# 4.bölüm

**4**

# ALGORİTMA TEMELLERİ

# (Koşullu yapılar ve döngüler)

Koşullu ifadeler ve Döngüsel yapılar tüm programlama dünyasının olduğu gibi VBA dünyasının da kilit noktalarından ikisidir. Bunlar aslında Algoritma dediğimiz kavramın omurgasını oluştururlar. Bu bölümde "x=y ise şunu yap, değilse bunu yap" tarzı sorgulamaları ve "bir şeyi yapmaya başla, bunu x defa yap veya şu şey olana kadar yap" gibi döngüsel yapıları göreceğiz.

## If'li Koşul Yapıları

Programlamanın temel taşlarından biri koşullu yapılarsa, koşullu yapıların temeli de IF bloklarıdır. Birçok programlama dilinde de zaten bu yapı kullanılır, sadece syntaxı farklı olabilmektedir.

Genel yapı şu şekildedir: **If....Then ....** Tabi bu yapı farklı görünümler kazanabilmektedir. Hepsine tek tek bakalım, sonra da örnekleri verelim.

### Temeller

#### Kullanım şekli

|  |  |
| --- | --- |
| Kullanım şekli | Syntax |
| Tek satırda yazım(kısa bir işlem yaptıracaksak) | If isim="Volkan" Then Goto devam |
| Tekli if bloğu. Birden fazla işlem yaptıracaksak If-End If bloğu arasında yazılır. | If isim="Volkan" Then    MsgBox "merhaba volkan"    Goto devam End If |
| Çoklu if bloğu: Eğer şöyleyse şunu yap yok değilse bunu yap, diğer durumlarda ise şunu yap. | If isim="Volkan" Then    MsgBox "Merhaba volkan" ElseIf isim="Serkan" Then    MsgBox "Merhaba serkan" Else    MsgBox "Merhaba değerli insan" End If |
| İç içe If: Eğer şöyleyse, ve yine eğer böyleyse şunları yap | If isim="Volkan" Then   If şehir="İstanbul Then    MsgBox "merhaba istanbullu volkan"   Else    MsgBox "merhaba anadolulu volkan"    End If Else    MsgBox "merhaba değerli insan" End If |
| IIF(tek satırda pratik değer atama) | YasDurum=IIF(yas<18;"Çocuk";Yetişkin") |

Şimdi hemen küçük bir örnek yapalım:

Sub basitif()

Dim segment As integer

Dim segmentAd as String

segment=Application.Inputbox("Segment kodunu girin",Type: =1)

If segment= 1 Then

segmentAd = "Bireysel"

Else

segmentAd = "Tüzel"

End If

MsgBox "Müşteri " & segmentAd & " segmenttedir"

End Sub

#### Operatörler

Temel operatörler şunlardır: eşitlik(=), eşit olmama(<>), büyük ve büyükeşit(>,>=) ile küçük ve küçükeşit(<,<=)

Bunlardan başka ne tür operatörler olduğunu aşağıda göreceğiz.

### İleri Seviye işlemler

#### Boolean değişkenler ve If blokları

"If X=4" mü diye bir sorgulama yaparken aslında X=4 sonucu True mu False mu(doğru mu yanlış mı) şeklinde bir sorgulama yapmış oluyoruz. Aksi belirtilmedikçe tüm sorgulamalarda sonucun True mu olduğunu sorgulamış oluruz, ve True ise Then ifadesinden sonraki kısım işletilir.

If bloklarında False sorgulaması da yapılabilir, bunun için açıkça If x= False diye sormak gerekir, tabi x'in boolean tipinde olması kaydıyla. Böylece ifadenin False olması durumunda da Then'den sonraki kısmın işletilmesi sağlanabilir. Unutulmamalıdır ki yaratılmış ancak henüz değer atanmamış Boolean değişkenlerin değeri False'tur. Bazen bu bilgi çok kullanışlı olmaktadır. Bi örnek yapalım: Kullanıcıdan ismini girmesini isteriz ve bu ismi, arasında ";" olacak şekilde 10 kere yazdırmak istiyoruz diyelim. Şimdi ilk başta hatalı bir kod yazalım. Bu kodda en başta da istenmeyen bir ";" olacak.

Sub booleanif1()

isim = InputBox("İsminizi girin")

For i = 1 To 10

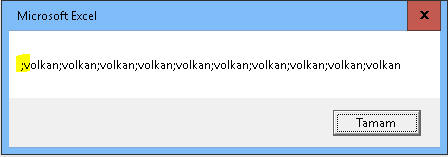
kelime = kelime + ";" + isim

Next i

MsgBox kelime

End Sub

Gördüğümüz gibi en baştaki ; fazladan oldu.



Ancak kodumuzda aşağıdaki gibi False sorgulamasını yaparsak istediğimizi elde ederiz.

Sub booleanif1()

Dim ilkDegerAtandımı As Boolean

isim = InputBox("İsminizi girin")

For i = 1 To 10

If ilkDegerAtandımı = False Then 'İlk etapta Falsetur, ve false ise bu kısım işletilir

kelime = isim

Else

kelime = kelime + ";" + isim

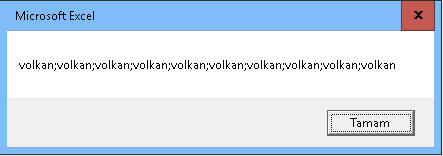
End If

ilkDegerAtandımı = True 'False'tan True'ya ilk geçişi burada yapıyoruz, sonrasında zaten 9 kez zaten True olan değere True atamış olacak

Next i

MsgBox kelime

End Sub



Başka bir örnek daha yapalım, bunu Interaktivite bölümünde de görmüştük, kullanıcı InputBox’a giriş yapmadan çıkarsa bunu yakalayalım. Detay için **Temeller>Interaktivite** konusuna bakınız.

