = 253
Y! = 3.141
Z& = 254144
```

Bu şekilde tanımlama yaptığınızda aynı değişkeni hep aynı şekilde yazmak zorundasınız. Yani A% nin değerini değiştirmek için A = 100 kullanmak sakıncalıdır.

DİKKAT:

```
A! = 22.125
A& = 46500
A% = 255
PRINT A!, A&, A%
PRINT A
```

Kullanıcı tanımlı Veri Tipleri

Kendi veri tipinizi belirleyebilirsiniz Bu size bellekte kaplayacak olan verilerinizi kullanmanızı kolaylaştıracaktır. Örneğin kişilerin adres bilgileri üzerinde işlem yapmak istiyorsanız, kişi için kendinizin belirlediği bir veri yapısı oluşturabilirsiniz.

```
TYPE Adrestipi
    Adresi AS STRING * 50
    PostaKodu AS STRING * 7
    Adi AS STRING * 30
    Telefonu AS STRING * 18
END TYPE
```

Bu açıklamalar sanırım yetersiz olacaktır başlangıç için. Daha detaylı bilgi almak için TYPE yazısı üzerinde iken F1 e basarak yardım alabilirsiniz.

Tanımlamış olduğumuz ADRESTIPI veri tipini kullanmak için bellekte yer açmalıyız. Bunun için

```
DIM Personel AS Adrestipi
```

komutunu kullanırız. Şimdi bu değişkenin(PERSONEL) elemanlarına değer aktaralım

```
Personel.Adresi = "Ankara Caddesi"
Personel.PostaKodu = "33522"
Personel.Adı = "Murat Velioğlu"
Personel.Telefonu = "0866-945 44 21"
```

Quick-Basic Kursu

Bu tip deęiřken kullanımı QBasic de yeni iseniz ve ya dięer BASIC dillere alıřkanlıęınız varsa garip gelecektir. Ama bu tip deęiřkenler dięer dillerde de kullanılmaktadır. řu an iin telařlanmanıza gerek yoktur. Zamanla bu yapıya alıřıp gerekli yerlerde rahatlıkla kullanacaksınız.

Veri Tipleri zet Tablosu

VERİ TİPİ	TANIMLAMA řEKİLLERİ	BELLEKTE KAPLADIęI ALAN (Byte)	AIKLAMA
INTEGER	DIM A AS INTEGER DIM A% DEFINT A	2 (16 bit)	-32768 ile 32767 sayıları arasındaki TAM sayıları tutabilir(2^{15}). Daha byk sayılar yazmaya alıřtıęınızda Owerflow(Tařma) hatası verir. Ksurlu rakamlar vermeye alıřtıęınızda yakın olan sayıya tamamlar. A% = 45.12 '--> 45 A% = 51.72 '--> 52
LONG	DIM A AS LONG DIM A& DEFLNG A	4 (32 bit)	-2147483648 ile 2147483647 sayıları arasındaki TAM sayıları tutabilir(2^{31}). Ksurlar INTEGERdeki gibidir.
SINGLE	DIM A AS SINGLE DIM A! DEFSNG A	4 (32 bit)	7 rakamdan oluřan ksurlu veya tam rakamları tutar. A! = 3.141145 Daha fazla rakam girildięinde bilimsel kullanım řekline dnřtrr. A! = 12345678 ' -> 1.234568E+07 E+07 demek .(nokta) yı 7 rakam saęa kaydınılacak demektir E-07 olsaydı sola kaydırma olacaktı. A! = 50000000000 ' -> 5E+10 A! = 1 / 25000000 ' -> 4E-08
DOUBLE	DIM A AS DOUBLE DIM A# DEFDBL A	8 (64 bit)	15 rakamdan oluřan ksurlu veya tam rakamları tutar. Dięer zellikler SINGLEdaki gibidir.
STRING	DIM A AS STRING DIM A\$ DEFSTR A	Her bir karakter iin 1 byte	
Kullanıcı Tanımlı		Tanımlanan geniřlięe gre deęiřir	

PF NOKTASI:

DIM BORCU AS LONG

řeklinde deęiřken tanımlandıktan sonra programın ilerleyen satırlarında

BORCU = 1250000

řeklinde kullanabiliriz. Ama karmařık programlar yazdıęınızda bu kullanım hangi deęiřkenin hangi veri tipinde olduęunu anlamanızda zorluk ıkarır. **Kolayı:** İmle deęiřken zerinde iken F1 tuřuna basarak deęiřkenin hangi modlde hangi veri tipinde kullanıldıęını gsteren ekranı grebilirsiniz.

Bölüm 3: CLS , INPUT ve PRINT

Bu bölümde ekranı silme, program çalışırken değişkenlere veri aktarma ve bu verileri yazdırmayı öğreneceğiz.

CLS komutu

Bu komut çalıştırıldığında daha önceden ekrana yazılmış olan yazılar silinir. Genelde programlar, ilk olarak ekranı temizleyerek, sonraki yazılacaklara temiz bir ekran hazırlar. Burada dikkat edilecek şey CLS den bir önceki zemin rengi ne ise ekran o renk ile doldurulur. CLS den önce bir renk ayarı yapılmamışsa ekran siyah renk ile doldurularak temizlenir.

```
COLOR , 4
CLS
PRINT "Merhaba"
```

Değişkenlere Veri aktarmak

Veri tiplerini açıklarken değişkenlerden bahsetmiştik. Değişkenler, bilgisayarın aklında tutması gereken şeyleri aktarırız. Değişkenler bir harf ile başlayıp 40 karaktere kadar bir isim alabilir. İçinde İngilizce harf ve rakamların dışında semboller olursa kabul edilmez.

```
4KAT
MÜŞTERİ
PERSONEL NO
MEDENİ_DURUMU
```

Yukarıdaki değişken için kullanılan isimler hata oluşturacaktır.

```
PI = 3.14
PRINT PI
```

1. satırda PI değişkenine 3.14 rakamını aktardık. Sonraki aşamalarda PI sayısına bir sayı aktarmazsak program sonuna kadar o değeri aklında tutar. Ve biz bunu gerekli olduğunda kullanırız. Örneğin 2. satırda olduğu gibi.

Yukarıda olduğu gibi değişkenlere veriyi direkt aktardığımız gibi bir fonksiyonun ya da işlemin sonucunu da aktarabiliriz.

```
SAYI = 125
KARE = SAYI * SAYI
PRINT KARE
```

1. satır artık yabancı gelmiyor size. SAYI değişkenine aklında 125 rakamını tut dedik. KARE değişkenine de SAYI değişkenin tuttuğu rakamı kendisiyle çarp ve sonucunu aklında tut dedik. Bu arada SAYI değişkeninin değerinde hiçbir değişiklik olmaz. Değişiklik olabilmesi için aktarma işlemi olması gerekir. = işaretinin sağındaki değer veya işlem solundaki değişkene aktarılır. Yukarıda olduğu gibi = in sağ tarafında SAYI nın kendisiyle çarpımından SAYI değişkeni etkilenmemektedir.

Quick-Basic Kursu

Değişkenlere değerler aktarılırken tanımlanan tipine uygun değerler aktarılmalıdır. Sınırını aşan veya uygun olmayan veri aktarımlar kabul edilmeyecektir. Aşağıdaki satırlar hata oluşturacaktır.

A% = 33652 'Sınırı aşmıştır

A& = 2154.43 ' Bu tür değişkenler küsurlu sayıları aklında tutamaz o yüzden sayıyı yuvarlayarak 2154 yapar. Bu satırda bilgisayar bir uyarı vermez.

A\$ = 65000 ' String değişkenlere değerler " (çift tırnak) lar ile aktarılır

A# = "Merhaba" ' Bu tür değişkene string veri aktarılamaz

INPUT Komutu

Değişkenlere A = 45 gibi bir satır yazarak bir değer aktarabiliyoruz. Bazen program çalışırken değişkenlere programı kullanan kişinin veri aktarması istenebilir. INPUT komutu ile istediğimiz değişkenlere program çalışırken değer aktarılabilir.

INPUT kullanılırken; INPUT un ardından verilecek mesaj tırnaklar arasında yazılır sonra (,) veya (;) konulur ve klavyeden yazdıklarımızı aktaracağımız değişken ismi yazılır.

```
CLS
INPUT "ADINIZ " ; AD$
INPUT "YAŞINIZ " , YAS%
PRINT "Sayın " ; AD$ ; YAS% ; " yaşındasınız."
```

Yaşımızı sorduğunda rakam dışında bir şey yazarsak, bir uyarı ile aynı soruyu tekrar sorulur.

DÜŞÜN: Burada virgül ile noktalı virgülün farkı ne?

PRINT Komutu

Ekrana bir mesaj veya bir değişkenin değerini yada bir fonksiyon yada işlemin sonucunu yazdırmak için kullanılır. PRINT yazmak yerine yalnızca ? yazın.

```
CLS
PRINT 3 * 8 + 12 ' Sonuç 36 olarak ekranda görünecek
YASI = 32
PRINT "Yaşı = " ; YASI ; " dir"
A% = 15
B! = 3.14
ADI$ = "Mesut"
PRINT A% , B! , ADI$
SA$ = "Akcan"
PRINT ADI$ ; SA$
```

DÜŞÜN: PRINT de kullanılan virgül ve noktalı virgülün farkı ne?
PRINT komutunda bir işlemin sonucunu da yazdıracağımızı söylemişim.

