






## *Hedefler*

Bu üniteyi çalıştıktan sonra;

-  Döngüleri öğrenmek,
-  Alt Programları öğrenmek,
-  Fonksiyonları öğrenmek,

### *İçindekiler*

- For-Next Döngüsü,
- For Each-Next Döngüsü,
- Do Loop Döngüsü
- Alt Programlar ve Fonksiyonlar,
- SUB Alt Programı,
- FUNCTION Alt Programı,
- Modüller,

## Döngüler

Uygulama geliştirirken, çoğunlukla dallanmalar ve tekrarlarla programlama yapılır. Programlama içerisinde birden fazla kez aynı kodu yazıyorsak, burada döngüye ihtiyacımız olabilir.

Örneğin ekrana yüz adet öğrencinin adını yazdırmak istersek, bunu satır satır yazmak, programlamanın ve programların var oluş amacına aykırı olurdu. Uygulamada bunu tek bir satırda yazıp, yüz kez tekrarlanmasını sağlayabiliriz. Bu bölümde tekrarlama işlemlerini yaptırabileceğimiz döngü komutlarını göreceğiz.

### For - Next Döngüsü

Belirli bir başlangıç değerinden başlayarak, bir bitiş değerine kadar verilen artış miktarı ile (artım değeri yazılmazsa 1 olduğu kabul edilir) artarak ilerleyerek, bir kodu tekrar tekrar çalıştıran bir döngü bloğudur. Genel formu aşağıdaki şekildedir.

```
FOR Döngü Değişkeni = İlk Değer TO Son Değer [STEP Artım]
    <işlemler>
[EXIT FOR]
    <işlemler>
NEXT [Döngü Değişkeni]
```

Kullanılan **Döngü Değişkeni** “İlk Değer” den başlar, **FOR**’ dan sonraki ilk **NEXT** e kadar tüm işlemler çalıştırılır. Program **NEXT** satırına geldiğinde, **Döngü Değişkeni Artım** miktarı kadar arttırılır (**Artım** miktarı verilmeyebilir, eğer verilmezse **Artım** miktarı 1 olarak kabul edilir). Eğer **Döngü Değişkeninin** yeni değeri “Son Değer” den küçükse, program tekrar **FOR** satırına dönerek, işlemler tekrarlanır. Bu işlem **NEXT** satırındaki yeni değer “Son Değer” den büyük oluncaya kadar devam eder, büyük olduğunda, program **NEXT** satırının altındaki satırdan çalışmasına devam eder.

Bazen program içinde istediğimiz sonucu aldıktan sonra **FOR-NEXT** döngüsü içerisinde çıkılmak istenebilir, o zaman **EXIT FOR** deyimini kullanabiliriz.

İççe **FOR-NEXT** döngüleri kullanılmak istendiğinde, döngülerin birbirini kesmemesi gerekir. Bu durumda, öncelikle en içteki döngü çalıştırılır, daha sonra en dıştaki döngüye kadar devam eder.

**Örnek - 1 :** Köşegen elemanları 1,2,3,.....,10 olan diğer elemanları 0 olan matris dizisi oluşturan satırlar (Burada Step 1 yazılmasa da olur).

```
For i = 1 To 10 Step 1
    For j = 1 To 10
        If i = j Then
            matris(i, j) = i
        Else
            matris(i, j) = 0
        End If
    Next j
Next i
```

**Örnek - 2 :** Bir **ListBox1**'in içine ; 1 ile 10 arasındaki sayıları yazacak, fakat yazarken 4 ve 7 rakamlarını atlayacak olan kod parçasını yazalım.

```
For i = 1 To 10
    If (i <> 7 And i <> 4) Then
        ListBox1.Items.Add(i)
    End If
Next
```

1
2
3
5
6
8
9
10

yada **Continue For** komutu ile, **For-Next** döngüsünün **Next** satırına gitmesi sağlanabilir.

```
For i = 1 To 10
    If (i = 7 Or i = 4) Then Continue For
    ListBox1.Items.Add(i)
Next
```