Sub booleanif2()

a = Application.InputBox("Yaşınızı girin", Type: =1)

If a = False Then 'Variant her değeri alabileceği için False ifadesi aynen yazılır

MsgBox "Bir giriş yapılmadan çıkmayı tercih ettiniz"

'Diğer kodlar buraya

Else

MsgBox "Giriş yapıldı"

End If

End Sub

**NOT**: Boolean ifadelerde True sorgulaması yaparken "**=True**" yazmaya gerek yoktur, o yüzden içinde = operatörü olmayan bir sorgulama şekli gördüğünüzde şaşırmayın. Aşağıda bir örnek var.

Sub yaskontrol()

Dim yas As Integer

Dim resitmi As Boolean

yas = Application.InputBox("yaşınızı girin", Type: =1)

If yas >= 18 Then resitmi = True 'else durumunda False atamaya gerek yok hiçbir atama olmaz, atanmamaış boolenalar da default değeri yani false değeirini alır

If resitmi Then '=True yazmadık, yazabilirdik de.

MsgBox "reşitsin"

Else

MsgBox "reşit değilsin"

End If

End Sub

Bu yöntemi sadece değişkenlerle değil Booelan tipli dönüşü olan fonksiyon veya propertylerle de kullanabiliriz.

If Application.DisplayAlerts Then '=True yazmamama gerek yok çünkü DisplayAlerts propertysi Booelan döndürür

'kodlar

End If

#### Mantıksal kontroller

If ile birlikte kullanılan bazı built-in kontrol yapıları vardır. Yani =, <, > ve is operatörlerinden başka bir de bu ifadelerle kontrol sağlanır, ki bunlar da aslında Boolean sorgulama şekilleridir.

**IsEmpty**: (Variant tipli) değişken boş mu, yani henüz bir değer atanmadı mı? Veya bir hücrenin içi boş mu?

**IsDate**: Değişken, tarihsel bir değişken mi? Pratikte, Inputbox’a girilen bir tarihin doğru formatta girilip girilmediğini kontrol etmek için kullanılır.

**IsNumeric**: Değişken sayısal mı? Inputbox’a sayısal girilmesi gereken bir verinin doğru girilip girilmediğini kontrol amaçlı kullanılabildiği gibi, bir hücredeki verinin sayısal veri içerip içermediğini kontrol etmek için de kullanılır. Gerçi Inputbox’ı Application.Inputbox şeklinde kullanıp bir de veri tipi olarak 1 seçersek zaten kullanıcıyı otomatikman sayısal girmeye zorlamış oluruz ama normal Inputbox’la sorulduğunda bunu kontrol amaçlı kullanabilirsiniz.

**IsNull**: Değişken, bir veri içeriyor mu?

Bütün bunları **Not** operatör ile ters mantıkta sorgulayabilirsiniz. Bu konuyu bir alt bölümde göreceğiz.

**Çeşitli örnekler:**

1.Örnek

'bulunduğunuz yerden ilk dolu hücreye kadar olan tüm boş hücreleri silmek için

'tabi bu örneği bir döngü içinde yapmak daha şık olurdu ancak burda örnek vermek

'adına If ve GoTo ile yapılmıştır

bas:

If IsEmpty(ActiveCell) Then

ActiveCell.EntireRow.Delete

ActiveCell.Offset(1,0).Select

GoTo bas

End If

2.Örnek

'Belli bir anda aktif hücrenin değerinin sayısal olmaması durumunda programı durduruyoruz

If IsNumeric(ActiveCell.Value) Then Exit Sub

#### If Not

Bazen bir kodu günlük konuşma dilinde olduğu gibi önce negatifini sorgulayarak yazmak isteyebiliriz. X=y değilse şu kod çalışsın. Biraz kötü bir örnek olacak ama mesela aşağıdaki kodda, sadece 18 yaşından küçükler için çalışacak bir kod yazmış oluyoruz.

'Öncül kodlar

If Not Yaş>18 Then

'işletilecek kod

End If

'diğer kodlar

Kötü bir örnek dedik çünkü  burda if Yaş<=18 diye yazsaydık daha az kodla yazılmış olurdu. Evet olurdu, ancak bazen öyle durumlar olacak ki günlük konuşma diline göre kod yazmak size daha rahat gelecek ve negatifi sorgulamak da daha mantıklı olacaktır.

Şimdi mybooelan adına bir Boolean tipli değişkeniniz olsun. Bu durumda **If Not myboolean** yazmak ile **If mybooolean=False** yazmak arasında bir fark yok gibi görünebilir ancak hem konuşma diline yakınlığı dolayısıyla anlaşılırlığı hem de performans açısından **Not**'lı versiyon biraz daha öndedir. Çünkü önce mybooleanını ne olduğunu hesaplıyor, bu 1; sonra bunun False mu olduğuna bakıyor, bu da 2. Halbuki **Not** diyince sadece 1 değerlendirme işlemi yapmış oluyorsunuz. Ancak bu performans farkı o kadar da büyük bir fark değil. Bu konu, daha çok okunaklılık ve kişisel tercihle alakalıdır.

**Not** yöntemi pratikte en çok Boolean karşılaştırmalarda kullanılır ve özellikle de Booelan değer döndüren bir fonksiyonu sonucu ile. Mesela DosyaMevcutmu diye bir fonksiyonunuz olsun, eğer mevcutsa True, değilse False döndürüyor olsun. Hergün saat 12de bir dosyanın oluşup oluşmadığına bakan scheduled bir kodunuz olsun. Eğer dosya henüz oluşmamışsa kullanıcılara mail atmadan çıksın, oluştuysa mail atsın.

