



Eskişehir Osmangazi Üniversitesi
Mühendislik Mimarlık Fakültesi
İnşaat Mühendisliği Bölümü

Visual Basic 6

Görsel Programlama Dili

Sunu Ders Notları

The screenshot shows the Visual Basic 6 IDE. The top window is titled 'Project1 - Form1 (Code)' and displays the following code:

```
'Option Explicit

Private Sub Form_Click()
    a = 2
    b = 3
    c = 4
    d = 5
    Print a * b / c * d
    Print a * b / (c * d)
    Print a * b / c / d
End Sub
```

The bottom window is titled 'Form1' and displays the output of the code:

```
7.5
0.3
0.3
```

Ahmet TOPÇU
Eskişehir, 1999

Temel Bilgiler

Hemen her ülkenin konuştuğu /yazdığı bir dil vardır. İnsanlar aralarında anlaşabilmek için ortak bir dil bilmek zorundadır. Bilgisayarın da anlayabildiği birden çok dil vardır. BASIC, FORTRAN, Pascal, C gibi. Visual Basic BASIC kodlama dilini, Delphi Pascal kodlama dilini kullanır.

Bilgisayarda program yazabilmek için onun anladığı dillerden bir veya birkaçını öğrenmek zorundayız. Konuştuğumuz dil harfler, rakamlar, kelimeler, cümleler içerir. Konuşulan/yazılan dillerde yüz binlerce kelime, yüzlerce cümle yapısı vardır. Ancak , günlük yaşamda üç beş bin kelime ile yetiniriz. Çok sık kullandığımız, az kullandığımız, çok nadir kullandığımız, hayatımız boyunca hiç kullanmadığımız ve hatta anlamını bilmediğimiz(sözlükte var olan) kelimeler vardır.

Bilgisayar dilleri de buna benzerlikler içerir. Bir kere, bilgisayarda kullanılabilen her dili bilmemiz gerekmez. Öğrenmekte olduğumuz dilin de her kelimesini, her kuralını bilmeğe gerek yoktur. Bilmediklerimizi gereğinde öğrenebiliriz.

Dersin bu bölümünde Visual Basic 6 programlama dilinin özünü örneklerle öğrenmeğe çalışacağız.

İçerik kısaca:

- Sabitler, değişkenler, operatörler, veri tipleri, değişken tanımlama
- Aritmetik ifadeler
- Deyimler(döngüler, şartlı ifadeler, ...)
- Standart fonksiyonlar, türetilen fonksiyonlar
- Alt programlar
- Önemli fonksiyonlar
- Grafik

Harfler:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

Rakamlar:

0123456789

Özel amaçlı işaretler:

+ - * / = < > ^ . , ; “ ‘ \$ % # () []

NOT:

Çç İı Ğğ Öö Üü Şş gibi Türkçe ye özgü karakterler düz yazı dışında kullanılamaz.

α β γ π gibi, matematikte alışık olduğumuz harfler kullanılamaz.

IV, LC gibi Romen rakamları kullanılamaz.

Bu karakterlerin geçtiği ifadelerde yukarıdaki harf ve rakamlarda en anlamlıları veya kombinasyonları kullanılır: α yerine **A** veya **Alfa**, β yerine **B** veya **Beta** gibi.

Sabitler:

Program yazılırken içeriği bilinen ve program çalışırken değişmeyen büyüklüklerdir.

Matematik

VB karşılığı

2467
-35789
8,69
-42,85x10⁶
127,17x10⁻⁵
10⁶⁹
25 Mart 2003
Doğru
Yanlış

2467
-35789
8.69
-42.85E6
127.17E-5
1.0E69
"25 Mart 2003"
True
False

Sayılar da virgül yerine nokta kullanılır

10 sayısının kuvveti, E harfi (Exponent anlamında) kullanılarak yazılır

Yazı tipi (alfa sayısal) sabitler çift tırnak içinde yazılır

Mantıksal sabitler için **True** ve **False** kelimeleri kullanılır

The screenshot shows the Visual Basic 6 IDE. The top window is 'Project1 - Form1 (Code)' with the 'Form' dropdown and 'Click' event selected. The code editor contains the following code:

```
'Option Explicit  
  
Private Sub Form_Click()  
Print 2467, -35789, 8.69  
Print -42850000#, 0.0012717, 1E+69  
Print "25 Mart 2003"  
Print True, False  
End Sub
```

The bottom window is 'Form1' showing the output of the program:

```
2467      -35789      8.69  
-42850000 0.0012717  1E+69  
25 Mart 2003  
True      False
```

Program kodu **Form1** in **Click** olayına yazıldı

Print deyimi veriyi form üzerine yazar

Program **RUN** veya **F5** tuşu ile çalıştırıldıktan sonra Form tıklanırsa yukarıdaki program kodu işletilerek sonuçlar Form üzerine yazılır

Değişkenler:

Program yazılırken içeriği bilinmeyen ve program çalışırken içeriği hesaplanması veya kullanıcı tarafından girilmesi gereken büyüklüklerdir. Harf ve/veya rakamlar kullanılarak isimlendirilir.

VB değişkeni	tipik kullanım örnek
x	x=26
x3	x3= -127.68
Sayı1	sayı1=154.09E5
maas	maas=432000000.00
adi\$	adi\$="Cenk"
Soyadi\$	Soyadi\$="TAŞÇIOĞLU"
b1	b1=True
b2	b2=False

Tam sayı değişkeni

Ondalık sayı değişkeni

Alfa sayısal değişken(String)

Mantıksal değişkenler

```
Project1 - Form1 (Code)
Form Click
'Option Explicit
Private Sub Form_Click()
x3 = -127.68
sayı1 = 15409000#
adi$ = "Cenk"
soyadi$ = "TAŞÇIOĞLU"
b1 = True
b2 = False
Print x3, sayı1
Print adi$, soyadi$
Print b1, b2
End Sub
```

Form1	
-127.68	15409000
Cenk	TAŞÇIOĞLU
True	False

Değişken adı:

- Mutlaka bir harf ile başlar
- Özel amaçlı nokta, virgül, parantez, artı, yıldız, boşluk gibi işaretleri içeremez.
- İndis içeremez.
- En fazla 255 harf ve/veya rakamdan oluşur.
- VB de kullanılan özel kelimeler değişken olarak kullanılamaz(örneğin **End**)

Aritmetik operatörler:

Toplama, çıkarma, çarpma gibi aritmetik işlemlerde kullanılan işaretlerdir.

İşlem	VB karşılığı
Toplama	+
Çıkarma	-
Çarpma	*
Bölme	/
Üs alma	^

Matematik	VB karşılığı
$a+3,7$	$a+3.7$
$x-7$	$x-7$
$3t$	$3*t$
$\frac{10,27}{k}$	$10.27/k$
$25,54^{0,632}$	$25.54^{0.632}$

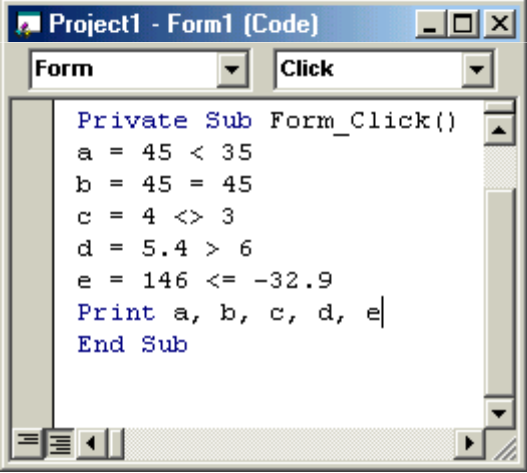
Karşılaştırma operatörleri:

Bir ifadenin bir diğer ifade ile karşılaştırılmasında kullanılan işaretlerdir. Aranana cevap : Biri diğerinden büyük mü? küçük mü? büyük veya eşit mi? farklı mı? Eşit mi? türündendir. Sonuç mantıksal, yani **True** (doğru) veya **False** (yanlış) olur.

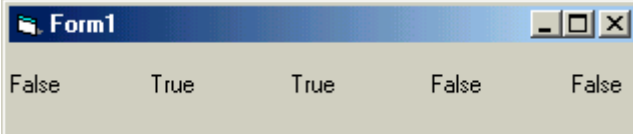
Matematik işlem	VB karşılığı
a=b	a=b
a>b	a>b
a<b	a<b
a≥b	a>=b
a≤b	a<=b
a≠b	a<>b

Karşılaştırmanın sonucu doğru ise **True**, yanlış ise **False** mantıksal değerini alır:

a=45 < 35	a=False olur
b= 45 = 45	b=True olur
c = 4 <> 3	c=True olur
d=5.4>6	d=False olur
e= 146 <= -32.9	e=False olur



```
Project1 - Form1 (Code)
Form Click
Private Sub Form_Click()
a = 45 < 35
b = 45 = 45
c = 4 <> 3
d = 5.4 > 6
e = 146 <= -32.9
Print a, b, c, d, e
End Sub
```



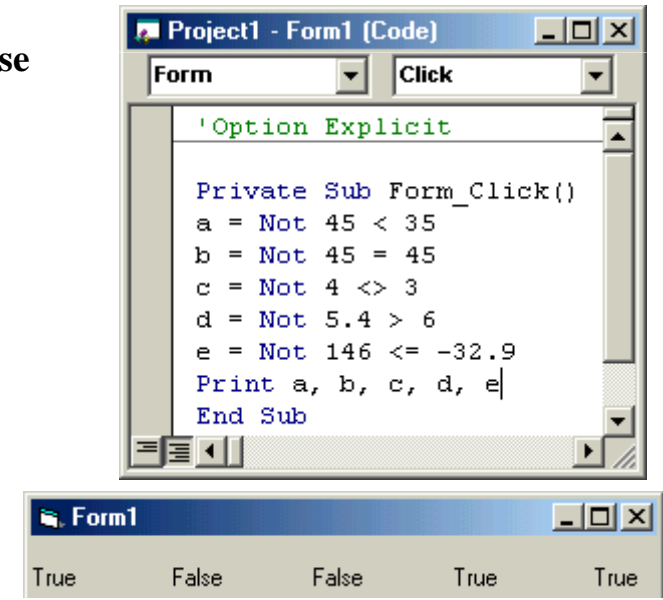
Form1
False True True False False

Mantıksal operatörler:

Mantıksal bir ifadeyi bir diğer mantıksal ifadeye bağlayan veya mantıksal bir ifadeyi tersine çeviren operatörlerdir. Sonuç gene mantıksal, yani **True** veya **False** olur. **Not** (değil), **And** (ve), **Or** (veya), **Xor** (tekli veya) kelimeleri matematikteki, \bar{a} , \wedge , \vee , \oplus mantıksal operatörlerin karşılığıdır.

Matematik	VB karşılığı	açıklama
\bar{a}	Not a	a False ise sonuç True ; True ise False olur
$a \wedge b$	a and b	Hem a ve hem de b True ise sonuç True ; aksi halde False olur
$a \vee b$	a or b	a veya b True ise sonuç True ; aksi halde False olur
$a \oplus b$	a xor b	a ve b False ise sonuç False a ve b True ise sonuç False a ve b de biri True diğeri False ise sonuç True olur

a=Not 45 < 35 → a=True olur
b=Not 45 = 45 → b=False olur
c=Not 4 <> 3 → c=False olur
d=Not 5.4>6 → d=True olur
e= Not 146 <= -32.9 → e=True olur



a= 45 < 35 and 45=45 a=False olur

b= 4<>3 and 5.4>5 b=True olur

c=146<=-32.9 and 0< -264.6 c=False olur

```

Project1 - Form1 (Code)
Form Click
Option Explicit
Private Sub Form_Click()
a = 45 < 35 And 45 = 45
b = 4 <> 3 And 5.4 > 5
c = 146 <= -32.9 And 0 < -264.6
Print a, b, c
End Sub
Form1
False True False

```

a= 45 < 35 xor 45=45 a=True olur

b= 4<>3 xor 5.4>5 b=False olur

c=146<=-32.9 xor 0<-264.6 c=False olur

```

Project1 - Form1 (Code)
Form Click
Option Explicit
Private Sub Form_Click()
a = 45 < 35 Xor 45 = 45
b = 4 <> 3 Xor 5.4 > 5
c = 146 <= -32.9 Xor 0 < -264.6
Print a, b, c
End Sub
Form1
True False False

```

Xor nadiren kullanılan(bit işlem, şifreleme gibi) bir operatördür. İlk aşamada öğrenmek için zaman harcamayınız!

Veri tipleri:

Matematikte; yerine göre; tam sayı, ondalık sayı gibi değişik tip veri kullanırız. Önceki konulardan da anlaşılacağı gibi, VB de de farklı veri tipleri vardır. Farklı veri tipi kullanmanın gerekçesi vardır: Belleği ekonomik kullanmak, elden geldiğince hızlı program yazmak gibi.

Bilgisayarda en küçük bellek birimi **Bit** adını alır. Bir bit sadece 0 veya 1 değerini alabilir. Bir **Byte** sekiz bitten oluşur ve 0 ile 255 arasında tam sayı değerler alabilir. Programcının genelde kullanabileceği en küçük bellek birimi **Byte** tır*. Sayısal hesap için Byte da, çok açıktır ki, yetersizdir. Bu nedenle farklı veriler için farklı bellek birimleri kullanılır. VB de kullanılan veri tipleri tabloda verilmiştir.

Veri tipi	Kullandığı Byte	İçerik aralığı	açıklama
Byte	1	0 den 255 kadar	Tam sayı
Integer	2	-32768 den 32767 kadar	Tam sayı
Long	4	-2 147 483 648 den 2 147 483 647 kadar	Tam sayı
Single	4	$\pm 3.402823E38$ den $\pm 1.401298E-45$ kadar	Ondalık sayı
Double	8	$\pm 1.79769313486232E308$ den $\pm 4.94065645841247E-324$ kadar	Ondalık sayı
Boolean	2	True veya False	Mantıksal
String	10 + karakter sayısı	Sadece harflerden, rakamlardan ve özel işaretlerden oluşan dizi depolanabilir	Alfa sayısal
Decimal	14	+/-79 228 162 514 264 337 593 543 950 335	Tam/ondalık sayı
Variant	16	Herhangi bir sayısal değer içerebilir, ancak aralığı Double ile aynı	Herhangi bir tip
Date	8	1 Ocak 100 gününden 31 Aralık 9999 gününe kadar	Tarih için
Currency	8	+/- 922 337 203 685 477.5807 (noktadan sonra sadece 4 hane)	Para hesabı için

•Bellek ve veri depolama birimlerinin(disket, hardisk, CD-ROM, ...) kapasitesi Byte in katları ile anılır.

kilo byte : 1 kB=1024 Byte

Mega Byte: 1 MB=1000 kB

Giga Byte : 1 GB=1000 MB

Aritmetik ifadeler:

Sabit ve deęişkenlerin aritmetik operatörler (toplama, çarpma, üs, ...) ve parantez kullanılarak bağlanmaları ile bir aritmetik ifade oluşur.

Matematik	VB karşılığı	Açıklama
$-2r+3,26$	$-2*r+3.26$	çarpma işareti daima yazılır
$ax^{2-b} - 267,009$	$a*x^{(2-b)}-267.009$	ilk önce parantez içi hesaplanır
$\left[\frac{b(a+b)^2}{3,8} - c \right]^5$	$(b*(a+b)^2/3.8-c)^5$	Sadece yuvarlak parantez kullanılır
$\sqrt[c]{a^b}$	$a^{(b/c)}$	
$-a^{\frac{1}{3}}$	$-a^{(1/3)}$	
$(-a)^{\frac{1}{3}}$	YOK!	Üs ondalık bir sayı ise, taban pozitif olmalı! (kompleks sayı da yok!)
$(-a)^3$	$(-a)^3$	Üs tam sayı ise taban negatif olabilir
$\sqrt{a_{ij}}$	$aij^{0.5}$	İndisli deęişken yoktur!.

İŞLEM SIRASI KURALI

Önce üs alma, sonra çarpma ve bölmeler ve daha sonra da toplama çıkarmalar yapılır. Çarpma ve bölme (veya toplama ve çıkarma) gibi eşdeğer işlemlerde hesap soldan sağa doğru *sıra ile* yapılır. Parantezli ifadeler (varsa), yukarıdaki kurala uygun olarak ve en içteki parantezli ifadeden başlanarak, *ilk önce* hesaplanır.

Tabloda verilen örnekler işlem sırasının ne denli önemli olduğunu vurgulamaktadır. her birinin özenle incelenmesi ve kavranması gerekir. Kararsızlık halinde yukarıdaki işlem sırası kuralının yeniden okunması ve program yazarak sonucun incelenmesi gerekir.

İşlem sırasını önemsemeyen programcının programı hemen hiçbir zaman doğru çalışmayacak, işin kötüsü programını binlerce defa gözden geçirse dahi, hatayı hiçbir zaman göremeyecektir.

Tablodaki ilk üç ifadenin program kodu ve çıktıları örnek olarak aşağıda verilmiştir. Sonuçları hesap makinesi ile kontrol ediniz.

```

Project1 - Form1 (Code)
Form Click
'Option Explicit

Private Sub Form_Click()
a = 2
b = 3
c = 4
d = 5
Print a * b / c * d
Print a * b / (c * d)
Print a * b / c / d
End Sub

```

```

Form1
7.5
0.3
0.3

```

VB ifade	Matematik karşılığı	açıklama
$a*b/c*d$	$\frac{ab}{c}d$	$\frac{ab}{cd}$ değil!
$a*b/(c*d)$	$\frac{ab}{cd}$	
$a*b/c/d$	$\frac{ab}{cd}$	
$a-b/c+2$	$a - \frac{b}{c} + 2$	$\frac{a-b}{c+2}$ değil!
$(a-b)/c+2$	$\frac{a-b}{c} + 2$	$\frac{a-b}{c+2}$ değil!
$(a-b)/(c+2)$	$\frac{a-b}{c+2}$	
$a/b/c$	$\frac{a}{\frac{b}{c}} = \frac{a}{bc}$	$\frac{ab}{c}$ değil!
a^b^c	$(a^b)^c = a^{bc}$	a^{b^c} değil
$a^(b^c)$	a^{b^c}	
$a*c/b^3$	$\frac{ac}{b^3}$	$(\frac{ac}{b})^3$ değil!
$(a*c/b)^3$	$(\frac{ac}{b})^3$	
$-a^b/c$	$-\frac{a^b}{c}$	$-a^{\frac{b}{c}}$ değil!
$-a^(b/c)$	$-a^{\frac{b}{c}}$	

Standart Fonksiyonlar

Matematikte karşılaşılan Sin x, Cos x, Log x, e^x gibi fonksiyonların birçoğu VB de tanımlıdır.

Visual Basic trigonometrik fonksiyonlarda radyan açı birimini kullanır. Örneğin, trigonometrik değeri hesaplanacak açı derece cinsinden ise $\pi/180$ ile çarpılarak radyana çevrilmelidir.

Int(x) örnekleri:

y=Int(2.9) den y=2
y=Int(2.1) den y=2
y=Int(-2.9) den y= -3
y=Int(-2.1) den y= -3
y=Int(2.0) den y=2
y=Int(-2.0) den y= -2

Fix(x) örnekleri:

y=Fix(2.9) den y=2
y=Fix(2.1) den y=2
y=Fix(-2.9) den y= -2
y=Fix(-2.1) den y= -2
y=Fix(2.0) den y=2
y=Fix(-2.0) den y= -2

Sgn(x) örnekleri:

y=Sgn(2000) den y=1
y=Sgn(-2000) den y= -1
y=Sgn(0) den y=0

Rnd(x) örnekleri:

y=Rnd(1) den y=0.7055475
y=Rnd(0) den y= 0.0195312
y=Rnd(-1) den y=0.224007
y=Rnd den y=0.7055475

Rasgele sayılar

Diğer örnekler:

$$y = \text{Ln} \left| x^3 - x \right| + e^{\sqrt{5x}}$$

İfadesinin VB karşılığı:

$$y = \text{Log}(\text{Abs}(x^3 - x)) + \text{Exp}(\text{Sqr}(5 * x))$$

VB fonksiyonu	Matematik anlamı	Açıklama
y=Sin(x)	y= Sin x	x radyan
y= Cos(x)	y= Cos x	x radyan
y= Tan(x)	y= Tan x	x radyan $x \neq \pm \pi/2$
y= Atn(x)	y= Arctan x	$-\pi/2 < y < \pi/2$
y= Log(x)	y= Ln x	Log _e x=Ln x x>0
y= Sqr(x)		x ≥ 0
y= Exp(x)	$y = e^x$	x < 709.782712893
y= Abs(x)	y= x	Mutlak değer
y= Sgn(x)	y= Sign x	x in işareti
y= Fix(x)		x in tam sayı kısmı
y= Int(x)		x in (yuvarlanmış) tam sayı kısmı
y= Rnd(x) veya y= Rnd		0 ≤ y < 1 arasında rasgele bir sayı

Diğer örnekler:

$$y = x \sin \alpha^3 + \sqrt{x+3} \cos^2 \beta$$

ifadesinin VB karşılığı, A= α ve B= β alınarak

α , β radyan biriminde ise:

$$y=x*\sin(A^3)+\text{Sqr}(x+3)*\cos(B)^2$$

α , β derece biriminde ise:

π sayısının karşılığı P=4*Atn(1) olmak üzere

$$y=x*\sin((A*P/180)^3)+\text{Sqr}(x+3)*\cos(B*P/180)^2$$

$$y=\text{Int}(x+0.5)$$

İfadesi x ondalık sayısını aşağı veya yukarı yuvarlar:

$$y=\text{Int}(2.3+0.5) \text{ den } y=2 \quad (\text{aşağı yuvarlama})$$

$$y=\text{Int}(2.8+0.5) \text{ den } y=3 \quad (\text{yukarı yuvarlama})$$

$$y=\text{Int}(10^n * x + 0.5) / 10^k$$

İfadesi x ondalık sayısının n. ondalık basamağını

yuvarlar. x=2.387654 sayısı için:

$$y=\text{Int}(10^1 * 2.387658 + 0.5) / 10^1 \text{ den } y=2.4$$

$$y=\text{Int}(10^2 * 2.387658 + 0.5) / 10^2 \text{ den } y=2.39$$

$$y=\text{Int}(10^3 * 2.387658 + 0.5) / 10^3 \text{ den } y=2.388$$

$$y=\text{Int}(10^4 * 2.387658 + 0.5) / 10^4 \text{ den } y=2.3877$$

$$y=\text{Int}(10^5 * 2.387658 + 0.5) / 10^5 \text{ den } y=2.38765$$

$\alpha=0^0$, $\alpha=45^0$, $\alpha=90^0$, $\alpha=135^0$ olduğuna göre, Tan $\alpha=?$

π sayısının karşılığı P=4*Atn(1) olmak üzere

$$y=\text{Tan}(0*P/180) \text{ den } y=0$$

$$y=\text{Tan}(45*P/180) \text{ den } y=1$$

$$y=\text{Tan}(90*P/180) \text{ den } y=1.63317787283838E+16$$

$$y=\text{Tan}(135*P/180) \text{ den } y= -1$$

VB hatalı sonuç veriyor !
HATA vermeliydi. Çünkü,
Tan $90^0 \rightarrow \infty$ dur. Bilgisayarda ∞ yoktur.

$$y=a+\text{Rnd}*b$$

İfadesi a sayısı ile b sayısı arasında rasgele bir sayı üretir:

$$a \leq y \leq b$$

$$y= -500+\text{Rnd}*1500 \text{ den } -500 \leq y \leq 1500 \text{ olur. Çok fazla veri}$$

gerektiren programların test aşamasında bu imkandan

yararlanılır.

$$y=\text{Int}(a+\text{Rnd}*(b-a)+0.5)$$

İfadesi a sayısı ile b sayısı arasında rasgele bir tam sayı üretir: a

$$\leq y \leq b$$

$$y= \text{Int}(1+\text{Rnd}*49) \text{ den } 1 \leq y \leq 49 \text{ olur(Loto sayıları)}$$

$$y= (0+\text{Rnd}*2) \text{ den } y=0 \text{ veya } 1 \text{ veya } 2 \text{ olur (spor toto sayıları)}$$

Türetilen fonksiyonlar:

Listesi verilen Standart fonksiyonlar teknik ve matematikte karşılaşılan Sinh x, Arcsin x, 10 tabanına göre Logaritma,... gibi fonksiyonları içermez.. Karşılığı olmayan bu tür fonksiyonlar matematik bağıntılar ve kurallara göre standart fonksiyonlar yardımıyla türetilirler.

Örnek:

$$\tan \frac{\pi}{4} = 1$$

$$\pi = 4 \text{Arc tan}(1)$$

VB karşılığı:

$$y=4*\text{Atn}(1)$$

$$y = \text{Sinh } x = \frac{e^x - e^{-x}}{2}$$

VB karşılığı:

$$y=(\text{Exp}(x)-\text{Exp}(-x))/2$$

Matematik	VB karşılığı
$y=\text{Log}_n x$	$y=\text{Log}(x)/\text{Log}(n)$
$y= \pi=3.1415\dots$	$y=4*\text{Atn}(1)$
$y= e=2.7182\dots$	$y=\text{Exp}(1)$
$y=\text{Sec } x$	$y= 1 / \text{Cos}(x)$
$y=\text{Cosec } x$	$y= 1 / \text{Sin}(x)$
$y=\text{Cotan } x$	$y=1 / \text{Tan}(x)$
$y=\text{Sinh } x$	$y= (\text{Exp}(x) - \text{Exp}(-x)) / 2$
$y=\text{Cosh } x$	$y=(\text{Exp}(x) + \text{Exp}(-x)) / 2$
$y=\text{Tanh } x$	$y=2 * \text{Exp}(x) / (\text{Exp}(x) + \text{Exp}(-x)) - 1$
$y=\text{Coth } x$	$y=2 * \text{Exp}(-x) / (\text{Exp}(x) - \text{Exp}(-x)) + 1$
$y=\text{Sech } x$	$y=2 / (\text{Exp}(x) + \text{Exp}(-x))$
$y=\text{Cosech } x$	$y=2 / (\text{Exp}(x) - \text{Exp}(-x))$
$y=\text{Arcsin } x$	$y= \text{Atn}(x / \text{Sqr}(-x * x + 1))$
$y=\text{Arccos } x$	$y=\text{Atn}(-x / \text{Sqr}(-x * x + 1)) + 2 * \text{Atn}(1)$
$y=\text{Arccot } x$	$y= \text{Atn}(x) + 2 * \text{Atn}(1)$
$y=\text{Arcsec } x$	$y=\text{Atn}(x/\text{Sqr}(x*x-1))+(\text{Sgn}(x)-1)*(2*\text{Atn}(1))$
$y=\text{Arccosec } x$	$y=\text{Atn}(x/\text{Sqr}(x*x-1))+(\text{Sgn}(x)-1)*(2*\text{Atn}(1))$
$y=\text{Arcsinh } x$	$y= \text{Log}(x + \text{Sqr}(x * x + 1))$
$y=\text{Arccosh } x$	$y= \text{Log}(x + \text{Sqr}(x * x - 1))$
$y=\text{Arctanh } x$	$y=\text{Log}((1 + x) / (1 - x)) / 2$
$y=\text{Arccoth } x$	$y= \text{Log}((X + 1) / (X - 1)) / 2$
$y=\text{Arcsech } x$	$y= \text{Log}((\text{Sqr}(-x * x + 1) + 1) / x)$
$y=\text{Arccosech } x$	$y= \text{Log}((\text{Sgn}(x) * \text{Sqr}(x * x + 1) + 1) / x)$

Adım adım bir program örneği:

Projenin tanımı: Bir işyerinde üç tür mal satılmaktadır. Her malın fiyatı ve katma değer vergi yüzdesi farklıdır. Müşterinin aldığı mal veya malların fatura tutarını hesaplayan programı yazınız.

Bilinenler: Her malın fiyatı ve katma değer vergi yüzdesi:

1. Malın fiyatı 80 Milyon TL, Katma değer vergi yüzdesi %8
2. Malın fiyatı 240 Milyon TL, Katma değer vergi yüzdesi %15
3. Malın fiyatı 743 Milyon TL, Katma değer vergi yüzdesi %18

Hesaplanacak: Katma değer vergisi(KDV) dahil toplam fatura tutarı.

Algoritma: Malların fiyatları sırasıyla f_1, f_2, f_3 ; KDV yüzdeleri sırasıyla k_1, k_2, k_3 olsun. Fatura tutarını f ile gösterelim:

$$f = f_1 \cdot \left(1 + \frac{k_1}{100}\right) + f_2 \cdot \left(1 + \frac{k_2}{100}\right) + f_3 \cdot \left(1 + \frac{k_3}{100}\right)$$

Program01: Formun **Click** olayına gerekli verileri kodlayarak **Print** deyimi yardımıyla fatura tutarını yazdıralım. VB çalıştırılarak form çift tıklanır, açılan pencerenin üst sağ penceresinden **Click** olayı seçilir ve kod yazılır. Program çalıştırılır, Form tıklanır, sonuç form üzerinde görülür.