## For Each - Next Döngüsü

**FOR-NEXT** döngüsünün özelleşmiş bir biçimidir. Özellikle liste, dizi veya grup içindeki her bir eleman için işlem gerçekleştirmeyi sağlar. Genel kullanımı aşağıdaki şekildedir.

```
FOR EACH Döngü Değişkeni IN Dizi/Grup
    <işlemler>
NEXT [Döngü Değişkeni]
```

**Örnek :** “numara” adındaki bir dizideki sayıları, **ListBox1**’ in içine göndermek için gereken kod parçasını yazalım.

```
Dim numara = {125, 138, 156, 205, 308, 947}
Dim dizi As Integer

For Each dizi In numara
    ListBox1.Items.Add(dizi)
Next
```

125  
138  
156  
205  
308  
947

## While - End While Döngüsü

Bu döngüde, belirli bir koşulun sağlandığı sürece döngü içindeki deyimler icra edilir. Genel kullanım şekli aşağıdaki gibidir.

```
WHILE <Koşul>

    <işlemler>

END WHILE
```

Buradaki **<Koşul>** gerçekleştiği sürece döngü devam eder. **Koşul** sağlanmadığında işlem **END WHILE** deyimini izleyen satıra geçer.

**Örnek :** 1'den 5' e kadar olan sayıları, *ListBox1*' in içine yazdırmak için kullanılan kod parçasını yazalım.

```
Dim a As Integer
a = 0
While a < 5
    a += 1
    ListBox1.Items.Add(a)
End While
```

1  
2  
3  
4  
5

## Do – Loop Döngüsü

DO-LOOP deyimi, bir döngü deyimidir. 5 farklı formu vardır.

**1. DO - LOOP :** Hiçbir koşul olmayan bir döngüdür. *DO-LOOP* arasındaki deyimler hiçbir koşul olmadan işlem görür. Döngüden çıkmak için *EXIT DO* deyimi kullanılabilir.

```
DO
    <işlemler>
    EXIT DO
    <işlemler>
LOOP
```

**Örnek :** 1' den 100' e kadar sayıların toplamını bulan kod satırları.

```
x = 0 : toplam = 0
Do
    If x > 99 Then
        Exit Do
    Else
        x = x + 1
        toplam = toplam + x
    End If
Loop
```

**Toplam**

5050

**2. DO UNTIL - LOOP : DO-LOOP'** un bu formunda, döngü içindeki deyimler, verilen **<Koşul>** yanlış olduğu sürece döngü çalıştırılır. Döngüden çıkmak için **EXIT-DO** deyimi kullanılır.

```
DO UNTIL <Koşul>
    <işlemler>
    EXIT DO
    <işlemler>
LOOP
```

**Örnek :** 1' den 100' e kadar sayıların toplamını bulan kod satırları.

```
x = 0 : toplam = 0
Do Until x > 99
    x = x + 1
    toplam = toplam + x
Loop
```

**Toplam**

5050

**3. DO WHILE - LOOP : DO-LOOP'** un bu formunda, döngü içindeki deyimler, verilen **<Koşul>** doğru olduğu sürece döngü çalışır. Döngüden çıkmak için **EXIT-DO** deyimi kullanılır.

```
DO WHILE <Koşul>
    <işlemler>
    EXIT DO
    <işlemler>
LOOP
```

**Örnek :** 1' den 100' e kadar sayıların toplamını bulan kod satırları.

```
x = 0 : toplam = 0
Do While x < 100
    x = x + 1
    toplam = toplam + x
Loop
```

**Toplam**

5050

**4. DO - LOOP UNTIL :** Döngü koşulunun, döngünün sonunda test edilen **DO-LOOP UNTIL** deyiminin genel yazılışı yandaki şekildedir. Koşulun döngünün sonunda kontrol edildiği bu döngü tipinde, döngü koşulu ne olursa olsun, her şartta döngüde bulunan satırlar en azından bir kez işletilir. Koşul yanlış olduğu sürece döngü çalışır.

```
DO
    <işlemler>
EXIT DO
    <işlemler>
LOOP UNTIL <Koşul>
```

**Örnek :** 1' den 100' e kadar sayıların toplamını bulan kod satırları.

```
x = 0 : toplam = 0
Do
    x = x + 1
    toplam = toplam + x
Loop Until x > 99
```