If Not DosyaMevcutmu(dosyadı) Then

Exit Sub

Else

'uzunca bir mail atma kodu

End If

Şimdi yine bir itiraz gelebilir, bu örnekte de aslında Not kullanmasak olabilir, ilk başa DosyaMevcutmu diye yazar mail işlemini Else öncesine koyabilir, Exit Sub'ı da Else sonrasına. Ancak gördüğünüz gibi mail işlemi oldukça uzun olabilir, mesela 50 satır. Böyle bir durumda ilk bloğa daha kısa olan kodu koymak daha mantıklı ve okunaklıdır. Böylece dosya oluşmadıysa ne yapılacağını çok net görebiliyorum, ancak başa dosya oluşma durumunu koyup Else'den sonra oluşmama durumunu koysaydık Else'den sonra ne olduğunu görmek için scrollbarı aşağı indirmek gerekecekti. Küçük bir detay ama bu tür pratiklikler size hep zaman kazandırır. Zaten makroların amacı da bu değil mi, bize her anlamda zaman kazandırmak. O yüzden kodlarımızı düzenlerken en okunaklı, bir zaman sonra tekrar içine baktığınıza anlaması en kısa sürecek şekilde düzenlemekte fayda var. Bu arada bu kod da pek tabiki **Not DosyaMevcutmu**  yerine **DosyaMevcutmu =False** şeklinde yapılabilirdi ama yine okunurluk açısından **Not**'lı halini gündelik dile daha çok benzetiyorum. Tercih sizin.

##### Mantıksal fonksiyonlarla Not kullanımı

Yine mantıksal fonksiyonların ters sorgulaması yapıldığında **False** ile sorgulamak yerine gündelik dile daha yakın olması açısından **Not** ile sorgularız. Ör: Kullanıcının boş mu dolu mu bir hücre seçtiğini kontrol edecek bir kod yazalım.

tekrar:

Set alan=Application.InputBox("Hücre seçin", Type: =8)

If Not IsEmpty(alan) Then 'boş değilse. If IsEmpty(alan)=False de olur ama biraz garip görünür

'işletilecek kodlar

Else

MsgBox "Boş bir hücre seçtiniz, lütfen dolu bir hücre seçin"

GoTo tekrar

End If

##### Nothing ve Not

Bir de Not'ın tamamen zorunlu olduğu bir durum var ki, o da Nothing ile birlikte kullanım şeklidir. Genel Syntax şöyledir: **If Not nesne Is Nothing.** Burada nesne Object tipindeki herşey olabilir. Range, Worksheet, Outlook Application v.s. Kelime kelime tercümesi "nesne hiçlik değilse" şeklinde yapılabilecek olan bu cümlecik aslında "nesne bir şeyse, yani birşey atanmışsa veya henüz boşaltılmadıysa" anlamında daha Türkçe olarak yorumlanabilir. Ancak VBA’de "**Birşey**" yani **Something** diye bir ifade olmadığı için bunun zıddı olan "**Not Nothing**" şeklinde yazılır.

Mesela Worksheet olayları bölümünde sıkça kullanacağımız bir **Intersect** metodu var. Worksheet değiştiğinde eğer değişen hücreler belli bir aralıkta mı diye kontrol ederiz, işte burda **Not ve Nothing** kombinasyonu kullanılır.

Ör: **If Not Intersect(Target, Range("C3: C4")) Is Nothing Then** aslında şu demektir: **If Intersect(Target, Range("C3: C4")) Is Something Then**, yani değişen hücre(Target) C3: C4 alanı ile kesişim kümesindeyse, yani kesişim kümesi boş küme değilse, yani "birşeyse". Worksheet olaylarında daha detaylı örnekler için ilgili konunun bulunduğu sayfalara göz atabilirsiniz.

Aşağıda da 3 değişkenden 2sine değer atanma durumu söz konusu, diğer açıklamalar commentlerde bulunuyor.

Sub ifnotnothing()

Dim hucre As Range

Dim ws As Worksheet

Dim wb As Workbook

Set hucre = ActiveCell

Set ws = ActiveSheet

If hucre Is Nothing Then 'if not nothing, hücrenin boş olup olmadığını sorgulamak değildir.

'hucre değişkenine bir değer atanıp atanmadığını sorgulamaktır. boş olup olmadığı IsEmpty ile sorgulanır

Debug.Print "hucre değişkenine atama yapılmamış"

Else

Debug.Print "hucre değişkenine atama yapılmış"

End If

If Not ws Is Nothing Then 'if ws is something yani birşey ise

Debug.Print "ws değişkenine atama yapılmış"

Else

Debug.Print "ws değişkenine atama yapılmamış"

End If

'sadece something olma durumunda çalışacak kod bloğu

If Not wb Is Nothing Then 'if wb is something yani birşey ise

Debug.Print "buraya girmeyecek"

End If

End Sub

##### Koşulu terse çevirme

Bu arada boolean tipli özellikleri tersine çevirmek için o özelliğin mevcut durumunu öğrenmek amacıyla If yapısını kullanmaya gerek yoktur; **Not** operatörü ile doğrudan tersine çevrilebilir. Mesela sayfadaki gridler açıkken kapatan, kapalıyken açan(bu işlemlere toggle işlemleri denir) bir buton yapıp quickaccessbara ekleyebilirsiniz.

If'li kod şöyle olacaktır.

Sub grid\_toggle()

If ActiveWindow.DisplayGridlines Then 'Dikkat ettiyseniz =True yazmadım, yazsam da bişey fark etmeyecekti

ActiveWindow.DisplayGridlines=False

Else

ActiveWindow.DisplayGridlines=True

End If

End Sub

Onun yerine basitçe şöyle de yazabilirsiniz:

Sub grid\_toggle()

ActiveWindow.DisplayGridlines = Not ActiveWindow.DisplayGridlines

End Sub

Tabiki, bunu yapabilmeniz için ilgili propertynin(özelliğin) hem okunabilir hem yazılabilir(Returns and Sets) karakterde olması lazım.

#### İçiçe If ve Bağlaçlar(Birden fazla koşul sorgulama)

Birden fazla koşulu sorgulamak için içiçe birkaç IF bloğu yazılabilir. Ör: Açılış Tarihi bu yıldan önce olan şubelerden sadece Bireysel Şube sayısını öğreneceğimiz bir kod yazalım.

If Subeaçılıştarihi< Year(Now) Then

If tip="Bireysel" Then

'işletilecek kodlar

End If

End If

Bunu yapmanın bir diğer yolu da **And** bağlacını kullanmak olacaktır.

If Subeaçılıştarihi< Year(Now) And tip="Bireysel" Then

'işletilecek kodlar

End If

And kullanmak sanki daha mantıklı gibi ama ikinci koşuldan önce ara bir işlem yapmak isterseniz içiçe IF kullanmanız gerekir.