VB tarafından **Click** olayı için otomatik olarak hazırlanan satır

Yazdığımız kodlar

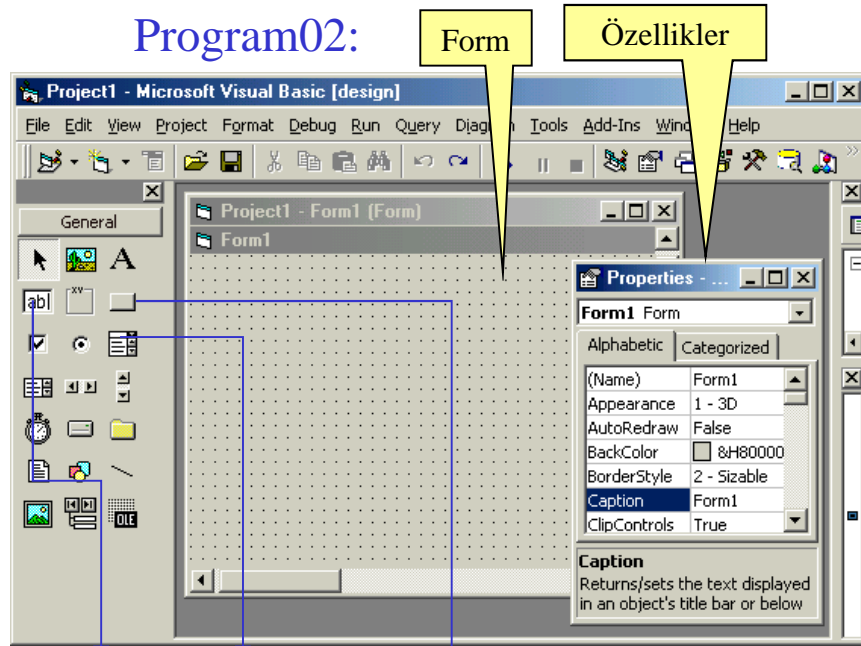
VB tarafından **Click** olayı için otomatik olarak hazırlanan satır

Fatura tutarı(sonuç)

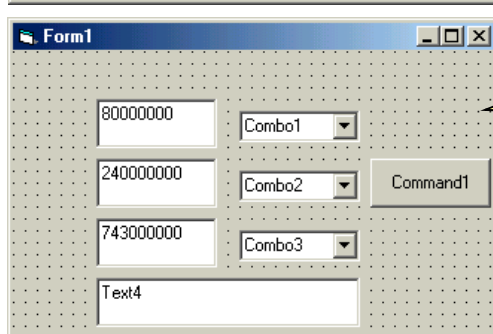
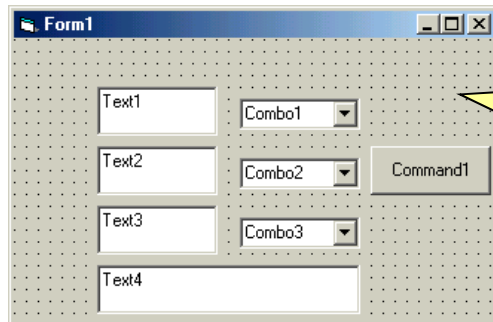
```
Project1 - Form1 (Code)
Form Click
End Sub
Private Sub Form_Click()
f1 = 80000000
f2 = 240000000
f3 = 743000000
k1 = 8
k2 = 15
k3 = 18
f = f1 * (1 + k1 / 100) + f2 * (1 + k2 / 100) + f3 * (1 + k3 / 100)
Print f
End Sub
Form1
1239140000
```

Program çok basit olmakla birlikte çok da kötüdür. Çünkü malın fiyatı ve KDV yüzdesi değiştikçe programın satırlarının da değişmesi gerekir. Kullanıcı(satıcı) ise programlama bilmez. Ayrıca bilse bile kullanım zahmetlidir ve değişiklik yaparken hatalara neden olabilir. Ayrıca program görsel(Visual) değildir.

Program02:



TextBox ComboBox CommandButton

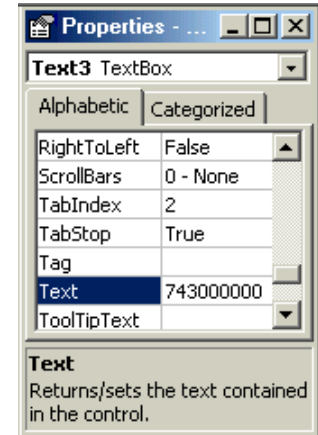
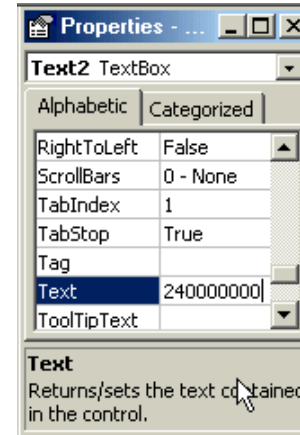
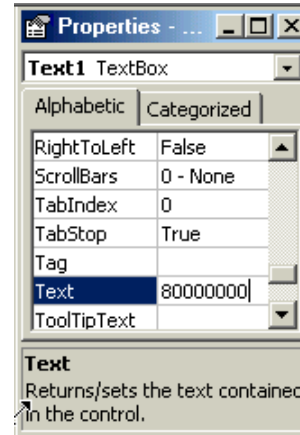


Form üzerine dört adet **TextBox**, üç adet **ComboBox** ve bir adet **CommandButton** yerleştirelim. Bunun için kontroller(**General**) penceresindeki bu kontrolleri çift tıklayalım. Her kontrolü sağa-sola, yukarı aşağı çekerek istediğimiz görünümü sağlayalım.

Bu kontrolleri neden kullandık? Text1, Text2 ve Text3 pencerelerine 1., 2. Ve 3. Malın fiyatını yazacağız. Combo1, Combo2, Combo3 ü 1., 2. ve 3. malın KDV yüzdeleri için; Command1 komut düğmesini **HESAPLA** komutu için ve nihayet Text4 penceresini fatura tutarını görüntülemek için kullanacağız.

Text1 penceresini tıklayarak seçelim. Özellikler(**Properties**) penceresinde **Text** satırını bularak 1. Malın fiyatını 80000000 olarak yazalım. Text1 penceresinin içeriği artık 80000000 olacaktır.

Aynı yolu izleyerek Text2, Text3 pencerelerine 2. Ve 3. Malın fiyatlarını 240000000 ve 743000000 olarak yazalım.



Çalışma sırasında formun görünümü

Form1

80000000 Combo1

240000000 Combo2 Command1

743000000 Combo3

Text4

Form1

80000000 Combo1

8
15
18

240000000 Command1

743000000 Combo3

Text4

Form1

80000000 8

240000000 15 Command1

743000000 15

Text4

Combo1 penceresini tıklayarak seçelim. Özelliklerden List satırını bulup tıklayalım. Aşağı doğru açılan pencereye 8 Ctrl Enter, 15 Ctrl Enter, 18 Ctrl Enter tuşlayalım (Ctrl Enter bir alt satıra geç anlamındadır). Bu değerler KDV yüzdesi seçenekleridir. Aynı yolu izleyerek Combo2 ve Combo3 ün List penceresine aynı KDV yüzdesini yazalım.

Properties - ...

Combo1 ComboBox

Alphabetic Categorized

HelpContextID 0

Index

IntegralHeight True

ItemData (List)

Left 2280

List (List)

Locked 8

15

18

List Returns/sets the list of items contained in a...

Properties - ...

Combo2 ComboBox

Alphabetic Categorized

HelpContextID 0

Index

IntegralHeight True

ItemData (List)

Left 2280

List (List)

Locked 8

15

18

List Returns/sets the list of items contained in a...

Properties - ...

Combo3 ComboBox

Alphabetic Categorized

HelpContextID 0

Index

IntegralHeight True

ItemData (List)

Left 2280

List (List)

Locked 8

15

18

List Returns/sets the list of items contained in a...

Program bu haliyle çalıştırılırsa (F5 tuşu ile) sol üstteki görüntü elde edilir. Henüz hiçbir kod yazmadığımız için bir işe de yaramaz. Peki, Hesapları yapacak kodu nereye ve nasıl yazacağız? HESAPLA komutunu nasıl vereceğiz?

Malların fiyatları ilgili pencerelerde görülmektedir. Bu değerler Text1.Text, Text2.Text ve Text3.Text özelliklerinde saklıdır. KDV yüzdesi de Combo ların Text özelliğinde saklıdır. Ancak birden çok KDV yüzdesi seçeneği olduğundan görüntülenmezler. Combo larda tıklayarak istediğimiz KDV yüzdesini seçeriz.

Command1 düğmesi tıklanınca HESAPLA komutu olarak algılanmasını ve gerekli hesapların yapılarak sonucun (fatura tutarı) Text4 penceresinde görüntülenmesini istiyoruz. Bunun için gerekli olan program kodlarını Command1 kontrolünün Click olayına yazmalıyız.

1.Programdan hatırlanacağı gibi, malların fiyatları f_1, f_2, f_3 ile; KDV yüzdeleri k_1, k_2, k_3 ile; hesaplanacak olan fatura tutarı da f ile gösterilmiştir. Bu nedenle:

Text1.text, Text2.text, Text3.text in içeriğini f_1, f_2, f_3 e

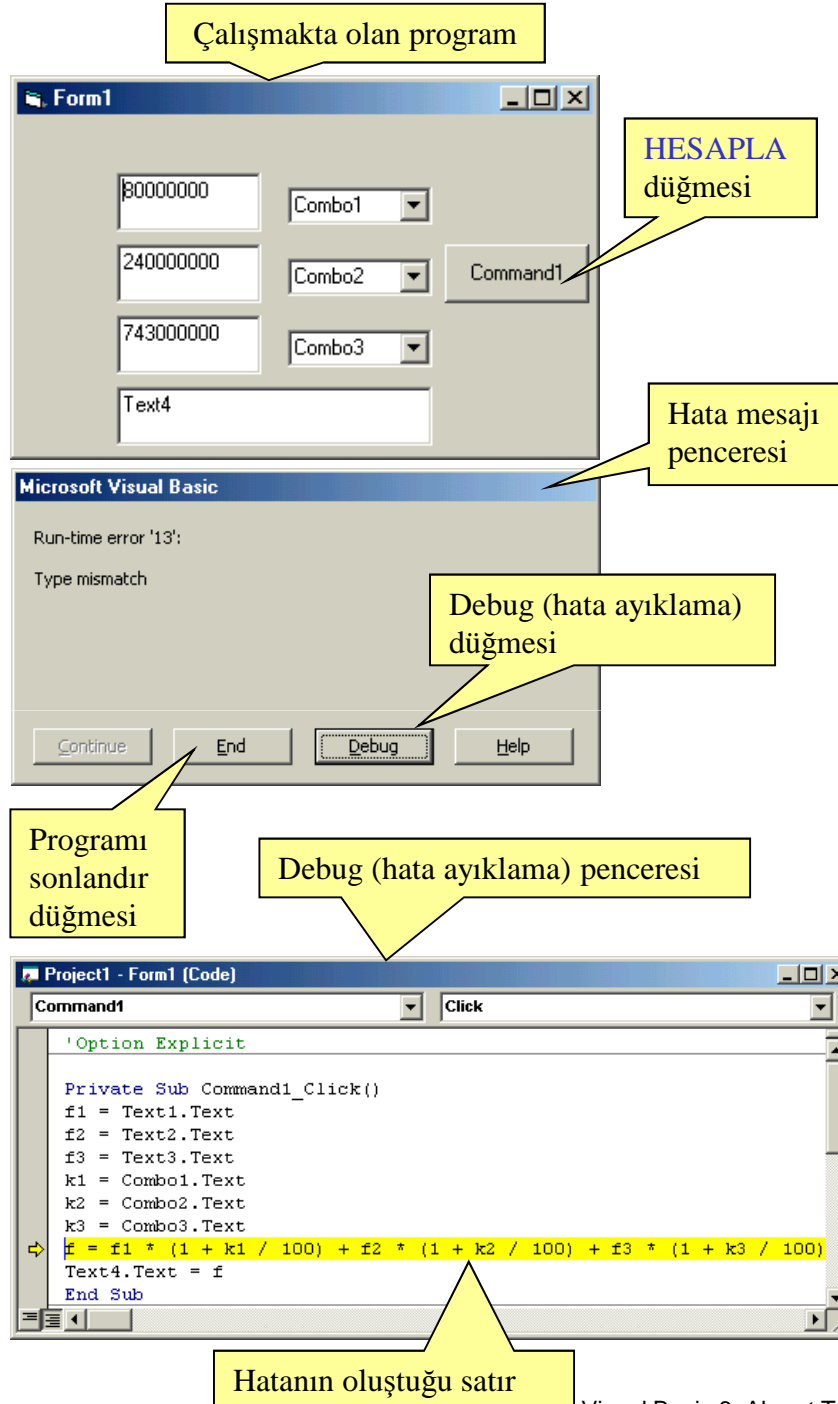
Combo1.text, Combo2.text, Combo3.text in içeriğini k_1, k_2, k_3 e

$$\text{aktarmamız ve } f = f_1 \cdot \left(1 + \frac{k_1}{100}\right) + f_2 \cdot \left(1 + \frac{k_2}{100}\right) + f_3 \cdot \left(1 + \frac{k_3}{100}\right)$$

ile fatura tutarını hesaplamamız gerekir. Hesaplanan f değerini, görüntülemek için, Text4.Text e aktarmalıyız.

Bu adımları içeren program kodunu Command1 kontrolünün Click olayına yazalım.

```
Project1 - Form1 [Code]
Command1 Click
'Option Explicit
Private Sub Command1_Click()
    f1 = Text1.Text
    f2 = Text2.Text
    f3 = Text3.Text
    k1 = Combo1.Text
    k2 = Combo2.Text
    k3 = Combo3.Text
    f = f1 * (1 + k1 / 100) + f2 * (1 + k2 / 100) + f3 * (1 + k3 / 100)
    Text4.Text = f
End Sub
```



Programımız artık çalışmaya hazırdır, **F5** tuşu ile çalıştıralım. Sol üstteki görüntüyü elde ederiz. Command1(HESAPLA) düğmesi tıklanınca bu düğmenin **Click** olayına yazdığımız program satırları işlenir.

Tıklayalım. Oda ne! Text4 penceresinde hesaplanan f değerini bekliyorduk. Karşımıza sol ortadaki pencere çıktı. Niçin? Araştıralım. Hata penceresindeki **Debug** düğmesini tıklayalım. Karşımıza hatanın nerede oluştuğunu gösteren sol alttaki pencere çıkar(hata renkli satırda oluşmuş). Kodlamayı doğru yapmıştık. Peki bu satırda yanlış olan ne? Anlamak için fareyi f1, k1,... değişkenlerinin üzerine sürükleyerek kısa süre bekletelim. Fare f1 üzerinde iken

```

k3 = Combo3.Text
⇒ f = f1 * (1 + k1 / 100) + f2 * (1 + k2 / 100) + f3 * (1 + k3 / 100)
   f1="80000000" = f
End Sub

```

k1 üzerinde iken

```

k3 = Combo3.Text
⇒ f = f1 * (1 + k1 / 100) + f2 * (1 + k2 / 100) + f3 * (1 + k3 / 100)
   Text4.Text[k1="Combo1"] = f
End Sub

```

görüntülenir. Bunlar, bu satır işlenirken bu değişkenlerin o andaki içeriğini gösterir. Diğer değişkenleri de benzer yolla öğrenebiliriz. Örneğin f için

```

k3 = Combo3.Text
⇒ f = f1 * (1 + k1 / 100) + f2 * (1 + k2 / 100) + f3 * (1 + k3 / 100)
   f=Empty,4.Text = f
End Sub

```

Empty (boş) görüntülenir.

Hatanın nedenini anladık. k1, k2 ve k3 değişkenlerinin içeriği sayısal değil, alfa sayısalıdır. Program işlemi yürütememektedir. Bu nedenle f de hesaplanamamıştır, içi boştur.

k1, k2 ve k3 değişkenlerini Combo lardan seçmediğimiz için bu hata oluşmuştur.

Çalışmakta olan programı **Run** penceresinden **End** tıklayarak sonlandırılm. **F5** tuşu ile programı yeniden çalıştırılm. Combo lardan KDV yüzdelerini seçelim. Sol üstteki görüntü oluşacaktır.

Artık hesap için tüm değerler bellidir. Command1 düğmesini tıklayalım. Sol alttaki görüntüyü elde ederiz. Hesaplanan değeri, yani satın alınan malların KDV dahil fatura tutarını, Text4 penceresinde görmekteyiz.

Verileri duruma göre değiştirebiliriz. Örneğin, bir müşteri ikinci maldan satın almazsa değerini sıfır gireriz. KDV yüzdesi değişirse, listede olmayan değeri Combo penceresine yazarız.

İlk minik görsel programımızı tamamladık. Ancak gene de yeterince profesyonel değil. Bunun bir çok nedeni var. Birkaç tane sayalım:

- Bu kutucukların anlamları nedir?
- Kullanıcı nasıl anlayacak?
- Programı nasıl sonlandıracağız?
- Programın bir adı yok mu?
- Command1 düğmesi neden Türkçe değil?

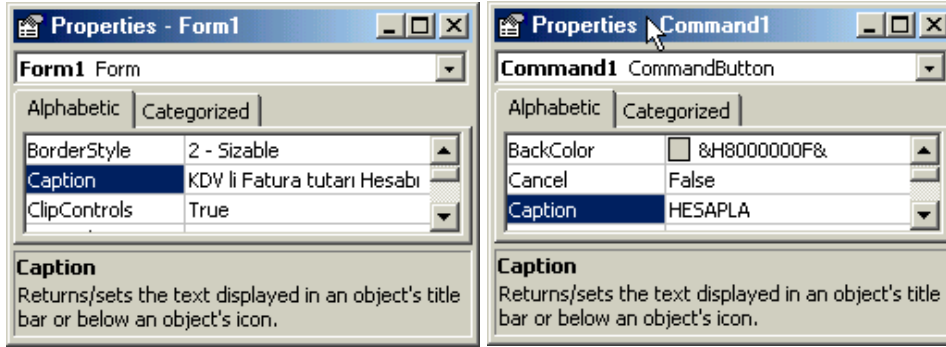
....

Bu soruların çoğunun cevabını biliyoruz, çünkü biz yazdık. Ya kullanacak olanlar?

Bu sorunları gidermek ve belki de tümüyle değiştirerek çok daha iyi bir program yazmanın sayısız yolu vardır. Bu, her programcının kendi hayal dünyasında farklı gelişir.

Ancak bilinmelidir ki en iyi programın daima daha da iyisi vardır. O halde bir yerde yetinmesini bilmeliyiz. Aksi halde aynı program üzerinde yıllarca çalışır, bir başka konuda yazamayız. Bu arada, kimse programdan yararlanmadan, teknoloji değişir, her şey sil yeni baştan başlar.

Program03:



The image shows the 'KDV li Fatura tutarı Hesabı' form in design mode. The form has a title bar with the text 'KDV li Fatura tutarı Hesabı'. It contains three text boxes for 'Mallar' (80000000, 240000000, 743000000), three dropdown menus for 'KDV yüzdesi' (Combo1, Combo2, Combo3), and two buttons labeled 'HESAPLA' and 'ÇIK'. A text box at the bottom left is labeled 'Fatura tutarı' and contains 'Text4'. The bottom right text is 'TL (KDV Dahil)'.

Form
(tasarım modunda)

The image shows the 'KDV li Fatura tutarı Hesabı' form in run mode. The 'Mallar' text boxes contain 80000000, 240000000, and 743000000. The 'KDV yüzdesi' dropdown menus are set to 8, 15, and 18. The 'HESAPLA' button is highlighted with a mouse cursor. The 'Fatura tutarı' text box now contains '1239140000'. The 'ÇIK' button is still visible.

Form
(çalışma modunda)

Önceki programı biraz daha iyileştirelim. Malları ve KDV yüzdelerini belirgin kılacak Label (etiket), programı durduracak CommandButton, ... ekleyelim. Programa bir ad verelim. Command1 i Türkçeleştirelim.

Programa bir ad verelim: Formu tıklayalım. Özelliklerden Caption satırını bulup programa bir ad yazalım: KDV li Fatura tutarı Hesabı. Bu ad formun üst tarafında görünecektir.

Command1 düğmesini Türkçeleştirelim: Bu düğmeği tıklayalım Özelliklerden Caption satırını bulup amacına yönelik olarak HESAPLA yazalım. Bu ad düğme üzerinde görünecektir.

Mallar ve KDV yüzdeleri için Label (etiket) yerleştirelim: General penceresinden **Label** kontrolünü çift tıklayarak ilk Label i yerleştirip mallara ait kutucukların üstüne hizalayalım. Bu Label in Caption ına Mallar yazalım.

General penceresinden **Label** kontrolünü çift tıklayarak ikinci Label i yerleştirip KDV yüzdelere ait kutucukların üstüne hizalayalım. Bu Label in Caption ına KDV yüzdesi yazısını yazalım.

İki label daha yerleştirip birini Text4 ün soluna hizalayarak Caption ına fatura tutarı yazalım. Diğerini Text4 ün sağına yerleştirip Caption a TL (KDV Dahil) yazalım.

Programı durduracak düğme ekleyelim: Forma bir adet **CommandButton** düğmesi yerleştirip hizalayarak büyüklüğünü ayarlayalım. Caption a ÇIK yazalım. Bu düğmenin **Click** olayına End satırını yazalım. Bu düğme tıklandığında **Click** olayı işlenensin, çalışan program sonlansın istiyoruz. Programın iyileştirilmiş son şekli ve çalıştırılmış hali solda görülmektedir. ÇIK düğmesi tıklandığında sonlanacaktır.

Ödevler:

1. Aşağıdaki boşlukları doldurunuz.

256 kB = Byte
256 MB = kByte = Byte
80 GB = MB = kByte = Byte

2. Aşağıdaki VB sabitlerinin bazıları hatalıdır, bulunuz, açıklayınız

13,32 -0.00097 276.824.746 340 000 000 E-4 3.0E±7 -1237 “Evet” $5\frac{3}{4}$

3. Aşağıdaki VB değişkenlerin bazıları hatalıdır, bulunuz, açıklayınız

K2z7 Print alfa Adres\$?soru t₂ İsim j/6

4. Aşağıdaki matematik ifadelerin VB karşılığını yazınız.

$$s = a^{n-1} \quad S = t^{x^{2n}} \quad a_i = -2,39\epsilon - \frac{\lambda}{2\alpha} \quad s = \frac{b}{3+2,09d} \sqrt{a^2 - b^2} \quad s = k_1 k_2 - \sqrt[4]{\alpha - 2\theta}$$
$$s = \frac{7}{8} (n - y)^{\sqrt{a}} \quad s = (1 + t) \left\{ 1 + a[b + c^2] - \left(\frac{t}{8a} - 1\right)^3 \right\} \quad s = \frac{a + b + c}{(2b + c)ab}$$

5. Aşağıdaki VB ifadelerin matematik karşılığını yazınız.

$$s = A * ((b + e) * d - b) \quad s = 1 / (1 / y + 1 / z) \quad s = (w * s + r) * E * 1.0E7 \quad s = (A * A + b^x)^2$$
$$s = X^{(-p/q)} \quad s = 1 / (m/n - 1/n/n) \quad s = c * c^h / E^5 / z^{(h/2) - 1} \quad s = p1 * L^{0.5} / e / d^3$$
$$s = (((a^{0.5}) * b)^{0.5} * c)^{0.5} \quad s = A^n^{0.5 - 1} \quad s = T1 / T2 / T3 * P^k * (1 - P)^m$$

6. Program03 ü kendiniz yazınız. Form üzerine programcının adını ve program versiyon tarihini ekleyiniz. Fare Text1 (1. malın fiyatı) üzerinde iken açılacak bir pencerede 1.malın fiyatı yazısı görüntülensin (**ToolTipText özelliğini kullanın**). Program F5 ile çalıştırıldığında Text4 (Fatura tutarı) penceresinde 0; Combo1, Combo2 ve Combo3 pencerelerinde 8, 15 ve 18 ön değerleri görünsün (**İlgili kontrollerin Text özelliğini kullanın**)
Yazdığınız programı çalıştırınız ve 1.malın fiyatına kazara bir harf ekleyerek HESAPLA düğmesini tıklayınız. Ne oldu? Niçin? Ne yapılabilir?

Değişken tipi tanımlama

VB6 da değişkenlerin tiplerinin tanımlanması zorunlu değildir, programcının isteğine bırakılmıştır. Değişken tip tanımlanması yapılmadığı takdirde tüm değişkenler otomatik olarak **Variant** tipinde olur. Tip tanımlamadan program yazılması bir çok sorun yaratır: Program yavaş çalışır, gereksiz çok bellek harcanır, değişkene atama yapılmadan işlem yapma(çarpma, bölme,...) hatasına düşülebilir. Bu nedenle VB6 da tüm değişkenlerin tipleri programcı tarafından tanımlanmalıdır.

Değişken tipi tanımlamayı zorunlu kılmak için iki değişik yol vardır.

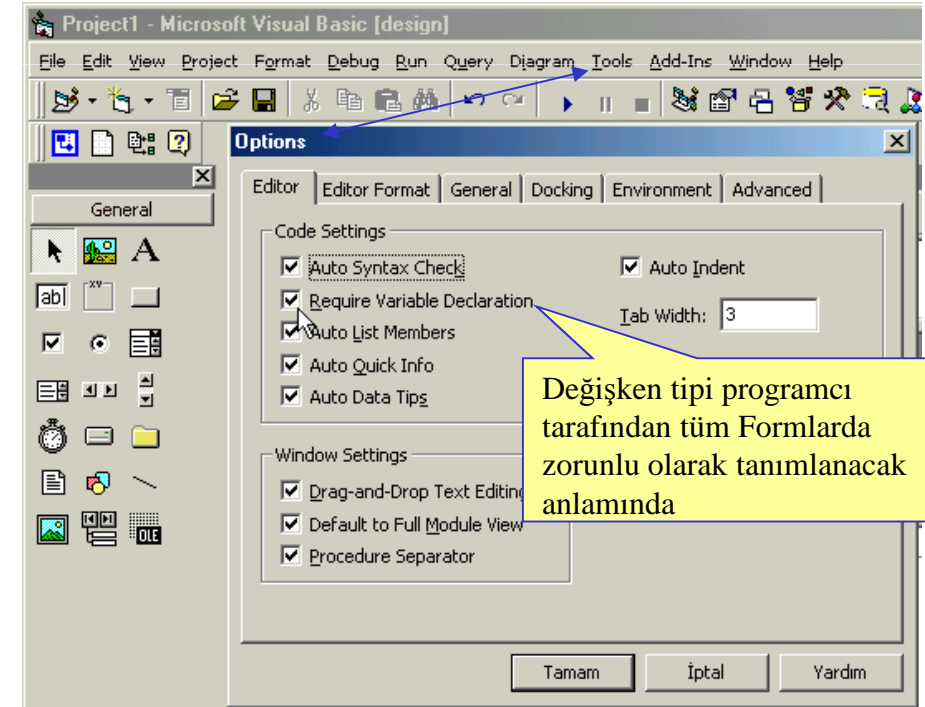
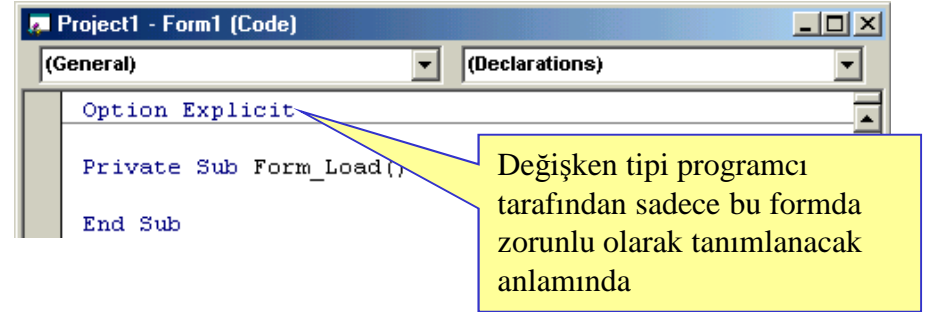
1.Programcı formun **General declarations** (genel tanımlamalar) kısmına **Option Explicit** (isteğe bağlı kesin) satırını kendisi ekler:

Bu yöntemle eklenen **Option Explicit** sadece bulunduğu form için geçerlidir, diğer formlarda geçersizdir. Programcı her forma eklemek zorundadır.

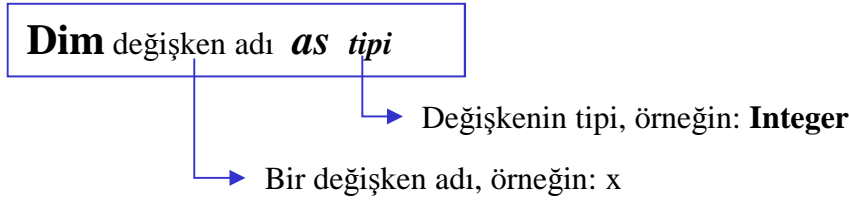
2.Programcı **Tools/Options** penceresinde **Require Variable Declarations** seçeneğini işaretler:

Bu yöntemle **Option Explicit** satırı VB6 tarafından her yeni forma otomatik eklenir. Ancak, daha önce kod yazılmış formlara eklemeyiz. Bu nedenle yeni bir projeye başlarken Hiçbir kod yazmadan **Require Variable Declarations** seçeneği işaretlenmeli **File/New Project** ile yeni bir proje başlatılmalıdır.

İyi programcı Option Explicit seçeneğini daima işaretler!



Option Explicit satırı eklenmiş programda artık hiçbir değişken tanımlanmadan kullanılamaz. Aksi halde program hata mesajı ile sona erer. Değişkenlerin tanımlanmasında değişik yollar vardır. En çok kullanılanı **Dim** (**D**imension=**boyut**) deyimidir:



Bir değişkene tipi ile bağdaşmayan değer atanamaz. **Integer** tipindeki bir değişkene -32768 den 32767 arasında bir tam sayı atanabilir. Tipi sayısal(**Integer, Double,...**) olan değişkenlere 0, string değişkenlere "" ön değeri VB6 tarafından otomatik olarak atanır.