```
PRINT 12+33 ' Ekrana toplamı(45) verir
PRINT 125+48*10 ' 605 yazar. Öncelik sırası: Parantez içi, * / + - dir
PRINT 12-(80/4-23)+54/9 ' sonuç: 21 ??
```

Quick-Basic Kursu

```
PRINT 1 + 1 ' Toplam olan 2 çıkar
PRINT "1" + "1"
' 11 çıkar. Çift tırnak içindeki sayılar sayı olarak görülmez.
' Burada sayı olmadığı için toplama işlemi değil
' tırnaklar içindekileri birleştirme işlemi uygulanır
PRINT "QUICK" + "basic" ' QUICKbasic
PRINT 8 + "elma" ' !!! Hata !!!
PRINT "8" + "elma" ' 8elma
PRINT 8 ; "elma" ' 8 elma
PRINT 8/2 ; "elma" ' 4 elma
```

Uygulama

```
CLS
PRINT "Çıkmak için 0 yaz"
basla:
INPUT "Bir sayı yaz "; SAYI
IF SAYI = 0 THEN END
CLS
PRINT "Verdiğiniz sayı "; SAYI
PRINT SAYI; " 'nın karesi "; SAYI ^ 2; " dır."
PRINT SAYI; " 'nın küpü "; SAYI ^ 3; " dır."
GOTO basla
```

PROBLEM:

Dairenin çevresini hesaplayıp ekrana yazan bir program yazın. Dairenin çevresi = Daire çapı * Pi sayısı

Bölüm 4: Kontrol blokları

Programlama dillerinde döngüler ve akış kontrol komutları çok sık kullanılır. Programları program yapan esas kısımlar bu komutlarla sağlanır. Qbasicde diğer programlama dillerindekine benzer yapıda döngüler ve mantıksal karşılaştırmalar yapılabilir.

IF ... THEN ... ELSE

Mantıksal karşılaştırma için kullanılır. Karşılaştırma işlemin sonucunda bir değer döner bu değer ya mantıksal DOĞRU dur ya da mantıksal YANLIŞ. Lise 1 deyken matematik dersinde 1 ve 0 lar ile, doğru ve yanlışlar ile işlemler yapardık. Birçok kişide ne işe yarıyor bunlar diye söylenip dururlardı. Demek ki bir gerekliliği varmış. İşte onlar burada gerekecek, isterseniz MANTIK ile ilgili kısımları bir daha gözden geçirin. :)

Mantıksal karşılaştırma için basit bir örnek:

```
IF A = 40 THEN B = 40
```

Burada A değişkenin değeri 40 ise B nin değeri de 40 olacaktır. Eğer A nın değeri 40 dan farklı ise bu satırın hiçbir etkisi olmayacaktır.

Bir başka kullanımı:

```
A = 25
IF A > 40 THEN M$ ="Sayı 40 dan büyük" ELSE M$= "Sayı 40 dan küçük"
PRINT M$
```

Üstte A değişkeninin değerinin 40 dan büyük olup olmadığı kontrol ediliyor. Mantıksal karşılaştırmaların sonucunda ancak iki değer dönebilir. DOĞRU veya YANLIŞ. Doğru olması durumunda THEN den sonraki işlem yapılır, YANLIŞ olması durumunda ise ELSE den sonraki işlem. A ya 25 aktardığımız için A>40 mantıksal karşılaştırmaların sonucu YANLIŞ olacaktır. Çünkü 25, 40dan büyük değil. Bu durumda M\$ a "Sayı 40 dan küçük" değeri aktarılır.

DÜŞÜN: Üstteki programı denedikten sonra A = 40 olsaydı sonuç ne olurdu? diye düşünüp cevabı bulmaya çalışın. Sonra Qbasicde deneyerek düşündüğünüzü kontrol edin.

```
IF A = 40 .... ' A , 40 a eşit mi?
IF A > 40 .... ' A , 40 dan büyük mü?
IF A < 40 .... ' A , 40 dan küçük mü?
IF A <> 40 .... ' A , 40 a eşit değilse
IF A <= 40 .... ' A , 40 a eşit ya da 40 dan küçük mü?
IF A >= 40 .... ' A , 40 a eşit ya da 40 dan büyük mü?
IF A .... ' A nın değeri 0 dan farklı mı?
IF NOT A .... ' A nın değili DOĞRU mu?
```

Quick-Basic Kursu

IF ile karşılaştırma yaptığımızda dönen değerlere göre çok sayıda komut yürüteceksek aşağıdaki yapıyı kullanırız. Bu şekilde kullanımda karşılaştırma bloğunu bitiren END IF kullanmak zorundayız.

```
IF A > 40 THEN
    'doğruysa yapılacaklar
    ....
ELSE
    'Yanlışsa yapılacaklar
    ....
END IF
```

Örnek :

```
INPUT "ADINIZ " ; AD$
IF AD$ = "MESUT" THEN
    PRINT "SİZİN ADINIZ MESUT"
ELSE
    PRINT "SİZİN ADINIZ MESUT DEĞİL"
END IF
```

İç içe IF.

```
INPUT A
IF A > 40 THEN
    IF A < 60 THEN
        PRINT "SAYI 40 ile 60 arasında"
    ELSE
        PRINT "SAYI 60 yada 60dan büyük"
    END IF
ELSE
    IF A = 40 THEN
        PRINT "SAYI 40a eşit"
    ELSE
        PRINT "SAYI 40dan küçük"
    END IF
END IF
```

Mantıksal karşılaştırmalar

Şimdiye kadar bir satırda bir değişkeni karşılaştırdık. Bazen birkaç değişkenin karşılaştırılıp sonucunun alınması gerekebilir.

```
INPUT A
IF A>10 AND A<20 THEN PRINT "SAYI 10 ile 20 arasında"
INPUT A
IF A<10 OR A>50 THEN PRINT "SAYI 10 ile 50 arasında değil"
' yani sayı 10 dan küçük ya da 50 den büyük
```

Mantıksal karşılaştırmalarda kullanılacak terimler: AND , OR , XOR , EQV, IMP, NOT

Değer	Mantıksal karşılaştırma ve sonucu
-------	-----------------------------------

X	Y	NOT T X	X AND D Y	X OR R Y	X XOR R Y	X EQ V Y	X IMP Y
1	1	0	1	1	0	1	1
1	0	0	0	1	1	0	0
0	1	1	0	1	1	0	1
0	0	1	0	0	0	1	1

1 -> Mantıksal Doğru

0 -> Mantıksal Yanlış

Bu terimlerden faydalanarak bitsel karşılaştırma da yapılabilir.

PRINT (155 AND 160)

yazdığımızda 128 sonucunu alırız. Çünkü 155 in 2 li sayı düzenindeki karşılığı 10011011, 160 inki ise 10100000. İkisini alt alta yazıp bitlerini AND ile karşılaştırdığımızda

```
10011011
10100000
-----
10000000
```

çıkar. O da 128 e eşittir. Desimal sayıları 2 li sayıya çevirmek için Hesap makinasını Bilimsel görünüme getirerek kullanabilirsiniz.. Dec = Decimal(10 lu sayı), Bin = Binary (2 li sayı), Hex = Hexadecimal (16 lı sayı)

ELSEIF, SELECT CASE

IF in bir başka kullanımı

```
INPUT "1 ile 3 arasında sayı girin " ; A
IF A = 1 THEN
    PRINT "SAYI = 1"
ELSEIF A = 2 THEN
    PRINT "SAYI = 2"
ELSEIF A = 3 THEN
    PRINT "SAYI = 3"
ELSE
    PRINT "HATALI SAYI"
END IF
```

Bunun yerine buna benzer yapıya sahip anlaşılabilirliği ve kodlaması kolay olan SELECT CASE yapısı kullanılır.

```
INPUT "1 ile 3 arasında sayı girin " ; A
SELECT CASE A
    CASE 1
        PRINT "SAYI = 1"
    CASE 2
        PRINT "SAYI = 2"
    CASE 3
        PRINT "SAYI = 3"
```


Quick-Basic Kursu

```
        CASE ELSE  
            PRINT "HATALI SAYI"  
END SELECT
```

Bölüm 5: Döngüler

Tekrar edilen işlemler için döngüler kullanılır. Böylece bilgisayarın sürekli yapacağı işlemler için aynı komutları bir daha yazmak zorunda kalmayız.

GOTO döngüsü

En basit döngü GOTO ile yapılan döngüdür.

```
basla:
A = A + 1
PRINT A
GOTO basla
```

Yukarıdaki program bir sonsuz döngü oluşturur. Durdurmak için CTRL-PAUSE tuşlarına basınız. Aşağıdaki şekilde değişiklik yaparsak döngüyü kontrol altına almış oluruz.

```
basla:
A = A + 1
IF A>20 THEN END
PRINT A
GOTO basla
```

PROBLEM 1 : 1 den 30 a kadar olan çift sayıların karelerini ekrana yazan bir program yazın.

FOR ... NEXT döngüsü

Belirli sayılarda işlemlerin tekrar etmesi için kullanılır.

```
FOR N = 1 TO 25
    PRINT N
NEXT N
```

1 den 25 e kadar olan sayıları yazacaktır. Her döngüde N değişkeninin değeri 1 artacaktır. Eğer ilk satırı

```
FOR N = 1 TO 25 STEP 4
```

yapacak olursak. N nin ilk değeri 1 olacak sonra her seferinde üzerine 4 eklenerek devam edecektir.