If Subeaçılıştarihi<Year(Now) Then

'ara işlemler

If tip="Bireysel" Then

'işletilecek kodlar

End If

End If

Koşul yapısı **Veya** şeklinde olacaksa da **Or** kullanılır. Ör: Şube tipi bireysel veya Karma ise şunları yap gibi.

If tip="Bireysel" or tip="Karma" Then

'işletilecek kodlar

End If

##### Çoklu If bloğu vs Ayrı If blokları

Bazen öyle anlar gelir ki, konuşma dilinde söylediğimiz şey sanki çoklu if kullanmamız gerektiğini ima eder, ancak yapılması gereken işlem ayrı if blokları kullanmak olabilir. Hemen bir örnek yapalım.

Diyelim ki, bir schedule prosedürünüz var ve Excel açılır açılmaz devreye giriyor.(Scheduling işlemleri için **Dört Temel Nesne>Application** konusundaki OnTime kısmına bakınız). O an saat 08: 01den büyükse A makrosu, 08: 11'den büyükse B makrosu v.s çalışsın istiyorsunuz. Söylerken kulağa çoklu if kullanılacakmış gibi geliyor ancak öyle yaptığımız durumda ilk koşul gerçekleşince kalanı işletilmez. Aslında burda düşüncemizi dile getirme şeklimiz de hatalı olabilir. O yüzden düşündüğümüz şeyi kelimelere daha doğru dökelim: A makrosu, o an saat 08: 01'den büyükse çalışsın, B makrosu saat 08: 11'den büyükse çalışsın v.s. Küçük bir nüans var, fark edebildiniz mi? Evet, önce makronun adını sonra zamanı dile getirdik. 712575

İlk versiyona göre kodumuzu şöyle hazırlardık ve hatalı bir işlem yapmış olurduk:

If TimeValue(Now) > TimeValue("08: 01: 00") Then

Application.OnTime Now + TimeValue("00: 02: 00"), Procedure: ="pyskontrol"

ElseIf TimeValue(Now) > TimeValue("08: 11: 00") Then

Application.OnTime Now + TimeValue("00: 02: 30"), Procedure: ="nrkontrol"

ElseIf TimeValue(Now) > TimeValue("08: 21: 00") Then

Application.OnTime Now + TimeValue("00: 03: 00"), Procedure: ="pdmkontrol"

ElseIf TimeValue(Now) > TimeValue("08: 31: 00") Then

Application.OnTime Now + TimeValue("00: 04: 00"), Procedure: ="pargkontrol"

End If

Doğru kod aşağıdaki gibi olacaktır.

If TimeValue(Now) > TimeValue("08: 01: 00") Then

Application.OnTime Now + TimeValue("00: 02: 00"), Procedure: ="pyskontrol"

End If

If TimeValue(Now) > TimeValue("08: 11: 00") Then

Application.OnTime Now + TimeValue("00: 02: 30"), Procedure: ="nrkontrol"

End If

If TimeValue(Now) > TimeValue("08: 21: 00") Then

Application.OnTime Now + TimeValue("00: 03: 00"), Procedure: ="pdmkontrol"

End If

If TimeValue(Now) > TimeValue("08: 31: 00") Then

Application.OnTime Now + TimeValue("00: 04: 00"), Procedure: ="pargkontrol"

End If

#### IIf

Bir değişkene, bir koşul sonucuna bakarak bir değer atamak istiyorsanız **IIf** yapısı çok kullanışlıdır ve kullanımı oldukça basittir.

Aşağıdaki kodda, segment kodu 1 ise Bireysel değilse Tüzel şeklinde bir atama yapılmaktadır.

Segment=IIf(segmentkodu=1,"Bireysel","Tüzel")

## Select Case ve Diğer koşullu yapılar

If yapısı tek koşul yapısı değildir. **Select Case** yapısı başta olmak üzere üç ayrı ifade daha vardır. Bu bölümde de bunlara bakacağız.

### Select Case

Kontrol edilen şey en az 3 farklı değer için kontrol ediliyorsa **If Then Else** yerine **Select Case** yapısı tercih edilebilir; zira okunurluk açısından böylesi daha iyi olacaktır. Genel kullanım şekli şöyledir.

Select Case SorgulananŞey

Case şuysa

şunu yap

Case buysa

bunu yap

Case ....

....

Case Else 'Diğer durumlarda da burayı işlet

....

End Select

Örneğimizi kompleks durumları da içerecek şekilde yapalım.

Dim sayı as Integer

sayı=Inputbox("Bir sayı girin")

Select Case sayı

Case Is < 0

MsgBox "Negatif bir sayı girdiniz"

Case 1 To 9

MsgBox "Pozitif bir rakam girdiniz" '0-9 arasındaki herşey rakamdır, 9un üstündekiler rakam değil sayıdır

Case Is > 0

MsgBox "Pozitif bir sayı girdiniz"

Case 0

MsgBox "0 girdiniz" '0 da bir rakamdır ama biz onu ayrı ele allaım dedik

Case Else

MsgBox "Lütfen geçerli bir sayı giriniz"

End Select

Gördüğünüz gibi bir değer aralığı için **x To y** şeklinde, bir şeyden küçük/büyük için ise **Is > 0** şeklinde yazıyoruz. Birkaç değeri yanyana yazacaksa da "," ile birbirinden ayırırız. Dosya formatına göre uzantı belirlenen aşağıdaki örnekte de bu durumu görüyoruz.