Örnekler:

Dim x as Long
Dim y as Double
Dim c as Double
Dim dugme as String
Dim t

Veya

Dim x as Long:Dim y as Double
Dim c as Double:Dim dugme as String
Dim t

: işareti kullanılarak
deyimler aynı satıra
yazılabilir

Veya

Dim x as Long, y as Double
Dim c as Double, Dim dugme as String
Dim t

x değişkeni **Long** tipinde tanımlanmıştır, 4 Byte kullanılacaktır, içeriği -2 147 483 648 ile 2 147 483 647 arasında bir tam sayı olabilir.

y değişkeni **Double** tipinde tanımlanmıştır, 8 Byte kullanılacaktır, içeriği ± 1.79769313486232E308 ile ± 4.94065645841247E-324 arasında ondalık sayı olabilir.

dugme değişkeni **String** tipinde tanımlanmıştır, her karakter için 1 Byte kullanılacaktır, içeriği örneğin, "Osmangazi Üniversitesi" olabilir.

t değişkeni tanımlanmış fakat bir tip belirtilmemiştir. Tipi otomatik olarak **Variant** olacaktır, içeriği her tip veri (sayısal veya **string**) olabilir.

Değişkenler programın neresinde tanımlanır?

1. Değişkenler formun **General declarations** bölümünde tanımlanabilir. Bu bölümde tanımlanan değişkenler formun tüm alt programlarında geçerli olur. Ancak diğer formlarda geçersiz olur.

Değişkenler zorunlu tanımlanacak

```
Option Explicit
Dim a As Integer
Dim b As Double, c As Double
Dim dugme As String

Private Sub Command1_Click()
a = 2: b = 3.5
c = a + b
dugme = "Command1->"
Print dugme, "a="; a, "b="; b, "c="; c
End Sub

Private Sub Command2_Click()
c = a + b
dugme = "Command2->"
Print dugme, "a="; a, "b="; b, "c="; c
End Sub
```

Program çalıştırıldı, önce Command1 tıklandı

Command1->	a=2	b=3.5	c=5.5
Command2->	a=2	b=3.5	c=5.5

Program çalıştırıldı, önce Command2 tıklandı

Command2->	a=0	b=0	c=0
Command1->	a=2	b=3.5	c=5.5
Command2->	a=2	b=3.5	c=5.5

Bu değişkenler tüm izleyen alt programlarda geçerlidir. a=0, b=0, c=0, dugme="" ön değerine sahiptir(VB6 otomatik atıyor).

Comand1 tıklandığında değişkenlere buradaki yeni değerler atanacak

Değerleri daha önce atanmış değişkenler bu alt programda kullanılacak

Program çalıştırılıp önce Command1 tıklandığında değişkenler yeni değerleri aldı

Sonra Command2 tıklandığında değişkenler Command1 de aldığı değerleri kullandı

Program çalıştırılıp önce Command2 tıklandığında değişkenler VB6 ön değerlerini kullandı

Sonra Command1 tıklandığında değişkenler yeni değerler aldı

Sonra Command2 tıklandığında değişkenler Command1 de aldığı değerleri kullandı

2. Değişkenler bir alt programda tanımlanabilir. Bu değişkenler sadece tanımlandığı alt programda geçerli olur.

Değişkenler zorunlu tanımlanacak

```
Project1 - Form1 (Code)
Command2 Click
Option Explicit

Private Sub Command1_Click()
Dim a As Integer
Dim b As Double, c As Double
Dim dugme As String
a = 2: b = 3.5
c = a + b
dugme = "Command1->"
Print dugme, "a="; a, "b="; b, "c="; c
End Sub

Private Sub Command2_Click()
c = a + b
dugme = "Command2->"
Print dugme, "a="; a, "b="; b, "c="; c
End Sub
```

Bu alt programda tanımlanan değişkenler ve içerikleri sadece bu alt programda geçerlidir

Bu alt programda kullanılan değişkenler tanımsızdır, içerikleri yoktur. Bu satırlar işlenirken hata oluşacaktır

Önce Command1, sonra Command2 tıklandı

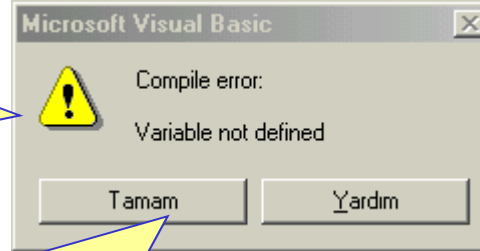
Form1

Command1-> a=2 b=3.5 c=5.5

Command1

Command2

Command2 tıklanınca hata oluşuyor, değişkenlerin tanımsız olduğu bildiriliyor



Tamam tıklanınca hatanın olduğu alt program gösteriliyor

Buradaki değişkenler tanımsız! Rastlanılan ilk tanımsız değişken belirginleştirilmiş.

```
Project1 - Form1 (Code)
Command2 Click
Option Explicit

Private Sub Command1_Click()
Dim a As Integer
Dim b As Double, c As Double
Dim dugme As String
a = 2: b = 3.5
c = a + b
dugme = "Command1->"
Print dugme, "a="; a, "b="; b, "c="; c
End Sub

Private Sub Command2_Click()
c = a + b
dugme = "Command2->"
Print dugme, "a="; a, "b="; b, "c="; c
End Sub
```

3. Değişkenler hem **General declarations** bölümünde hem de alt programlarda tanımlanabilir. **General declarations** bölümünde tanımlananlar tüm alt programlarda geçerli olurken, alt programlarda tanımlananlar sadece tanımlandığı alt program içinde geçerli olurlar.

The screenshot shows the Visual Basic code editor for 'Project1 - Form1 (Code)'. The code is as follows:

```
Option Explicit
Dim a As Integer
Dim dugme As String

Private Sub Command1_Click()
Dim b As Double, c As Double
a = 2: b = 2.5
c = a * b
dugme = "Command1->"
Print dugme, "a="; a, "b="; b, "c="; c
End Sub

Private Sub Command2_Click()
Dim b As Double, c As Double
a = 3: b = 3.5
c = a * b
dugme = "Command2->"
Print dugme, "a="; a, "b="; b, "c="; c
End Sub
```

The callouts explain the scope of the variables:

- Top Callout:** **a** ve **dugme** değişkenleri Form1 e ait tüm alt programlarda tanımlı olacak. her yerde kullanılabilir, yeni değer atanabilir
- Left Callout 1:** Buradaki **b** ve **c** değişkenleri sadece bu alt programda geçerli
- Right Callout 1:** Buradaki **a** ve **dugme** değişkenlerinin içerikleri, yeni bir değer atanmadıkça, her alt programda geçerli. **b** ve **c** nin içeriği sadece burada geçerli
- Left Callout 2:** Buradaki **b** ve **c** değişkenleri sadece bu alt programda geçerli
- Right Callout 2:** Buradaki **a** ve **dugme** değişkenlerinin içerikleri, yeni bir değer atanmadıkça, her alt programda geçerli. **b** ve **c** nin içeriği sadece burada geçerli

The bottom window shows the form 'Form1' with two buttons, 'Command1' and 'Command2'. The output of the commands is displayed in a text box:

Command1->	a= 2	b= 2.5	c= 5
Command2->	a= 3	b= 3.5	c= 10.5
Command1->	a= 2	b= 2.5	c= 5
Command2->	a= 3	b= 3.5	c= 10.5

Public değişkenler:

Bir değişkenin projedeki tüm formlarda geçerli olması istenirse **Dim** deyimi yerine **Public** (herkese açık) deyimi kullanılır. **Public** deyimi ile tanımlanan değişken, hangi formda tanımlanırsa tanımlansın, projedeki tüm formlarda geçerli olur. **Public** deyimi daima formun **General declarations** bölümüne yazılır.

Public değişken adı *as* tipi

Değişkenin tipi, örneğin: Integer

Bir değişken adı, örneğin: x

Bu değişkenler tüm formlarda geçerli olacak. Adları aynı, ancak içerikleri farklı olacak

Bilgilerine erişebilmek için Form2 yükleniyor, ancak görüntülenmez

Form1 ve Form2 deki değişkenlere değer atanıyor

Form1 bilgileri Form1 e yazdırılıyor

Form2 bilgileri Form1 e yazdırılıyor

Form1 bilgileri Form1 e yazdırılıyor

Form2 bilgileri Form1 e yazdırılıyor

Form1 bilgileri Form1 içinden Form2 ye yazdırılıyor

Form2 bilgileri Form1 içinden Form2 ye yazdırılıyor

```
Project1 - Form1 [Code]
Command2 Click
Option Explicit
Public a As Integer
Public bilgi As String

Private Sub Form_Load()
Load Form2
bilgi = "Form1->"
a = 1
End Sub

Private Sub Command1_Click()
Print Form1.bilgi; Form1.a
Print Form2.bilgi; Form2.a
End Sub

Private Sub Command2_Click()
Form2.Show
Print Form1.bilgi; Form1.a
Print Form2.bilgi; Form2.a
Form2.Print Form1.bilgi, Form1.a
Form2.Print Form2.bilgi, Form2.a
End Sub

Project1 - Form2 (Code)
Form Load
Option Explicit
Public a As Integer bilgi As String

Private Sub Form_Load()
bilgi = "Form2->"
a = 2
End Sub

Form2
Form1-> 1
Form2-> 2
```

Form1

Form1-> 1
Form2-> 2
Form1-> 1
Form2-> 2

Command1
Command2

Static deęişkenler:

Alt programlarda **Dim** deyimi ile tanımlanan deęişkenler alt program sona erdiğinde bellekten atılırlar, içeriklerine bir daha erişilemez. Bir deęişkenin içeriğinin, form bellekte kaldığı sürece, saklı tutulması istenirse deęişken **Dim** yerine **Static** (kalıcı) deyimi ile tanımlanır.

Örneğin: bir alt program kaç kez kullanıldı? Bir düğme kaç kez tıklandı?

Aşağıda soldaki program **Command1** ve **Command2** düğmelerinin kaç kez tıklandığını belirlemektedir. Sayıcı deęişkeni **Static** deyimi ile tanımlandığından önceki deęer unutulmaz ve her tıklamada sayıcı bir artırılır. Sağdaki programın **Command2 Click** olayında bu deęişken **Dim** deyimi ile tanımlandığı için, sayıcı deęişkeni her tıklamada 0 (sıfır) ön deęerini aldığından, **Command2** düğmesinin kaç kez tıklandığı belirlenemez.

The image shows two side-by-side code editors and two form windows. The left code editor shows two subroutines, both using `Static sayıcı As Integer`. The right code editor shows `Command1_Click` using `Static sayıcı As Integer` and `Command2_Click` using `Dim sayıcı As Integer`. The left form window shows a list of 5 messages: 3 for Command1 and 2 for Command2. The right form window shows a list of 5 messages: 3 for Command1 and 1 for Command2. Yellow callout boxes highlight the static variable declarations and the output messages.

```
Option Explicit

Private Sub Command1_Click()
Static sayıcı As Integer
sayıcı = sayıcı + 1
Print "Command1 " & sayıcı & ". kez tıklandı !"
End Sub

Private Sub Command2_Click()
Static sayıcı As Integer
sayıcı = sayıcı + 1
Print "Command2 " & sayıcı & ". kez tıklandı !"
End Sub
```

```
Option Explicit

Private Sub Command1_Click()
Static sayıcı As Integer
sayıcı = sayıcı + 1
Print "Command1 " & sayıcı & ". kez tıklandı !"
End Sub

Private Sub Command2_Click()
Dim sayıcı As Integer
sayıcı = sayıcı + 1
Print "Command2 " & sayıcı & ". kez tıklandı !"
End Sub
```

Command1 1. kez tıklandı!
Command1 2. kez tıklandı!
Command1 3. kez tıklandı!
Command2 1. kez tıklandı!
Command2 2. kez tıklandı!

Command1 1. kez tıklandı!
Command1 2. kez tıklandı!
Command1 3. kez tıklandı!
Command2 1. kez tıklandı!
Command2 1. kez tıklandı!

Değişkenler üzerine Özet:

İyi programlar yazmak isteyen programcı değişkenleri çok iyi kavramalıdır: Nerede? niçin? hangi tip? Bu konu kavranmadan da program yazabilirsiniz. Ancak programınız, binlerce kod satırı içeriyorsa, üzerindeki hakimiyetinizi kısa sürede kaybedeceksiniz, güvensiz çalışacak (bazen doğru, bazen yanlış), gereksiz bellek harcayacaksınız, zamanla neyin ne olduğunu artık sizde anlayamayacaksınız. Bu nedenle, değişkenlerin tanımı aşağıda özetlenecektir.

The image shows two code editors side-by-side. The left editor is titled 'Project1 - Form1 (Code)' and the right is 'Project1 - Form2 (Code)'. Both show code for a 'Click' event. The code in Form1 includes variables 'a', 'b', 'i1', 'j1', 'k', and 't'. The code in Form2 includes variables 'a', 'b', 'i2', 'j2', 'k', and 't'. Callouts explain the scope of these variables: 'Bu değişkenler form1 in içerisinde her yerde geçerli, diğer formlarda geçersiz' (These variables are valid everywhere in form1, not in other forms), 'Bu değişkenler form2 nin içerisinde her yerde geçerli, diğer formlarda geçersiz' (These variables are valid everywhere in form2, not in other forms), 'Bu değişkenler projedeki tüm formlarda geçerli' (These variables are valid in all forms in the project), 'Bu değişkenler sadece bu alt programda geçerli, alt program sonlandığında bellekten atılacaklar' (These variables are only valid in this sub-program, they will be removed from memory when the sub-program ends), and 'Bu değişkenler sadece bu alt programda geçerli. Form bellekte olduğu sürece içerikleri saklı kalır.' (These variables are only valid in this sub-program. The contents are saved in memory as long as the form is in memory).

Rem (Remark=açıklama, not) veya ' işaretini izleyen satırdaki her karakter programın çalışmasını etkilemeyen, programcının kendine veya programı inceleyenlere yönelik açıklamalardır. Programda olmasa da olur. Ancak program okunurken görsel anlaşılabilirliği artırdığından yararlıdır, sıkça kullanılır.

- Option Explicit** deyimini mutlaka kullanılmalı!
- Değişkenlerin tipi öngörülen içeriklerinin tipi ve aralığına uygun olmalı! Örneğin: 0-255 arasındaki tam sayılar için **Byte**, -32768 ile 32767 arasındaki sayılar için **Integer**, alfa sayısal içerikler için **String**.
- General declarations** bölümünde değişken tanımından elden geldiğince kaçınılmalı!
- Değişkenler elden geldiğince alt programlarda tanımlanmalı!
- Static** değişkenler zorunlu olmadıkça kullanılmamalı!

Kontrollerin özelliklerine erişim:

Her formun ve kontrolün çok sayıda özelliği vardır. Formun **Caption**, **Font** ve **Height** özellikleri örnek olarak verilebilir. Bazı özellikler tek değerli iken bazıları da çok değerlidir. Örneğimizde **Caption** ve **Height** tek değerli özelliklerdir. Ancak **Font** özelliği çok değerlidir, diğer bir deyişle **Font** özelliğinin de alt özellikleri vardır. Bunu **Font** değiştirmek istediğimizde açılan pencereden hemen anlayabiliriz. Fontun tipi, yazı biçimi, boyutu alt özelliklerdir. Program tasarım modunda iken bu özellikleri ilgili pencereden seçeriz.

Peki program çalışma modunda iken, bu özelliklere nasıl erişebilir, içerikleri başka değişkenlere nasıl aktarılabilir veya değiştirilebilir?

Genel olarak, bir formun özelliğine yeni bir değer atamak aşağıdaki gibidir:

Form adı.özellik.alt özellik=yeni değer

Örnek:

```
Form1.caption="Deneme programı"
```

```
Form1.height=500
```

```
Form1.font.name="Arial"
```

```
Form1.font.italic=True
```

```
Form1.font.size=16
```

Satırları Form1 in başlığını, yüksekliğini ve font özelliklerini program çalışırken (sanki tasarım anında özellikler penceresinden yapılmış gibi) değiştirir.

Bir kontrolün özelliğine yeni bir değer atamak aşağıdaki gibidir:

Form adı.kontrol adı.özellik.alt özellik=yeni değer

Örnek:

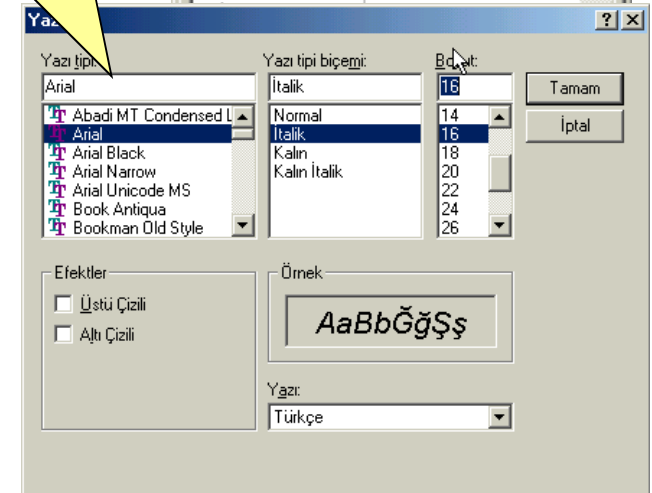
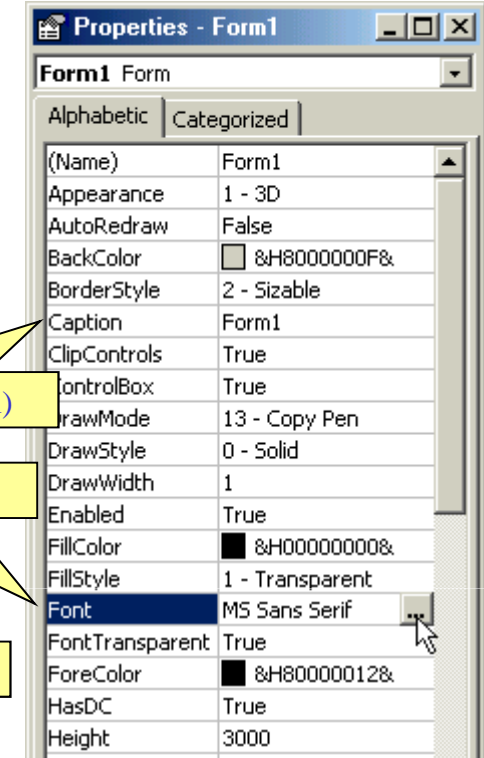
```
Form1.Command1.Enabled=False
```

Özelliği erişilmek istenen form veya kontrol kodlamanın yazıldığı form ile aynı ise form adı yazılmayabilir.

Caption özelliği(tek değerli)

Font özelliği(çok değerli)

Font özelliğinin alt özellikleri



Örnek:

Sağdaki program iki formu olan bir projeye aittir. Her ikisinin üzerinde Command1 kontrolleri vardır. Bu kontrollerin özellikleri tasarım modunda seçilmiştir.

Form1 içine yazılan kod satırları ile Form1 üzerindeki Command1 kontrolünün(Tamam) özellikleri Form2 üzerine, Form2 üzerindeki Command1 kontrolünün özellikleri de Form1 üzerine yazdırılmıştır.

Kod satırlarını ve çıktıları inceleyerek kavramaya çalışınız!

Ödev

The screenshot shows a Visual Basic 6 code editor window titled "Project1 - Form1 [Code]". The code is written in VBA and is currently selected on the "Click" event of the "Command1" control. The code is as follows:

```
Option Explicit

Private Sub Command1_Click()
Form2.Cls
Form2.Print "Aşağıdaki bilgiler Form1.Command1 in özellikleridir:"
Form2.Print Command1.Caption
Form2.Print Command1.Height
Form2.Print Command1.Width
Form2.Print Command1.Name
Form2.Print Command1.Font.Name
Form2.Print Command1.Font.Size
Form2.Print Command1.Font.Italic

Form1.Cls
Print "Aşağıdaki bilgiler Form2.Command1 in özellikleridir:"
Print Form2.Command1.Caption
Print Form2.Command1.Height
Print Form2.Command1.Width
Print Form2.Command1.Name
Print Form2.Command1.Font.Name
Print Form2.Command1.Font.Size
Print Form2.Command1.Font.Italic
End Sub

Private Sub Form_Load()
Form2.Show
End Sub

Private Sub Command2_Click()
Unload Form1
End Sub

Private Sub Form_Unload(Cancel As Integer)
Unload Form2
End Sub
```

Below the code editor, two windows are visible. The "Form1" window displays the output of the code when the "Command1" button is clicked on Form1. The output is:

```
Aşağıdaki bilgiler Form2.Command1 in özellikleridir:
Command1
615
1815
Command1
Arial
14.25
True
```

The "Form2" window displays the output of the code when the "Command1" button is clicked on Form2. The output is:

```
Aşağıdaki bilgiler Form1.Command1 in özellikleridir:
Tamam
375
855
Command1
MS Sans Serif
8.25
False
```

Matris deęişkenler(indisli deęişken, Dizi):

Şimdiye kadar görülen deęişkenlere basit deęişken adı verilir. Basit deęişkenler tek bir deęer içerirler. Uygulamada karşılaşılan çoęu problemin sadece basit deęişkenler ile kodlanması hem zahmetli hem de karmaşık ve çok uzun programların oluşmasına neden olur. Bir kişinin kimlik bilgilerini düşünün. Kişinin adı, doğum tarihi, doğum yeri, ana adı, baba adı, ... gibi çok sayıda bilgi için ayrı ayrı basit deęişken kullanılması durumunda her bir kişi için onlarca deęişken tanımlamak gerekecektir.

Bir amaca yönelik liste veya tablolar da çok satırlı çok kolonlu bilgilerdir. Listenin her bir elemanı için bir deęişken tanımlamak uygun olmayacaktır. Aşağıdaki öğrenci notları listesini ele alalım.

Adı	1.Ara sınav	2.Ara sınav	Final	Sayısal not	Harf notu
Yavuz AK	48	20	14	24	FF
Mine KARA	92	76	40	62	BB
Cem ARAS	88	100	94	94	AA

Bu tablo isimler, sayılar gibi farklı veri tipleri içermektedir. Bu tabloyu aşağıdaki gibi düzenleyelim:

Yavuz AK
Mine KARA
Cem ARAS

48	20	14	24
92	76	40	62
88	100	94	94

FF
BB
AA

Oluşan bu üç tabloya birer ad verelim:

A=

Yavuz AK
Mine KARA
Cem ARAS

B=

48	20	14	24
92	76	40	62
88	100	94	94

C=

FF
BB
AA

A, B ve C ye matematikte matris adı verilir. A ve C matrisleri 3 satırlı bir kolonlu, B matrisi 3 satırlı 4 kolonludur. A ve C alfa sayısal(**String**), B matrisi tam sayı(**Integer**) bilgiler içermektedir.

Bu matrislerin içeriklerini depolamak için A, B ve C gibi sadece üç değişken adı yeterlidir. Ancak her matrisin birden çok elemanı olduğundan değişkenin kaç satırı ve kaç kolonu olduğunun da belirtilmesi gerekir.

VB6 da bu tanımlama **Dim** deyimi ile aşağıdaki gibi yapılır:

Dim A(3,1) as string, B(3,4) as Integer, C(3,1) as string

Veya, satır veya kolon sayısı bir ise verilmesi zorunlu olmadığından:

Dim A(3) as string, B(3,4) as Integer, C(3) as string

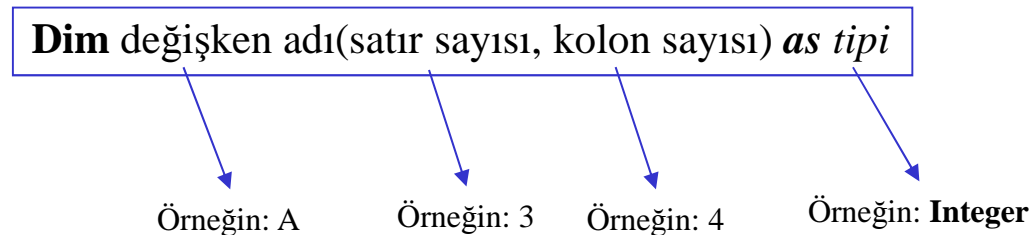
Böylece A değişkeninin **String** tipinde 3, B değişkeninin **Integer** tipinde 12 ve C değişkeninin de **String** tipinde 3 elemanı olacağı programa bildirilmiş olur. A ve C bir B iki boyutlu matrislerdir. Benzer şekilde üç boyutlu matrisler de tanımlanabilir. Ancak, üç boyutlu matrislere uygulamada nadiren rastlanır.

Bu matrislerin elemanları; doğrudan atanarak, **InputBox**, **Textbox**, **Listbox** veya benzer başka bir yolla okunarak veya bazıları (örneğin sayısal not ile Harf notu) hesaplanarak depolanır. İkinci öğrencinin bilgileri için

A(2)="Mine KARA" : B(2,1)=92 : B(2,2)=76 : B(2,3)=40
B(2,4)=0.25*B(2,1)+0.25*B(2,2)+0.50*B(2,3) ' Ara sınav ağırlıkları %25, final %50
'
'
C(2)="BB"

yazılabilir.

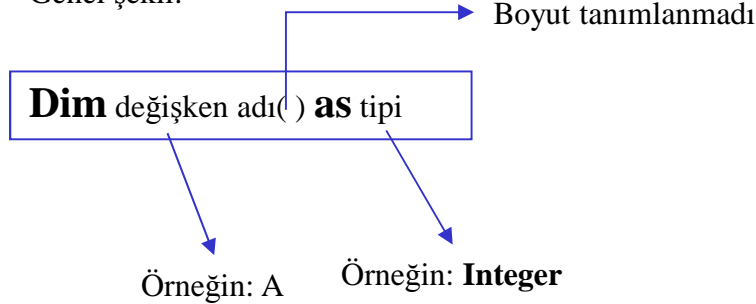
Matris değişken genel olarak aşağıdaki gibi tanımlanır:



Dinamik matris:

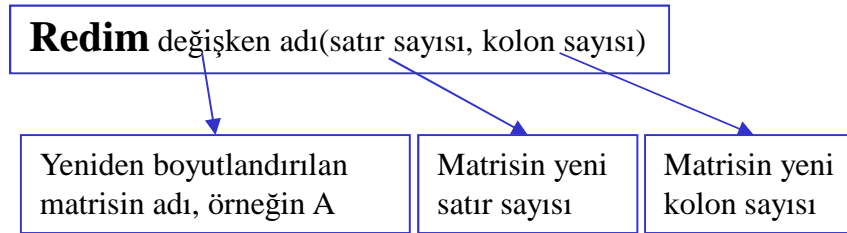
Bazı özel durumlarda matrisin boyutu program hazırlanırken bilinmez, kullanım anında belli olur. Bu durumda **Dim** veya **Public** deyimi ile dinamik matris tanımı yapılır. Bir matris, satır ve kolon sayısı verilmeden, parantez içi boş bırakılarak tanımlanırsa dinamik matris olur. Boyutu programın gerek duyulan her yerinde çok kez değiştirilebilir.

Genel şekli:



Dinamik matris, kullanılmadan önce, **Redim** (yeniden boyutlandır) deyimi ile yeniden boyutlandırılır.

Genel şekli:



Dim a() as double ' a dinamik matristir. Boyutu gerek duyulduğunda Redim ile tanımlanacak

'

Redim a(2,4) ' a matrisinin boyutu 2 satır 4 kolon olarak yeniden tanımlandı

'

Redim a(10) ' a matrisi bir boyutlu 10 elemanlı olarak yeniden tanımlandı

'

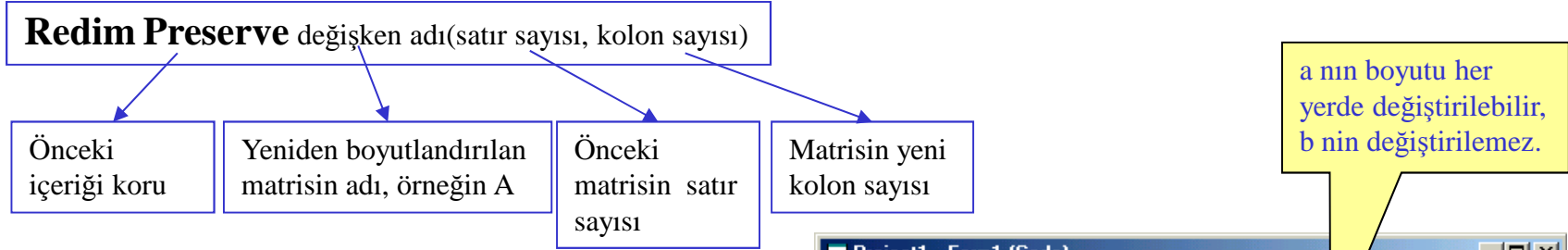
Redim a(7,3) ' a matrisinin boyutu 7 satır 3 kolon olarak yeniden tanımlandı

'

Redim Preserve deyimi:

Dinamik deęişkenlerin içerięi her **Redim** kullanıldığında silinir. Nadiren de olsa, içerięi daha önce tanımlanmış matrisin boyutu deęiştirilmek, ancak içerięi kaybolmasın istenir. Bunu sağlayabilmek için **Redim** deyimine **Preserve** (öncekileri koru) kelimesi eklenir.

Genel şekli:



Preserve kelimesinin kullanılması durumunda , satır sayısı aynı kalmak kaydıyla, matrisin önceki boyutu büyütüp küçültülebilir. Ancak, matris bir boyutlu ise gene bir, iki boyutlu ise iki boyutlu kalmalıdır. Bir boyutlu iki boyutluya veya iki boyutlu bir boyutluya dönüştürülemez. Matrisin küçültülmesi durumunda bazı elemanların önceki içerikleri kaybolacaktır.

Redim Preserve deyimini dikkatli kullanılmalı, VB6 help(yardım) okunarak kısıtlamalar anlaşılmalıdır.

```
Project1 - Form1 [Code]
Command1 Click
Option Explicit
Dim a() As Integer 'dinamik matris
Dim b(3, 5) 'dinamik olmayan matris
Dim i As Integer, j As Integer
Private Sub Command1_Click()
Dim n As Integer, m As Integer 'satır, kolon sayısı
ReDim a(3, 4) 'a boyutlandırıldı
a(2, 4) = 24: Print a(2, 4)
Print
'....
n = 3: m = 5
ReDim Preserve a(n, m) 'a boyutlandırıldı, içerięi kaybolmaz
Print a(2, 4)
ReDim Preserve a(n, 2) 'a boyutlandırıldı, bazı elemanlar kaybolacak
Print a(2, 4)
'....
End Sub
```

Bu eleman artık tanımsız. HATA!