```
FOR N = 25 TO 1 STEP -1
```

yazılacak olursa 25 den 1 e doğru N nin değeri her seferinde 1 azaltılır.

NEXT in arkasına değişkeni yazmak şart değildir ama okunurluğu kolaylaştırmak için yazmakta fayda vardır.

FOR ları içi içe koyarak da kullanılabilir. İçerdeki FOR un NEXT i dışarıdakinin NEXT inden sonra gelmemesine dikkat etmelisiniz.

Quick-Basic Kursu

```
FOR N = 1 TO 10
    FOR M = 1 TO 4
        PRINT N * M ,
    NEXT M
    PRINT
NEXT N
```

FOR döngüsünden çıkma gereği olursa EXIT FOR ile çıkılabilir.

PROBLEM 2 : Problem 1 i FOR..NEXT döngüsüyle yapın

DO ... LOOP döngüsü

```
DO
    PRINT A
    A = A + 1
LOOP
```

Bu da sonsuz döngü oluşturacaktır. Döngüyü kırmak, kontrol altına almak için çeşitli yollar var

EXIT DO ile döngüden çıkmak:

```
DO
    IF A > 40 THEN EXIT DO
    PRINT A
    A = A + 1
LOOP
```

WHILE kullanarak döngüyü kontrol altına almak:

```
'1 . program
CLS
DO WHILE A < 40 ' A, 40 dan küçükİKEN döngüye devam
    PRINT A
    A = A + 1
LOOP
'2 . program
CLS
DO
    PRINT A
    A = A + 1
LOOP WHILE A < 40 ' A, 40 dan küçükİKEN döngüye devam
```

Yukarıdaki iki programı denediğinizde hiçbir fark göremeyeceksiniz. Şimdi ilk satırlarına A = 45 komutunu ekleyip deneyin ve farkı anlamaya çalışın. Eğer WHILE ı LOOP un yanına koyarsak döngüde şart aranmaksızın en az 1 kere döner. DO nun yanına konulursa döngü başlamadan şart kontrol edilir, şart uygun değilse döngü gerçekleşmez..

UNTIL kullanarak döngüyü kontrol altına almak:

```
CLS
```

Quick-Basic Kursu

```
DO UNTIL A > 40 ' şart DOĞRU olanA KADAR dön. A , 40 dan büyük olana kadar devam
    PRINT A
    A = A +1
LOOP
```

WHILE için verdiğim açıklamalar bunda da geçerli UNTIL i DO nun yanına yada LOOP un yanına koyabiliriz.

Kısaca WHILE, şartın DOĞRU olmasında; UNTIL, şartın YANLIŞ olmasında döngüye devam eder.

PROBLEM 3: Problem 1 i DO..LOOP döngüsüyle yapın

WHILE ... WEND döngüsü

```
CLS
WHILE A < 40 ' A , dan küçük İKEN devam
    A = A + 2
    PRINT A
WEND
```

WHILE...WEND, DO...LOOP un bir alternatifidir. Ama DO LOOP kadar kullanışlı değildir. Çünkü DO LOOP da karşılaştırma şartını başta veya sonda verebiliyoruz.

PROBLEM 4: Problem 1 i WHILE..WEND döngüsüyle yapın

Bölüm 6: Diziler

Dizilere neden ihtiyaç duyulur? Çünkü binlerce değişkeni tanımlamak uzun sürer ve kullanışlı olmaz. Örneğin 12 ay için ayrı ayrı değişken tanımlamak yerine yalnız bir boyutlu değişken tanımlamak yeterlidir.

DIM AY(12) gibi.

Dizi tanımlama. (Hafızada yer açma)

```
DIM AY1 AS STRING
DIM AY2 AS STRING
DIM AY3 AS STRING
..
..
```

Gördüğünüz gibi bu uzayıp gidecek. Bunun yerine

```
DIM AYLAR(12) AS STRING
```

yeterli olacaktır. Bu tanımladığımız dizinin kullanımı

```
AYLAR(1) = "OCAK"      : AYLAR(2) = "ŞUBAT"
AYLAR(3) = "MART"      : AYLAR(4) = "NİSAN"
AYLAR(5) = "MAYIS"     : AYLAR(6) = "HAZİRAN"
AYLAR(7) = "TEMMUZ"    : AYLAR(8) = "AĞUSTOS"
AYLAR(9) = "EYLÜL"     : AYLAR(10) = "EKİM"
AYLAR(11) = "KASIM"    : AYLAR(12) = "ARALIK"
buay% = VAL(LEFT$(DATE$, 2))
PRINT AYLAR(buay%)
```

OPTION BASE, LBOUND, UBOUND

Normalde, yani belirtmezseniz, boyutlu değişken tanımladığınızda; ilk boyut no 0 (Sıfır) olur. DIM A(10) dediğimizde 10 değil 11 adet değişken tanımlamış oluruz. Ama istersek ilk boyut numarasını kendimiz belirleyebiliriz. (Tabii 1 yada 0 olarak)

OPTION BASE 1

yazdığımızda bundan sonra tanımlanacak boyutlu değişkenler yani dizilerin ilk boyut numarası 1 olur.

DIM A(10) yazarsak 10 değişken tanımlamış oluruz.

Tanımlanmış bir dizinin ilk boyut numarasını öğrenmek için LBOUND, son boyutunu öğrenmek için UBOUND kullanılır.

```
OPTION BASE 1
DIM A(10) AS INTEGER
PRINT "İlk boyut no: " ; LBOUND(A)
PRINT "Son boyut no: " ; UBOUND(A)
```

Örnek:

Quick-Basic Kursu

```
CLS
DIM A(10) AS INTEGER
FOR N = 0 TO 10
    A(N) = N * 10
NEXT

FOR N=LBOUND(A) TO UBOUND(A)
    PRINT A(N) ,
NEXT
```

Bir başka özellikte boyut numaralarını kendimiz belirleyebilmemiz.

DIM A(10 TO 20) yazdığımızda A'nın ilk boyut numarası 10 son numarası 20 olacaktır. Bundan sonra A(5) = 40 ya da A(22) = 65 yazacak olursanız hata oluşacaktır. Çünkü değişken dizi sınırları dışında. (Subscript out of range)

Çok boyutlu değişken tanımlama

Şimdiye kadar tek boyutlu değişkenler tanımladık.

DIM A(5, 6, 10)

yazarak üç boyutlu bir dizi oluşturabiliriz. Buna göre bellekte (5 x 6 x 10) + 3 adet değişken için yer açıldı. +3 adet 0. değişkenler için. Yine bir başka kullanım olarak

DIM A(5, 3 TO 12, 5 TO 15) gibi bir tanımlama da yapabiliriz.

```
OPTION BASE 1
CLS
DIM SAYI(10,10) AS INTEGER
FOR N = 1 TO 10
    FOR M = 1 TO 10
        SAYI(N,M)=N * M
    NEXT
NEXT

'Çarpım tablosu bellekte oluştu
INPUT "1. sayıyı girin " ; A
INPUT "2. sayıyı girin " ; B
'Çarpım tablosundan sonucu alıyoruz
'Dikkat edin verilen sayıları çarpıyoruz
PRINT SAYI(A,B)
```

REDIM, ERASE

Diziler için bellekte ayırdığımız alanı genişletmek için kullanırız.

```
CLS
REM $DYNAMIC
' üst satır, değişkenlerin dinamik olacağını belirtir.
DIM A(15)
PRINT UBOUND(A)
REDIM A(30) ' Yeniden boyutlandı
PRINT UBOUND(A)
ERASE A ' Bellekte kapladığı alanı boşalt, sil
PRINT UBOUND(A) ' HATA. Çünkü dizi bellekten atıldı
```

Bölüm 7: Goto, Gosub, Sub, Function

Programlarımızdaki kodlar arttıkça veya programın işlevleri arttıkça bazı tekrar eden işlemler gerekli olabilir. Ya da içinde birkaç şey değiştirerek aynı işlemler yapmak gerekir. Bunun için alt programlar kullanılır. Ayrıca Fonksiyonlar ve Alt programlar kullanmak programınızın okunurluğunu kolaylaştıracaktır.

GOTO

Goto komutunu daha önce döngü oluştururken de görmüştük. Aslında Goto ve gosub komutlarına çok az ihtiyaç duyacaksınız. Sadece diğer basic dillerine uyumlu olsun diye konulmuş. Program kodunun herhangi bir yerinde işleyişini bırakıp başka bir noktadan çalışması için kullanılır. Goto ve ardında bir etiket yada satır numarası yazılmalıdır.

```
PRINT "MERHABA DÜNYA"  
GOTO 10  
PRINT "BU SATIRI GÖREBİLECEK MİSİNİZ?"  
10 PRINT "PROGRAM SONA ERDİ"  
END
```

GOSUB

Gosub ile alt programlar oluşturabiliriz. Gosub 'u da kullanmanıza gerek yoktur. Onun yerine SUB kullanmak daha iyi olacaktır. Gosub da goto gibi programın işleyişini bırakıp başka bir noktadan başlaması sağlanır. Fakat farklı olarak RETURN komutunu görünce kaldığı yere geri dönerek çalışmasına devam eder. Alt programa, istediğimiz yerden istediğimiz kadar atlayabiliriz. GOSUB ve ardından bir etiket ya da satır numarası yazılmalı. Gosub ile atladığımız yerde RETURN bulunmazsa geri dönüş olmaz.

```
CLS  
GOSUB CIZGICIZ  
PRINT "MERHABA DÜNYA"  
GOSUB CIZGICIZ  
PRINT "QUICK BASIC"  
GOSUB CIZGICIZ  
PRINT "PROGRAMLAMA DİLİ"  
GOSUB CIZGICIZ  
END ' programı burada sonlandırmazsak  
    ' alt program da çalışır ve hata oluşur  
  
CIZGICIZ:  
PRINT "-----"  
RETURN  
'2. program  
CLS  
PRINT "çift sayılar(1-100)"  
DO  
    A = A + 1  
    IF (A MOD 2) = 0 THEN GOSUB CIFTSAYI  
LOOP UNTIL A = 100  
END
```

Quick-Basic Kursu

```
CIFTSAYI:
PRINT A;
RETURN
```

SUB

GwBasic gibi diğer dillerde Sub veya Function özellikleri yoktur. O yüzden program kodları büyüdükçe okumak ve kontrol etmek epey zor olacaktır. İlk başta Sub ve Functionların faydalarını anlamak zor olabilir. Alıştığınızda bir defa yapacağınız işlemler için bile bunları kullanacaksınız belki de.

Sub yapısı: [ile] arası şart değil gerekirse kullanılır

```
SUB altprogram_ismi (varsa parametreleri) [STATIC]
..
..
[EXIT SUB] ' alt programdan çıkılmak istenirse
..
..
END SUB ' alt program sonu
```

Bir sub veya function eklemek için EDIT menüsündeki New Sub... ya da New Function dan faydalanabilirsiniz ya da direkt yazabilirsiniz. Alt programı eklediğinizde bunlar ayrı bir sayfa olarak görünür. Sub veya Function listesini görmek ve istediğinizi incelemek için F2 tuşuna basın.

Uygulama:

Menüden **File / New Program** ile yeni bir projeye başlayın. **SUB yazıyaz** yazıp ENTER' e bastığımızda hemen iki alt satıra **END SUB** yazıldığını göreceksiniz. Kodlarımızı SUB ile END sub satırları arasında olmalı. Arasına şunları ekleyin