Select Case ActiveWorkbook.FileFormat

Case "-4143", "-4158", 6, 56 'normal xls, txt, csv veya Excel2007deki 97-2003 xls'i mi

FileExtStr = ".xls"

Case 50, 51, 52 'xlsx, xlsb veya xlsm ise

FileExtStr = ".xlsx"

Case Else

MsgBox "Bu dosya formatı bu makronun çalışması için uygun değil. xls, xlsx, xlsb, xlsm, txt veya csv dosyalarıyla çalışmalısınız"

Exit Sub

End Select

**NOT**: En az 3 değer için sorgulama olması durumunda Select Case tercih edilmeli dedik, ancak sorgulanan şey her defasında değişiyorsa yine If Then Else kullanılmalıdır. Ör:

Sub ifmiselectmi()

If a > 10 Then

'şunları yap

ElseIf b > 20 Then

'şunları yap

ElseIf c > 50 Then

'şunları yap

ElseIf d > 100 Then

'şunları yap

Else

'şunları yap

End If

End Sub

### Diğer iki koşul yapısı

#### Choose

1'den başlayıp artarak giden bir sırada ilerleyen bir değer grubu varsa, ve bir değişkene değer atamak istiyorsanız, bunlar için If veya Select Case yapısı kullanmak yerine **Choose** ifadesinin kullanmak çok daha pratik olacaktır. Genel Syntax'ı şöyledir.

**Syntax: Choose(değişken,değer1,değer2,......,değerx)**

Bu örnekte değişken=1 değerini alıyorsa dönen değer Değer1, değişken=2 ise dönen değer=Değer2 olur ve bu şekilde ilerler.

Aylar için bu özellik rahatlıkla kullanılabilir. Örneğin, ay isimlerinden oluşan bir klasör grubunuz var diyelim. Kullanıcıya ay ismini girdirmek yerine ay numarasını isteyerek istediğiniz ay adını elde edebilirsiniz.

Sub chooseornek()

Dim ayno As Integer

Dim ayadı As String

ayno = Application.InputBox("ay no giriniz", Type: =1)

ayadı = Choose(ayno, "Ocak", "Şubat", , , , , "Aralık")

'kodların devamı

Debug.Print ayadı

End Sub

Not: Eğer eleman sayısından fazla bir index girilirse **null** döndürür. Bu örnekte 13 girmek gibi. Tabi ayadı değişkenimiz String tanımlandığı için kod hata alır. Ancak String yerine variant tanımlanırsa Immediate Window'a Null yazdığını görebilirsiniz.

#### Switch

**If Else End If**'in tek satır karşılığı nasıl **IIF** ise, Select Case'in de tek satır versiyonu Switchtir. Genel kullanımı şekli şöyledir:

x=Switch(Şart 1,Sonuç 1,Şart 2,Sonuç 2,....,Şart n,Sonuç n)

Sub switchornek()

kanalkodu = 1

kanaladı = Switch(kanalkodu = 1, "Şube", kanalkodu = 8, "İnternet", kanalkodu = 16, "Mobil")

MsgBox kanaladı

End Sub

Okunurluğu artırmak adına şu şekilde de düzenlenebilir:

Sub switchornek()

kanalkodu = 1

kanaladı = Switch(kanalkodu = 1, "Şube", \_

kanalkodu = 8, "İnternet", \_

kanalkodu = 16, "Mobil")

MsgBox kanaladı

End Sub

## For...Next... Döngüleri

VBA dünyasının en temel döngü tipi For döngüleridir. Bir önceki koşullu yapılar konusunda söylediğimiz gibi döngüler de programlamanın olmazsa olmazlarıdır. Onlar olmasaydı belki 3 satırlık kodu onlarca hatta yüzlerce satır şeklinde yazmamız gerekirdi.

For döngülerinin 2 tipi vardır:

* Basit For Next döngüsü
* For Each döngüleri

İlk olarak basit For döngüsü ile başlayalım

### For...Next...

Genel kullanım şekli şöyledir:

For i=başlangıçsayısı to bitişsayısı

'i'ye bağlı veya i'den bağımsız olarak şunu yap

Next i

Gördüğünüz gibi, sayaç olarak kullandığımız i değişkenini döngü içinde de parametrik olarak kullanabiliriz veya kullanmayabiliriz. İkisine de örnek verelim. Önce i'yi döngü içinde parametrik olarak kullanacağımız örnek olsun. Bu örnekte, A1'den A10'a kadar olan hücrelere sırayla i'nin kendisini yazmış oluruz.

For i=1 to 10

Cells(i,1)=i

Next i

Şimdi de parametrik olmayan örneğe bakalım, bunda ise Excel’e 10 kez bip sesi çıkarttırıyoruz.

For i=1 to 10

Beep

Next i

i sayacı genelde 0 veya pozitif(ki bu da genelde 1dir) bir sayı olmakla birlikte negatif bir sayı da olabilir, ama bunun pratikte pek kullanıldığını görmedim.

Sayacımız normalde 1er 1er artar, ancak istersek **Step** ifadesi ise 2'şer, 3'er de arttırabiliriz. Hatta sayacı geriye doğru da saydırabiliriz, ki bunun için Step ifadesinden sonra negatif bir sayı gelir.

For i = 10 To 1 Step -2

Cells(i / 2, 1) = i

Next i

Tüm sayfalarda gezindiğimiz bir kod:

For i = 1 To Sheets.Count

Worksheets(i).Select

'diğer kodlar

Next i

Şimdi bir başka örnekte de ilk hücreden en alt hücreye doğru ilerleyelim.

For i = 2 To Cells(1, 1).End(xlDown).Row

Cells(i,1).Select

'Diğer kodlar buraya

Next i

#### İçiçe For Döngüleri

For döngülerini de tıpkı koşullu yapılarda olduğu gibi iç içe geçmiş şekilde kullanabiliriz. Aşağıdaki örnekte boş olan hücreleri sarıya boyayalım ve sayısını bulalım:



Öncelikle şunu belirtmekte fayda var. Tek olsun içiçe olsun tüm döngülerde For satırını yazdıktan sonra hemen onunu bitişi olan Next satırını da yazın, aradaki kodları da girintili yazın. İçiçe For Next olacaksa bu ikinci For'u da yine girintili yazın. Şimdi kodumuza bakalım.