A=

11	12	13	14	15
21	22	33	44	55
31	32	33	34	35

Redim Preserve a(3,7)

A=

11	12	13	14	15	0	0
21	22	33	44	25	0	0
31	32	33	34	35	0	0

Redim Preserve a(3,3)

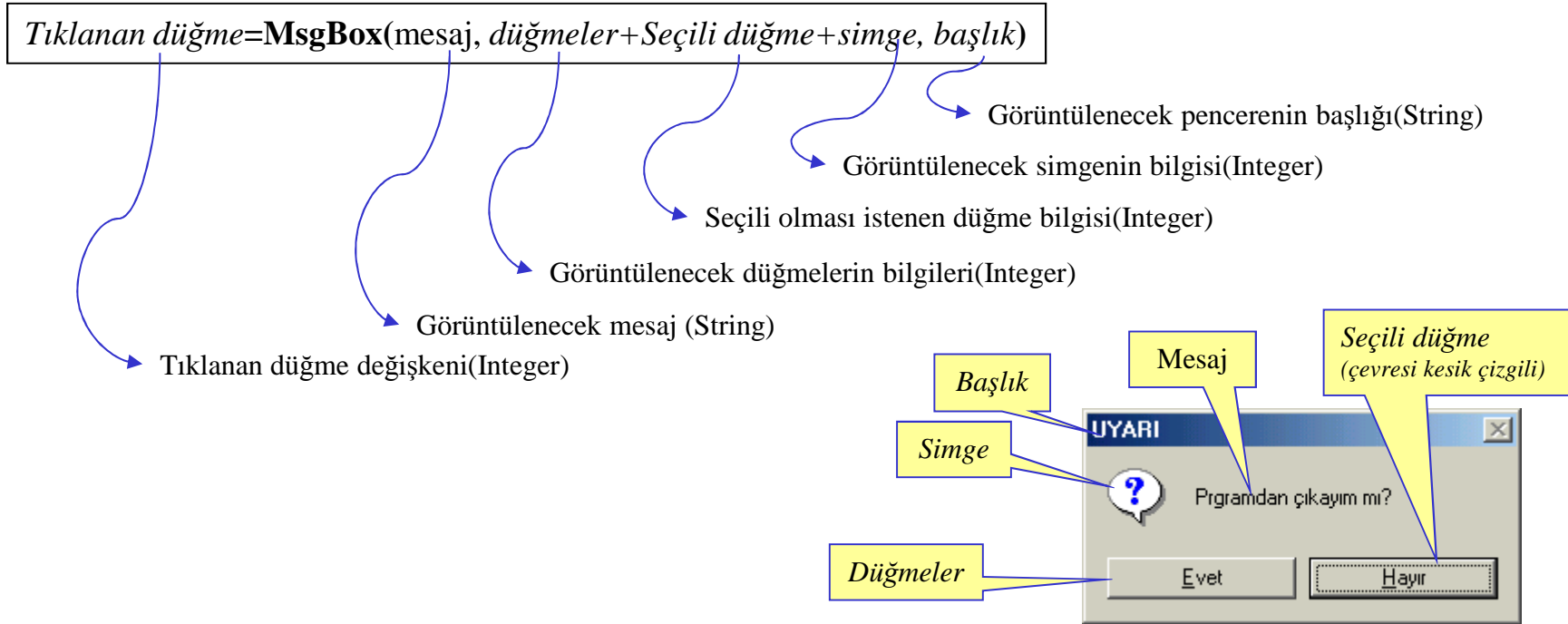
A=

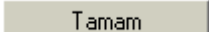
11	12	13
21	22	33
31	32	33

Kısa Giriş/Çıkış Fonksiyonları







Ekranda açılan bir pencerede bir mesaj görüntüleyerek(örneğin: uyarı mesajı) kullanıcının reaksiyonunu almak veya kullanıcıdan kısa veri almak(örneğin: şifre) için görsel programlarda **MsgBox** (Mesaj kutusu) ve **InputBox** (veri kutusu) fonksiyonları sıkça kullanılırlar.

MsgBox Fonksiyonu:




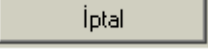

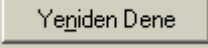
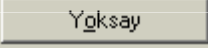


Açılan pencereye **mesaj** ile verilen yazıyı, pencerenin üstüne **başlık** ile verilen yazıyı yazar. Ayrıca **düğmeler** ile belirtilmiş düğmeleri ve **simge** ile belirtilmiş simgeyi görüntüler. Simge kullanıcının dikkatini çekmek için kullanılır. Kullanıcı mesajı okuduktan sonra karar vererek düğmelerden birini tıklar. **Tıklanan düğme** değişkeninin içeriği kontrol edilerek kullanıcının hangi düğmeği tıkladığı öğrenilir ve buna göre program akışı yönlendirilir. İtalik yazılı büyüklükleri kullanıp kullanmamak programcının isteğine bağlıdır, verilmesi zorunlu değildir. İtalik olmayan ifadeler verilmesi zorunlu olan en az bilgidir. **Düğmeler** bilgisi verilmezse sadece  görüntülenir.





düğmeler

Görünmesi istenen düğme	Verilmesi gereken düğmeler bilgisi
	vbOKonly
	vbYesNo
	vbOkCancel
	vbYesNoCancel
	vbAbortRetryIgnore
	vbRetryCancel

Tıklanan düğme

Kullanıcının tıkladığı düğme	Tıklanan düğme değişkeninin içeriği
	vbOk
	vbYes
	vbNo
	vbCancel
	vbAbort
	vbRetry
	vbIgnore

simge

Görünmesi istenen simge	Verilmesi gereken simge bilgisi
	vbCritical
	vbQuestion
	vbExclamation
	vbInformation

seçili düğme

Seçili olması istenen düğme	Verilmesi gereken seçili düğme bilgisi
1. düğme	vbDefaultButton1
2. Düğme	vbDefaultButton2
3. Düğme	vbDefaultButton3
4. Düğme	vbDefaultButton4

Örnek: Form/Program Kapatmak

Form üzerine yerleştirilen **Command1** düğmesinin **Caption** i ÇIK tır. Bu düğme tıklandığında bir **MsgBox** penceresinde kullanıcının uyarılması isteniyor. **MsgBox** penceresinin Başlığı UYARI, Mesaj Çıkayım mı? olsun. Evet Hayır düğmeleri ile soru işareti görüntülensin ve yanlışlıkla çıkılmasını önlemek için hayır düğmesi seçili olsun. Kullanıcı evet düğmesini tıklarsa program sonlandırılınsın hayır düğmesini tıklarsa program çalışmaya devam etsin.

Tasarım formunun görünümü ve program kodu aşağıda verilmiştir. Kodlar **Command1** (ÇIK) düğmesinin **Click** olayına yazılmıştır. Program çalıştırılıp ÇIK düğmesi tıklandığında **MsgBox** penceresi görüntülenecektir. Kullanıcı Evet düğmesini tıklarsa program sonlandırılacak aksi halde **MsgBox** penceresi kapanarak program çalışmaya devam edecektir.

Tasarım formu

MsgBox fonksiyonu çağrılıyor

Evet düğmesi tıklandıysa formu bellekten at

Program buradaki menüden kapatılırsa MsgBox penceresi görüntülenmez

Program buradan kapatılırsa MsgBox penceresi görüntülenmez

Tıklandığında MsgBox penceresi görüntülenir

```
Option Explicit

Private Sub Command1_Click()
' Programı sonlandırma isteği
' kullanıcı MsgBox üzerindeki
'EVET düğmesini tıklarsa program sonlanacak
Dim cevap As Integer, mesaj As String
Dim dugmeler As Integer, baslik As String
mesaj = "Çıkayım mı?"
dugmeler = vbYesNo + vbDefaultButton2 + vbQuestion
baslik = "UYARI"
cevap = MsgBox(mesaj, dugmeler, baslik)
If cevap = vbYes Then Unload Form1
End Sub
```

MsgBox fonksiyonunun ÇIK düğmesinin Click olayına kodlanmış olması iyi bir yol değildir. Çünkü kullanıcı programı formun sol üst veya sağ üst köşesinden kapadığında MsgBox penceresi görüntülenmeyecektir!

Son satırdaki If Then deyiminin yapısı için bir sonraki Kontrol deyimleri bölümüne bakınız!

Örnek: Form/Program Kapamak(iyileştirilmiş)

Form nereden kapatılırsa kapatılsın **Msgbox** penceresinin görüntülenmesini istiyoruz. Bunu sağlayabilmek için **MsgBox** ile ilgili kodların Formun **Unload** olayına yazılması gerekir.

Ödev

Programı kapa (bellekten at) isteği **unload** olayını tetikler.

Form nereden kapatılırsa kapatılsın **Unload** olayı tetiklenir.

MsgBox fonksiyonu çağrılıyor

Evet düğmesi tıklandıysa **Cancel=0** (kapa isteği uygulansın), aksi halde **Cancel=-1** (kapa isteği uygulanmasın)

```
Private Sub Command1_Click()  
Unload Form1  
End Sub  
  
Private Sub Form_Unload(Cancel As Integer)  
' Cancel=0 ise Unload işlemini gerçekleştirir  
' Cancel=-1 ise Unload işlemini iptal eder  
' Programı sonlandırma isteği  
' kullanıcı MsgBox üzerindeki  
'EVET düğmesini tıklarsa program sonlanacak  
Dim cevap As Integer, mesaj As String  
Dim duymeler As Integer, baslik As String  
mesaj = "Çıkayım mı?"  
duymeler = vbYesNo + vbDefaultButton2 + vbQuestion  
baslik = "UYARI"  
cevap = MsgBox(mesaj, duymeler, baslik)  
If cevap = vbYes Then Cancel = 0 Else Cancel = -1  
End Sub
```

Önceki Boyut
Iaşı
Boyut
_ Simge Durumuna Küçült
Ekranı Kapla
X Kapat Alt+F4

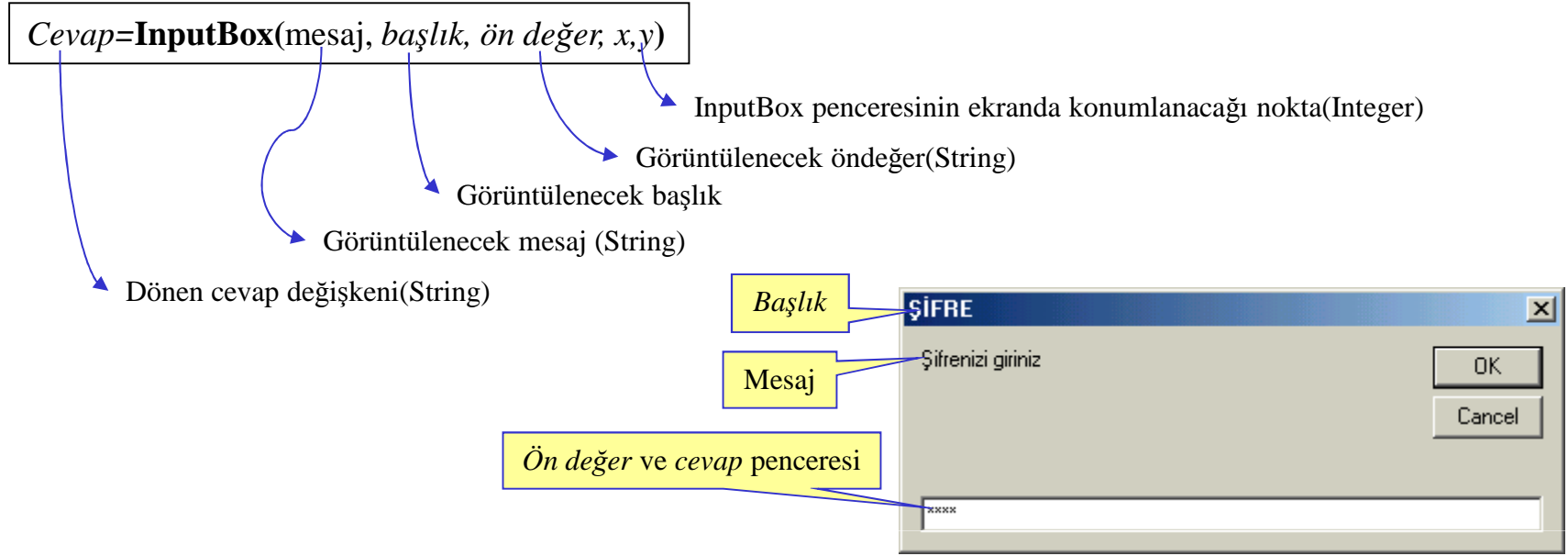
Form1
ÇIK düğmesini tıkla!
ÇIK Ka

Form1
ÇIK düğmesini tıkla!
ÇIK

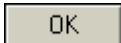
UYARI
Çıkayım mı?
Evet Hayır

Son satırdaki **If Then Else** deyiminin yapısı için bir sonraki Kontrol deyimleri bölümüne bakınız!

InputBox Fonksiyonu:



InputBox penceresi ekranın koordinatları x, y olan noktasında açılır. Pencereye **mesaj** ile verilen yazıyı, pencerenin üstüne **başlık** ile verilen yazıyı yazar. **Ön deęer** ve **cevap** penceresinde **Ön deęer**in içerięi görüntülenir. Kullanıcı mesajı okuduktan ve **cevap** bilgisini girdikten sonra düęmelerden birini tıklar.

 tıklanırsa veya Enter tuşlarırsa : **Cevap** penceresindeki bilgi **Cevap** deęişkenine aktarılır.

 tıklanırsa veya Esc tuşlarırsa **Cevap**="" olarak döner (boş **String**).

Cevap deęişkeninin içerięi kontrol edilerek program akışı yönlendirilir.

İtalik yazılı büyüklükleri kullanıp kullanmamak programcının isteęine baęlıdır, verilmesi zorunlu deęildir.

İtalik olmayan ifadeler kullanımı zorunlu olan en az bilgidir.

Örnek: Basit bir şifre sorgulaması

Program yüklenirken ve form görüntülenmeden önce kullanıcıya şifresi sorulsun istiyoruz. Bunun için **InputBox** penceresini kullanalım. Kullanıcı doğru şifreyi girerek **InputBox** üzerindeki **Ok** düğmesini tıkladığında form açılsın, kullanıcı programı kullanabilsin. Yanlış şifre girmesi durumunda program yüklenmesin. Şifremiz **Deve** olsun. Gerekli kodları formun **Load** olayına yazacağız.

The image illustrates the implementation of a password prompt in Visual Basic 6. It shows the design view of a form with a 'ÇIK' button, the code view for the 'Form_Load' event, and three simulated dialog boxes representing the password prompt. The code uses the **InputBox** function to prompt the user for a password. If the password is correct ('Deve'), the main form is displayed. If the password is incorrect, a **MsgBox** is shown with the message 'Şifre yanlış!, Program kapatılacak !'.

Tasarım formu

Öngörülen şifre

InputBox fonksiyonu çağırılıyor

MsgBox fonksiyonu çağırılıyor

InputBox penceresi

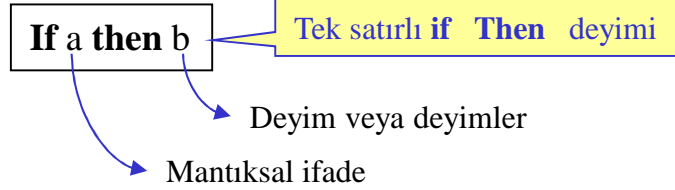
Form açılmaz

Form açılır

```
Option Explicit
Private Sub Form_Load()
Dim sifre As String, cevap As String
sifre = "Deve"
cevap = InputBox("Şifreyi giriniz", "ŞİFRE", "*****")
If cevap <> sifre Then
MsgBox ("Şifre yanlış!, Program kapatılacak !")
End 'Programı başka mesaj vermeksizin kapat
End If
End Sub
```

Kontrol Deyimleri

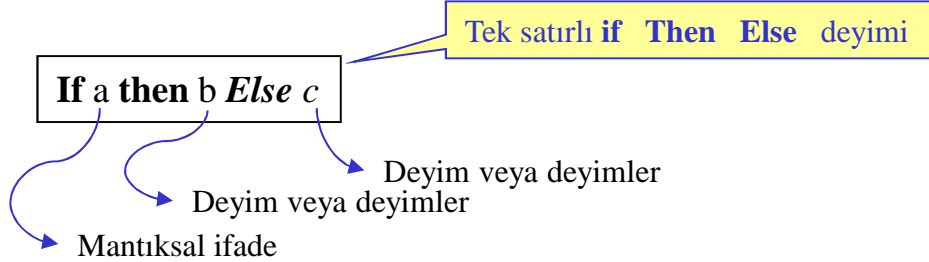
Program kodları normal olarak sıra ile satır satır işlenirler. Çoğu kez mantıksal bir ifadenin doğru veya yanlış olmasına bağlı olarak programın bazı kod parçalarının işlenmesi bazılarının da işlenmemesi istenir. Bu amaca yönelik, kullanımı benzer ancak yapıları farklı **If ... Then Else ...** ve **Select Case** kontrol deyimleri vardır.



Eğer a mantıksal ifadesi **True** (doğru) ise b ile verilen deyim veya deyimler işlenir, **False** (yanlış) ise işlenmez

Örnek: `If x<0 then x=Abs(x):t=x^0.5`

Eğer x negatif ise o zaman x in mutlak değeri x değişkenine aktarılır. x in kare kökünü t ye aktarılır.



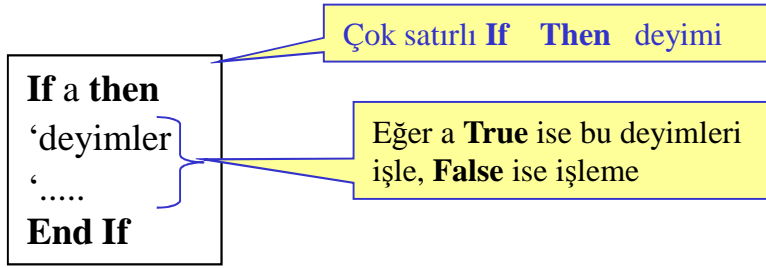
Eğer a mantıksal ifadesi **True** ise b ile verilen deyim veya deyimler işlenir, **False** ise c ile verilen deyim veya deyimler işlenir

Örnek: `If x<0 then x=Abs(x):t=x^0.5 Else t=x^0.5`

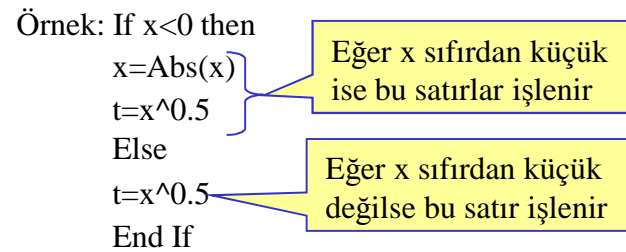
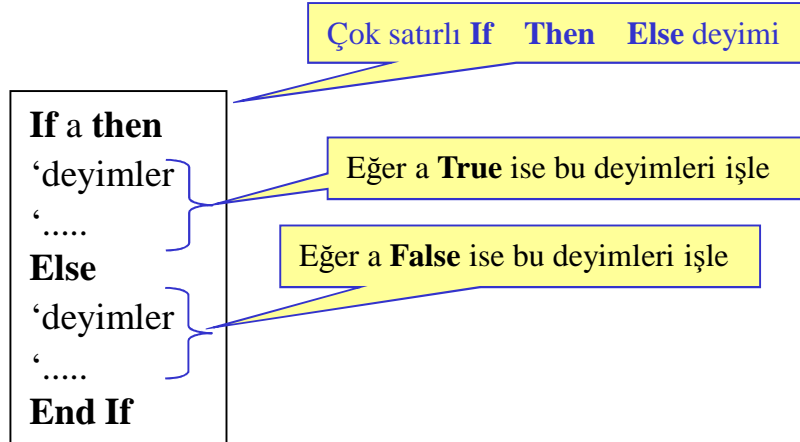
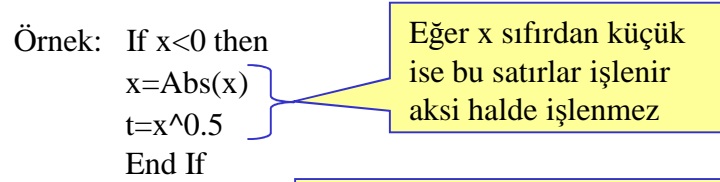
Eğer x negatif ise o zaman **Then** kelimesini izleyen deyimler, aksi halde **Else** kelimesini izleyen deyim işlenir.

`If cevap = vbYes Then Cancel = 0 Else Cancel = -1`

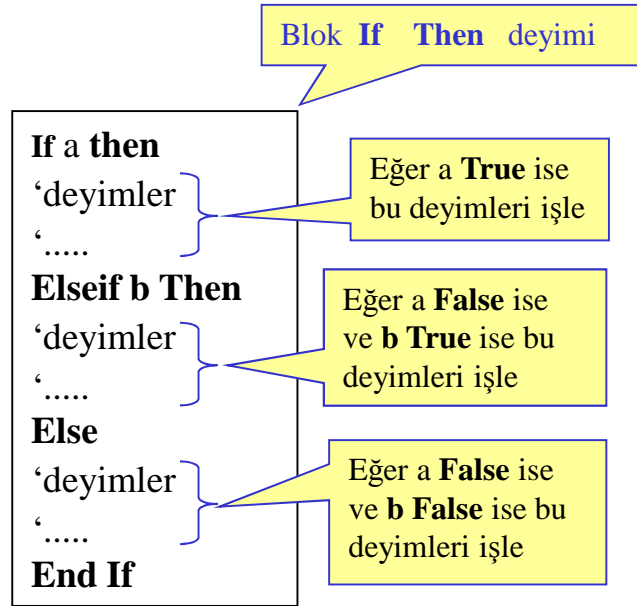
Eğer Cevap= vbYes ise Cancel=0 aksi halde Cancel= -1 olur



Eğer a mantıksal ifadesi **True** ise **If** satırı ile **End If** satırı arasındaki tüm deyimler işlenir, **False** ise işlenmez.



End If kelimelerinin ayrı yazıldığına dikkat ediniz !



Elseif kelimesinin bitişik yazıldığına dikkat ediniz !
End If kelimelerinin ayrı yazıldığına dikkat ediniz !

Bu yapının kullanımı programın anlaşılmasını zorlaştırır. Kullanımından kaçınılmalıdır. Bu yapı yerine **Select Case** deyimi kullanılmalıdır!

Örnek:

```

If cevap="Evet" then
  mesaj="Cevap değişkeni=Evet olduğundan bu satırları işledim"
  MsgBox(mesaj)
Elseif cevap="Hayır" then
  mesaj="Cevap değişkeni=Hayır olduğundan bu satırları işledim"
  MsgBox(mesaj)
Else
  mesaj="Cevap değişkeni ne Evet nede Hayır olduğundan bu satırları işledim"
  MsgBox(mesaj)
End If

```

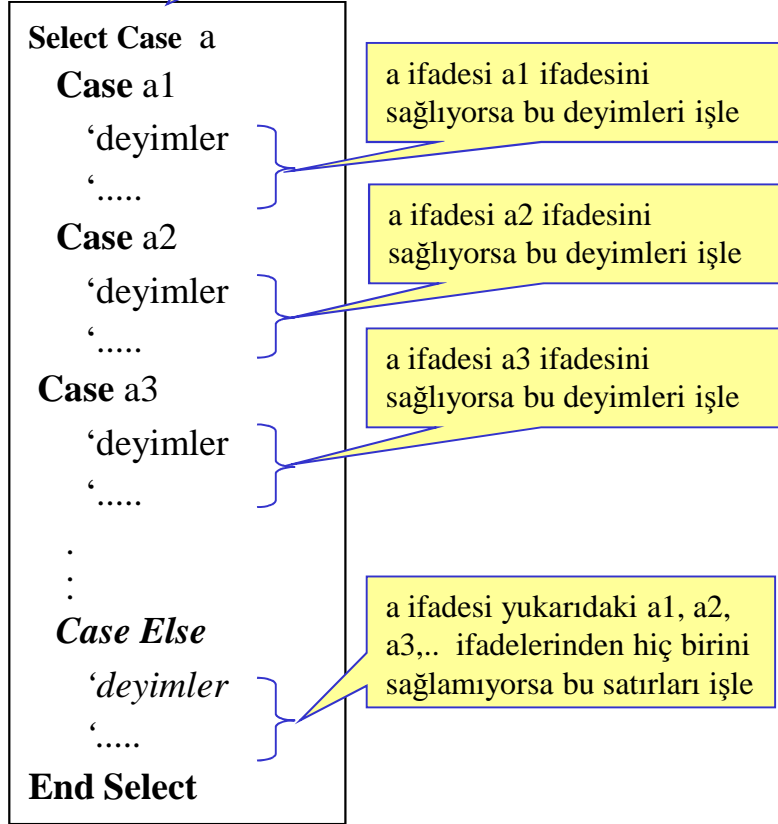
Aşağıdaki program parçası yukarıdaki ile aynı işi yapar:

```

If cevap="Evet" then
  mesaj="Cevap değişkeni=Evet olduğundan bu satırları işledim"
  MsgBox(mesaj)
End if
if cevap="Hayır" then
  mesaj="Cevap değişkeni=Hayır olduğundan bu satırları işledim"
  MsgBox(mesaj)
end If
If cevap<>"Evet" and cevap<>"Hayır" then
  mesaj="Cevap değişkeni ne Evet nede Hayır olduğundan bu satırları işledim"
  MsgBox(mesaj)
End If

```

Select Case deyimi



Case Else parçası kullanılmayabilir. Güvenli programlama açısından, daima kullanılması önerilir.

a: herhangi bir aritmetik veya **String** ifade

a1, a2, a3, ... aşağıdaki yapıda olabilir:

aritmetik ifade

String ifade

aritmetik ifade1 **TO** aritmetik ifade2

IS aritmetik operatör aritmetik ifade

aritmetik İfade1 < aritmetik ifade2 olmalı !

Örnek:

1) Herhangi bir sayının aşağıdaki durumu belirlenmek istensin.

Negatif mi?

Sıfır mı?

Pozitif mi?

Buna ait kod parçası:

```
Select Case sayi
  Case Is <0 'negatif mi?
    Print sayi; "negatif"
  Case Is =0 'sıfır mı?
    Print sayi; "sıfır"
  Case Is >0 'pozitif mi?
    Print sayi; "pozitif"
End Select
```

Veya

```
Select Case sayi
  Case Is <0 'negatif mi?
    Print sayi; "negatif"
  Case Is =0 'sıfır mı?
    Print sayi; "sıfır"
  Case Else 'pozitif olmalı !
    Print sayi; "pozitif"
End Select
```

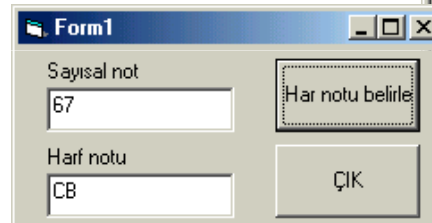
Ödev

2) Bir dersten alınan sayısal notun harf notu karşılığının aşağıdaki gibi olması isteniyor.

Sayısal not ≥ 90 ise harf notu AA
Sayısal not 85-89 arasında ise harf notu BA
Sayısal not 75-84 arasında ise harf notu BB
Sayısal not 65-74 arasında ise harf notu CB
Sayısal not 50-64 arasında ise harf notu CC
Sayısal not 45-49 arasında ise harf notu CD
Sayısal not 37-44 arasında ise harf notu DD
Sayısal not ≤ 36 arasında ise harf notu FF

Veri sayısal mı?
kontrol et

Veri sayısal değilse hata mesajı
ver ve alt programdan çık



Form üzerine 2 **Label**, 2 **TextBox** ve 2 **CommandButton** kontrolü yerleştirilmiştir. Kodlama **Command1** (Harf notu belirle) tıklama olayına yazılmıştır.

```
Project1 - Form1 (Code)
Command1 Click
Option Explicit
Private Sub Command1_Click()
Dim sayisalnot As Single, harfnotu As String
Dim Cevap As Byte ' MsgBox fonksiyonu için gerekli
If Not IsNumeric(Text1.Text) Then
Cevap = MsgBox("Sayı vemelisiniz !", vbExclamation, "HATA")
Exit Sub
End If
sayisalnot = Text1.Text
Select Case sayisalnot
Case Is < 0
harfnotu = "Sayısal not hatalı !"
Case 90 To 100
harfnotu = "AA"
Case 85 To 89
harfnotu = "BA"
Case 75 To 84
harfnotu = "BB"
Case 65 To 74
harfnotu = "CB"
Case 50 To 64
harfnotu = "CC"
Case 45 To 49
harfnotu = "CD"
Case 37 To 44
harfnotu = "DD"
Case Is <= 36
harfnotu = "FF"
Case Else
harfnotu = "Sayısal not hatalı !"
End Select
Text2.Text = harfnotu
End Sub
```

Sayısalnot < 0 ise

$0 \leq \text{Sayısalnot} \leq 100$ ise

Sayısalnot > 100 ise

3) Verilen isimlerden Türkçe ye özgü harfler ile başlayanlar ve diğer harfler ile başlayanlar ayrı pencerelerde görüntülensin isteniyor.

Form üzerine iki **label**, bir **Textbox**, iki **Listbox** ve iki **Commandbutton** yerleştirilmiştir. **Text1** isim girilmesi için, **List1** Türkçe harfle başlayanların görüntülenmesi için ve **List2** de diğer harfler ile başlayanların görüntülenmesi için kullanılacaktır. **Text1** penceresine girilen isim Belirle(**command1**) tıklandığında ilgili pencereye aktarılacaktır.