**Toplam**

5050

**5. DO - LOOP WHILE :** Döngü koşulunun, döngünün sonunda test edilen **DO-LOOP WHILE** deyiminin genel yazılışı yandaki şekildedir. Koşulun döngünün sonunda kontrol edildiği bu döngü tipinde, döngü koşulu ne olursa olsun, her şartta döngüde bulunan satırlar en azından bir kez işletilir. Koşul doğru olduğu sürece döngü çalışır.

```
DO
    <işlemler>
EXIT DO
    <işlemler>
LOOP WHILE <Koşul>
```

**Örnek :** 1' den 100' e kadar sayıların toplamını bulan kod satırları.

```
x = 0 : toplam = 0
Do
    x = x + 1
    toplam = toplam + x
Loop While x < 100
```

**Toplam**

5050



## Alt Programlar ve Fonksiyonlar

Visual Basic .NET'te yazılan programların çoğu belli bir olay için yazılan alt programlardı. Örneğin Form' a eklediğimiz bir butonuna tıklanınca ne yapılması gerektiği ;

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As _  
    System.EventArgs) Handles Button1.Click  
    .....  
    .....  
    .....  
End Sub
```

satırları arasına yazılıyordu. Bunlar olay alt programları olarak isimlendirilmektedir. Ancak tüm alt programlar olay alt programları değildir. Bir olay tarafından başlatılmaktan ziyade, program tarafından çağrıldıklarında belli işi yapan alt programlar ve fonksiyonlar da çok kullanılmaktadır.

Bazen programlarda bir işlemin çok kez yapılması gerekebilir. Her seferinde bu program parçasını tekrar yazmak hem programı uzatmakta hem de karışıklığa yol açmaktadır. Bunu engellemek için alt programlar kullanılır. Ayrıca programın yapısal olmasını sağlamak ve birbiriyle ilgili komutları ve programın bir bölümünü istenilen isim altında toplamak için de kullanılır. Genel amaçlı Genel amaçlı 2 alt program kullanılır. Bunlar **FUNCTION** ve **SUB** alt programlarıdır.

Alt program, bir program bloğunun isimlendirilmiş hali olarak tanımlanabilir. Kodun tamamı yazılacağına, alt programın adı çağrılarak aynı iş gerçekleştirilmiş olur.

Fonksiyon da alt program gibi isimlendirilmiş bir kod bloğudur. Ancak fonksiyon, alt programdan farklı olarak çağrıldığı yere bir değer döndürür.

### SUB Alt Programları :

Alt programlar, belirli işlemleri gerçekleştiren anlamlı ve ilişkili kod satırlarını bir isim altında bloklamaya yararlar. Bir **SUB** Alt programın genel yapısı aşağıdaki şekildedir;

```
SUB isim (Parametreler) [As VeriTipi]
.....
.....
Program Satırları
.....
.....
END SUB
```

Program satırlarının olay alt program satırlarından hiçbir farkı yoktur. SUB alt programı ;

**İsim** (Parametreler)

şeklinde çağrılır.

Alt programlar kontrol akışını etkilerler. Visual Basic bir alt program adıyla karşılaştığı zaman, programın o anda kaldığı yeri hatırlar ve adına rastladığı alt programı çalıştırmak için harekete geçer. Alt programı tamamlayınca, programda kaldığı yere döner ve çalışmaya oradan devam eder. Visual Basic yüzlerce alt program çağırabilir. Bir program tarafından çağırılmış bir alt programın kendisi de bir alt program çağırabilir ve bu aşağıdaki şekilde gösterildiği gibi devam eder.

**Örnek** : Programın farklı yerlerinde, farklı mesajlar vermemize yardımcı olacak alt program.

**Çağrılan Alt Program (Programın en üstüne "Public Class Form1" satırının altına yazılır)**

```
Sub Mesaj(ByVal msj As String, ByVal butonlar As Integer)
    MsgBox(msj, butonlar)
End Sub
```

Bu alt program gerektiği yerde aşağıdaki gibi çağrılabilirler;

```
.....

.....

.....