Sub bosluksay()

Dim adet As Integer

For i = 1 To Range("A1").End(xlToRight).Column

For k = 1 To Range("A1").End(xlDown).Row

If IsEmpty(Cells(k, i)) Then

Cells(k, i).Interior.Color = vbYellow

adet = adet + 1

End If

Next k

Next i

MsgBox "toplamda " & adet & " adet boş hücre var"

End Sub

ve sonuç:



### For Each...Next...

For Each yapısı bir obje grubu ve bir dizi(veya dizimsi) içindeki elemanlar içinde gezinmek için kullanılır.

For Each nesne in nesnegrubu

'nesneyle ilgili bir işlem

Next nesne

For Each elaman in dizi

'dizi elemanıyla ilgili işlem

Next eleman

Diziler hakkında ön bilgi almak için **Dizimsiler>Diziler** konusuna bakabilirsiniz.

Şimdi yukardaki boşluk saydırma örneğini For Each ile nasıl yapacağımıza bir bakalım.

Sub bosluksay2()

Dim adet As Integer

Dim hucre As Range, alan As Range

Set alan = Range("A1").CurrentRegion

For Each hucre In alan

If IsEmpty(hucre) Then

hucre.Interior.Color = vbYellow

adet = adet + 1

End If

Next hucre

MsgBox "toplamda " & adet & " adet boş hücre var"

End Sub

Bence For Each ile sanki daha kolay gibi görünüyor. O yüzden böyle bir görevde ben ForEach'i kullanırdım. Ancak eğer döngü içinde i veya k parametrelerinden birini kullanarak da bir işlem yapmam gerekseydi, o zaman basit For döngüsü kullanırdım, gerçi yine For Each kullandığımızda hücrenin row ve column özellikleri ile i ve k'yı dolaylı olarak elde edebilirim ama basit For'da bunlar zaten elimde olacaklardı. Kısacası duruma göre hangisini kullanacağınıza siz karar vereceksiniz. Basit For'un esnek başlangıç ve bitiş değerleri ile hareket yönündeki esnekliği avantaj sayılabilirken, For Each için daha hızlı olmasını avantaj olarak vurgulayabiliriz.

For Each yapısını, Range'teki hücrelere ek olarak, Workbooktaki sayfalar ve tüm workbooklar arasında dolaşma şeklinde de bol miktarda kullanıyoruz.

'workbooklarda dolaşma

Dim wb As Workbook

For Each wb In Workbooks

MsgBox wb.FullName

Next wb

'worksheetlerde dolaşma

Dim ws As Worksheet

For Each ws In ActiveWorkbook.Sheets

MsgBox ws.Name

Next ws

Yine dizi veya dizimsilerin elemanları içinde gezinirken de her iki yapı kullanılabilir. Basit For'da **1 to elemansayısı** yapısı kullanılırken ForEach'te **each eleman in dizi** şeklinde kullanırız. Şimdi kısa bir örnek de dizilerle ilgili yapalım, daha detaylı örnekleri Diziler konusuna bırakalım.

Mesela sabit bir bölge kodu listeniz olsun ve bunların her birini bir diziye atayalım, sonra da bu bölge kodunu kullanarak bir işlem yapalım.

Sub forarray()

Dim blg() As Variant

blg = Array(5001, 5002, 5003, 5004, 5005, 5006, 5007, 5008, 5009, 5010)

For Each b In blg

Workbooks.Open ("C: \bölge raporları\" & b & "-netice raporu.xlsx")

'diğer işlemler

Next b

End Sub

Akılda bulundurulması gereken önemli bir husus, ForeEach kullanıldığında read-only bir özellik gösterir. Yani bu yöntemle dizi elemanlarını değiştiremezsiniz. Elamanları değiştirmek istiyorsanız basit For döngüsü kullanmanız lazım. [http://Excelmacromastery.com/VBA-for-loop/](http://excelmacromastery.com/vba-for-loop/) sitesinden aldığım bu iki örnekte fark çok açıkça anlatılmış.

Sub UseForEach()

' Diziyi yaratıp 3 değer ekliyoruz

Dim arr() As Variant

arr = Array("A", "B", "C")

Dim s As Variant

For Each s In arr

' s'nin atadığı değeri değiştirmeye çalışıyoruz

s = "Z"

Next s

' Ama değişmemiş olduğunu görüyoruz

For Each s In arr

Debug.Print s

Next s

End Sub

Basit for ile değişimi görelim

Sub UsingForWithArray()

Dim arr() As Variant

arr = Array("A", "B", "C")

Dim i As Long

For i = LBound(arr) To UBound(arr)

arr(i) = "Z"

Next

For i = LBound(arr) To UBound(arr)

Debug.Print arr(i)

Next

End Sub

Son bir örnek daha yapalım. Bu sefer, toplu bir goalseek(hedef ara) işlemi yapalım.

Sub toplugoalseek()

Dim h As Range

For Each h In Range("E8: E2294")

h.Select

h.goalseek Goal: =h.Offset(0, 4).Value2, ChangingCell: =h.Offset(0, 2)

Next h

End Sub

### Döngüden Çıkış

Yukarıdaki kod örneklerinden de gördüğünüz üzere, For döngüleri genellikle kaç kez çalıştırılacağını bildiğiniz durumlarda kullanılır. Ör: **For i=1 to 10** dersek, 10 kez çalışacağını biliriz veya **For Each item in dizi** dersek dizideki tüm elemanlar için çalışacağını biliriz. Tabiki üst limitin değişken olduğu bazı durumlarda, o limite başka VBA kodları ile de ulaşabilir ve yine For döngülerini kullanabiliriz, mesela en alt satırın kaç olduğunu bilmiyoruzdur ama Range("A1").End(xlDown).Row ile bunu bilip döngüyü buraya kadar götürebiliriz, gerçi bu tür durumlarda diğer döngü türlerini kullanmak daha kullanışlıdır.

Ancak bazen, döngüyü erken terk etmeniz gerekebilir. Yani döngüye girdiniz diye ille sonuna kadar gitmeniz gerekmiyor. Bunun için **GoTo** ifadesi kullanılabileceği gibi, **Exit For** yapısı da kullanılabilir. GoTo kullanımında doğrudan belli bir etikete yönlendirilirken Exit For ile döngünün hemen arkasından devam edilir.