```
PRINT "=====
PRINT "==  MERHABA  =="
PRINT "=====
```

F2 tuşuna basın Altprogramları listeleyen ekran gelecek **Untitled** kaydedilmemiş basic dosyamız oluyor ve alt programların anası oluyor kaydettiğimizde bu isim değişir. **yazıyaz** ise alt program ismi. Alt tarafta seçili olan ile ilgili işlemler var.

Edit in Active: Düzenlemek için aç

Edit in Split: Düzenleme ekranına ayrı bir bölme olarak aç

Delete : Modülü yani alt programı sil

Move : Modülü açık olan başka bir basic dosyaya taşı. Qbasic de aynı anda çok sayıda dosya açılabilir.

Siyah şerit Untitled üzerindeyken ENTER'e basın.

Şimdi oluşturduğumuz altprogramı çağıracağız. Çağırarak için yalnızca yazıyaz

Quick-Basic Kursu

yazabiliriz. Ama programın okunurluğu açısından

CALL yazıyaz

ile çağırmak en doğrusu olacaktır. F5 ile programımızı çalıştırdığımızda çalıştığını göreceksiniz. Kodları şu hale getirip çalıştırın. Buradaki kodlama kolaylığını öğrenmeye çalışın

```
CLS
yaziyaz
yaziyaz
```

Parametre kullanımı:

Alt programımız çalışırken ona bazı değerler gönderip farklı şekillerde çalışmasını sağlayabiliriz. Az önceki SUB da parametre yoktu. Parametre veri değişken tipinin ne olacağını belirtmekte fayda var. Belirtilmezse SINGLE olarak kabul edilir. Bazı örnekler:

SUB ekranayaz (satir AS INTEGER, sutun AS INTEGER)
aynı satırı şöyle de yazabiliriz
SUB ekranayaz (satir%, sutun%)

'Örnek program:

```
CLS
cizgi$ = STRING$(60, "-")
yaziortala 1, çizgi$
yaziortala 2, "Merhaba"
yaziortala 3, "Bugün Qbasic de Goto, GoSub, Sub ve Function'u öğrendim"
'farklı bir kullanım: CALL ile alt programı çağırma
CALL yaziortala(5, "Qbasic Öğrenmek çok zevkli")
yaziortala 6, çizgi$

SUB yaziortala (satir AS INTEGER, yazi AS STRING)
    uzunluk% = LEN(yazi)
    LOCATE satir, (80 - uzunluk%) / 2
    PRINT yazi
END SUB
```

PROBLEM:

1. Yukarıdaki programı alt program kullanmadan yapmaya çalışın.
2. Alt program kullanarak sağa yaslı yazı yardırmayı deneyin
yazisagayasla 4 , "Merhaba"
gibi

FUNCTION

Functionların yapısı SUB lar gibidir. Yukarda açıklananlar bunda da geçerli. Function'un farkı verilen değerler üzerinde işlem yapıp bir sonuç ile geri döndürmesi. Qbasic in kendi yapısındaki birçok komutun Function özelliği vardır. Örneğin : $X = \text{SQRT}(81)$ yazdığımızda verilen 81 sayısı SQRT(karekök alma) fonksiyonu tarafından işlenir ve sonuç olarak 9 döner. Bu sonuç X değişkenine atanır.

Şimdi biz kendimiz bir fonksiyon oluşturalım

```
X = karesi(15)
PRINT X
PRINT "20 nin karesi = "; karesi(20)
PRINT "1.4 ün karesi = "; karesi(1.4)
FUNCTION karesi (sayi AS DOUBLE)
    DIM sonuc AS DOUBLE
    sonuc = sayi * sayi
    karesi = sonuc
END FUNCTION
```

Function da tek satır ile sonucu alabilirdik. Anlaşılır olması bakımından uzun yazıldı. Kısa olarak:

```
FUNCTION karesi (sayi AS DOUBLE)
    karesi = sayi * sayi
END FUNCTION

'2. örnek:
CLS
PRINT enbuyuksayi(15, 25)
FUNCTION enbuyuksayi (sayi1, sayi2)
    IF sayi1 > sayi2 THEN
        enbuyuksayi = sayi1
    ELSE
        enbuyuksayi = sayi2
    END IF
END FUNCTION

'3. örnek
DIM sayi(5)
CLS
sayi(0) = 20
sayi(1) = 30
sayi(2) = 66
sayi(3) = 88
sayi(4) = 36
sayi(5) = 23
x = ortalama(sayi())
PRINT x
FUNCTION ortalama (sayilar())
    FOR n = LBOUND(sayilar) TO UBOUND(sayilar)
        t = t + sayilar(n)
    NEXT
    ortalama = t / n
END FUNCTION
```

STATIC, SHARED, COMMON

STATIC:

Bazen tanımladığımız değişkenin değerini kaybetmeden SUB ve FUNCTION içinde de kullanmak gerekli olabilir. Alt program içinde değişkeni STATIC ile tanımlarsak değişken değerini kaybetmez aklında tutar. Yoksa her fonksiyon ve sub başlangıcında tanımlanan değişkenlerin değerleri sıfırlanır. Örnek:

```
CLS
yaziortala "merhaba"
yaziortala "Qbasicde Function ve Sub Kullanmak kodlamayı kolaylaştırıyor"
yaziortala "Öğrenmem gereken daha çok şey var sanırım."
SUB yaziortala (yazi AS STRING)
    STATIC satir AS INTEGER
    satir = satir + 1
    LOCATE satir, (80 - LEN(yazi)) / 2
    PRINT yazi
END SUB
```

Burada dikkat ederseniz SUB a satır numarasını göndermedik 0 dan başladı her SUB başlamasında 1 arttı ve yazılar alt alta ortalı olarak yazıldı. STATIC i DIM ile değiştirin farkı göreceksiniz.

SHARED:

Bazen de bir değişkenin değerini kaybetmeden tüm SUB ve FUNCTION içinde geçerli olması istenebilir. Buna değişkeni global (genel, her yerde geçerli) tanımlama diyoruz. Alt program içinde yapılan değişken tanımlamaları Local tanımlama(yerel, sadece alt program içinde geçerli)diyoruz. Global tanımlamayı Ana program içinde yapmalıyız

```
DIM SHARED satir AS INTEGER
CLS
satir = 2
LOCATE satir: PRINT "Merhaba"
yaziortala "Qbasic öğreniyorum"
satir = satir + 1
LOCATE satir: PRINT "Çalışan başarır"
yaziortala "İstiyorsan başarırırsın"
SUB yaziortala (yazi AS STRING)
    satir = satir + 1
    LOCATE satir, (80 - LEN(yazi)) / 2
    PRINT yazi
END SUB
'basit bir örnek daha
DIM SHARED a
CLS
a = 5
PRINT a
CALL karesi
PRINT a
PRINT ussu(3)
PRINT a
PRINT ussu(4)
SUB karesi
    a = a * a
END SUB
```

Quick-Basic Kursu

```
FUNCTION ussu (kuvvet%)  
    a = a ^ kuvvet%  
    ussu = a  
END FUNCTION
```

COMMON:

Common, shared gibidir ama daha genel bir tanımlama yapılır. Qbasic ile CHAIN komutuyla başka bir bas dosyaya bağlantı kurabiliriz. Değişken Common ile tanımlama yapılırsa değeri bağlantı kurulan bas dosyada da geçerli olur. Zaten başlangıçta tek bas dosya üzerinde çalışacağınızdan bu gerekli olmayacak.

Bölüm 8: Karakterlerle ilgili işlemler

Kursun bu bölümünde karakterle ilgili komutları inceleyeceğiz.

PRINT

Verileri ekrana yazdırmak için kullanıyoruz.

PRINT ifade yada değişken , ;

ifade olarak bir fonksiyon ya da matematiksel işlem olabilir. Virgül verileri belli atlama noktalarına dizerek yazar, Noktalı virgül ise verileri boşluk bırakmadan yazdırır.

```
A$ = "Quick" : B$ = "Basic" : C$ = "v4.5"
PRINT A$ , B$ , C$
PRINT A$ ; B$ ; C$
PRINT A$ + B$ + C$ ' üstteki satırla aynı işi görür
PRINT A$ ; " " ; B$ ; " " ; C$
PRINT LEN(A$)
PRINT (4545 - 256) * 24 / 2 ^ 3 + 20
```

Matematiksel işlem uygulandığında işlem öncelik sırası: parantez içi , ^ , * ve / , + ve -

PRINT 10 - 3 * 2 işleminin sonucu 14 değil 4 dür. Çünkü * öncelikli olduğundan 3 * 2 işlemi önce yapılır. 10 dan 3 ü çıkarıp 2 ile çarpmak isterseniz

PRINT (10 - 3) * 2 olarak yazmalısınız.

TAB komutu ile sonraki yazacağımız metnin başlangıç kolonunu belirtiriz

PRINT "1 - " ; TAB(5) ; "Qb 4.5"; TAB(20); "DOS"

PRINT "2 - " ; TAB(5) ; "Visual basic " ; TAB(20); "WIN"

SPC komutu ile de arada bırakılacak boşluk belirtilir.

PRINT "1 - " ; SPC(3) ; "Qbasic"; SPC(4); "DOS"

PRINT "2 - " ; SPC(5) ; "Visual basic " ; SPC(4) ; "WIN"

PRINT USING

Metin ya da rakamları belirtilen biçimde yazar. Biçimleme için özel karakterler kullanılır.

Sayısal değerler için

: sayının konumunu

. (nokta) : küsuratı

, (virgül) : binler ayırıcı

+ (arti) : sayının negatif veya pozitif işaretinin konumu

- (eksi) : negatif sayılarda, sayıdan sonra - konur

\$\$: Dolar işareti ekler. PRINT USING "\$\$###.##" ; 458.62

** : Rakamların başına * koyarak her rakamın aynı genişlikte olmasını sağlar

Quick-Basic Kursu

****\$** : ** ve \$ bileşik

^^^ : Sayıları üssü olarak gösterir. PRINT USING "##.##^^^"; 234.56

Sözel değerler için

& : Değişkenlerin yerleştirileceği yeri belirler
PRINT USING "Gülen & ile ağlayan &"; "ayva"; "nar"

! (ünlem) : Metnin ilk karakterini verir
PRINT USING "!!"; "Mesut"; "Akcan"

_ (alt eksi) : Bu tablodaki özel karakterlerden birini yazdırmak istersek önüne _ konur
PRINT USING "! 1500_!"; "qbasic"

Herhangi bir karakter : (Bu tabloda verilen karakterlerin dışında karakter) Olduğu gibi aynı konumda yazılır