    Mesaj("Devam Etmek İstiyor musunuz ?", 2)

.....

    Mesaj("Onaylamak için Tamam butonuna basınız !", 1)

.....
```

#### **Dikkat**



Burada 1 rakamı; Mesaj kutusunda, Tamam ve İptal butonları çıkması için, 4 rakamı Evet ve Hayır butonları çıkması için yazılır. 0 ile 5 arasında her rakamın bir anlamı vardır.

### **FUNCTION Alt Programları :**

Alt programlar kendilerine verilen görevi yaparlar ve işleri biter. **FUNCTION** alt programının tanımlanması da SUB alt programlarının tanımlanmasına benzer. FUNCTION' lar ise belirli bir işlevi yerine getirirler ve geriye bir bilgi döndürürler. FUNCTION' ların çağırılmaları SUB alt programlarına benzemez. FUNCTION isimleri ancak bir ifade içinde veya bir komut cümlesi içinde operand olarak yer alır. Genel kullanımı aşağıdaki şekildedir;

FUNCTION Fonksiyon ismi (Parametreler) [As VeriTipi]

.....

.....

Program Satırları

.....

.....

[RETURN DEĞER]

END FUNCTION

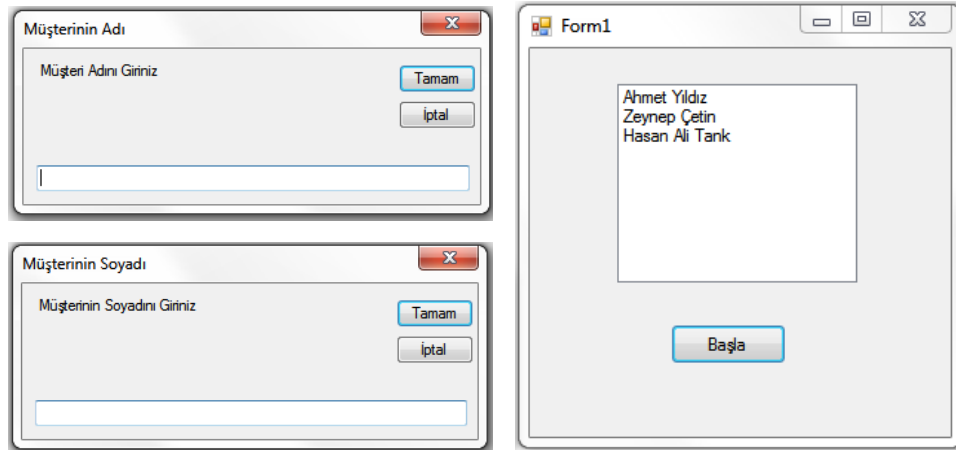
**Örnek :** Inputbox'lar ile girilen müşterilerin ad ve soyadları birleştirilip ListBox'a ekleniyor.

Fonksiyon kısmı en üstte oluşturuluyor.

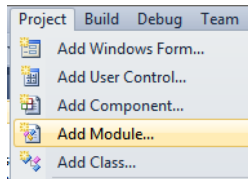
```
'Foksiyon satırları en üste "Public Class Form1" yazısının altına yazılmalıdır.  
Function birlestir(ByVal ad As String, ByVal soyad As String) As String  
    Dim isim As String  
    isim = ad & " " & soyad  
  
    'Ad ve Soyad birleştirildikten sonra, "isim" değişkeni ile ana programa  
    döndürülüyor.  
    Return isim  
End Function
```

Daha sonra Buton1' in içine program kodları yazılıyor.

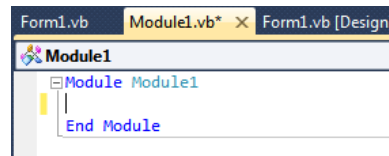
```
Dim ad, soyad As String  
10:  
    'Program başladığında, dışarıdan Müşterinin Adının ve Soyadının  
    girilmesini istiyor  
    ad = InputBox("Müşteri Adını Giriniz", "Müşterinin Adı", "")  
    soyad = InputBox("Müşterinin Soyadını Giriniz", "Müşterinin Soyadı", "")  
  
    'Eğer Müşterinin Adı yada Soyadı boş girilmişse program sonlandırılıyor.  
    If (ad = "" Or soyad = "") Then End  
  
    'Sorun yoksa, Ad ve Soyad "birleştir" fonksiyonu ile birleştiriliyor ve  
    ListBox1'e ekleniyor.  
  