Mesela yukardaki örneğimizi, boşluğa ilk rastlanılan hücrenin adresini verecek şekilde değiştirelim.

Sub bosluksay3()

Dim a As String

Dim hucre As Range, alan As Range

Set alan = Range("A1").CurrentRegion

For Each hucre In alan

If IsEmpty(hucre) Then

a = hucre.Address

Exit For

End If

Next hucre

If a <> vbNullString Then

MsgBox "ilk olarak " & a & " adresinde boşluğa rastlanmıştır"

Else

MsgBox "herhangi bir boş hücre bulunmamaktadır"

End If

End Sub

## Do ve While Döngüleri

For döngülerini incelerken gördük ki, bu döngüler genellikle çevrelenen kod parçasının kaç kez çalıştırılacağının bilindiği durumlarda kullanılmaktaydı.

Üst limitin bilinmediği diğer durumlarda ise çoğunlukla **Do** ve **While** döngüleri kullanılır. While ifadesi ile genelde "**Do While**" kalıbı içinde geçer ve bu aslında While döngüsü değil bir Do döngüsüdür. **While Wend** döngüsü artık çok kullanılan bir döngü olmayıp bu bölümdeki  ve site genelindeki kodlarımızı Do döngüleriyle halletmeye çalışacağız. En son kısa bir While Wend döngü örneği de yapıp konuyu bitireceğiz.

Do döngüleri de For döngüleri gibi dizi ve dizimsilerde bol miktarda kullanılır, ayrıntılı bilgi için For Döngüleri ve Diziler konularına bakabilirsiniz.

### Do Döngüleri

Do döngülerinin 2 ana, 4 alt tipi vardır.

* **Do While**
  + **Do While Şart şuysa.......Loop**: Şart gerçekleşmezse çevrelenmiş kod hiç çalışmayabilir
  + **Do ....... Loop While Şart şuysa**: Şart gerçekleşse de gerçekleşmese de **en az 1 kez çalışır**
* **Do Until**
  + **Do Until Olay......Loop**: Olay olana(şart gerçekleşene) kadar çalışır, döngüye girildiğinde olay zaten olmuşsa(şart gerçekleşmişse) çevrelenmiş kod hiç çalışmaz
  + **Do ....... Loop Until Olay**: Olay olana(şart gerçekleşene) kadar çalışır; Çevrelenmiş kod **en az 1 kez çalışır**

Gördüğünüz üzere, **şart eğer Do satırındaysa, kod hiç çalışmayabiliyor, ama Loop satırındaysa en az 1 kez çalışır.**

Hemen bir örnek yapalım. Kullanıcıdan karesi alınacak bir sayı girmesini isteyelim ve eğer kullanıcı geçerli bir sayı yerine başka birşey mesela bir harf girerse başa dönmesini sağlayalım. Bunu bir If ve GoTo ile de yapabilirdik ama bu sefer Do ile yapacağız.

Sub dowhile1()

'bu en az 1 kere çalışır, ve sayı girene kadar bize aynı soruyu sorar

Do

sayı = InputBox("Karesi alınacak bir sayı girin")

Loop While Not IsNumeric(sayı)

MsgBox sayı & " sayısının karesi şudur: " & sayı \* sayı

End Sub

Sub dowhile2()

'Bu ise soruyu hiç sormaz. Çünkü önce şart kontrolünü yapar,

'şart sağlanmadığı için döngüye girmeden çıkar

'Şart sağlanmaz çünkü say değişkeni varianttır ve

'Variantlara henüz değer atanmadıysa 0 değerini alır, yani IsNumeric sorgusu True'dur

'başında da bir Not operatörü olduğu için şart sağlanmaz.

Do While Not IsNumeric(sayı)

sayı = InputBox("Karesi alınacak bir sayı girin")

Loop

MsgBox sayı & " sayısının karesi şudur: " & sayı \* sayı

End Sub

Aşağıda ise Do Until yapısına ait 2 örnek bulunuyor.

Sub dountil1()

'sayfa sayısı 5 olana kadar işlem yapar. 5ten çoksa siler azsa ekler.

Application.DisplayAlerts = False

If Sheets.Count < 5 Then

Do Until Sheets.Count = 5

Sheets.Add After: =Sheets(Sheets.Count)

Loop

ElseIf Sheets.Count > 5 Then

Do Until Sheets.Count = 5

Sheets(Sheets.Count).Delete

Loop

End If

End Sub

Sub dountil2()

'en az bir kere çalışır, taki sayfa sayısı 1 olana kadar

Do

Sheets(2).Delete 'Her defasında hep 2.sayfa silinir, ta ki tek sayfa kalana kadar

Loop Until Sheets.Count = 1

End Sub

#### While mı Until mi?

Hangi durumda hangisini kullanmalıyız?

İkisini de her durumda kullanabilirsiniz. Tek farkı, birini diğerinin tersi mantıkla yazmak. Geliştiriciler neden böyle bir ayrıma girmişler emin değilim ama sanırım konuşma dilindeki kullanım tercihlerimizi göz önünde bulundurmuş olabilirler. Mesela çocuğumuza "yemeğin bitene kadar masadan kalkmak yok" da diyebiliriz, "yemeğin bitmediği sürece masadan ayrılmak yok" da, ikisi de aynı kapıya çıkar. VBA'de de durum pek farklı değil: En alt satıra inene kadar çalış da diyebiliriz, "aktif satır no<son satır no" olduğu sürece çalış da.

#### Döngüden Çıkış

For döngülerinde olduğu gibi Do Loop’larından çıkış için de Exit ifadesini kullanırız. Mesela aşağıdaki döngüde, boş bir hücreye rastlanıldığında döngüden çıkılır.

i=1

Do While i < 1000

If IsEmpty(Cells(i,1)) Then

Exit Do

End If

i = i + 1

Loop

#### İçiçe Loop

Diyelim ki bir bölgenin şubelerini hacimlerine göre boy sırasına dizdiniz, buna göre her bölgenin en yüksek hacimli rakamını yazdırmak istiyorsunuz, yani bir nevi Excel’de var olmayan(2016da MAXIFS geldi) bir MAXIF fonksiyonunu Sub prosedür olarak ele alacağız ve bunun tersi olan MINIF'i.