```
DIM AY AS DOUBLE
CLS
AY(0) = 454121.3654 : AY(1) = -6845.587 : AY(2) = 982.6
FOR N% = 0 TO 2
    PRINT USING "###,###.##+"; AY(N%)
NEXT
```

STR\$

Bellekte sayı olarak tutulan bir değeri metne(**STR**ing) dönüştürür.

STR\$(sayısal değer ya da değişken)

Örnek:

```
CLS
A% = 1986 : B% = 15
PRINT "Bu yıl ";
PRINT A% + B%;
PRINT " yılındayız"
PRINT "Bu yıl ";
PRINT STR$(A%) + STR$(B%);
PRINT " yılındayız"
```

Yukarıdaki kodları denediğinizde A% ile B% yi toplayabilirken bunu **STR**ing e yani metne dönüştürdüğünde toplamak yerine birleştirdi. Çünkü daha önce PRINT komutunda da görmüştük string verilerde sayısal işlem yapılamaz.

```
PRINT 1 + 1 ' sonuç: 2
PRINT "1" + "1" 'sonuç: 11 , ama sayısal 11 değil
```

VAL

Bellekte metin olarak tutulan rakamı, sayısal işlemlerde de kullanılabilecek halde sayı değerine (**VAL**ue) dönüştürür

VAL(sözel değer ya da değişken)

Örnek:

```
INPUT "Doğum yılınız"; T$
PRINT "Yaşınız "; 2001 - VAL(T$)
```

Örnekte metin olarak girdiğimiz T\$ sayısal işleme sokulamaz VAL ile dönüşüm sağlanarak sayısal işlem yapılmıştır.

STRING\$

Aynı karakterden çok sayıda kullanmanız gerekiyorsa kullanılır.
STRING\$(sayı,karakter)

```
PRINT STRING$(2000,"*") ' ekranı * ile doldurur
PRINT STRING$(80,"=") ' Bir satırı = ile doldurur
```

SPACE\$

Değişkene istenilen sayı kadar boşluk karakteri aktarır

```
FOR I=1 TO 5
  X$=SPACE$(I)
  PRINT X$;"MESUT"
NEXT
```

ASC

Bir karakterin **ASCII** kod karşılığını verir
ASC(karakter)

```
PRINT ASC("A")
PRINT ASC("MESUT") 'bu satırda sadece ilk harf olan M harfi dikkate alınır
```

CHR\$

ASCII koduna denk gelen karakteri(**CHAR**acter) verir.
CHR\$(ascii kod)

```
PRINT CHR$(65) 'ekrana A çıkar
FOR N% = 32 TO 255 ' 32 den 255 e kadar olan
  PRINT CHR$(N%) 'ascii karakterleri yaz
NEXT
```

HEX\$, OCT\$

HEX\$: Verilen sayıyı 16'lık sayı sistemine(**HEX**adecimal) dönüştürür
HEX\$(sayı)

OCT\$: Verilen sayıyı 8'lik sayı sistemine(**OCT**al) dönüştürür
OCT\$(sayı)

```
PRINT HEX$(2001) ' 7D1
PRINT OCT$(2001) ' 3721
```

INKEY\$

Klavyeden girilen karakteri okur. ESC tuşunun ASCII kodu 27 dir.

Quick-Basic Kursu

```
DO
  A$ = INKEY$
  LOCATE 5, 5: PRINT A$
LOOP UNTIL A$ = CHR$(27) 'Çıkmak için ESC tuşuna basın
```

2. örnek: Çok basit bir editör

```
DO
  A$ = INKEY$
  PRINT A$;
LOOP UNTIL A$ = CHR$(27)
```

LCASE\$, UCASE\$

LCASE\$: Verilen metni küçük harflere(Lower **CASE**) dönüştürür.

UCASE\$: Verilen metni büyük harflere(Upper **CASE**) dönüştürür. Her iki komutta da yalnızca İngilizce harfler dikkate alınır.

```
PRINT LCASE$("MERHABA Dünya")
PRINT LCASE$("Merhaba Dünya")
```

LEFT\$, RIGHT\$, MID\$

LEFT : Metnin belli sayıda sol(**LEFT**)undaki karakterleri alır

RIGHT: Metnin belli sayıda sağ(**RIGHT**)ındaki karakterleri alır

MID: Metnin belli sayıda orta(**MIDDLE**)sındaki karakterleri alır

```
PRINT LEFT$("QuickBasicV4.5",5)
PRINT LEFT$("QuickBasicV4.5",4)
PRINT MID$("QuickBasicV4.5",6,5) '6. karakterden itibaren 5 karakter
```

MID\$ in farklı kullanımı: Burada verilen metin içinde değişiklik yapar

```
A$ = "O adam 25 yaşında"
MID$(A$,3)="kadın"
PRINT A$
```

LEN

Verilen metnin kaç karakterden oluştuğunu verir

```
A$ = "MESUT AKCAN"
G% = LEN(A$)
PRINT G% '11 , çünkü boşluklar da sayılır
FOR N%=1 TO G%
  PRINT LEFT$(A$,N%)
NEXT
```

LTRIM\$, RTRIM\$

LTRIM\$: Metnin solundaki boşlukları siler

RTRIM\$: Metnin sağındaki boşlukları siler

Quick-Basic Kursu

```
PRINT LTRIM$(RTRIM$( "      MESUT AKCAN      " ) )
```

Bölüm 9: Rakamlarla ilgili işlemler

ABS

Verilen sayının mutlak(**ABS**olute) değerini verir. Sayı negatif ya da pozitif olsa da sonuç pozitif olur.

```
PRINT ABS (-127)
PRINT ABS (254)
```

INT

Sayının tam kısmını verir. Sayı negatif ise bir küçük sayıyı verir.

```
A = 12.86: B = - 12.86
PRINT INT(A) , INT(B)
```

FIX

Sayının sıfıra yakın olan tam kısmını verir.

```
A = 45.9: B = -45.1: C = -45.8
PRINT FIX(A) , FIX(B) , FIX(C)
```

RANDOMIZE

Rasgele sayı üreticisini hazırlar.

RND

0 ile 1 arasında rasgele bir sayı üretir. RND komutu program her çalıştırılışında aynı sayıları üretir. Bundan kurtulmak için aşağıda verilen örnekte olduğu gibi RND den önce RANDOMIZE TIMER çalıştırılır. TIMER o anki saate göre saniye cinsinden bir sayı üretir

```
DEFINT A-Z
RANDOMIZE TIMER: CLS
rs = RND * 10 + 1
PRINT "1 - 10 arası bir sayı girin ";
10 INPUT s
a = a + 1
IF s < 1 OR s > 10 THEN PRINT "!! hatalı sayı !!": GOTO 10
IF s <> rs THEN PRINT "Bilemediniz, tekrar deneyin...": GOTO 10
PRINT a; "denemede bildiniz..."
```

DÖNÜŞTÜRME FONKSİYONLARI

Bellekte değişken değeri olarak tutulan sayıyı farklı formatlara dönüştürmek için bazı fonksiyonlar kullanılır.

CDBL : Sayıyı DOUBLE formata dönüştürür.

Quick-Basic Kursu

```
A% = 15454 ' Bellekte 2 baytlık yer tutuyor
B# = CDBL(A%) ' Şimdi 8 baytlık Double formata dönüştü ve B değişkenine
aktarıldı
PRINT LEN(A%), LEN(B#)
```

CINT : Sayıyı INTEGER formata dönüştürür. Sayı küsurlu ise; küsur .4 den büyükse yukarı değilse aşağı yuvarlanır ve küsur atılır.

```
A = 1245.85 : PRINT CINT(A)
```

CLNG : Sayıyı LONG formata dönüştürür. Sayı küsurlu ise CINT gibi.

CSNG : Sayıyı SINGLE formata dönüştürür

MATEMATİKSEL FONKSİYONLAR

TAN : Radyan olarak verilen açının **TAN**jantını verir.

ATN : Radyan olarak verilen açının **ArkTAN**jantını verir.

```
PI = 4 * ATN(1)
PRINT PI
```

COS : Radyan olarak verilen açının **kosinüsünü** verir. Radyanı dereceye dönüştürmek için pi / 180 ile çarpın.

```
PI = 3.141593 : D = 30 ' derece
R = D * (PI / 180)
PRINT COS(R)
```

SIN : Radyan olarak verilen açının **sinüsünü** verir.

EXP : e sabitinin ($\sim = 2.718282$) üstünü alır

LOG : Sayıyı doğal **LOG**aritmasını hesaplar.

MOD : İki sayının bölümü sonucunda kalanı verir. A = 45682 : PRINT A MOD 4 ' A nın 4 e bölümünde kalan sayı.

SGN : Sayının işaretini belirtir. Sayı; 0 ise 0, pozitif ise 1, negatif ise -1 değerini verir. ? SGN (-28)

SQR : Sayıyı karekökünü verir. ? SQR(81)

LEN , ASC, VAL

[Önceki bölümde](#) bunları görmüştük

Bölüm 10: Dosya İşlemleri-1

Bilgileri değişkenlere aktararak bellekte tutabiliyoruz ve onlarla ilgili işlemler yapabiliyoruz. Belleğe aktarılacak ve işlenilecek bilgiler çok sayıda ve belli bir düzende ve değişme ihtimali olan bilgileri program kodları içine yazmak mantıklı olmayacaktır. Qbasic'in kullanacağı bellek de sınırlıdır. Örneğin beş bin öğrencisi bulunan bir okulda öğrencilerin kayıtlarının tutulacağını düşünün. Bu bilgiler klavyeden RAM belleğe aktarılabilse bile bilgisayarı kapattığımızda bilgiler kaybolacaktır. Bu yüzden kalıcı bellekte kalmasını istediğimiz bilgiler için dosyalar kullanılır.

Üç türlü dosya açma yöntemi vardır. Sıralı, Rasgele erişimli ve Binary

OPEN

Disk üzerinde dosya oluşturmak ve değişiklik yapmak için OPEN komutu kullanılır.