    ListBox1.Items.Add(birlestir(ad, soyad))  
    'Program tekrar başa dönerek (10.satıra) aynı işlemler tekrarlanıyor.  
    GoTo 10
```



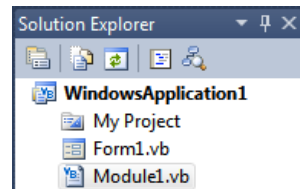
## Modüller :



Gerek **Function** gerekse **SUB** alt programları projedeki formlara bağlı olarak



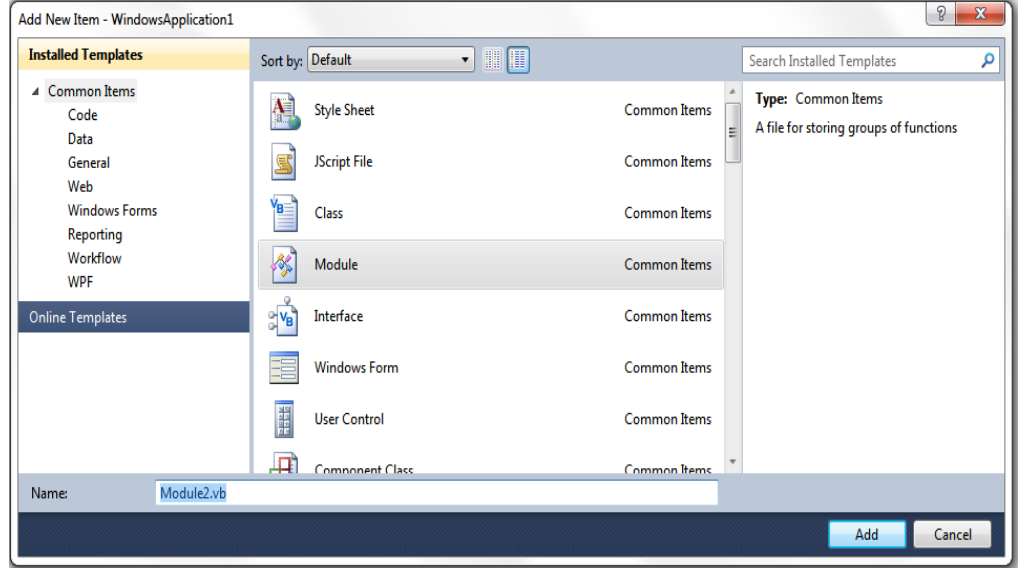
yazılıyor. Halbuki **Modül** kullanarak bu alt programları formdan bağımsız olarak oluşturmak mümkündür. Bağımsız olarak **Modül** içine yerleştirilen alt programlar, hem projede hem de geliştirilecek başka projelerde kullanılabilir.



Bir standart modül dosya uzantısı **.vb** olan ve programın her tarafından kullanılabilen değişken ve alt programlar içeren bir dosyadır. Formlar gibi **Modüller de Solution Explorer'da** ayrı olarak listelenirler.

Bir program içinde **Modül** oluşturmak için, **Project** menüsünden **Add Module** seçeneği tıklanır. Ekranı aşağıdaki pencere gelecektir. **Module** şablonu seçilir. **Name** kutusunda istediğimiz (\*.vb) ismi verebiliriz.

**Örnek : FUNCTION** konusundaki aynı örneği burada yapmak istersek, en üstte yazdığımız **FUNCTION-END FUNCTION**



satırları ile birlikte aradaki satırlar **Modülün** içine yazılır.

```
Module Module1
    Function birlestir(ByVal ad As String, ByVal soyad As String) As String
        Dim isim As String
        isim = ad & " " & soyad

        'Ad ve Soyad birleştirildikten sonra, "isim" değişkeni ile ana programa döndürülüyor.
        Return isim
    End Function
End Module
```