Ödev

Text1.text içeriği büyük harflere çevriliyor ve büyük değişkenine aktarılıyor

buyuk değişkeninin ilk harfi ilkharf değişkenine aktarılıyor

Türkçe harflerle başlayanlar belirleniyor ve Text1.text içeriği List1 penceresinin içeriğine ekleniyor

Diğer harfler ile başlayanlar List1 penceresinin içeriğine ekleniyor

```
Private Sub Command1_Click()  
Dim buyuk As String, ilkharf As String  
buyuk = UCase(Text1.Text)  
ilkharf = Left(buyuk, 1)  
Select Case ilkharf  
Case "Ç" To "ç", "Ğ", "İ", "Ö", "Ş", "Ü"  
List1.AddItem Text1.Text  
Case Else  
List2.AddItem Text1.Text  
End Select  
End Sub
```

Form1

şükrü

Belirle

Türkçe harfle başlayanlar

Diğer harflerle başlayanlar

çağla şükrü

ayten hasan

ÇIK

4) Aşağıdaki programın text1 penceresine cebinizdeki paranın miktarını yazarak tamam düğmesini tıklayın (Eğlenceli bu örneğin amacı **If Then Else** deyimi ve **MsgBox** fonksiyonunun değişik kullanımını vurgulamaktır. Dikkatle inceleyiniz)

Ödev

Para tipi

‘ işareti ve sonrası (satır sonuna kadar) olmasa da olur!

Aşağıdaki satırları işleme! Alt programdan çık!

Çok satırlı mesaj!

Soru işareti

İkinci düğme seçili

Evet ve hayır düğmeleri

```

Project1 - Form1 (Code)
Command1 Click
Option Explicit

Private Sub Command1_Click()
Dim miktar As Currency 'girilen para miktarı
Dim cevap As Integer, dugmeler As Integer 'Msgbox da kullanılacak değişkenler
Dim mesaj As String, baslik As String 'Msgbox da kullanılacak değişkenler
If Not (IsNumeric(Text1.Text)) Then MsgBox ("Bir sayı vermelisiniz !"): Exit Sub
miktar = Text1.Text
If miktar < 0 Then MsgBox ("Eksi!, Yani borçlu musunuz?"): Exit Sub
If miktar = 0 Then MsgBox ("Vah vah. Meteliksiz?"): Exit Sub
If miktar < 5000000# Then MsgBox ("İyi, günü belki idare eder !"): Exit Sub
If miktar >= 5000000# And miktar <= 10000000# Then MsgBox ("İyi, günü idare eder !"): Exit Sub
If miktar >= 10000000# And miktar <= 20000000# Then MsgBox ("Çok iyi, günü kesin idare eder !"): Exit Sub
mesaj = "Aman kimse görmesin, borç ister!" + Chr(10) + Chr(13) + "Bana on milyon verebilir misin?"
dugmeler = vbYesNo + vbDefaultButton2 + vbQuestion
baslik = "UYARI/SORU"
cevap = MsgBox(mesaj, dugmeler, baslik)
If cevap = vbYes Then MsgBox ("Sağol !") Else MsgBox ("Seni gidi cimri!")
End Sub

Private Sub Command2_Click()
Unload Form1
End Sub

Private Sub Form_Unload(Cancel As Integer)
' programı sonlandırma isteği
' kullanıcı MsgBox üzerindeki EVET düğmesini tıklarsa program sonlanacak
If MsgBox("Çıkayım mı?", vbYesNo + vbDefaultButton2 + vbQuestion, "UYARI") = vbNo Then Cancel = -1
End Sub

```

5) Bir önceki programın Select Case deyimi ile yazılmış şekli.

Ödev

The screenshot displays the Visual Basic 6 IDE. The main window is titled "Project1 - Form1 (Code)" and shows the following code:

```
Option Explicit

Private Sub Command1_Click()
Dim miktar As Currency 'girilen para miktarı
Dim cevap As Integer, duymeler As Integer 'Msgbox da kullanılacak değişkenler
Dim mesaj As String, baslik As String 'Msgbox da kullanılacak değişkenler
If Not (IsNumeric(Text1.Text)) Then MsgBox ("Bir sayı vermelisiniz !"): Exit Sub
miktar = Text1.Text
Select Case miktar
Case Is < 0
MsgBox "Eksi!, Yani borçlu musunuz?": Exit Sub
Case 0
MsgBox "Vah vah. Meteliksiz?": Exit Sub
Case Is < 5000000#
MsgBox "İyi, günü belki idare eder !": Exit Sub
Case 5000000# To 10000000#
MsgBox "İyi, günü idare eder !": Exit Sub
Case 10000000# To 20000000#
MsgBox "Çok iyi, günü kesin idare eder !": Exit Sub
Case Else
mesaj = "Aman kimse görmesin, borç ister!" + Chr(10) + Chr(13) + "Bana on milyon verebilir misin?"
duymeler = vbYesNo + vbDefaultButton2 + vbQuestion
baslik = "UYARI/SORU"
cevap = MsgBox(mesaj, duymeler, baslik)
If cevap = vbYes Then MsgBox "Sağol !" Else MsgBox "Seni gidi cimri!"
End Select
End Sub

Private Sub Command2_Click()
Unload Form1
End Sub
```

The runtime window shows a form titled "Form1" with a text box containing "50E6" and two buttons: "Tamam" and "ÇIK". A message box titled "UYARI/SORU" is displayed with the text "Aman kimse görmesin, borç ister! Bana on milyon verebilir misin?" and buttons "Evet" and "Hayır". Another message box titled "Project1" is displayed with the text "Sağol !" and a "Tamam" button.

Ödev: basit bir oyun programı

Yanda görülen form üzerinde 9 adet CommandButton düğmesi, iki Label ve bir TextBox vardır. Label1 üzerinde Hazine burada yazmaktadır. Hazineyi bulama programı yazılacaktır.

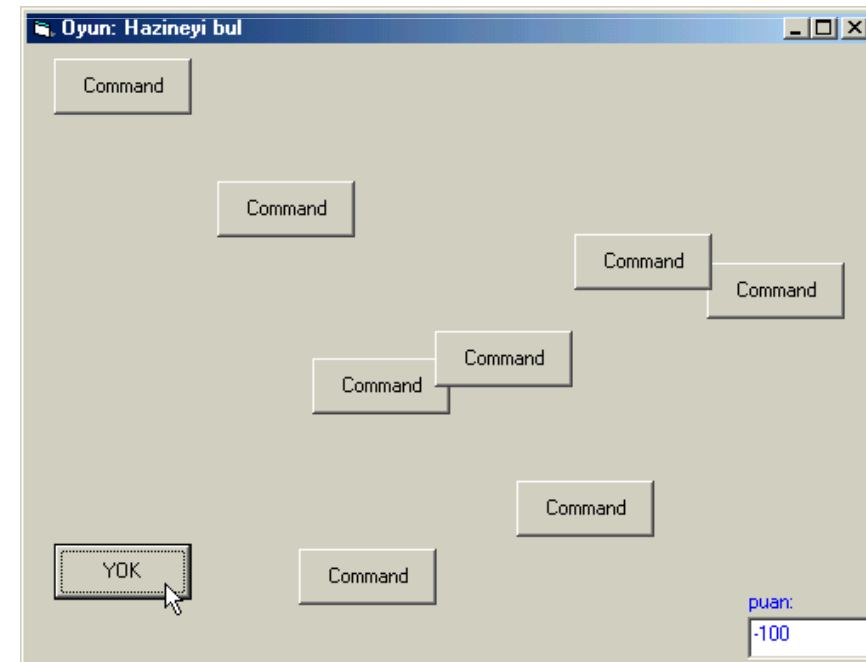
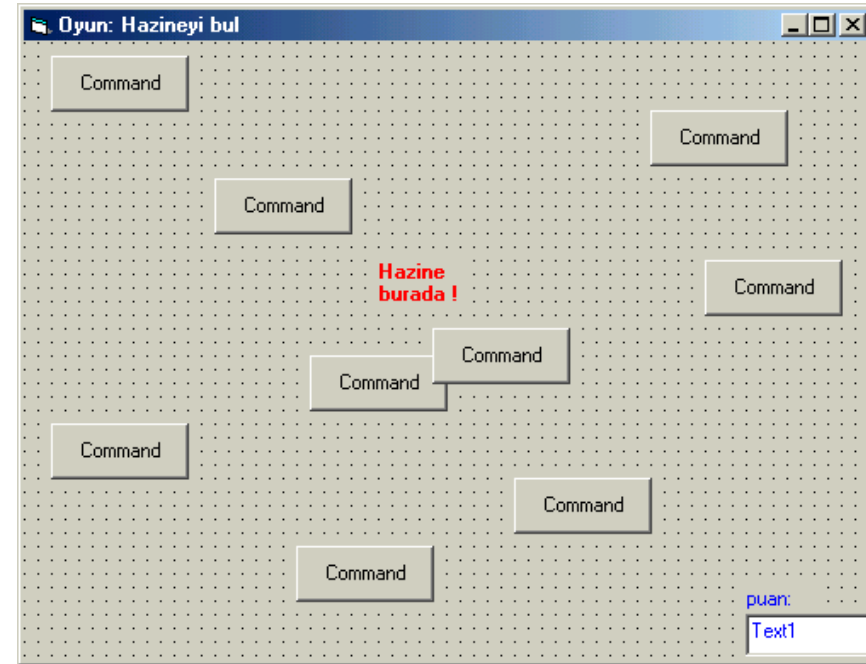
Oyunun kuralı:

Program çalıştırıldığında Command düğmeleri form üzerinde rasgele konumlanacaktır. Label1 bu düğmelerden rasgele birinin altına saklanacaktır. Oyuncu hazinenin hangi düğme altında olduğunu tahmin ederek düğmeyi tıklayacaktır. Hazine tıklanan düğme altında değilse oyuncu 100 puan kaybedecek ve düğme üzerine YOK yazılacaktır. Hazine düğme altında ise oyuncu 900 puan kazanacak ve command düğmesi sağa kayarak Hazine burada yazısı görünecektir. Ayrıca kullanıcıya *Aferin hazineyi buldunuz* mesajı verilecektir. Oyuncu mesaj penceresinin tamam düğmesini tıkladığında oyun yeniden başlayacak ancak puanı sıfırlanmayacaktır.

Her tıklamada düğmelerin yeri değişecektir. Hazine bulunduğu anda alınan puan sıfırlanmayacak oyuncu devam ettikçe puanı artacak veya azalacaktır.

Alınan puan text1 penceresinde görülecektir.

Düğmeler rasgele konumlanacağı için bazıları üst üste düşer. Kullanıcı formu tıklayarak yeni bir yerleşim oluşturacaktır. Bu durumda puan ve düğmelerin üstündeki yazılar değişmeyecektir.



Döngü Deyimleri

Matematikte karşılaştığımız

$$a = \sum_{i=1}^n i^2, \quad b = \prod_{i=1}^n i$$

İfadelerinden birincisi birden n ye kadar tam sayıların karelerinin toplamı, ikincisi de 1 den n ye kadar tam sayıların çarpımı, yani n! (n faktoriyel) anlamındadır. Birincisi; belli bir alt sınırdan(=1) belli bir üst sınıra kadar(=n) tam sayıların karelerinin türetilmesi ve toplama işleminin tekrarlanması gerektirir. İkincisi; belli bir alt sınırdan(=1) belli bir üst sınıra kadar(=n) tam sayıların türetilmesi ve çarpma işleminin tekrarlanması gerektirir.

Programlama dillerinde bu ve benzeri problemleri kodlayabilmek için döngüler kullanılır. Döngü, programın bir parçasının belli sayıda tekrar tekrar çalışmasıdır. Hemen her programda az yada çok kullanılan değişik döngü yapıları vardır. VB6 da aşağıdaki döngüler kullanılır:

For Next

Do While Loop

Do Loop While

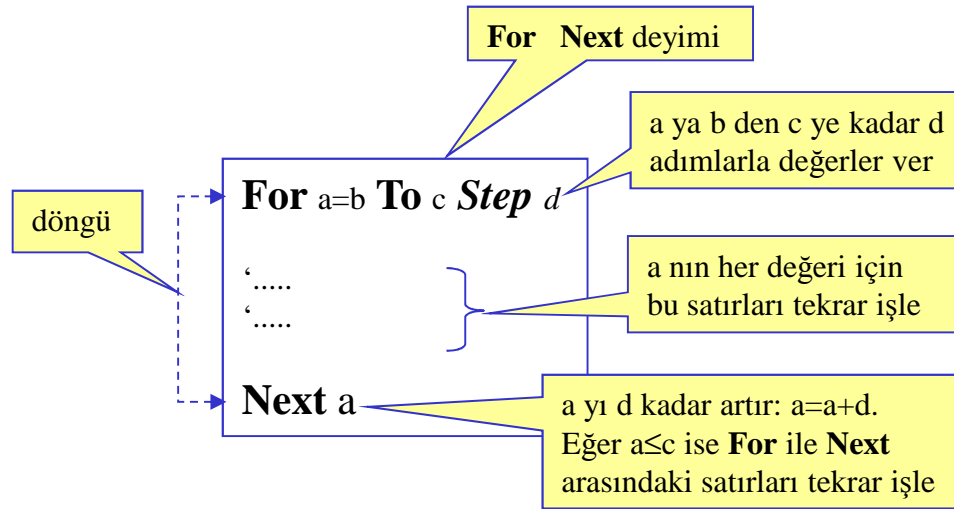
Do Until Loop

Do Loop Until

While Wend

Bu döngüler programın bir veya birden çok satırının, belli bir şart sağlanıncaya kadar, tekrar işlenmesini sağlarlar.

For Next döngüsü:



a: değişken
b: a'nın alacağı ilk değer(sayı veya aritmetik ifade)
c: a'nın alacağı son değer(sayı veya aritmetik ifade)
d: a'nın adımı (sayı veya aritmetik ifade).

- d pozitif veya negatif olabilir. Pozitif ise a değişkeni b den c ye d adımlarla artarak, negatif ise azalarak değerler alır.
- a'nın her değeri için **For Next** satırları arasındaki satırlar tekrar işlenir.
- a>c olduğunda döngü sona erer.
- d=1 ise **Step d** kısmı yazılmayabilir. Bu durumda adım otomatik olarak hep 1 alınır.

• b, c ve d nin değerlerinin döngü içinde değişmemesi gerekir(bunlara atama yapılmamalı!).

Goto veya **on Goto** gibi deyimler ile döngü içine atlanmamalı!

Örnekler:

$a = \sum_{i=1}^{100} i^2$ ifadesinden a değerini hesaplayan program parçası:

' 1 den 100 kadar tam sayıların karelerinin toplamı

Dim i as byte, a as double

a=0

For i= 1 to 100 step 1

a=a+i*i

Next i

Veya

' 1 den 100 e kadar tam sayıların karelerinin toplamı

Dim i as byte, a as double

a=0

For i= 1 to 100

a=a+i*i

Next i

$b = \prod_{i=1}^{25} i$ ifadesinden b değerini hesaplayan program parçası:

' 1 den 25 e kadar tam sayıların çarpımı (25!)

Dim i as byte, b as double

b=1

For i= 1 to 25 step 1

b=b*i

Next i

Veya

' 1 den 25 e kadar tam sayıların çarpımı(25!)

Dim i as byte, b as double

b=1

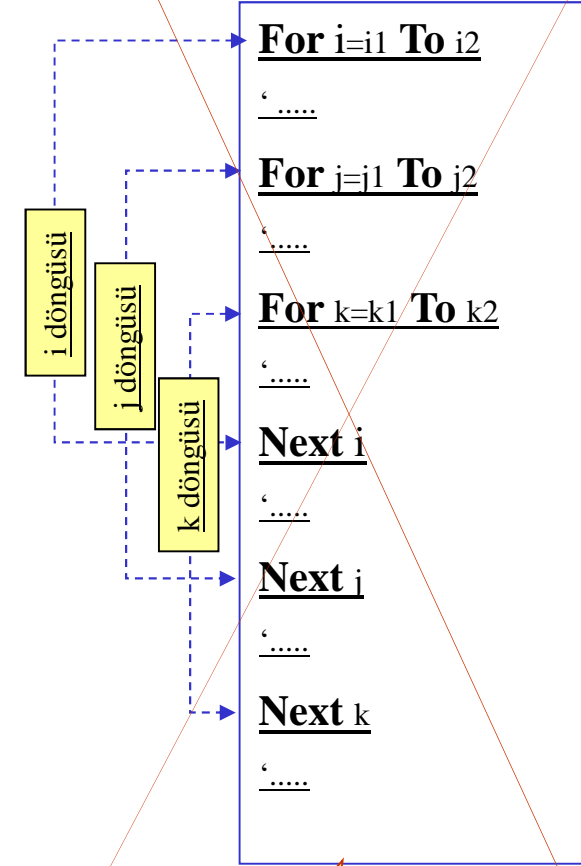
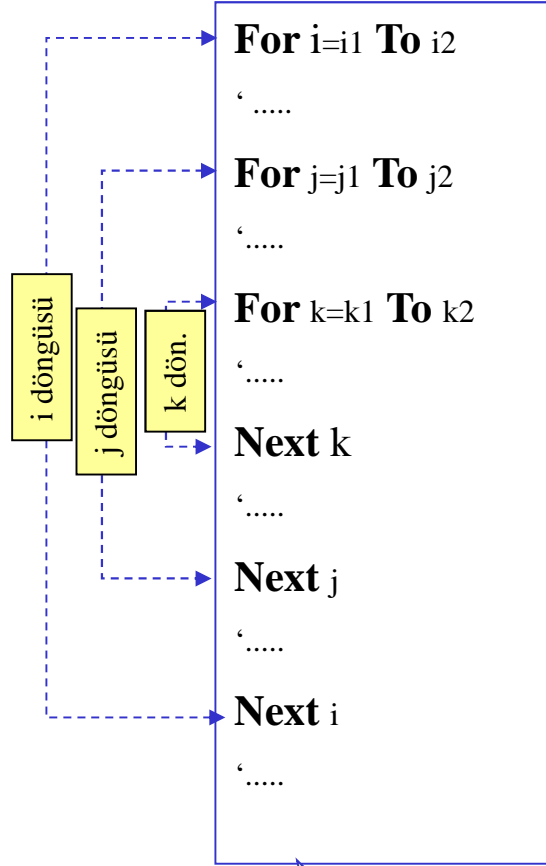
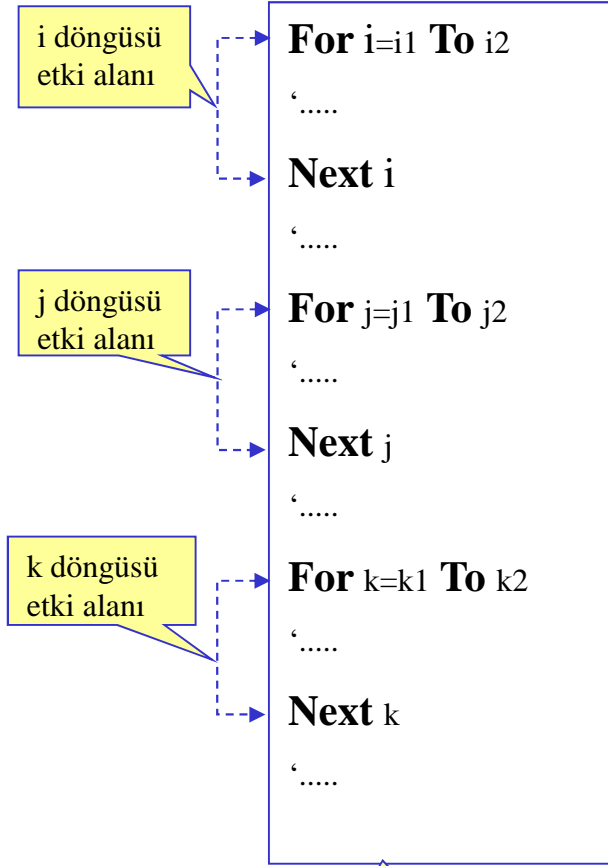
For i= 1 to 25

b=b*i

Next i

İç içe For Next döngüleri:

Bir programda birbirinden bağımsız çok sayıda **For Next** deyimi olabildiği gibi iç içe döngüler de olabilir. Ancak, iç içe döngülerin etki alanları kesişmemelidir.



Örnekler:

$$a = \sum_{i=1}^4 \sum_{j=1,3,\dots}^7 (i + j)^2$$

Değerini hesaplayan program parçası:

Dim a as double, i as byte, j as byte

a=0

For i= 1 to 4

For j=1 to 7 step 2

a=a+(i+j)^2

Next j

Next i

$$a = \left(\sum_{i=1}^4 i \right) \left(\sum_{j=1,3,\dots}^7 j^2 \right)$$

Değerini hesaplayan program parçası:

Dim a as double, b as double

Dim i as byte, j as byte

a=0

For i= 1 to 4

a=a+i

Next i

b=0

For i=1 to 7 step 2

b=b+j^2

Next i

a=a*b

$$a = \sum_{i=1}^4 \left[i \sum_{j=1,3,\dots}^7 (i + j)^2 \right]$$

Değerini hesaplayan program parçası:

Dim a as double, b as double

Dim i as byte, j as byte

a=0

For i= 1 to 4

b=0

For j=1 to 7 step 2

b=b+(i+j)^2

Next j

a=b*i

Next i

A=

10	12	14	16	18
20	22	24	26	28
30	32	34	36	38

A Matrisinin elemanlarını türeterek form

üzerine yazan program parçası:

Dim a as double, i as byte, j as byte

For i= 10 to 30 step 10

For j=0 to 8 step 2

a=i+j

Print a,

Next j

print

Next i

3x5 boyutlu bir A matrisinin elemanlarını -500 ile +500 arasında rasgele sayılar ile türeterek form üzerine yazan program parçası:

Dim a(3,5) as double

Dim i as byte, j as byte

For i= 1 to 3

For j=1 to 5

a(i,j)=-500+1000*Rnd

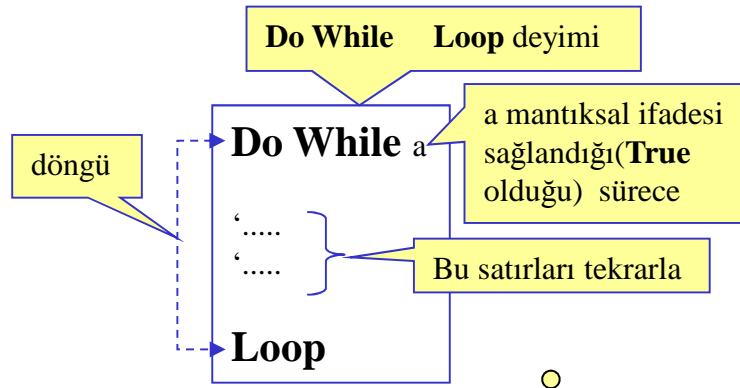
Print a (i,j),

Next j

print

Next i

Do While Loop:



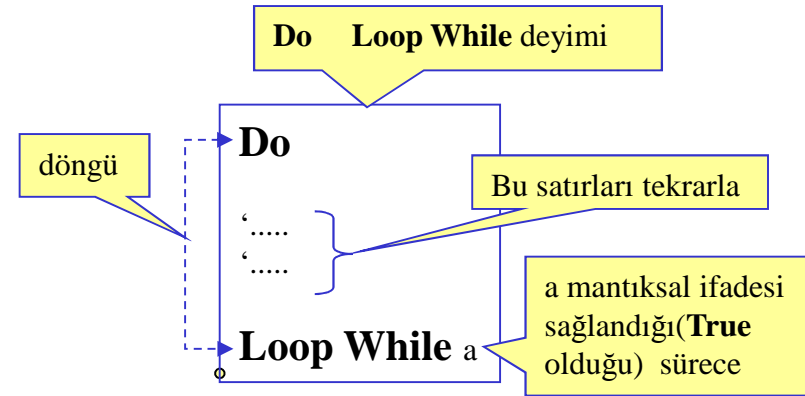
a: Mantıksal ifade

A mantıksal ifadesi **True** olduğu sürece **Do While** ile **Loop** arasındaki satırlar tekrar tekrar işlenir. Döngü içerisinde a mantıksal ifadesini **False** yapacak bir atama yapılması gerekir. Aksi halde **döngü sonsuza dek tekrarlanır**. Bu da programın kilitlenmesi veya hata ile kırılmasına neden olur.

Aralarındaki farka dikkat:

a **False** ise soldaki döngüye hiç girilmez. sağdakine bir kez girilir.

Do Loop While :



a: Mantıksal ifade

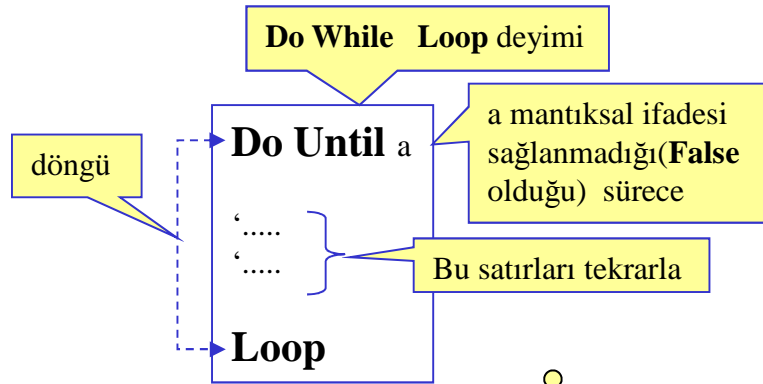
Döngü içi işlenir. A mantıksal ifadesi kontrol edilir. a **True** olduğu sürece **Do** ile **Loop While** arasındaki satırlar tekrar tekrar işlenir. Döngü içerisinde a mantıksal ifadesini **False** yapacak bir atama yapılması gerekir. Aksi halde **döngü sonsuza dek tekrarlanır**. Bu da programın kilitlenmesi veya hata ile kırılmasına neden olur.

Örnekler: $a = \sum_{i=1}^{100} i^2$ değerini hesaplayan program parçası:

```
' 1 den 100 kadar tam sayıların karelerinin toplamı
Dim i as byte, a as double
a=0:i=0
Do While i<100 ' i 100 den küçük olduğu sürece tekrarla
i=i+1:a=a+i*i
Loop
```

```
' 1 den 100 kadar tam sayıların karelerinin toplamı
Dim i as byte, a as double
a=0:i=0
Do
i=i+1:a=a+i*i
Loop While i<100' i 100 den küçük olduğu sürece tekrarla
```

Do Until Loop:



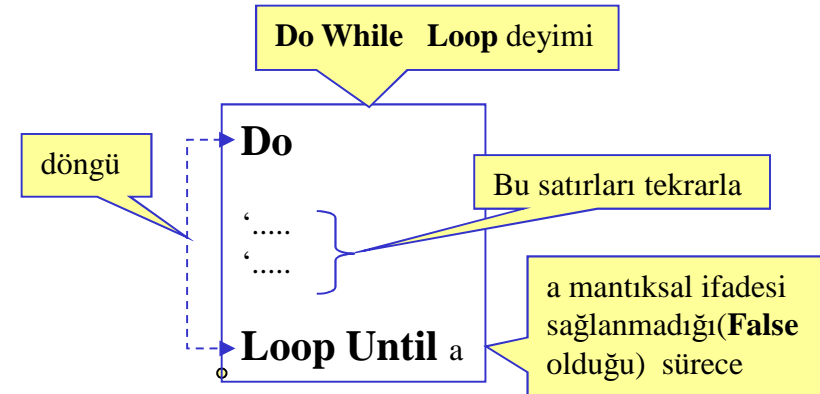
a: Mantıksal ifade

A mantıksal ifadesi **False** olduğu sürece **Do Until** ile **Loop** arasındaki satırlar tekrar tekrar işlenir. Döngü içerisinde a mantıksal ifadesini **True** yapacak bir atama yapılması gerekir. Aksi halde **döngü sonsuza dek tekrarlanır**. Bu da programın kilitlenmesi veya hata ile kırılmasına neden olur.

Aralarındaki farka dikkat:

a True ise soldaki döngüye hiç girilmez. sağdakine bir kez girilir.

Do Loop Until:



a: Mantıksal ifade

Döngü içi işlenir. A mantıksal ifadesi kontrol edilir. a **False** olduğu sürece **Do** ile **Loop Until** arasındaki satırlar tekrar tekrar işlenir. Döngü içerisinde a mantıksal ifadesini **True** yapacak bir atama yapılması gerekir. Aksi halde **döngü sonsuza dek tekrarlanır**. Bu da programın kilitlenmesi veya hata ile kırılmasına neden olur.

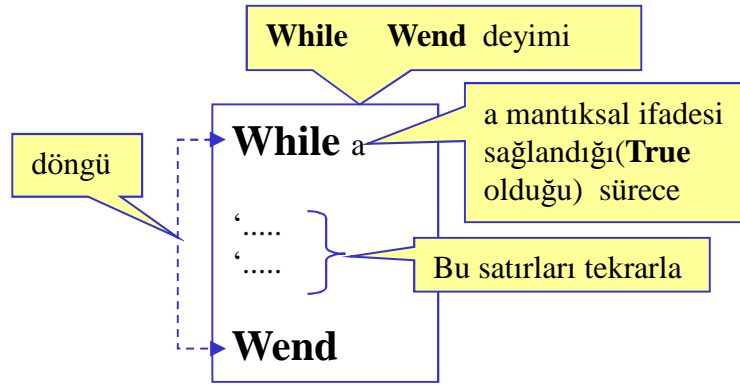
Örnekler: $a = \sum_{i=1}^{100} i^2$

değerini hesaplayan program parçası:

```
‘ 1 den 100 kadar tam sayıların karelerinin toplamı  
Dim i as byte, a as double  
a=0:i=0  
Do Until i=100 ‘ i 100 oluncaya kadar tekrarla  
i=i+1:a=a+i*i  
Loop
```

```
‘ 1 den 100 kadar tam sayıların karelerinin toplamı  
Dim i as byte, a as double  
a=0:i=0  
Do  
i=i+1:a=a+i*i  
Loop Until i=100 ‘ i 100 oluncaya kadar tekrarla
```

While Wend:



a: Mantıksal ifade

a mantıksal ifadesi **True** olduğu sürece **While** ile **Wend** arasındaki satırlar tekrar tekrar işlenir. Döngü içerisinde a mantıksal ifadesini **False** yapacak bir atama yapılması gerekir. Aksi halde **döngü sonsuza dek tekrarlanır**. Bu da programın kilitlenmesi veya hata ile kırılmasına neden olur.