```
OPEN dosyaadı [FOR açma tipi] [ACCESS erişim tipi] [lock] AS [#]  
dosyanumarası [LEN=kayıt genişliği]
```

dosyaadı : "Sürücü : \ klasör\ dosyaadı" şeklinde açacağınız dosyayı tanımlayan metin
açma tipi : INPUT, OUTPUT, APPEND, RANDOM ya da BINARY tiplerinden biri
erişim tipi : Ağ ortamında dosyayı başkaları da açacaksa dosyaya erişim tipini belirleyebilirsiniz. READ(okuma), WRITE(yazma) veya READ WRITE (okuma ve yazma, RANDOM ve BINARY dosyalarda geçerlidir)
lock : dosya kilitleme
dosya numarası : açılan her dosyaya bir numara verilmelidir.
kayıt genişliği : Rasgele erişimli dosyalarda kayıt genişliğini belirtmek içindir.

Sıralı Erişimli Dosya Oluşturma ve Yazma

Çoğunlukla az sayıda veya belirli bir kayıt formatı olmayan metin içerikli dosya işlemleri için kullanılır. Örneğin DOSdaki EDIT programı bu tür dosyaları oluşturur veya açar. Sıralı erişimli dosya açmak istediğimizde bunu yalnızca OKUMA veya yalnızca YAZMA amaçlı olabilir. Dosya ne amaçla açılmışsa o amaçla kullanılır. Okumak için açtığımız dosyaya yazma yapamayız. Ancak dosya kapatıldıktan sonra yazma amaçlı olarak açıp yazdırabiliriz.

```
OPEN "c:\ORNEK\dosya.txt" FOR OUTPUT AS # 1
```

Bu satırı açıklayalım. **C:** sürücüsü içinde **ORNEK** klasörü içine **dosya.txt** adında yeni bir dosya oluşur. **OUTPUT** (YAZMA amaçlı) olarak dosya oluşur, yani okuma amaçlı olarak kullanılamaz.

Eğer **C:** sürücünüzde **ORNEK** klasörünüz yoksa "**Path Not Found(Yol bulunamadı)**" hatası oluşacaktır. Dos komut satırına çıkarak **MD C:\ORNEK** yazarak bu klasörü oluşturup tekrar çalıştırmayı deneyin.

Quick-Basic Kursu

AS # 1 ise açtığımız dosyanın dosya numarasının 1 olduğunu belirtiyoruz. Bu numara dosya yazma ve okuma için kullanılacaktır. Başka dosya açacağımızda, eğer dosya kapatılmadıysa aynı numarayı kullanamayız.

Evet bu satırı çalıştırdığımızda klasörümüz varsa içinde **dosya.txt** oluşacaktır.

Dikkat ! : Eğer dosya daha önceden varsa yeniden oluşacağı için önceki dosyanın varolan içeriği silinecektir

2. satırı ekleyelim

```
PRINT #1 , "Edep öğrenilmeden ilim öğrenilmez."
```

PRINT komutunu biliyorsunuz. Ekrana yazı yazmak için kullanılıyordu. Varsayılan yazma yeri ekran olduğu için yazma yeri yazılmadığında ekrana yazar. Bu satırda ekrana değilde #1 numara ile açılan dosyaya yazdırılacağını söylüyoruz. Gördüğünüz gibi burada dosya adını filan yazmamıza gerek yok, açık olan dosyanın numarası yeterli.

Şimdi kaydettiğiniz klasöre ulaşın. Windowsdaysanız dosya üzerinde çift tıklayarak, DOS da iseniz

EDIT C:\ORNEK\dosya.txt

yazarak dosya içeriğini görebilirsiniz. Windowsda dosya içeriğine bakarsanız Türkçe karakterlerin yerine farklı karakterler olduğunu göreceksiniz. Bu normaldir.

3. satırı ekleyelim.

```
PRINT "Dosya uzunluğu : " ; LOF(1)
```

LOF : Dosya numarasıyla belirtilen dosyanın uzunluğunu byte cinsinden verir

4. ve 5. satırı ekleyelim.

```
PRINT #1, "bir" , "iki" ; "yedi" ; TAB(30) ; "on"  
WRITE #1, "bir" , "iki" , "yedi", "on"
```

WRITE komutu; verileri, arasına virgül koyarak dosyaya(yazma amaçlı açılan) yazar. PRINT komutunun ekrandaki etkisi ne ise aynen o şekilde dosyaya yazar. PRINT #de veriler arasına virgül , noktalı virgül , TAB SPC USING komutları da eklenebilir.

6. satır

```
CLOSE #1
```

Quick-Basic Kursu

Numarası belirtilen dosyayı kapatır. Bu satırı yazmasak da program çalışır ve hata vermez. Çünkü program sonlandığında programın açtığı tüm açık dosyalar kapatılır. Ama siz yine de alışkanlık olarak dosya ile ilgili işlem bittiğinde kapatma komutunu yazmayı unutmayın. Numara belirtilmezse tüm dosyalar kapatılır. Açık olan birden fazla dosyayı kapatmak için CLOSE #1, #3,#6 gibi bir komut kullanılabilir. Kapatılan dosya üzerinde işlem yapılamaz.

Dikkat ! : Dosyalar açıkken elektrik kesintisi veya kilitleme durumunda açık olan dosyalarınıza zarar gelebilir.

Sıralı Erişimli Dosyadan Okuma

Aşağıdaki programı yazıp çalıştırın. Bu kodlar bir dosya oluşturacak

```
OPEN "c:\ORNEK\veriler.txt" FOR OUTPUT AS #1
WRITE #1, "Mesut", "Akcan", 1968, "Adana"
CLOSE #1
```

Üstteki programın oluşturduğu dosya üzerinde okuma işlemi yapacağız

```
OPEN "c:\ORNEK\veriler.txt" FOR INPUT AS #1
```

Dosyayı okuma amaçlı açacağımızı INPUT ekiyle belirtiyoruz. Eğer dosya yoksa **File Not Found**(Dosya bulunamadı) hatası oluşur

```
INPUT #1, A$, S$, DT%, DY$
```

INPUT komutunu hatırlarsanız daha önceden klavyeden veri girmek amacıyla kullanılmıştı. Şimdi ise dosyada kayıtlı olan veriler okunup belirtilen değişkenlere aktarılıyor.

```
CLS : PRINT A$, S$, DT%, DY$
```

Değişkenlere aktarılan değerler ekrana yazdırılıyor.

```
CLOSE
```

Dosya(lar) kapanıyor.

Satır Satır okuma

Daha çok düz metin içerikli dosyaları okumak için kullanılan bir yöntemdir. Dosya okuma amaçlı olarak açılır.

Kullanımı:

LINE INPUT # dosya numarası , String Değişken adı

Komut yürütüldüğünde dosyadaki ilk satırı komple okuyarak belirtilen değişkene aktarır.

Aşağıdaki örneği inceleyiniz.

```
CLS : dosya$ = "c:\bootlog.txt"
OPEN dosya$ FOR INPUT AS #1
```

Quick-Basic Kursu

```
PRINT dosya$; " dosyası içeriği:"
PRINT "-----"
DO WHILE NOT EOF(1)
    a = a + 1
    IF a > 21 THEN a = 0: SHELL "pause"
    LINE INPUT #1, K$
    PRINT K$
LOOP
```

Dosyaya Ekleme Yapma

```
OPEN "c:\ORNEK\veriler.txt" FOR APPEND AS #1
```

Daha önceden kayıtlı dosyaya yeni bilgiler eklemek gerekirse APPEND kullanılır. OUTPUT kullanılırsa eski bilgiler silinir.

```
WRITE # 1 , "Ahmet" , "Akgül" , 1972 , "Ankara"
CLOSE
```

Şimdi dosyadaki bilgileri okuyup ekrana yazalım

```
DEFINT A-Z
NO = FREEFILE: CLS
OPEN "c:\ORNEK\veriler.txt" FOR INPUT AS #NO
DO UNTIL EOF(NO)
    INPUT #NO, A$, S$, DT, DY$
    PRINT A$, S$, DT, DY$
LOOP
SEEK #NO, 1
'baştan tekrar okunuyor
PRINT "-----"
INPUT #NO, A$, S$, DT, DY$
PRINT A$, S$, DT, DY$
CLOSE #NO
```

FREEFILE dosya numarası olarak kullanılmayan bir numara verir. Özellikle çok sayıda dosya açtığınızda hangi numaraların kullanıldığını bulmak zor olabilir. Bu durumlarda işe yarar bir komut.

EOF : Dosya sonu demektir(**End of File**) Dosyadaki kayıtların bitip bitmediği bu komutla anlaşılır.

SEEK : Okumaya başlanacak konumu(byte) belirtir. Başlangıçta 1 dir. Bu komutu kullanmasaydık okuma konumu dosya sonuna geldiği için **Input past end of file**(Okuma dosya sonuna taşıtı) hatası çıkardı.

UYGULAMA

5 Personeli olan bir kurumda personel kayıtları bir dosyaya aktarılacak. Aktarılacak bilgiler: Personelin; adı, soyadı, doğum yeri, yaşı, maaşı. Bilgiler klavyeden aktarılacak

```
DEFINT A-Z: DEFSTR P: DIM m AS LONG
OPEN "c:\ornek\personel.txt" FOR OUTPUT AS #1
```


Quick-Basic Kursu