Örnekler:

$a = \sum_{i=1}^{100} i^2$ değerini hesaplayan program parçası:

```
' 1 den 100 kadar tam sayıların karelerinin toplamı
Dim i as byte, a as double
a=0:i=0
While i<100 ' i 100 den küçük olduğu sürece tekrarlar
i=i+1:a=a+i*i
Wend
```

Veya:

```
' 1 den 100 kadar tam sayıların karelerinin toplamı
Dim i as byte, a as double
a=0:i=1
While i<=100 ' i 100 den küçük veya 100 olduğu sürece tekrarlar
a=a+i*i: i=i+1:
Wend
```

Veya:

```
' 1 den 100 kadar tam sayıların karelerinin toplamı
Dim i as byte, a as double
a=0:i=100
While i>0 ' i 0 dan büyük olduğu sürece tekrarlar
a=a+i*i: i=i-1
Wend
```

Veya:

```
' 1 den 100 kadar tam sayıların karelerinin toplamı
Dim i as byte, a as double
a=0:i=101
While i>1 ' i 1 den büyük olduğu sürece tekrarlar
i=i-1:a=a+i*i
Wend
```

Döngüden Çıkmak:

Bazı durumlarda, *nadiren de olsa*, döngü tamamlanmadan çıkmak gerekebilir. Örneğin: Hata oluşması veya kullanıcının vazgeçmesi gibi. Bu durumda aşağıdaki çıkış deyimleri kullanılabilir.

Exit For : For Next döngüsünden çıkmak için

Exit Do : Do Loop döngüsünden çıkmak için

While Wend döngüsünden çıkış deyim yoktur!

Öneriler:

- Başlangıç, bitiş ve adımı kodlama sırasında belli olan döngülerde sadece **For Next** kullanınız.
- Diğer durumlarda **Do Loop** kullanınız
- While Wend** deyiminden kaçınınız.
- Döngülerde sayıcı olarak i, j, k ve l değişkenlerini tercih ediniz.
- Döngülerde sayıcı olarak ondalık değişken kullanmaktan kaçınınız.

Örnek:

1 den 2 ye kadar 0.1 adımlarla 11 adet sayı türeterek yazdıralım.

Command1 sayıcı olarak a ondalık sayı değişkeni kullanmaktadır.

Sonuç hatalıdır, son sayı olan 2 türetilmemiştir.

Command3 sayıcı olarak a ondalık sayı değişkeni kullanmaktadır.

Sonuç hatalıdır, son sayı olan 2 türetilmemiştir.

Command2 sayıcı olarak i tam sayı değişkeni kullanmaktadır. **Sonuç doğrudur**, beklenen tüm sayılar türetilmiştir.

Dikkat:

Döngülerde ondalık sayı kullanılması durumunda, döngü sayısının öngörülenden eksik veya fazla olma olasılığı vardır. Bunun nedeni, ondalık sayılardaki yuvarlama hatalarıdır.

```
Option Explicit

Private Sub Command1_Click()
    Dim a As Double
    ' 1 de 2 ye kadar 0.1 adımlarla
    ' a sayısını türet ve yaz
    For a = 1 To 2 Step 0.1
    Print a;
    Next a
    Print
End Sub

Private Sub Command2_Click()
    Dim a As Double, i As Byte
    ' 1 de 2 ye kadar 0.1 adımlarla
    ' a sayısını türet ve yaz
    a = 1
    For i = 1 To 11
    Print a;
    a = a + 0.1
    Next i
    Print
End Sub

Private Sub Command3_Click()
    Dim a As Double
    ' 1 de 2 ye kadar 0.1 adımlarla
    ' a sayısını türet ve yaz
    a = 1
    Do Until a > 2
    Print a;
    a = a + 0.1
    Loop
    Print
End Sub
```

Command1 den

Command2 den

Command3 den

Çalışma formu

Tasarım formu

Örnek: Sin(x) hesabı

Sin(x) fonksiyonunu aşağıda verilen seri yardımıyla hesaplayan programı yazalım.

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} \dots$$

x açısı radyan birimindedir. x in kuvveti en fazla 99 olacak ve hesaba katılacak terim 10^{-16} dan küçük kalıncaya kadar hesaba devam edilecektir.

Yukarıdaki seride, teorik olarak, sonsuz terim alındığında en doğru sonuç bulunur. Ancak bilgisayarda sayıların sonsuz büyük alınması mümkün değildir. Bilgisayarda tanımlanan en büyük sayı daima sonlu bir sayıdır.

Program **Text1** in **Change** olayına kodlanmıştır ve yeterince açıklama içermektedir. x değıştikçe Sin(x) değeri **Text2** penceresinde ve son terimdeki x in üssü **Label2** de görüntülenecektir.

Sin(x) fonksiyonu VB6 nın standart fonksiyonudur. Seri ile hesabına gerek yoktur. Ancak buradaki amacımız farklı. Bu minik program ile kodlama tekniğinin bazı ince noktalarına açıklık getirmek istiyoruz:

İyi program, iyi kodlanmış ve yeterince test edilmiş programdır. Bir iki doğru sonuç programın doğru çalışacağı anlamına gelmez.

Verilen programın bazı sonuçlarına bakalım:

0, 30, 45, 60, 90, 135, 180, 270, 360, 3600 ve 36000 derece açılarının program ile hesaplanan değeri ve olması gereken gerçek değeri aşağıdaki tabloda verilmiştir.

```
Project1 - Form1 (Code)
Text1 Change
Option Explicit

Private Sub Text1_Change()
    Dim x As Double ' açı (derece biriminde)
    Dim sonTerim As Double ' hesaplanan son terimin değeri
    Dim sinx As Double ' Sin(x) değeri
    Dim isaret As Integer ' terimlerin işareti
    Dim i As Byte, j As Byte ' döngü sayıcıları
    Dim fak As Double ' Faktoriyel
    Dim pi As Double ' Pi sayısı
    '---- veri harf içeriyorsa geri dön
    If Not IsNumeric(Text1.Text) Then Exit Sub
    '-----
    x = Text1.Text
    pi = 4 * Atn(1) ' Pi sayısı 3.14.....
    x = x * pi / 180 ' açıyı randyana çevir
    sinx = x ' Sin(x) in başlangıç değeri
    isaret = -1 'terimin işaretini tersine çevir
    For i = 3 To 99 Step 2
        '--- Faktoriyel hesabı(i!)
        fak = 1
        For j = 1 To i
            fak = fak * j
        Next j
        '-----
        sonTerim = x ^ i / fak 'son terim
        sinx = sinx + isaret * sonTerim ' Sin(x) değeri
        isaret = -isaret 'terimin işaretini tersine çevir
        If sonTerim < 1E-16 Then Exit For 'yeterince küçükse çık
    Next i
    Text3.Text = sinx
    Label2.Caption = "Son terimin üssü " & i & " oldu."
End Sub
```

x(derece)	Son terimde x in üssü	Programın hesapladığı Sin(x)	Gerçek değer Sin(x)	Hata
0	3	0	0	0
30	15	0.5	0.5	0
45	17	0.707106781186547	0.707106781186547	0
60	19	0.866025403784438	0.866025403784439	≈ 0
90	23	1	1	0
135	27	0.707106781186548	0.707106781186548	0
180	29	3.33116540488902E-16	0	≈ 0
270	35	-1	-1	0
360	41	3.30124476784746E-15	0	≈ 0
3600	99	-3.17709894423748E+21	0	ÇOK BÜYÜK
36000	99	-1.09458451122058E+121	0	ÇOK BÜYÜK

Tablonun incelenmesinden, büyük açıların Sinüs değerlerinin doğru hesaplanamadığı hemen anlaşılmaktadır. Halbuki açının değeri ne olursa olsun program doğru sonuç vermeliydi. Peki, bunun sebebi nedir?

Açı büyüdükçe yüksek dereceden kuvvetleri de çok büyümektedir. Son terimin kuvveti için programda ön görülen 99 değeri yetersiz kalmaktadır. Açıkçası program 360 dereceye kadar doğru, daha büyük açılarda hatalı sonuç vermektedir.

Peki, ne yapılabilir? 99 değerini büyütme çözüm değildir. Çünkü büyük sayıların çok yüksek dereceden kuvvetleri ve faktoriyel hesabı sayı taşmasına ve yuvarlama hatalarına neden olur, program kırılır.

Çözüm, Sinüs fonksiyonunun özelliğinde gizlidir. Bilindiği gibi, Sinüs fonksiyonunun periyodu 360 derecedir. Bunun anlamı, açının 360 derece ve katlarının Sinüsü aynıdır. O halde Sinüsü hesaplanacak açının 360 derece tam katları atılabilir. Böylece açı ne verilirse verilsin, program daima 0-360 derece arasındaki karşılığını hesaplamış olur.

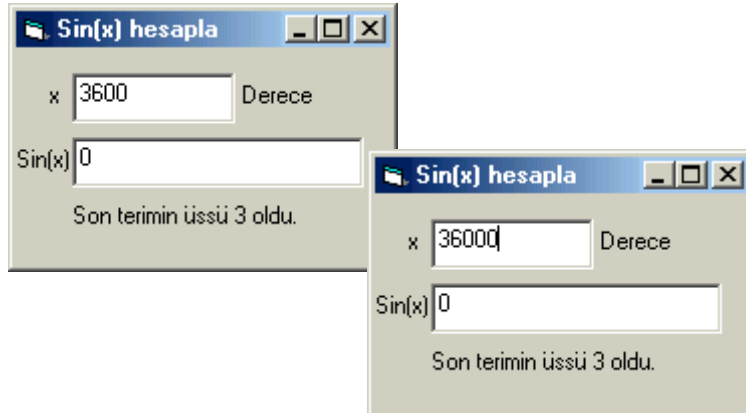
Örnek: Sin(x) hesabı (iyileştirilmiş)

Sin(x) fonksiyonunu hesaplayan önceki programa eklenen bir satır ile 360 derecenin tam katları Sinüsü hesaplanacak x açısından atılmıştır. Program artık büyük açılar için de doğru sonuç vermektedir.

Önceki programda; büyük açılarda x in üssü için 99 sayısı yetmez iken, iyileştirilmiş programda sadece 3 sayısı yeterli olmaktadır.

Bu küçük düzeltme neler kazandırdı:

- Program her açı için doğru sonuç verecek
- Büyük sayıların yüksek dereceden kuvvetleri ve faktoriyeli hesaplanmayacağı için sayı taşması olmayacak
- Yuvarlama hataları en aza indirgenmiş olacak
- Daha az terim ile daha hızlı sonuç bulunacak.



```
Project1 - Form1 (Code)
Text1 Change
Option Explicit

Private Sub Text1_Change()
Dim x As Double ' açı (derece biriminde)
Dim sonTerim As Double ' hesaplanan son terimin değeri
Dim sinx As Double ' Sin(x) değeri
Dim isaret As Integer ' terimlerin işareti
Dim i As Byte, j As Byte ' döngü sayıcıları
Dim fak As Double ' Faktoriyel
Dim pi As Double ' Pi sayısı
'---- veri harf içeriyorsa geri dön
If Not IsNumeric(Text1.Text) Then Exit Sub
'-----

x = Text1.Text
x = x - Int(x / 360) * 360 ' 360 in tam katlarını at
pi = 4 * Atn(1) ' Pi sayısı 3.14.....
x = x * pi / 180 ' açıyı radyana çevir
sinx = x ' Sin(x) in başlangıç değeri
isaret = -1 'terimin işaretini tersine çevir
For i = 3 To 99 Step 2
'---- Faktoriyel hesabı(i!)
fak = 1
For j = 1 To i
fak = fak * j
Next j
'-----
sonTerim = x ^ i / fak 'son terim
sinx = sinx + isaret * sonTerim ' Sin(x) değeri
isaret = -isaret 'terimin işaretini tersine çevir
If sonTerim < 1E-16 Then Exit For 'yeterince küçükse çık
Next i
Text3.Text = sinx
Label2.Caption = "Son terimin üssü " & i & " oldu."
End Sub
```

Eklenen satır

Ödev:

1. Koordinatları $A(x_1, y_1)$ ve $B(x_2, y_2)$ olan iki nokta arasındaki uzaklık

$$s = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

ile hesaplanır. Bu bağıntıdan yararlanarak sağda verilen A şehri ile I şehri arasındaki yolun uzunluğunu hesaplayan programı hazırlayınız.

Not: Program esnek olmalıdır. Şehir sayısını değişken tutunuz. Koordinatları bir matrise okuyunuz.

2. $ax^2+bx+c=0$ denkleminin denkleminde a, b, c sayılarını okuyan ve denklemin köklerini hesaplan programı hazırlayınız.

3. n tane sayının

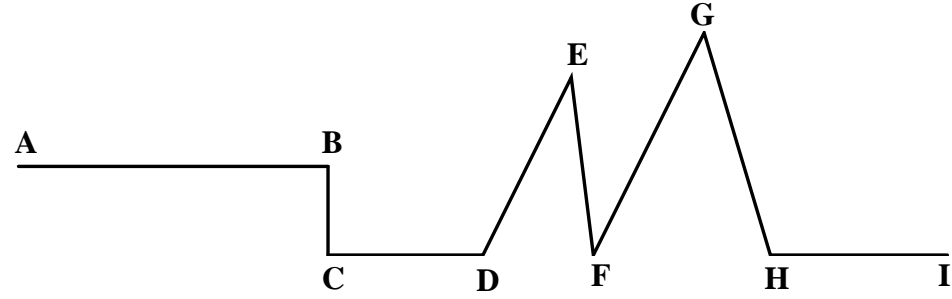
- Ortalamasını
 - Mutlak değerce en büyük olanını
 - Mutlak değerce en küçük olanını
- bulan programı hazırlayınız.

4. n tane sayıyı okuyan ve

- Negatif olanların adedini ve ortalamasını
- Pozitif olanların adedini ve ortalamasını
- Sıfır olanların adedini

hesaplayan programı hazırlayınız.

Not: Program esnek olmalıdır. Sayıları bir matrise okuyunuz.



Şehirlerin koordinatları

	A	B	C	D	E	F	G	H	I
x	0	70	70	105	125	130	160	175	215
y	0	0	-20	-20	20	-20	30	-20	-20

5. 1 den n ye kadar sayıların tam sayıların karelerini, küplerini, kareköklerini, 10 tabanına göre logaritmalarını ve e tabanına göre logaritmalarını hesaplayarak ikinci bir form üzerine aşağıdaki gibi yazan programı hazırlayınız.

Sayı	karesi	Küğü	LOG(10 tabanlı)	Ln(e tabanlı)

:				
:				
:				

6. 0 ile n derece arasındaki açılarının 5 er derece ara ile Sinüs, Cosinüs, Tanjant, Cotanjat değerlerini hesaplayarak ikinci bir form üzerine aşağıdaki gibi yazan programı hazırlayınız.

Açı x	Sin(x)	Cos(x)	Tan(x)	Cot(x)

:				
:				
:				

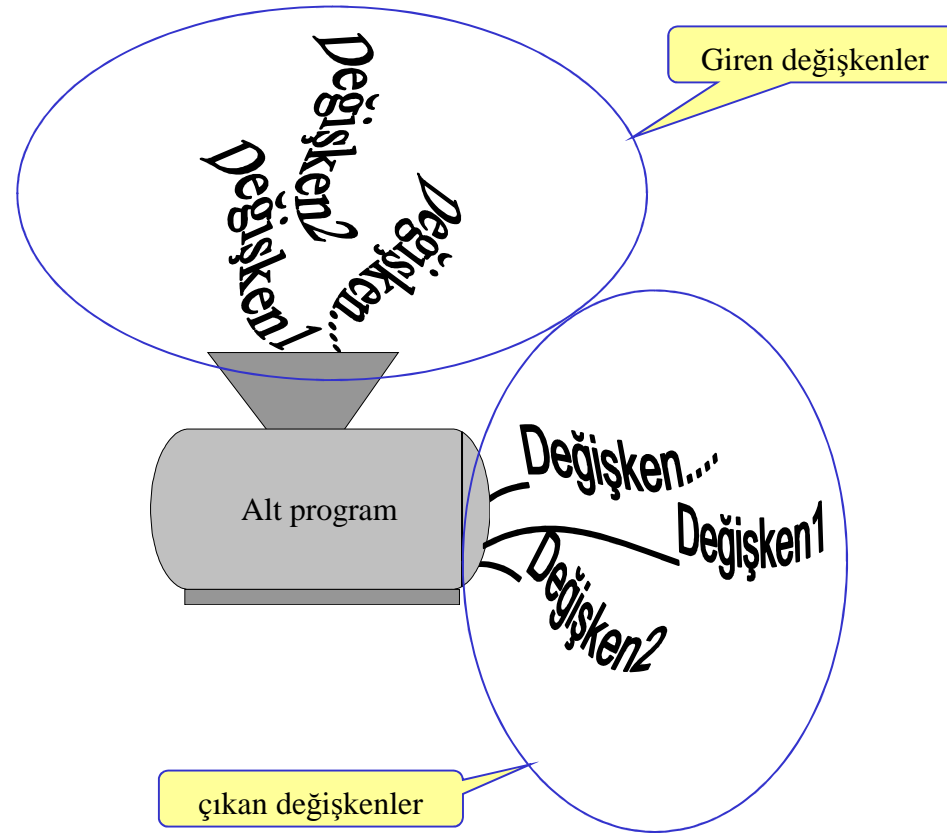
Kullanıcı Alt programları

Alt program nedir?

Benzetim:

Alt Program bir kıyma makinesi gibidir.

- Giren değişkenler vardır, içerikleri çağıran programdan gelir.
- Bunlar alt programda kullanılır, işlenirler.
- Çıkan değişkenler vardır, içerikleri çağıran programa aktarılır.



Büyük programlar binlerce kod satırı içerirler. Tüm satırların arka arkaya tek parça olarak yazılması hem programın anlaşılmasını zorlaştırır hem de aynı kodu içeren program parçası çok kez tekrarlanır. Bu sorunu gidermek için, profesyonel programlar çok sayıdaki alt programlar olarak yazılır. Bir problemi çözen program kodu alt program içine bir kez kodlanır ve projenin gerek duyulan her yerinde o problemi çözmek için alt program çağrılır.

Bir VB6 program yapısı incelendiğinde, kontrollere ait farklı olayların farklı alt program olarak oluşturulduğu görülür. Örneğin bir Command1 düğmesinin tıklama olayının kendine ait alt programı

```
Private Sub Command1_Click()  
    '.....  
  
End Sub
```

görünümündedir.

Tasarım modunda iken Command1 tıkladığında, VB6 ilk ve son satırı otomatik olarak hazırlar. Programcı bu iki satır arasına kendi kodlarını yazar.

Çalışma modunda iken Command1 her tıkladığında daima sadece bu iki satır arasındaki kodlar işlenir, projenin diğer kodları işlenmez.

Bu iki satırın üstünde ve altında belki binlerce satır olmasına rağmen, bu satırlar arasındaki kodlar tüm diğer satırlardan bağımsız çalışmaktadır. Bu nedenle buradaki program parçasına **alt program** adı verilir.

Yukarıdaki örnek alt program Command1 tıklama olayına aittir. Peki biz herhangi bir olaydan bağımsız olan, kendi alt programımızı yazmak istersek nasıl kodlayacak ve gerektiğinde nasıl çağıracağız?

VB6 da bu amaca yönelik iki farklı alt program yapısı vardır:

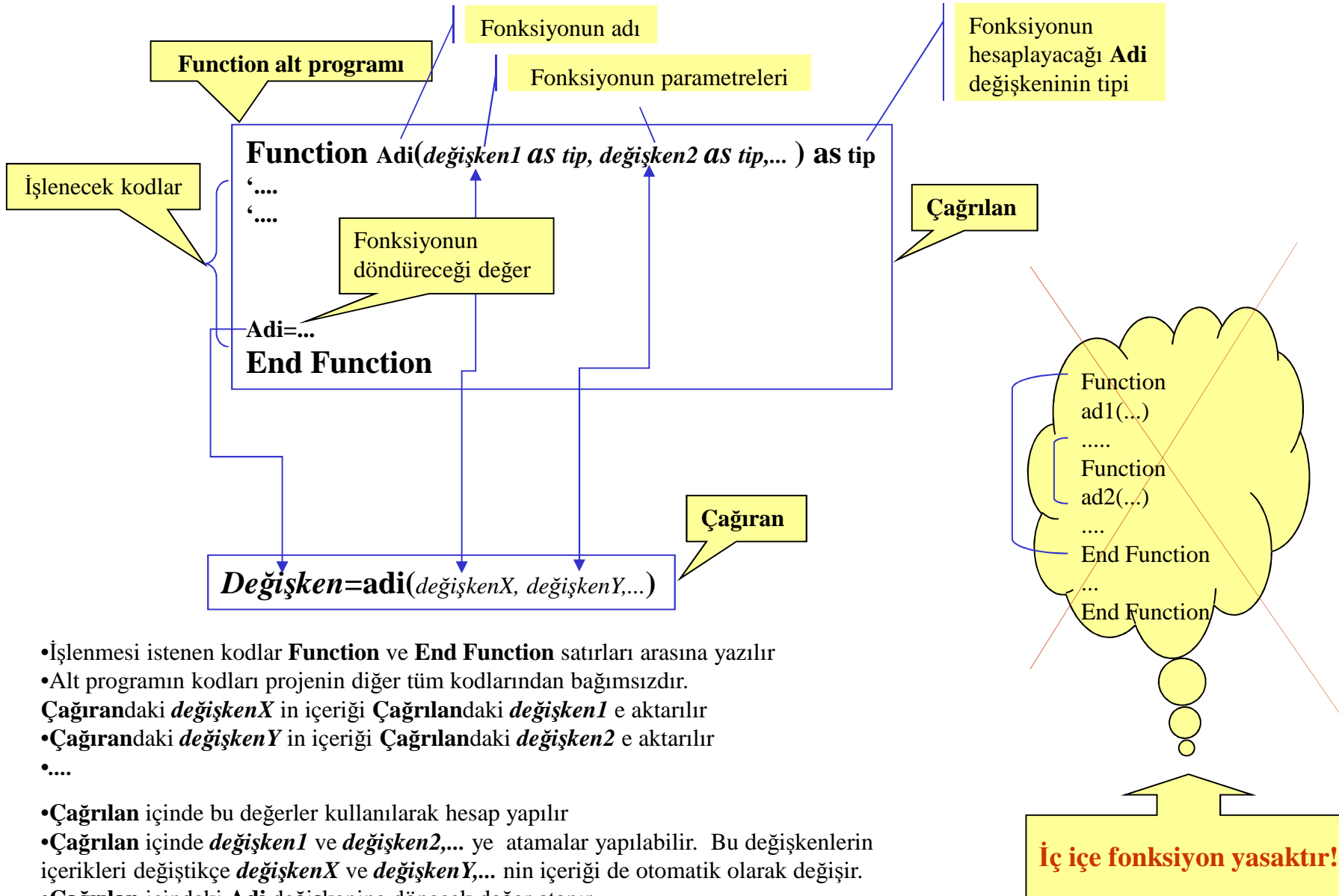
1. Function

2. Sub

alt program yapısı. **Function** alt programı tek bir değer hesaplarken **Sub** alt programı birden çok değer hesaplar. **Function** bir işlem içinde de adı ile çağrılabilir. **Sub** alt programı **Call** deyimi ile ayrı bir satırda çağrılmak zorundadır. Aralarındaki önemli farklar bundan ibarettir.

Aşağıda genel yapıları ve gerektiğinde nasıl çağrılacakları açıklanacaktır.

Function alt program yapısı ve çağırılması:



- İşlenmesi istenen kodlar **Function** ve **End Function** satırları arasında yazılır
- Alt programın kodları projenin diğer tüm kodlarından bağımsızdır.
- **Çağırıldaki** *değişkenX* in içeriği **Çağırıldaki** *değişken1* e aktarılır
- **Çağırıldaki** *değişkenY* in içeriği **Çağırıldaki** *değişken2* e aktarılır
-
- **Çağırılan** içinde bu değerler kullanılarak hesap yapılır
- **Çağırılan** içinde *değişken1* ve *değişken2,...* ye atamalar yapılabilir. Bu değişkenlerin içerikleri değiştikçe *değişkenX* ve *değişkenY,...* nin içeriği de otomatik olarak değişir.
- **Çağırılan** içindeki **Adi** değişkenine dönecek değer atanır
- **Çağırılan** alt programının görevi sona erer
- **Çağırılan** içindeki **Adi** değişkeninin içeriği **Çağırıldaki** *Değişken* e aktarılır.

Örnek:

x pozitif tam sayısının faktoriyelini (x!) hesaplayan bir fonksiyon yazalım. Bu fonksiyonu çağırarak

$$c = \frac{n!}{m!(n-m)!}$$

Değerini hesaplayalım. Burada n, m tam sayılardır ve $n \geq m$ dir.

Sayısal örnek:

$$n=7, m=4$$

$$c = \frac{7!}{4!(7-4)!} = \frac{7!}{4!3!}$$

$$c = \frac{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7}{(1 \cdot 2 \cdot 3 \cdot 4)(1 \cdot 2 \cdot 3)} = \frac{5040}{24 \cdot 6} = 35$$

- Yazacağımız faktoriyel fonksiyona bir ad verelim, örneğin **Fak**.
- Fak** in tipi **Long** olmalı (büyük sayı olabilir)
- Fak** fonksiyonu x tam sayı(integer) değişkenini parametre olarak alacak.

- x faktoriyeli hesaplayan gerekli kodlar

Function Fak(x as integer) as Long

End Function

satırları arasına yazılacak.

- C değerini hesaplayan kodlar **Text1** in **Change** olayına yazılacak ve bu kodlar **Text2** nin içeriği değiştiğinde tetiklenecek.
- C değeri **Text3** de görüntülenecek.

Fak fonksiyonu

Fak fonksiyonu üç kez çağrılıyor

Text1_Change olayı çağrılıyor

```
Project1 - Form1 (Code)
Text2 Change
Option Explicit
Function fak(x As Integer) As Long
' x faktoriyel(x!) hesaplayan alt program
Dim i As Integer
fak = 1
For i = 1 To x
fak = fak * i
Next
End Function
Private Sub Form_Load()
'ön değerler
Text1.Text = 1: Text2.Text = 1: Text3.Text = 1
End Sub
Private Sub Text1_Change()
Dim n As Integer, m As Integer
Dim f1 As Long, f2 As Long, f3 As Long
Text3.Text = "" ' sonuç penceresini boşalt
'---verilerde harf varsa çık
If Not IsNumeric(Text1.Text) Then Exit Sub
If Not IsNumeric(Text2.Text) Then Exit Sub
'---
n = Text1.Text: m = Text2.Text
n = Abs(n): m = Abs(m) ' negatif ise pozitif yap
If n < m Then Exit Sub ' n>=m değilse çık
f1 = fak(n) ' n! hesapla, f1 e aktar
f2 = fak(m) ' m! hesapla, f2 e aktar
f3 = fak(n - m) ' (n-m)! hesapla, f3 e aktar
Text3.Text = f1 / (f2 * f3) ' sonuç
End Sub
Private Sub Text2_Change()
Call Text1_Change ' Text1_Change olayını çağır
End Sub
```

Örnek:

Buradaki program fonksiyonun işlem içinde nasıl çağrılacağını göstermektedir.

Önceki programda $n!$, $m!$, $(n-m)!$ değerleri **Fak** fonksiyonu üç kez çağrılarak hesaplanmış ve bu üç değer $f1$, $f2$, $f3$ değişkenlerine aktarılmıştı.

Sağdaki program aynı amaçla yazılmış, ancak $f1$, $f2$ ve $f3$ değişkenleri kullanılmamıştır. **Fak** fonksiyonu doğrudan, gerektiği yerde, işlem içinde çağrılmıştır.

Fak fonksiyonu işlem içinde çağrılıyor

```
Project1 - Form1 (Code)
Text2 Change
Option Explicit
Function fak(x As Integer) As Long
' x faktoriyel(x!) hesaplayan alt program
Dim i As Integer
fak = 1
For i = 1 To x
fak = fak * i
Next
End Function
Private Sub Form_Load()
'ön değerler
Text1.Text = 1: Text2.Text = 1: Text3.Text = 1
End Sub
Private Sub Text1_Change()
Dim n As Integer, m As Integer
Text3.Text = "" ' sonuç penceresini boşalt
'---verilerde harf varsa çık
If Not IsNumeric(Text1.Text) Then Exit Sub
If Not IsNumeric(Text2.Text) Then Exit Sub
'----
n = Text1.Text: m = Text2.Text
n = Abs(n): m = Abs(m) ' negatif ise pozitif yap
If n < m Then Exit Sub ' n>=m değilse çık
Text3.Text = fak(n) / (fak(m) * fak(n - m))
End Sub
Private Sub Text2_Change()
Call Text1_Change ' Text1_Change olayını çağır
End Sub
```

Form1

n: Text1

m: Text2

n!/(m!(n-m)!): Text3

Form1

n: 7

m: 4

n!/(m!(n-m)!): 35


Ödev:

Standart fonksiyonlar içinde olmayan bazı fonksiyonlar **Function** alt programları olarak sağda kodlanmış, ancak test edilmemiştir.

Yararlı bulduğunuz başka fonksiyonları da ekleyerek test ediniz. Projenizi FONKSIYONLAR adı altında diskete/CD ye saklayınız.

Önünüzde uzun bilgisayarlı bir yaşam var. Deneyimlerinizi biriktirin!

Zamanla buradaki fonksiyonları zenginleştirin. Gereksinim duydukça yeni projelerinize yapıştırıp kullanabilirsiniz.



```
Project1 - Form1 (Code)
(General) DerToFah

Option Explicit
Dim x As Double

Function Pi() As Double
' Pi=3.14... sayısı
Pi = 4 * Atn(1)
End Function

Function Es() As Double
' e=2.718... sayısı
Es = Exp(1)
End Function

Function Log10(x As Double) As Double
' x in 10 tabanına göre logaritması
If x <= 0 Then
MsgBox "HATA: Argüman pozitif olmalı!- Log10(x)"
Exit Function
End If
Log10 = Log(x) / Log(10)
End Function

Function Sinh(x As Double) As Double
' Sinüs hiperbolik
Sinh = 0.5 * (Exp(x) - Exp(-x))
End Function

Function Cosh(x As Double) As Double
' Cosinüs hiperbolik
Cosh = 0.5 * (Exp(x) + Exp(-x))
End Function

Function DerToRad(x As Double) As Double
' x derece açının radyan biriminde karşılığı
DerToRad = x * Pi / 180
End Function

Function RadToDer(x As Double) As Double
' x Radyan açının Derece biriminde karşılığı
RadToDer = x * 180 / Pi
End Function

Function FahToDer(x As Double) As Double
' x Fahrenheit sıcaklığın derece biriminde karşılığı
FahToDer = 5 / 9 * (x - 32)
End Function

Function DerToFah(x As Double) As Double
' x Derece sıcaklığın Fahrenheit biriminde karşılığı
DerToFah = 9 / 5 * x + 32
End Function
```

Ödev:

Kenarları a, b ve c olan bir üçgenin alanı

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

Bağıntısı ile hesaplanabilir. Bu bağıntıya HERON formülü denir. Burada s üçgenin yarı çevresidir:

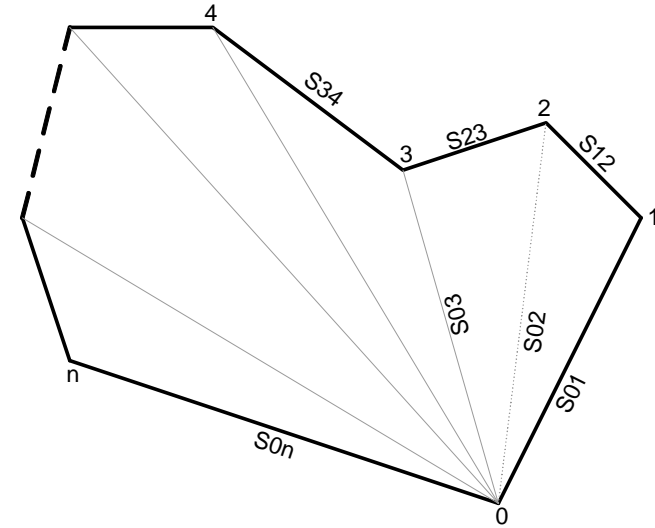
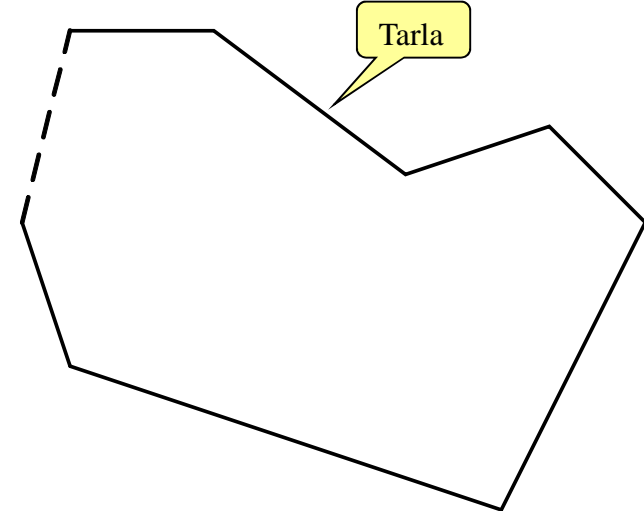
$$s = \frac{a+b+c}{2}$$

Bu bağıntılardan yararlanarak şekli görülen bir tarlanın alanı hesaplanacaktır. Bu amaca yönelik olarak tarla noktaları numaralanmış, üçgenlere bölünmüş ve kenar uzunlukları ölçülmüştür. Veriler aşağıdaki gibidir.

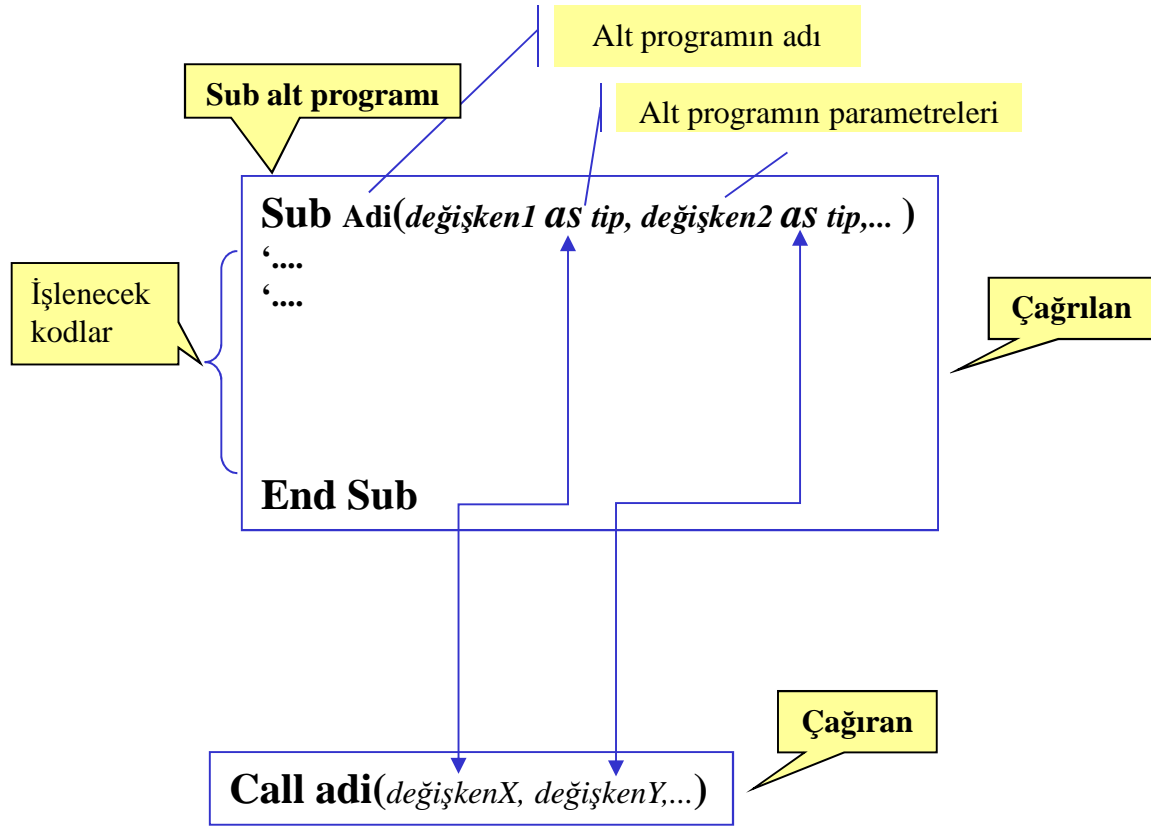
n=7

Kenar	Uzunluk(m)
S01	66.9
S02	80.1
S03	72.4
S04	116.6
S05	134.3
S06	116.6
S07	94.9
S12	28.3
S23	31.6
S34	50.0
S45	30.0
S56	41.20
S67	31.6

Üçgen alanı hesaplayan bir alt program(fonksiyon) yazınız. Bu alt fonksiyonu kullanarak tarla alanını hesaplayınız.

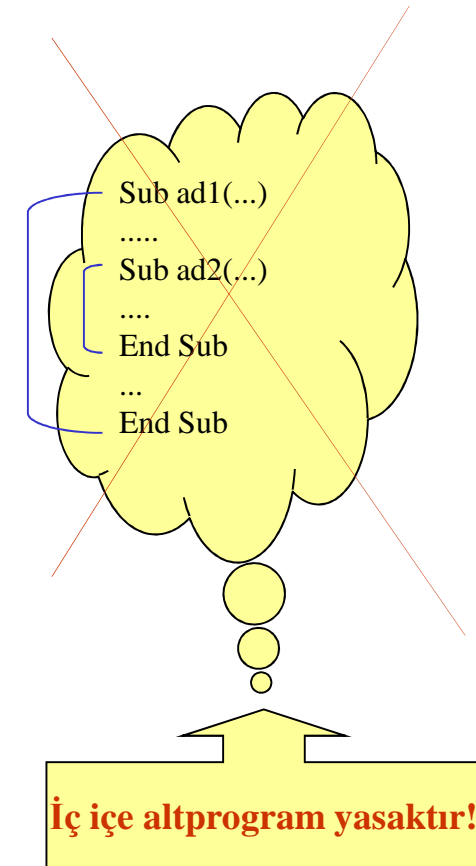


Sub alt program yapısı ve çağırılması:



- İşlenmesi istenen kodlar **Sub** ve **End Sub** satırları arasına yazılır
- Alt programın kodları projenin diğer tüm kodlarından bağımsızdır.
- Çağırıldaki *değişkenX* in içeriği Çağırıldaki *değişken1* e aktarılır
- Çağırıldaki *değişkenY* in içeriği Çağırıldaki *değişken2* e aktarılır
-

- Çağırılan içinde bu değerler kullanılarak hesap yapılır
- Çağırılan içinde *değişken1* ve *değişken2,...* ye atamalar yapılabilir. Bu değişkenlerin içerikleri değiştikçe *değişkenX* ve *değişkenY,...* nin içeriği de otomatik olarak değişir.
- Çağırılan alt programının görevi sona erer



Örnek:

R çapı verilen dairenin alanını ve çevresini hesaplayan alt program yazalım.

Alan: $A=\pi R^2/4$

Çevre: $S=\pi R$

- Yazacağımız alt programa bir ad seçelim, örneğin **DaireAveS**.
- DaireAveS** alt programı kodlanırken çapın içeriği bilinmez. Alt programda Çapın adı **cap** ve tipi de **single** olsun.
- DaireAveS** alt programı **cap** değişkenini kullanarak alanı ve çevreyi hesaplayacak. Bu değerlerin değişkeni **alan** ve **cevre**, tipleri de **single** olsun.
- DaireAveS** alt programına **cap** giren değer olacaktır.
- DaireAveS** alt programından **alan** ve **cevre** dönen değerler olacaktır.

- DaireAveS** alt programı için gerekli kodlar

```
Sub DaireAveS(cap as single, alan as single, cevre as single)
```

```
End sub
```

satırları arasına yazılacaktır.

Form üzerinde de **Text1**, **Text2** ve **Text3** kutuları olsun.

Çap değerini **Text1** kutusuna girmeyi, hesaplanacak olan alan ve çevre değerlerini de **Text2** ve **Text3** kutusunda görüntülemeyi planlayalım. **Text1** in içeriği değiştiğinde **Text2** ve **Text3** de değişsin istiyoruz.

O halde **DaireAveS** alt programını **Text1** in **Change** olayında çağırmalıyız.

Call DaireAveS(R,A,s) satırı ile alt program çağırılmaktadır. **R** değeri alt programa giden değerdir ve içeriği alt programın **cap** değişkenine aktarılır. Alt program içinde **Alan** ve **cevre** değişkenleri hesaplanır. **Alan** değişkeninin içeriği **A** değişkenine **cevre** değişkeninin içeriği **s** değişkenine aktarılır. **A** ve **s** alt programdan dönen değerlerdir.

Alt program

Alt program çağırılıyor

```
Project1 - Form1 (Code)
Text1 Change

Option Explicit
Function Pi()
    ' pi sayısı=3.14....
    Pi = 4 * Atn(1)
End Function

Sub DaireAveS(cap As Single, _
    Alan As Single, _
    cevre As Single)
    ' çapı verilen dairenin Alanını
    ' ve çevresini hesaplar
    Alan = Pi * cap * cap / 4
    cevre = Pi * cap
End Sub

Private Sub Text1_Change()
    Dim R As Single, A As Single, s As Single
    If Not IsNumeric(Text1.Text) Then Exit Sub
    R = Text1.Text
    If R < 0 Then
        MsgBox "Çap hatalı!"
        Exit Sub
    End If
    Call DaireAveS(R, A, s)
    Text2.Text = A
    Text3.Text = s
End Sub
```

Form1

Çap: Text1

Alan: Text2

Çevre: Text3

Form1

Çap: 10

Alan: 7.853981

Çevre: 31.41593

```

Project1 - Form1 (Code)
Command1 Click
Option Explicit
Function Pi() As Double
' Pi=3.14... sayısı
Pi = 4 * Atn(1)
End Function
Sub celikozellikleri(cap As Integer, _
                    alan As Single, _
                    cevre As Single, _
                    kutle As Single)
'cap inşaat çeliğinin çapı(mm)
'alan " " alanı(mm2)
'cevre " " çevresi(mm)
'kutle " " kütlesi(kg/m)

alan = Pi * cap * cap / 4
cevre = Pi * cap
kutle = 7950 * alan * 0.000001
End Sub

Private Sub Command1_Click()
Dim R As Integer, a As Single, c As Single, k As Single
Dim adet As Byte, i As Byte
adet = List1.ListCount ' List1 deki eleman(çelik)sayısı
List2.Clear ' List2 penceresini sil
For i = 0 To adet - 1
If List1.Selected(i) Then
R = List1.List(i)
Call celikozellikleri(R, a, c, k)
List2.AddItem Format(a, "000") _
               + Format(c, "0") _
               + Format(k, "0.00")
Else
List2.AddItem " "
End If
Next i
End Sub

```

Alt program

çağrı

Formatlı çıktı

Örnek:

İnşatta kullanılan yuvarlak çeliklerin çapları 6, 8, 10,..., 24, 25, 26, 28 mm dir (standart). Yapıların hesabında yoğun olarak bu çeliklerin alan, çevre ve bir metresinin kütlesine gereksinim duyulur.

Verilen programda bu değerler alt programda hesaplanmaktadır. Çeliğin kütlesi 7950 kg/m³ dür.

Form üzerinde Listbox1, Listbox2 ve Command1 kontrolleri vardır. Listbox1 in List özelliğine tasarım sırasında çaplar yazılmıştır. Çalışma modunda List1 den istenen çaplar seçilir ve Hesapla düğmesi tıklanırsa:

- Alt program çağrılır.
- Seçili çaplara ait alan, çevre ve kütle alt programda hesaplanır.
- Sonuçlar listbox2 penceresinde görüntülenir.

Programı inceleyiniz ve test ediniz.

Çaplar(mm):	Alan(mm ²)	Çevre(mm)	Kütle(kg/m)
<input type="checkbox"/> 6			
<input type="checkbox"/> 8			
<input type="checkbox"/> 10			
<input type="checkbox"/> 12			
<input type="checkbox"/> 14			
<input type="checkbox"/> 16			
<input type="checkbox"/> 18			
<input type="checkbox"/> 20			
<input type="checkbox"/> 22			
<input type="checkbox"/> 24			
<input type="checkbox"/> 25			
<input type="checkbox"/> 26			
<input type="checkbox"/> 28			

Çaplar(mm):	Alan(mm ²)	Çevre(mm)	Kütle(kg/m)
<input type="checkbox"/> 6			
<input checked="" type="checkbox"/> 8	<input type="checkbox"/> 050	25	0.40
<input checked="" type="checkbox"/> 10	<input type="checkbox"/> 079	31	0.62
<input checked="" type="checkbox"/> 12	<input type="checkbox"/> 113	38	0.90
<input checked="" type="checkbox"/> 14	<input type="checkbox"/> 154	44	1.22
<input checked="" type="checkbox"/> 16	<input type="checkbox"/> 201	50	1.60
<input type="checkbox"/> 18			
<input checked="" type="checkbox"/> 20	<input type="checkbox"/> 314	63	2.50
<input type="checkbox"/> 22			
<input type="checkbox"/> 24			
<input type="checkbox"/> 25			
<input type="checkbox"/> 26			
<input type="checkbox"/> 28			

Örnek: Matris işlemleri

TOPLAMA, ÇIKARMA:

$$\underline{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{21} & \dots & a_{2m} \\ \dots & & & \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \quad \underline{B} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \dots & & & \\ b_{n1} & b_{n2} & \dots & b_{nm} \end{bmatrix}$$

n satırlı m kolonlu \underline{A} ve \underline{B} matrislerinin toplamı $\underline{C}=\underline{A}+\underline{B}$ ile gösterilir. \underline{C} nin elemanları \underline{A} ve \underline{B} nin karşılıklı elemanlarının toplamıdır, $c_{ij} = a_{ij} + b_{ij}$. \underline{C} matrisinin boyutu da $n \times m$ olur.

n satırlı m kolonlu \underline{A} ve \underline{B} matrislerinin farkı $\underline{C}=\underline{A} - \underline{B}$ ile gösterilir. \underline{C} nin elemanları \underline{A} ve \underline{B} nin karşılıklı elemanlarının farkıdır, $c_{ij} = a_{ij} - b_{ij}$. \underline{C} matrisinin boyutu da $n \times m$ olur.

ÇARPMA:

$$\underline{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{21} & \dots & a_{2m} \\ \dots & & & \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \quad \underline{B} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1s} \\ b_{21} & b_{22} & \dots & b_{2s} \\ \dots & & & \\ b_{m1} & b_{m2} & \dots & b_{ms} \end{bmatrix}$$

n satırlı, m kolonlu \underline{A} ve m satırlı s kolonlu \underline{B} matrislerinin çarpımı $\underline{C}=\underline{A} \underline{B}$ ile gösterilir. \underline{C} nin i satırında ve j kolonundaki elemanı \underline{A} nin i satırındaki elemanların \underline{B} nin j kolonundaki elemanlar ile karşılıklı çarpılıp toplanması ile bulunur:

$$C_{ij} = a_{i1}b_{j1} + a_{i2}b_{j2} + \dots + a_{im}b_{jm}$$

\underline{C} matrisinin boyutu $n \times s$ olur.

```
'-----C=A+B (Matris toplama işlemi)-----  
For i = 1 To n  
For j = 1 To m  
c(i, j) = a(i, j) + b(i, j)  
Next j  
Next i  
'-----
```

```
'-----C=A-B (Matris çıkarma işlemi)-----  
For i = 1 To n  
For j = 1 To m  
c(i, j) = a(i, j) - b(i, j)  
Next j  
Next i  
'-----
```

```
'-----C=A*B (Matris Çarpma işlemi)-----  
For i = 1 To n  
For j = 1 To s  
c(i, j) = 0  
For k = 1 To m  
c(i, j) = c(i, j) + a(i, k) * b(k, i)  
Next k  
Next j  
Next i  
'-----
```

Ödev

Matris işlemleri ile ilgili programın formu ve tam kodu verilmiştir. Çok sayıda alt program içerdiğinden, yazarak test etmeniz, kendinize özgü değişiklikler yapmaya çalışmanız size büyük deneyim kazandıracaktır.

Command1
Command2
Command3
Command4
Command5

Command6

Command7

Command8

Option Explicit

Dim a() As Double, b() As Double, c() As Double 'dinamik matris tanımı

Sub matrisoku(MATRIS() As Double)

' MATRIS matrisinin elemanlarını InputBox fonksiyonu ile okur

Dim n As Integer, m As Integer

Dim i As Integer, j As Integer, eleman As String

n = UBound(MATRIS, 1) ' MATRIS in satır sayısı

m = UBound(MATRIS, 2) ' MATRIS in kolon sayısı

For i = 1 To n

For j = 1 To m

Do

eleman = InputBox(Str(i) + Str(j) + " Elemanını gir:", "Matris", "")

If eleman = "" Then Exit Sub ' vazgeçildiyse çık

Loop Until sayimi(eleman) ' sayı mı? kontrol et

MATRIS(i, j) = eleman

Next j

Next i

End Sub

Function sayimi(GelenText As String) As Boolean

' GelenText bir sayı ise True aksi halde False döner

If Not IsNumeric(GelenText) Then

beep: beep: beep

MsgBox "Veri: " + GelenText + " hatalı!"

sayimi = False

Else

sayimi = True

End If

End Function

Function matrisOkunmusmu() As Boolean

' a ve b matrisi okunmuşsa true aksi halde False döner

matrisOkunmusmu = True

If Text7.Text = "" Or Text8.Text = "" Then

matrisOkunmusmu = False

beep: beep

MsgBox " Önce matrisler okunmalı!"

End If

End Function

```

Private Sub Command6_Click()
' A matrisinioku
Dim n As Integer, m As Integer
If Not sayimi(Text1.Text) Then Exit Sub
If Not sayimi(Text2.Text) Then Exit Sub
n = Text1.Text ' a matrisinin satır sayısı
m = Text2.Text ' a matrisinin kolon sayısı
ReDim a(n, m)
Call matrisoku(a)
Call matrisyaz(a) 'a matrisini önce text9 a yaz
Text7.Text = Text9.Text: Text9.Text = "" 'text9.text i kopyala ve sil
End Sub

```

```

Private Sub Command7_Click()
' B matrisinioku
Dim n As Integer, m As Integer
If Not sayimi(Text3.Text) Then Exit Sub
If Not sayimi(Text4.Text) Then Exit Sub
n = Text3.Text ' b matrisinin satır sayısı
m = Text4.Text ' b matrisinin kolon sayısı
ReDim b(n, m)
Call matrisoku(b)
Call matrisyaz(b) 'b matrisini önce text9 a yaz
Text8.Text = Text9.Text: Text9.Text = "" 'text9.text i kopyala ve sil
End Sub

```

```

Sub matrisyaz(MATRIS() As Double)
' MATRIS in elemanlarını matris görünümünde Text9 a yazar.
Dim n As Integer, m As Integer
Dim i As Integer, j As Integer
n = UBound(MATRIS, 1) ' MATRIS in satır sayısı
m = UBound(MATRIS, 2) ' MATRIS in kolon sayısı
Text9.Text = ""
For i = 1 To n
For j = 1 To m
Text9.Text = Text9.Text + Str(MATRIS(i, j)) + " "
Next j
Text9.Text = Text9.Text + Chr(13) + Chr(10) ' satır sonu için Enter ve LF ekle
Next i
End Sub

```

```

Private Sub Command1_Click()
'C=A+B hesapla, sonucu yaz
Dim n As Integer, m As Integer
If Not matrisOkunmusmu Then Exit Sub
n = UBound(a, 1) ' a nın satır sayısı
m = UBound(a, 2) ' a nın kolon sayısı
' A nın ve B nin boyutlar uyumlu mu?
If n <> UBound(b, 1) Or m <> UBound(b, 2) Then
beep: beep
MsgBox "A ve B matrisinin boyutları farklı, toplanamazlar !"
Exit Sub
End If

Text5.Text = n: Text6.Text = m
ReDim c(n, m)
Call MatrisTopla(a, b, c)
Call matrisyaz(c) ' C matrisini yaz
End Sub

```

```

Private Sub Command2_Click()
'C=A-B hesapla, sonucu yaz
Dim n As Integer, m As Integer
If Not matrisOkunmusmu Then Exit Sub
n = UBound(a, 1) ' a nın satır sayısı
m = UBound(a, 2) ' a nın kolon sayısı
' A nın ve B nin boyutlar uyumlu mu?
If n <> UBound(b, 1) Or m <> UBound(b, 2) Then
beep: beep
MsgBox "A ve B matrisinin boyutları farklı, çıkartılamazlar !"
Exit Sub
End If

Text5.Text = n: Text6.Text = m
ReDim c(n, m)
Call MatrisCikar(a, b, c)
Call matrisyaz(c) ' C matrisini yaz
End Sub

```

```

Private Sub Command3_Click()
'C=B-A hesapla, sonucu yaz
Dim n As Integer, m As Integer
If Not matrisOkunmusmu Then Exit Sub
n = UBound(a, 1) ' a nın satır sayısı
m = UBound(a, 2) ' a nın kolon sayısı
If n <> UBound(b, 1) Or m <> UBound(b, 2) Then
beep: beep
MsgBox "A ve B matrisinin boyutları farklı, çıkartılamazlar !"
Exit Sub
End If

Text5.Text = n: Text6.Text = m
ReDim c(n, m)
Call MatrisCıkar(b, a, c)
Call matrisyaz(c) ' C matrisini yaz
End Sub

```

```

Private Sub Command4_Click()
'C=A*B hesapla, sonucu yaz
Dim n As Integer, m As Integer, s As Integer
If Not matrisOkunmusmu Then Exit Sub
n = UBound(a, 1) ' a nın satır sayısı
m = UBound(a, 2) ' a nın kolon sayısı
s = UBound(b, 2) ' b nin kolon sayısı
' A nın ve B nin boyutlar uyumlu mu?
If m <> UBound(b, 1) Then
beep: beep
MsgBox "A nın kolon sayısı B nin satır sayısından farklı, çarpılamazlar !"
Exit Sub
End If

Text5.Text = n: Text6.Text = s
ReDim c(n, s)
Call MatrisCarp(a, b, c)
Call matrisyaz(c) ' C matrisini yaz
End Sub

```

```

Private Sub Command5_Click()
'C=B*A hesapla, sonucu yaz
Dim n As Integer, m As Integer, s As Integer
If Not matrisOkunmusmu Then Exit Sub
n = UBound(b, 1) ' b nın satır sayısı
m = UBound(b, 2) ' b nın kolon sayısı
s = UBound(a, 2) ' a nin kolon sayısı
' A nın ve B nin boyutlar uyumlu mu?
If m <> UBound(a, 1) Then
beep: beep
MsgBox "B nin kolon sayısı A nın satır sayısından farklı, çarpılamazlar !"
Exit Sub
End If

Text5.Text = n: Text6.Text = s
ReDim c(n, s)
Call MatrisCarp(b, a, c)
Call matrisyaz(c) ' C matrisini yaz
End Sub

```

```

Sub MatrisTopla(a() As Double, _
                b() As Double, _
                c() As Double)
Dim n As Integer, m As Integer
Dim i As Integer, j As Integer
n = UBound(a, 1) ' a nın satır sayısı
m = UBound(a, 2) ' a nın kolon sayısı
'-----C=A+B (Matris toplama işlemi)-----
For i = 1 To n
For j = 1 To m
c(i, j) = a(i, j) + b(i, j)
Next j
Next i
'-----
End Sub

```

```

Sub MatrisCikar(a() As Double, _
    b() As Double, _
    c() As Double)
Dim n As Integer, m As Integer
Dim i As Integer, j As Integer
n = UBound(a, 1) ' a nın satır sayısı
m = UBound(a, 2) ' a nın kolon sayısı
'-----C=A-B (Matris çıkarma işlemi)-----
For i = 1 To n
For j = 1 To m
c(i, j) = a(i, j) - b(i, j)
Next j
Next i
'-----
End Sub

```

```

Sub MatrisCarp(a() As Double, _
    b() As Double, _
    c() As Double)
Dim n As Integer, m As Integer, s As Integer
Dim i As Integer, j As Integer, k As Integer
'If Not matrisOkunmusmu Then Exit Sub
' A nın ve B nin boyutlar uyumlu mu?
n = UBound(a, 1) ' a nın satır sayısı
m = UBound(a, 2) ' a nın kolon sayısı
s = UBound(b, 2) ' b nin satır sayısı
'-----C=A*B (Matris Çarpma işlemi)-----
For i = 1 To n
For j = 1 To s
c(i, j) = 0
For k = 1 To m
c(i, j) = c(i, j) + a(i, k) * b(k, i)
Next k
Next j
Next i
'-----
End Sub

```

```

Private Sub Form_Load()
'ön atamalar
Text1.Text = 2 ' a matrisi satır sayısı
Text2.Text = 2 ' a matrisi kolon sayısı
Text3.Text = 2 ' b matrisi satır sayısı
Text4.Text = 2 ' b matrisi kolon sayısı
End Sub

```

```

Private Sub Form_Unload(Cancel As Integer)
' programı sonlandırma isteği
Dim mesaj As String, dugmeler As Integer, baslik As String
Cancel = -1 ' sonlandırma!
mesaj = "Programdan Çıkayım mı?"
dugmeler = vbYesNo + vbDefaultButton2 + vbQuestion
baslik = "UYARI"
beep: beep: beep
If MsgBox(mesaj, dugmeler, baslik) = vbYes Then Cancel = 0 'sonlandır
End Sub

```

```

Private Sub Text1_Change()
Text7.Text = "" ' a matrisi penceresini sil
Text9.Text = "" ' c matrisi penceresini sil
End Sub

```

```

Private Sub Text2_Change()
Call Text1_Change
End Sub

```

```

Private Sub Text3_Change()
Text8.Text = "" ' b matrisi penceresini sil
Text9.Text = "" ' c matrisi penceresini sil
End Sub

```

```

Private Sub Text4_Change()
Call Text3_Change
End Sub

```

```

Private Sub Command8_Click()
Unload Form1
End Sub

```

Visual Basic'de Standart Fonksiyonlar

Visual Basic'de çok kullanılan standart fonksiyonlar vardır. Bu fonksiyonların bir kısmı alfa-nümerik fonksiyon bir kısmı da nümerik fonksiyonlardır.

İlk olarak bu fonksiyonlardan alfa-nümerik (karaktersel) fonksiyonları görelim.

Len() Fonksiyonu :

Bu fonksiyon yardımı ile alfa-nümerik değişkenlerin veya ifadelerin karakter sayısını, nümerik ifadelerin ise bellekte kapladıkları alan miktarını (**byte**) öğrenebiliriz.

Örnekler :

ders="Visual Basic"

Uzunluk=Len(ders)

Uzunluk=Len("Visual Basic")

'uzunluk 12 olur.

'uzunluk 12 olur.