```
FOR n = 1 TO 10
CLS : PRINT STRING$(40, "-")
PRINT n; ". personelin bilgilerini giriniz"
PRINT STRING$(40, "-")
INPUT "Adı"; pa
INPUT "Soyadı"; ps
INPUT "Doğum yeri"; pd
INPUT "Yaşı"; y
INPUT "Maaşı"; m
WRITE #1, pa, ps, pd, y, m
NEXT
CLOSE
PRINT STRING$(40, "-")
PRINT "... işlem tamam ..."
```

PROBLEM : Bilgisayarınızdaki C: kök klasöründeki AUTOEXEC.BAT ve CONFIG.SYS dosya içeriğini ekrana yazan program yazınız. **Dikkat** : Bu dosyaların yedeğini almadan bunu denemeyin.

Bölüm 11: Dosya İşlemleri-2

Bu bölümde Random dosya oluşturma, dosyaya yazma ve okumayı öğreneceğiz.

Rasgele Erişimli(RANDOM) Dosya Oluşturma

Belli bir kayıt yapısı olan dosya türleri için uygundur. Dosyadaki tüm kaydı birden okuma yada yazma gibi bir problem olmadığından dolayı bellek için de uygundur. Çok sayıda kayıt işlemi yapılabilir. Sınır, bilgisayarınızın boş harddisk alanı kadardır. İstenilen kayıt numarası verilerek sadece o kayıt okunabilir veya yazılabilir. Böylece hızlı bir şekilde verilere ulaşılabilir.

Random dosyada kayıtlar için bir kayıt yapısı oluşturulmalıdır. Type End Type ile kayıt yapısı belirlenir

```
TYPE kayıt
    no AS INTEGER
    adi AS STRING * 10
    soyadi AS STRING * 10
END TYPE
```

Her bir kayıt için kullanılacak değişkenler belirtildi. Gördüğümüz gibi STRING * 10 diye yazarak string değişkenin bellekte kaç karakter kaplayacağını da belirtiyoruz. String harici değişken türlerinin standart genişlikleri olduğu için belirtilmez

Byte cinsinden genişlikler	
STRING * n	n
INTEGER	2
LONG	4
SINGLE	4
DOUBLE	8

Tanımladığımız kayıt tipine uygun bir değişken için bellekte yer açalım. Sonraki satır random dosya açmak için

```
DIM ogr AS kayıt
OPEN "c:\ornek\ogrenci.txt" FOR RANDOM AS #1 LEN = LEN(ogr)
```

Rasgele erişimli dosya oluşturmak için RANDOM eki kullanılır. Dosya yoksa yeniden oluşturulur. Varsa herhangi bir problem oluşmaz, dosya silinip yeniden oluşmaz. Açık dosyayı kapatmaya gerek olmadan YAZMA ve OKUMA amaçlı olarak kullanılabilir. Farklı olarak LEN komutu kullanılıyor. Burada her bir kaydın sabit diskte ne kadarlık yer kaplayacağını belirliyoruz. LEN = LEN(ogr) ile ogr değişkeninin bellekte ne kadar yer kapladığını hesaplayıp ona eşitlemesini sağlıyoruz. İstersek buraya LEN=22 de yazabilirdik. Çünkü no 2, adi 10, soyadi 10 byte'lık yer kapladığı için toplam 22 olacaktı.

Rasgele Erişimli Dosyaya Yazma

Üstteki kodları yazıp çalıştırdığınızda boş bir dosya oluşur. Şimdi açık olan dosyaya kayıt yapacağız. Önce bellekte açtığımız **ogr** değişkenini dolduralım. **ogr** değişkeninin tipi daha önce TYPE komutuyla belirlenmişti. Değişken adı yanına bir nokta koyup tipe uygun değişken adlarından(alt değişken) birini yazıyoruz. **ogr.adi** gibi. Tüm alt değişkenleri doldurmak şart değildir. Doldurulmazsa; sayılar 0, stringler boş olarak diske yazılır. Doldurulmaması, diskte kapladığı alını değiştirmez.

```
ogr.no = 5124
ogr.adi = "Ahmet"
ogr.soyadi = "Akgül"
PUT #1, 1, ogr
```

Sıralı erişimli dosyada YAZMAK için PRINT ya da WRITE kullanılıyordu. Rasgele erişimli dosyada ise yazmak için PUT komutu kullanılır.

PUT # dosya numarası , kayıt no , değişken

şeklinde kullanılır. Kayıt numarası belirtilmezse; herhangi bir okuma ya da yazma yapılmamışsa, ilk kayıt olarak yazılır. Okuma ya da yazma yapılmışsa bir sonraki kayda yazılır. Numara belirtilirse o kayda yazılır.

Üstteki kodları yazıp çalıştırın. **c:\ornek\ogrenci.txt** dosyası oluşacak ve içeriğine ilk kaydı girecek. Bir hex editör ya da viewer ile o dosyaya bakacak olursak

yukarıdaki kodları görürüz.. Sağ bölme kodların ASCII karşılığı, sol bölme ise 16'lı sayı sistemine(HEX) göre her bir byte'ın değeri. İlk 2 karakter(byte) integer değişken için ayrılan alan. Burada 04 ve 14 var. Sayı değerini hesaplamak için $256 * \&H14 + \&H04$ kodlarını kullanabiliriz. Sonuç 5124 edecektir. Yani bizim kaydettiğimiz sayı. Integer değişkenin alabileceği maksimum değer 32767 dir. Öğrenci numarası bu sayıdan daha büyük olma ihtimali varsa; no değişkenini daha büyük sayıları tutabilen LONG olarak tanımlanmalıdır. Sonraki 10 karakter **ogr.adi** için. Gördüğümüz gibi ad 5 karakterli olduğu halde 10 karakterlik alan ayrılmış. Aynı **ogr.soyadi** 'nda olduğu gibi. Böylece, Tanımlanan TYPE e göre bir seferlik kayıta 22 byte'lık alan harcanmış olur. Her kayıta bu 22'nin katları olarak artar.

```
ogr.no = 625
ogr.adi = "Mehmet"
ogr.soyadi = "Ateş"
PUT #1, 3, ogr
```

Oluşan dosyayı incelediğimizde üstteki kodları görürüz. Burada farklı olarak PUT komutunu verirken kayıt numarasını da verdik. Ama 2. kaydı yaptığımız halde 3. kayıt alanına kayıt yaptık. Yani 2. kayıt alanını atlamış olduk. Atlamış olduğumuz alanlar da 0 değerleriyle dolduruldu(22 adet).

Rasgele erişimli demenin nedeni de budur. Kayıt için bir sıra takip etmek şart değildir. Rasgele bir numaraya bile kayıt yapılabilir. Bundan sonraki PUT komutunda kayıt numarası belirtilmezse 4 numaralı kayıt olur. Ama biz şimdi bu boş kalan 2. kayıt alanını dolduralım.

Quick-Basic Kursu

```
ogr.no = 5
ogr.adi = "Selami"
ogr.soyadi = "Güneş"
PUT #1, 2, ogr
```

[illegible]

Rasgele Erişimli Dosyadan Okuma

Dosyadan okuma için GET komutu kullanılır. Üstteki kodların devamına aşağıdaki kodları ekleyin. Dosyayı kapatmamız ve yeniden açmamız gerekmiyor.

```
GET #1, 3, ogr
CLS : PRINT "Öğrenci,"
PRINT "NO : "; ogr.no
PRINT "ADI : "; ogr.adi
PRINT "SOYADI : "; ogr.soyadi
```

İlkönce dosyadan GET komutuyla 3. kayıttaki bilgileri bellekte oluşturulan **ogr** değişkenine aktarıyoruz. Sonra **ogr** değişkeninin alt değişkenlerini PRINT ile yazdırıyoruz. Kayıt numarasını vermeseydik yine aynı sonucu alırdık. Çünkü bir önceki işlemde 2 numaralı kayda yazdırma yapmıştık. İşlem yapıldıktan sonra bir sonraki kayda atlanacağından 3 numaralı kayıt çağrılır. Şimdi aşağıdaki kodları ekleyip çalıştırın.

```
ku = LOF(1) \ LEN(ogr) : CLS
FOR n = 1 TO ku
    GET #1, n, ogr
    PRINT n; ". kayıttaki öğrencinin,"
    PRINT "No: "; ogr.no
    PRINT "Adı: "; ogr.adi
    PRINT "Soyadı : "; ogr.soyadi
    PRINT STRING$(30, "-")
NEXT
CLOSE : END
```

LOF(1) ile dosyanın boyutunu byte cinsinden alıyoruz, her bir kayıt uzunluğuna bölüyoruz. Kayıt uzunluğu 22 olduğu için LEN(ogr) yerine 22 de yazabilirsiniz. Burada dikkati çeken / değilde \ kullanılması. \ kalansız bölmeler için kullanılır. Normalde kalan olmaması gerekir. Dosya uzunluğunu kayıt uzunluğuna böldüğümüzde kayıt sayısı çıkar. For döngüsü ile tüm kayıtlar okunur ve ekrana basılır.

Tüm kodlar aşağıda

```
TYPE kayit
    no AS INTEGER
    adi AS STRING * 10
    soyadi AS STRING * 10
END TYPE
```

Quick-Basic Kursu