Dim tamsayi as integer

Uzunluk=len(tamsayi)

Tamsayi=32767

Uzunluk=Len(tamsayi)

'uzunluk=2 olacaktır.

'uzunluk=2 olacaktır.

Dim tamsayi As Integer, uzuntamsayi As Long

Dim ondalikli As Single, uzunondalikli As Double

Dim degisken As Variant

Print "tamsayı :" & Len(tamsayi)

Print "Uzun Tamsayı:" & Len(uzuntamsayi)

Print "ondalikli:" & Len(ondalikli)

Print "uzunondalikli:" & Len(uzunondalikli)

Print "Değişken:" & Len(degisken)

degisken = 1234

Print "Değişken:" & Len(degisken)

degisken = 1234567

Print "Değişken:" & Len(degisken)

0

4

7

Yan tarafta verilen örnekte **Integer** değişkenin içeriği ne olursa olsun uzunluğu **2**, **uzun tamsayının** uzunluğu **4**, **ondalıkli** sayının **4**, **uzun ondalıklı** sayının **8** dir. **Variant** değişkeninin uzunluğu ise aldığı değere göre değişmektedir. En büyük uzunluk değeri **20** olacaktır.

Left() Fonksiyonu :

Left(değişken, sol baştan itibaren alınacak karakter sayısı)

Bu fonksiyon verilen değişkenin içinde soldan başlayarak istenilen karakterde kısmı ayırmak için kullanılır. Değişken çeşitli tiplerde olur geriye dönen değer isteğe göre (bazı noktalara dikkat ederek) belirlenebilir.

Örnek :

sehir="Eskişehir"

İlkharf= Left(sehir,1)

İkikiharf=Left(sehir,2)

'ilkharf E olur.

'ikikiharf Es olur.

Dim degisken As String

Dim ad As Integer

degisken = "a1234"

ad = **Left**(degisken, 2)

Print ad

HATALI

Eğer string bir değişken sayısal (integer, single, double..) herhangi bir değişkene atanırsa hata oluşur.

Dim i As Integer

Dim ad As String

i = 1234

ad = **Left**(i, 2)

Print ad

DOĞRU

Eğer sayısal (integer, single, double..) herhangi bir değişken, string değişkene atanırsa hata oluşmaz.

Right() Fonksiyonu :

Right(değişken, sağ baştan itibaren alınacak karakter sayısı)

Bu fonksiyon, verilen değişkenin içinde sağdan başlayarak istenilen karakterde kısmı ayırmak için kullanılır. Değişken çeşitli tiplerde olur geriye dönen değer isteğe göre (bazı noktalara dikkat ederek) belirlenebilir.

Örnek :

sehir="Eskişehir"

sonharf= Right(sehir,1)

sonikiharf=Right(sehir,2)

'sonharf r olur.

'sonikiharf ir olur.

Dim degisken As String

Dim ad As Integer

degisken = "1234a"

ad = **Right**(degisken, 2)

Print ad

HATALI

Eğer string bir değişken sayısal (integer, single, double..) herhangi bir değişkene atanırsa hata oluşur.

Dim i As Integer

Dim ad As String

i = 1234

ad = **Right**(i, 2)

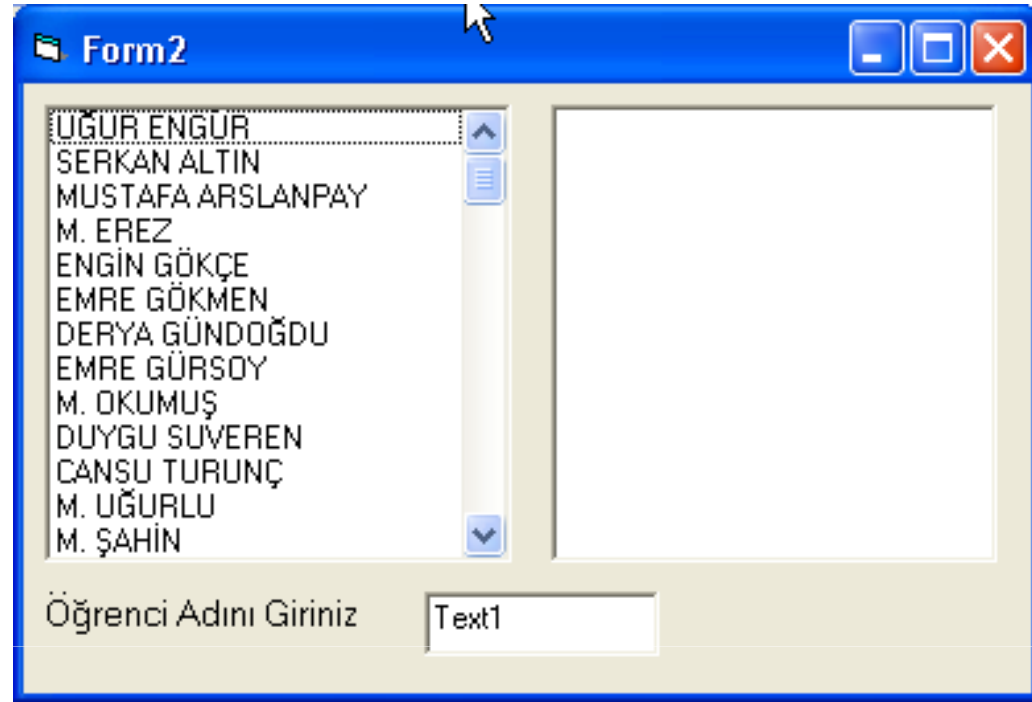
Print ad

DOĞRU

Eğer sayısal (integer, single, double..) herhangi bir değişken, string değişkene atanırsa hata oluşmaz.

Örnek Program:

Yanda formu görünen programda sol tarafta bulunan listbox1 nesnesine öğrenci isimleri programcı tarafından girilmektedir. Kullanıcı programı çalıştırdıktan sonra text1 nesnesine herhangi bir harf girdiğinde o harfle başlayan öğrenci isimlerini listbox2 nesnesine aktarmaktadır.



Form2

UĞUR ENGÜR
SERKAN ALTIN
MUSTAFA ARSLANPAY
M. EREZ
ENGİN GÖKÇE
EMRE GÖKMEN
DERYA GÜNDOĞDU
EMRE GÜR SOY
M. OKUMUŞ
DUYGU SUVEREN
ÇANSU TURUNÇ
M. UĞURLU
M. ŞAHİN

Öğrenci Adını Giriniz Text1

```
Private Sub Text1_Change()  
Dim i As Integer  
List2.Clear  
For i = 0 To List1.ListCount - 1  
    If Left(List1.List(i), Len(Text1.Text)) = UCase(Text1.Text) Then  
        List2.AddItem (List1.List(i))  
    End If  
Next  
End Sub
```

Mid() Fonksiyonu :

Mid(değişken, başlangıç karakteri, alınacak karakter uzunluğu)

Bu fonksiyon kendisine verilen string veya variant tipindeki değişken içerisinde bir kısmı çıkarıp ayırmak için kullanılır.

Örnek :

```
Dim numara As Variant, kayityili As String
```

```
Dim ogretim As String
```

```
numara = 151420001045#
```

```
kayityili = Mid(numara, 5, 4)
```

Kayityili="2000"

```
If Mid(numara, 9, 1) = "1" Then
```

```
ogretim = "birinci"
```

```
Else
```

```
ogretim = "ikinci"
```

```
End If
```

```
Print "Öğrenci " & kayityili & " yılında " & ogretim & " öğretime kayıt yaptırmıştır."
```

LTrim() Fonksiyonu :

LTrim(değişken)

Bu fonksiyon, verilen değişke içinde sol tarafında bulunan boşluk karakterlerini siler. Örnek üzerinde incelersek:

Örnek :

```
Dim sehir As String
```

```
sehir = " Eskisehir"
```

```
sehir = LTrim(sehir)
```

```
Text1.Text = sehir
```

Eskişehir kelimesinden önce 5 kez space tuşu ile boşluk bırakılmış.

Sonuç **sehir = "Eskisehir"** olur.

RTrim() Fonksiyonu :

RTrim(değişken)

Bu fonksiyon, verilen değişke içinde sağ tarafında bulunan boşluk karakterlerini siler. Örnek üzerinde incelersek:

Örnek :

```
Dim sehir As String
sehir = "Eskisehir "
sehir = RTrim(sehir)
Text1.Text = sehir
```

Eskişehir kelimesinden sonra 5 kez space tuşu ile boşluk bırakılmış.

Sonuç **sehir = "Eskisehir"** olur.

LTrim() fonksiyonu yukarıda bahsedilen nedenlerden dolayı kullanılır. Ayrıca her iki fonksiyonda değişkenlerin bellekte gereğinden fazla yer kaplamaları önler.

Trim() Fonksiyonu :

Trim(değişken)

Bu fonksiyon, yukarıda bahsedilen her iki fonksiyonun birleşimi gibidir. Değişken içinde bulunan boşluk karakterlerini her iki taraftan siler.

Örnekler :

```
Dim mesaj As String
mesaj = "Bu fonksiyonlar kelime aralarındaki boşlukları silmez."
Text1.Text = Trim(mesaj)
```

Bu fonksiyon genellikle dosyadan bilgi okunduğunda veya kullanıcının girmiş olduğu değerleri işlerken kullanılır. Bu şekilde yanlış girilmiş değerler dosyaya yada database'e kaydedildiğinde sorgu yaparken problemler çıkabilir (Sıralama işlemleri veya herhangi bir harf ile başlayan kayıtların bulunması gibi).

LCase() Fonksiyonu :

LCase(değişken)

Bu fonksiyon kendisine verilen string tipindeki değişken içerisinde bulunan ifadede BÜYÜK yazılmış harfleri, küçük harfe çevirir.

Örnek :

Dim mesaj As String

mesaj = "Tüm Harfler Küçük Olacak!"

Text1.Text = LCase(mesaj)

Text1.Text="tüm harfler küçük olacak!"

Bu fonksiyon kullanılırken dikkat edilmesi gereken bir nokta vardır. Visual Basic İngilizce olması nedeniyle sadece türkçeye özgü harflerde problem yaratacaktır.

Sadece İ ve ı harflerinde problem var!

Örnek :

Dim mesaj As String

mesaj = "İÇİNDE TÜRKÇE HARF BULUNAN İFADELER"

Text1.Text = LCase(mesaj)

Text1.Text="İçinde türkçe harf bulunan ifadeler"

UCase() Fonksiyonu :

UCase(değişken)

Bu fonksiyon kendisine verilen string tipindeki değişken içerisinde bulunan ifadede küçük yazılmış harfleri, BÜYÜK harfe çevirir.

Örnek :

Dim mesaj As String

mesaj = "Tüm Harfler Büyük Olacak!"

Text1.Text = UCase(mesaj)

Text1.Text="TÜM HARFLER BÜYÜK OLACAK!"

Örnek :

Dim mesaj As String

mesaj = "İçinde türkçe harf bulunan ifadeler"

Text1.Text = UCase(mesaj)

Text1.Text = "İÇİNDE TÜRKÇE HARF
BULUNAN İFADELER"

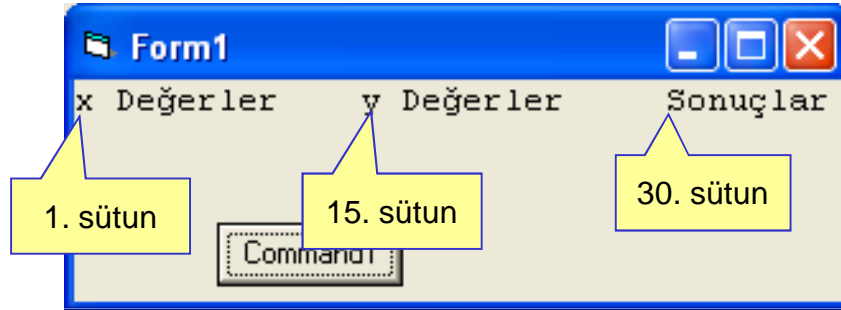
Tab() Fonksiyonu :

Tab(sayı)

Bu fonksiyon form üzerine print komutu ile yazı yazdırılırken veya yazıcıya her hangi bir yazı yazdırılmak istendiğinde kullanılır. Her bir harfin bir kolona yazıldığı düşünülürse Tab fonksiyonundan sonra yazdırılacak yazı verilen sayı kolondan başlayacaktır.

Örnek :

```
Private Sub Command1_Click()  
Print "x Değerler"; Tab(15); "y Değerler"; Tab(30); "Sonuçlar"  
End Sub
```



Bu fonksiyon kullanılan font seçeneklerine duyarlıdır.

Bu fonksiyonun doğru çalışabilmesi için tüm Tab ifadeleri kendinden önce yazılanların son sütunundan daha büyük bir sayı olması gereklidir.

String() Fonksiyonu :

String(Tekrarlama sayısı, Karakter)

Bu fonksiyon verilen karakteri tekrarlama sayısı kadar yazdırmak istediğimizde kullanılabilir.

Örnek :

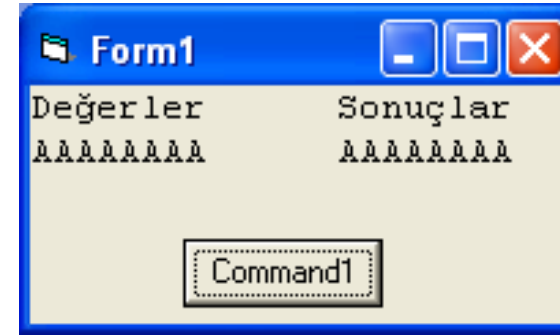
```
Private Sub Command1_Click()  
Print "Değerler"; Tab(15); "Sonuçlar"  
Print String(8, "-"); Tab(15); String(8, "-")  
End Sub
```



String fonksiyonunda karakter ifadesi olarak tek bir karakter girilmelidir. Herhangi bir kelime veya hece girilmesi durumunda sadece ilk harf çoğaltılacaktır.

Örnek :

```
Private Sub Command1_Click()  
Print "Değerler"; Tab(15); "Sonuçlar"  
Print String(8, "ALİ"); Tab(15); String(8, "ALİ")  
End Sub
```



Str() Fonksiyonu :

Str(Sayısal değer)

Bu fonksiyon verilen herhangi bir tipteki sayısal değeri string tipindeki değişkene çevirir.

Örnek :

```
Sayi=Str(1234.567) Sayi = "1234.567" olur.  
Dim A as Integer  
A = 900 Sayi = "900" olur.  
Sayi = Str(A)
```

Join() Fonksiyonu :

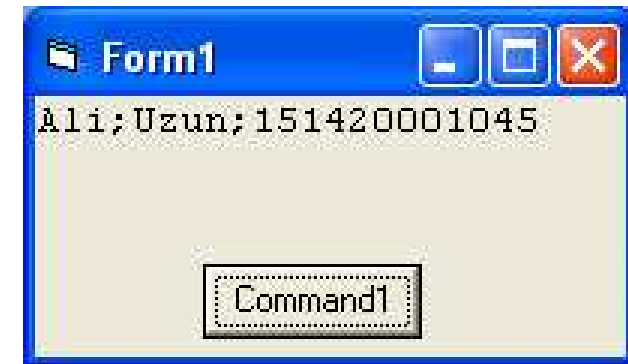
Join(Birleştirilecek vektör, ayraç karakteri)

Bu fonksiyon tek boyutlu bir vektörde depolanmış değerleri tek bir değişkene ayraç olarak belirtilen karakterle birbirinden ayrılarak aktarılmasını sağlar.

Örnek :

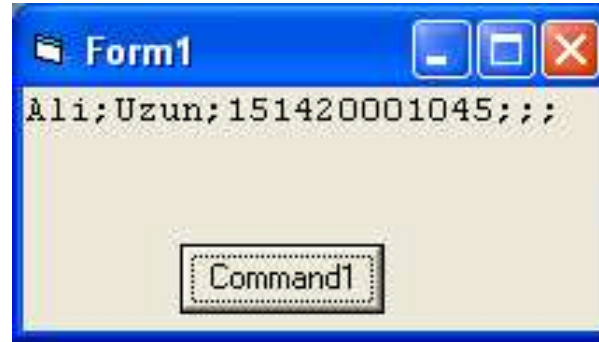
```
Private Sub Command1_Click()  
Dim a As String, b(2) As String  
b(0) = "Ali":b(1) = "Uzun":b(2) = "151420001045"  
a = Join(b, ";")  
Print a  
End Sub
```

Birleştirme işlemi vektör olarak verilen değişkenin 0. elemanından başlar, Dim komutu ile verilen değere kadar yapılır.



Örnek :

```
Private Sub Command1_Click()  
Dim a As String, b(5) As String  
b(0) = "Ali": b(1) = "Uzun": b(2) = "151420001045"  
a = Join(b, ";")  
Print a  
End Sub
```



Split() Fonksiyonu : Split(değişken, ayraç karakteri, en büyük eleman sayısı, karşılaştırma şekli)

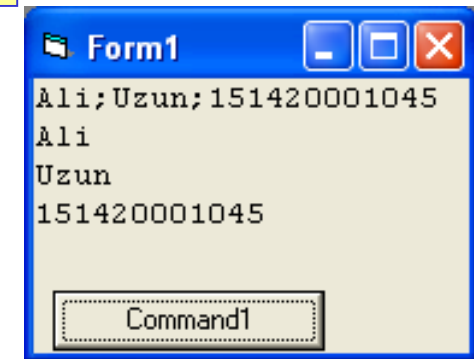
Bu fonksiyon Join fonksiyonunun tam tersi işi yapar. Belirli bir karakter ile birbirinden ayrılmış tek bir değişkenin içinde depolanmış verileri tek boyutlu bir vektör değişkene aktarır.

Örnek :

```
Private Sub Command1_Click()  
Dim a As String, b(2) As String, c() As String  
Dim ic As Variant  
b(0) = "Ali": b(1) = "Uzun": b(2) = "151420001045"  
a = Join(b, ";")  
Print a  
c = Split(a, ";")  
For Each ic In c  
Print ic  
Next  
End Sub
```

Bu fonksiyon kullanılarak değer aktarılacak değişken dinamik matris olarak tanımlanmalıdır.

Eğer bir matris değişkeninin boyutu bilinmiyorsa ve tüm elemanların kullanılması gerekli ise For-Next döngüsü bu şekilde kullanılmalıdır. Matris elemanlarının teker teker sıra ile aktarılacağı değişken mutlaka variant olarak tanımlanmalıdır.



Her iki fonksiyonda ayraç karakteri kullanılmaz ise kelime aralarındaki boşluklar ayraç olarak tanımlıdır.

Date(), Time(), Now() Zaman Fonksiyonları :

Bu fonksiyonlar sistemin fonksiyonun kullanıldığı andaki tarih, saat bilgilerini öğrenmemizi sağlar.

Örnek :

```
Private Sub Command1_Click()  
Text1.Text = Date  
Text2.Text = Time  
Text3.Text = Now  
End Sub
```

18.05.2003
08:42:44
18.05.2003 08:42:44
Command1

Time fonksiyonu 0-1 aralığını 86400 parçaya böler.
1 gün 86400 sn'dir. Gün içindeki her bir an 0-1 arasında bir sayıya karşılık gelir.

Örnek :

```
Private Sub Command1_Click()  
Dim baslangic As Double  
Dim i As Long, j As Integer  
baslangic = CDbI(Time)  
Text1.Text = baslangic  
For i = 1 To 60000  
For j = 1 To 50  
DoEvents  
Next j  
Next i  
Text2.Text = CDbI(Time)  
Text3.Text = CDbI((Time - baslangic) * 24 * 60 * 60)  
'24 saat 60 dak 60 san  
End Sub
```

0,315150462962963
0,315219907407407
6,000000000000094
Command1

İşleme başlama zamanının Double değişkene aktarılmış hali

İşlemin bitiş zamanının Double değişkene aktarılmış hali

İşlem boyunca geçen süre saniye cinsinden değeri.

Shell() Fonksiyonu : **Shell**(Çalıştırılacak programın adı ve yolu, *çalıştırılma modu*)

Bu fonksiyon bilgisayarda kurulu bulunan herhangi bir çalışabilir programı Visual Basic ortamından çalıştırılabilmesini sağlar.

Örnek :

```
Private Sub Command1_Click()  
Dim cevap As Integer  
cevap = Shell("notepad.exe", 1)  
AppActivate cevap  
End Sub
```

Çalışma modu programın ana penceresinin normal, ekranı kaplamış veya simge durumunda olup olmayacağını belirler.

SendKeys() Fonksiyonu : **SendKeys** (gönderilecek tuş veya tuşlar, *bekleme*)

Bu fonksiyon o an da aktif olan pencereye sanki klavyeden bir tuşa basılıyormuş gibi tuş veya tuşlar gönderir.

Örnek :

```
Private Sub Command1_Click()  
Dim cevap As Integer, mesaj As String  
cevap = Shell("notepad.exe", 1)  
AppActivate cevap, True  
mesaj = "NotePad'e yazı yazdırma!"  
SendKeys (mesaj):SendKeys "{enter}"  
SendKeys ("ikinci satır"):SendKeys "{enter}"  
SendKeys (Time):SendKeys "%{F4}"  
End Sub
```

Bekleme true yada false değerlerini alabilir. Eğer true değeri verilmişse Visual Basic programında bir alttaki satırdaki işleme geçilebilmesi için gönderilen tuşun işlenmiş olması gereklidir.

ÖDEVLER

Aşağıdaki özelliklerdeki programı tasarlayınız.

Kullanıcıdan tek bir inputbox ile sırayla öğrencinin adını, soyadını, 12 haneli öğrenci numarasını, doğum tarihini (Gün. Ay. Yıl) isteyecek ve her bir bilgiyi ayrı bir textbox nesnesine yazacak,

Öğrencinin kayıt tarihinde yaklaşık yaşını hesaplayıp, öğretimini (1. öğretim, 2. öğretim), bölümünü belirleyen ve bu bilgileri bir cümle içinde kullanacak,

Örnek: Öğrenci 2000 yılında 19 yaşında İnşaat mühendisliğinin birinci öğretimine kayıt yaptırmıştır.

Öğrencinin o anda kaç yaşında olduğunu (yıl, ay, gün) olarak hesaplayıp görüntüleyecek,

İstenen bilgilerden herhangi birinde hata varsa bilgileri tekrar isteyecek.

Örnek: 13.03.1980 yılında doğmuş bir kişinin 20.05.2003 tarihine kadar yaşadığı gün sayısı gun ise

Dim dogum as date

dogum = "13.03.1980"

Gun = date – dogum

Yaşadığı yılı bulmak için gun değerini 365'e bölüp tam kısmını almak gereklidir.

DOSYA İŞLEMLERİ

Bir çok programda yapılan işlerin daha sonra tekrar kullanılabilmesi için kaydedilmesi gerekmektedir. Bu kaydedilen dosyalar gerektiği zaman program tarafından açılıp gerekli bilgiler okunabilir, değiştirilebilir veya silinebilir. Visual Basic'te bu işlemleri yapmak için birkaç yol vardır. Bu yollardan en gelişmiş ve güncel olanı veritabanı (database) bileşenlerini kullanmaktır. Veritabanı; kullanıcının gereksinim duyduğu bilgilerin belirli formatlarda kalıcı olarak saklandığı dosyalardır. Günümüzde bir çok veritabanı programı mevcuttur. Visual Basic'te bu veritabanı programları ile uyumlu bir şekilde çalışabilir. Bahsedilen programlarla yaratılan veritabanı dosyaları V.B. tarafından açılabilir, işlenebilir, düzenlenebilir ve kaydedilebilir.

Bir başka veri depolama yöntemi klasik basic komutları kullanılarak yapılabilir. Çeşitli formatlarda dosyalar yaratılabilir gerekli bilgiler bu dosyalara kaydedilebilir, değiştirilebilir veya silinebilir.

Dosyaya yazma işlemleri

Visual Basic'te dosya kayıt/okuma işlemleri object yaratarak da yapılabilir. Bunun işlemleri sırayla inceleyelim:

Yazma veya okuma için nesne yaratma:

Set **FileSystemObject** ismi = CreateObject("Scripting.FileSystemObject")

FileSystemObject ismi: Program içinde bu nesneyi tanımlayacak

Object tipindeki değişken adıdır.

Dosya yazma ve okuma işlemleri için kullanılacak bir nesne yaratır.

Set **Text dosyası** ismi = **FileSystemObject** ismi .CreateTextFile(**Dosya Adı**, **Kayıt Şekli**)

Text dosyası ismi: Yaratılan Text dosyasını program

inde tanımlayacak Object tipindeki değişken adıdır.

Sadece yazma işlemi için kullanılacak bir text dosyası nesne yaratır.

Dosya Adı: Yaratılacak dosyanın tam yolu ve adı. (örnek "A:\deneme.txt") Diskette deneme isminde ve txt uzantılı bir dosya yaratır.

Kayıt Şekli: True veya False değerlerini alabilir. Eğer True değeri verilmiş ise daha önceden böyle bir dosya var veya yaratılmış ise bu dosyadaki tüm bilgiler silinerek yerlerine yeni bilgiler yazılacaktır. Eğer False değeri verilirse daha önceden böyle bir dosya yaratılmış ise program hata mesajı verecektir ve yazma işlemi yapılmaz.

Text dosyası ismi.write (metin)

Yaratılan text dosyasına yazma işlemini gerçekleştirir. **metin** değişken olabildiği gibi tırnak işareti içinde yazılan herhangi bir yazı da olabilir

Text dosyası ismi.WriteLine satır sayısı

Text dosyası ismi.WriteLine (metin)

Text dosyası ismi.Close

Yaratılan text dosyasına kapatır.

Şimdi yukarıda bahsedilen komutları bir program içerisinde görelim.

Örnek :

```
Private Sub Command1_Click()  
Dim nesne As Object, dosya As Object, i As Integer  
Set nesne = CreateObject("Scripting.FileSystemObject")  
Set dosya = nesne.CreateTextFile("C:\testfile.txt", True)  
dosya.write ("Bu dosya VB tarafından yaratılmıştır.")  
dosya.WriteLine 1  
dosya.write ("x değeri x^2 değeri")  
dosya.WriteLine 1  
For i = 1 To 100  
dosya.WriteLine (i & String(20 - Len(Str(i)) - Len(Str(i ^ 2)), " ") & i ^ 2)  
Next i  
dosya.Close  
End Sub
```

Yandaki program kodu C: üzerinde textfile.txt isminde bir dosya yaratır ve bu dosyanın ilk satırına

Bu dosya VB tarafından yaratılmıştır.

Bir alt satıra

x değeri x^2 değeri

Ve bu satırın altına da 1 ile 100 arasındaki sayılar ve karelerini yazar.

Set Text dosyası ismi= FileSystemObject ismi.OpenTextFile("c:\testfile.txt", dosya açama modu, kayıt şekli, format)

FileSystemObject ismi: Program içinde bu nesneyi tanımlayacak

Object tipindeki değişken adıdır.

dosya açama modu: 1 → Dosyayı sadece okunabilir açar

2 → Dosya yazma için açılır.

8 → Dosya yazılacak bilgiler dosyanın sonuna eklenmek üzere açılır.

Dosya yazma ve okuma işlemlerinin her ikisi içinde kullanılabilir text dosyası yaratır.

Kayıt şekli : True → Dosya daha önceden yaratılmışsa üzerine yazılır.

metin=Text dosyası ismi.ReadLine ()

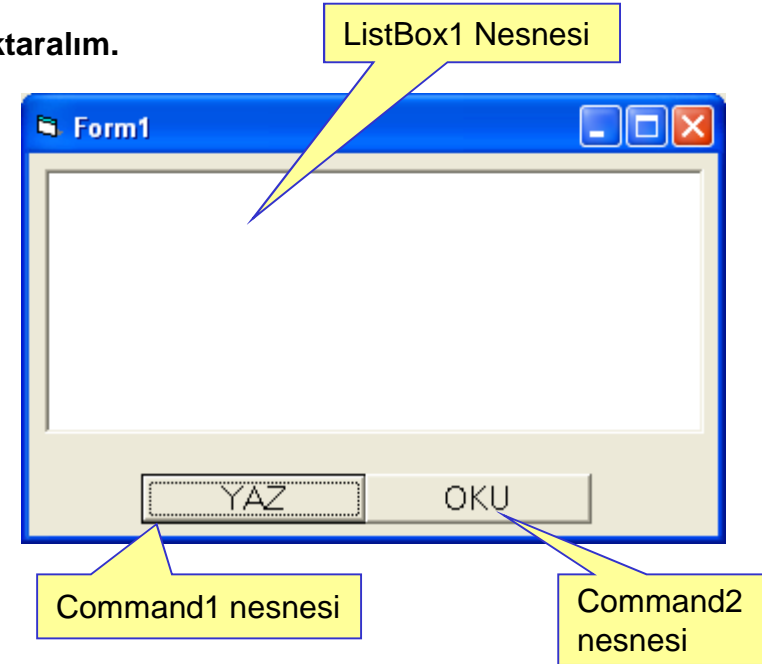
Yaratılan text dosyasından **satır satır** okuma işlemini gerçekleştirir. **metin** string tipindeki bir değişken olmalıdır.

Şimdi yukarıda yaratıp yazdığımız dosyayı okuyup listbox'a aktaralım.

Örnek :

```
Private Sub Command2_Click()  
Dim nesne As Object, dosya As Object, i As Integer  
Set nesne = CreateObject("Scripting.FileSystemObject")  
Set dosya = nesne.opentextfile("c:\testfile.txt", 1, False)  
Do While dosya.AtEndOfStream <> True  
List1.AddItem (dosya.readline)  
Loop  
dosya.Close  
End Sub
```

Açılan text dosyasının sonuna gelinceye kadar oku



ÖDEV

İsim soyisim, telefon numarası, adres, e-mail ve not(kişisel bilgi) bilgilerinin yer aldığı bir adres defteri programı tasarlayınız.

Program verilen bilgileri dosyaya yazacak ve gerektiğinde tekrar dosyadan okuyacak.