```
DIM ogr AS kayıt

OPEN "c:\ornek\ogrenci.txt" FOR RANDOM AS #1 LEN = LEN(ogr)

ogr.no = 5124
ogr.adi = "Ahmet"
ogr.soyadi = "Akgül"
PUT #1, 1, ogr

ogr.no = 625
ogr.adi = "Mehmet"
ogr.soyadi = "Ateş"
PUT #1, 3, ogr

ogr.no = 5
ogr.adi = "Selami"
ogr.soyadi = "Güneş"
PUT #1, 2, ogr

GET #1, , ogr
CLS : PRINT "Öğrenci,"
PRINT "NO : "; ogr.no
PRINT "ADI : "; ogr.adi
PRINT "SOYADI : "; ogr.soyadi

ku = LOF(1) \ 22: CLS
FOR n = 1 TO ku
    GET #1, n, ogr
    PRINT n; ". kayıttaki öğrencinin,"
    PRINT "No: "; ogr.no
    PRINT "Adı: "; ogr.adi
    PRINT "Soyadı : "; ogr.soyadi
    PRINT STRING$(30, "-")
NEXT
CLOSE : END
```

Bölüm 12: Dosya işlemleri-3

Yalnız metin içermeyen dosyalar vardır. Örneğin; resim(BMP,JPG,GIF ...) ses(WAV,MP3 ...), video(AVI, MOV, MPG ...) gibi dosyalar. Bu tür dosyalar içinde metin aranmaz ve metinle ilgili işlemler de yapılmaz. Ama örneğin bir gif dosya yapısını öğrendiniz ve dosya içindeki resmi ekrana basmak istiyorsunuz. Bunun için dosyayı binary olarak açıp içindeki kodları resim kodlarına dönüştürmelisiniz. Çalışma sistemi RANDOM dosya gibidir ancak önceden bir kayıt yapısı belirleme gerekliliği yoktur. GET ve PUT ile dosyanın istenilen kısmını okuyabilir, değişiklik yapabilirsiniz.

Binary(ikili) Dosya Oluşturma ve Yazma

```
OPEN "c:\ornek\ikili.bin" FOR BINARY AS #1
```

Random dosyada olduğu gibi dosya yoksa oluşturulur. Varsa sorun çıkmaz. Üstteki kodu denerseniz, yok ise boş bir dosya oluşur.

```
A$ = "Mesut"  
PUT #1, , A$
```

A\$ değişkeni içeriğini PUT ile dosyaya yazdırdık. Kayıt numarası belirtmedik. Herhangi bir okuma veya yazma yapılmadığı için ilk 5 byte yazdırılır.

```
A$ = CHR$(13) + CHR$(10) + "Akcan"  
PUT #1, , A$
```

İkinci satır oluşturuluyor. Sıralı erişimli dosyada satır atlamak için ikinci bir PRINT komutu kullanılıyordu ama burada byte byte işlem yapıldığı için satır atlama kodlarını da bizim girmemiz gerekir. CHR\$(13) + CHR\$(10) dan oluşan iki karakter sonraki satıra atlamayı sağlar. ENTER tuşunun yaptığını yapar

Üstteki kodları deneyip dosya içeriğini inceleyin sonra aşağıdaki kodları deneyin. Burada kayıt yeri olarak 2 yazılı. Değişken içeriği dosyanın 2. baytından başlayarak doldurulur. Tabii ki o kısımdaki eski bilgiler değişmiş olur.

```
PUT #1, 2, A$  
CLOSE
```

Binary Dosya Okuma

Dosyadan byte byte bilgi okumak için GET komutu kullanılır.
GET # dosyano, bytekonum, değişken
şeklinde kullanılır. Konum belirtilmezse PUT da olduğu gibidir.

Binary dosyalarda çalışırken byte olarak okuduğumuz verilerden bir kısmı sayı içeriyor olabilir. Örneğin bir resim dosyasında ilk 3 bayt resim türünü,sonraki 2 byte renk derinliğini, sonraki 15 byte tanımlama vs.. olabilir. Bunları tek tek GET komutuyla diskten okutmak yerine gerekli bilgi

Quick-Basic Kursu

bir seferde bir STRING deęişkene atanır. MID\$ ile gerekli bytelar deęişkenden alınır. Fakat sayısal deęerler byte'a dönüştüğünde farklı string deęerlere sahip olur. Bir önceki derste dosyayı HexViewer ile incelediğimizde görmüştük. Bu tür bilgileri dönüştürmek için QBASIC içindeki hazır fonksiyonlar kullanılır.

STRING'İ SAYIYA

CVI : 2 byte'lık String'i Integer sayı deęerine dönüştürür. Örneğin

PRINT CVI("[x"]) --> Sonuç : 30811

CVS : 4 byte'lık String'i Single sayı deęerine dönüştürür.

CVL : 4 byte'lık String'i Long sayı deęerine dönüştürür.

CVD : 8 byte'lık String'i Double sayı deęerine dönüştürür.

SAYIYI STRING'E

MKI\$: Integer sayı deęerini String'e dönüştürür. Örneğin

PRINT MKI\$(30811) --> Sonuç : [x

MKS\$: Single sayı deęerini String'e dönüştürür.

MKL\$: Long sayı deęerini String'e dönüştürür.

MKD\$: Double sayı deęerini String'e dönüştürür.

Okuma ya da yazma konumu

LOC fonksiyonu ile yazılacak ya da okunacak kayıt numarası öğrenilir. Binary dosyada byte olarak konumu, random dosyada ise kayıt numarasını verir.

```
OPEN "c:\ornek\test.txt" FOR BINARY AS #1
a$ = MKI$(5214)
PUT #1, 8, a$
PRINT LOC(1)
CLOSE
```

UYGULAMA

BMP formatında bir resim bulun ve **c:\ornek** klasörüne kopyalayın. Dosyanın adını **resim.bmp** olarak deęiştirin. Aşağıdaki kodları yazıp çalıştırın. Bu uygulama resim dosyanız hakkında bazı bilgileri ekrana yazacaktır.

Quick-Basic Kursu

```
CLS : dosya$ = "c:\ornek\resim.bmp"
OPEN dosya$ FOR BINARY AS #1
'PRINT "Dosya boyutu : "; LOF(1)
baslik$ = SPACE$(14): boyut$ = SPACE$(4)
GET #1, 1, baslik$: GET #1, 15, boyut$
bmptur = CVI(boyut$)
IF bmptur = 40 THEN ' 12 ise OS/2 BMP
    baslikbilgi$ = SPACE$(40)
    GET #1, 15, baslikbilgi$
    renkd = CVI(MID$(baslikbilgi$, 15, 4))
ELSE
    PRINT "Windows BMP resim dosyası değil": END
END IF
dosyag = CVL(MID$(baslik$, 3, 4))
PRINT "Dosya Genişliği:"; dosyag; "byte"
PRINT "Renk derinliği :"; renkd; "bit"
baslgen& = CVL(MID$(baslikbilgi$, 1, 4))
resGenislik = CVL(MID$(baslikbilgi$, 5, 4))
PRINT "Genişlik: "; resGenislik
resYukseklk = CVL(MID$(baslikbilgi$, 9, 4))
PRINT "Yükseklik: "; resYukseklk
PRINT
IF bmptur = 40 THEN
    PRINT "Sıkıştırma: ";
    skstrma = CVL(MID$(baslikbilgi$, 17, 4))
    IF skstrma = 0 THEN PRINT "Yok"
    IF skstrma = 1 THEN PRINT "Run Length - 8
Bits"
    IF skstrma = 2 THEN PRINT "Run Length - 4
Bits"
    kullRenk = CVL(MID$(baslikbilgi$, 33, 4))
    PRINT "Kullanılan renk sayısı:"; kullRenk
END IF
CLOSE
```

Dosyayı Silme

Dos komut satırında bir dosyayı silmek için DEL komutu kullanılır. Ama Qbasic içinden dosyayı silmek istersek ne olacak?

Qbasic içinde de dosya silmek için bir komut var ama DEL değil KILL (öldürmek) Kullanımı:

KILL "sürücü:\dosyayolu\dosyaadı"

```
KILL "c:\ornek\resim.bmp"
```

Dikkat ! Bu şekilde sildiğiniz dosyaya özel programlar kullanmadan ulaşmanız mümkün değildir.

Dosya adını değiştirme

NAME komutu ile dosya adı değiştirilebilir.

```
NAME "test.txt" AS "veri.dat"
```


Konum deęiřtirme

Dos komut satırında CD ile bulunduęunuz klasörü öğrenebilirsiniz. Bu klasörden başka klasöre geçiř yapmak için Qbasicde CHDIR komutu kullanılır. Sürücü ve yol belirtilir

```
CHDIR "D:\PROGRAMLAR"
```

Klasördeki Dosyaların Listesi

Dos'da belirtilen klasördeki dosyaların listesini DIR ile alırız. Qbasicde ise FILES komutu kullanılır.

FILES dosya türü tanımı

```
FILES ' bulunduęun klasörde tüm dosyalar
FILES "*.bmp" ' BMP resim dosyaları
FILES "c:\ornek\*.txt" ' belirtilen klasörde txt dosyalar
FILES "?.*" ' Dosya adı tek harfli, uzantısı B ile başlayan dosyalar.
```

Yeni Klasör Oluřturma

Dos'da MKDIR ya da MD komutu ile klasör oluřturulur. Qbasicde ise MKDIR ile

```
MKDIR "C:\ORNEK"
```

Eęer klasör yoksa yeni bir tane oluřur. Klasör varsa **Path/File access error** hatası belirir.

Klasörü Silme

Dos'da RMDIR ya da RD ile klasör silinir. Qbasicde ise RMDIR ile

```
RMDIR "C:\ORNEK"
```

Klasör içerięi boş deęilse **Path/File access error** hatası belirir. Klasörü silmek için önce içindekileri silmek gerekir.

```
KILL "C:\ORNEK\*.*"
RMDIR "C:\ORNEK"
```

Dosya Kopyalama

Qbasicde dosya kopyalamak için bir komut yoktur. Kısayoldan SHELL komutu ile tüm dos komutlarını kullanabiliriz ama Shell komutu kullanmadan dosyayı nasıl kopyalarız? Örnek kodlar ařaęıda. Çok büyük dosyalarda bu kodlar sorun çıkarır. Büyük dosyalar için ek kodlar gerekir. Bu sorunu çözmek de size ödev olsun :)

```
OPEN "c:\ornek\ikili.bin" FOR BINARY AS #1
g& = LOF(1) : a$ = SPACE$(g&)
```

Quick-Basic Kursu

```
GET #1, , a$  
CLOSE  
  
OPEN "c:\ornek\kopya.bin" FOR BINARY AS #1  
PUT #1, , a$  
CLOSE  
PRINT "Dosya kopyalandı"
```