Tabiki bunları bir UDF ile yapmak daha şık olacaktır ancak içiçe Loop örneği görmek adına bu örnek faydalı olacaktır.

Sub maxif()

'maxı yazacağın yere gel, ordayken çalıştır ve liste sıralı olsun

Set kriter = Application.InputBox("ana değişken kriterinin olduğu sütundan bir hücre seç", Type: =8)

Set rakam = Application.InputBox("maksimumun arandığı sütunu seç", Type: =8)

ks = ActiveCell.Column - kriter.Column

rs = ActiveCell.Column - rakam.Column

Do

Maks = ActiveCell.Offset(0, -rs).Value

Set ilkyer = ActiveCell

Do While ActiveCell.Offset(0, -ks).Value = ActiveCell.Offset(1, -ks).Value

If ActiveCell.Offset(1, -rs).Value > Maks Then Maks = ActiveCell.Offset(1, -rs)

ActiveCell.Offset(1, 0).Select

Loop

Set sonyer = ActiveCell

Range(ilkyer, sonyer).Value = Maks

ActiveCell.Offset(1, 0).Select

Loop Until ActiveCell.Offset(0, -1).Value = ""

End Sub

Sub minif()

'mini yazacağın yere gel, ordayken çalışıtır ve liste sıralı olsun

Set kriter = Application.InputBox("ana değişken kriterinin olduğu sütundan bir hücre seç", Type: =8)

Set rakam = Application.InputBox("minimumun arandığı sütunu seç", Type: =8)

ks = ActiveCell.Column - kriter.Column

rs = ActiveCell.Column - rakam.Column

Do

Mini = ActiveCell.Offset(0, -rs).Value

Set ilkyer = ActiveCell

Do While ActiveCell.Offset(0, -ks).Value = ActiveCell.Offset(1, -ks).Value

If ActiveCell.Offset(1, -rs).Value <= Mini Then Mini = ActiveCell.Offset(1, -rs)

ActiveCell.Offset(1, 0).Select

Loop

Set sonyer = ActiveCell

Range(ilkyer, sonyer).Value = Mini

ActiveCell.Offset(1, 0).Select

Loop Until ActiveCell.Offset(0, -1).Value = ""

End Sub

### While Wend döngüsü

Başta da söylediğimiz gibi bu döngü tipi artık pek kullanılmıyor, sadece eski yazılmış bir kod karşımıza geldiğinde bilelim diye hala var. Ama MSDN'nin bize söylediği gibi, Do Döngüsü daha fonksiyonel ve esnektir ve yeni kodlarda bu kullanılmalıdır.

While döngülerinde sadece başta şart kontrolü yapılır, yani en az 1 kere çalışma olayı bunda seçime bağlı değildir. Bu da demektir ki döngü içindeki kodumuzun hiç çalışmaması mümkündür. Ayrıca Do ve For dönülerindeki gibi döngüden çıkış ifadesi(Exit) de yoktur.

Aşağıdaki örnekte 1'den 10 kadar olan sayıların toplamını alıyoruz.

Sub topla()

While sayı <= 10

toplam = toplam + sayı

sayı = sayı + 1

Wend

MsgBox toplam

End Sub

### Çeşitli Örnekler

#### Bir hücre grubuna harfleri yazdırma

Sub harfyaz()

Dim i As Integer

i = 1

Do While i < 27

Cells(i, 1) = Chr(i + 64) 'A harfinin Ascii kodu 65tir

i = i + 1

Loop

End Sub

#### İlk sayfa hariç tüm sayfaları silme

Do

Sheets(2).Delete 'Her defasında hep 2.sayfa silinir, ta ki tek sayfa kalana kadar

Loop Until Sheets.Count = 1

#### Database'den tek tek tarihler için data çekip başka sayfada alt alta getirme

Diyelim ki, bir nedenle tarihsel datasını tek tek çekmeniz gereken bir sorgu var. Ör: 1 Ocak için ayrı, 2 Ocak için ayrı,.... v.s Bunu bir SQL ortamında yapmanız ağır bir iş gelebilir. Neyse ki VBA ve döngüler sayesinde bu ağır iş bile kolaylaşabilir.

Sub dowhileornek()

Dim t As Date

t = "01.01.2016"

Do

'SQL stringini oluşturuyorum

s1 = "Select \* from PARG.ARG\_TNETICE\_MUSTERI\_GER" & Chr(13)

s2 = "where BAGLI\_URUN\_ID in (165,166,167,168,169,170,171,172,173,174,175,176,177,178,179,180,181,182,183,184)"

s3 = "and TARIH=" & t & " and KPI\_ID=1001 and RAPOR\_TURU=2"

s4 = "and musteri\_id in (Select ESLESEN\_TARAF\_ID from PDWH.DWH\_TSAKLAMA where YONLENDIREN\_SUBE <> 1234 and TARIH=t)" & Chr(13)

With ActiveWorkbook.Connections("DWH").ODBCConnection

.BackgroundQuery = True

.CommandText = Join$(Array(s1 & s2 & s3 & s4))

'Diğer kodlar

End With

ActiveWorkbook.Connections("Query from PDWH\_USR").Refresh

t = t + 1 'Tarihi 1 artırıyorum

ActiveCell.CurrentRegion.Select

Selection.Copy

Sheets(2).Select

ActiveCell.Offset(1, 0).Select

ActiveSheet.Paste 'ilk çalışan sorgu sonucunu 2.sayfaya ekleyip bir alt satıra iniyorum

Loop Until t = "15.02.2016" '15 şubata kadar yani 45 kez sorgu çalışacak

'SQL ortamınıda bunu yapmak çok yorucu olabilirdi

End Sub

#### Belirli adette sayfası olan dosya yaratma

Diyelim ki, kodumuzun bir yerinde 5 sayfalı bir workbook yaratmamız gerekiyor. Normalde Excel’in default ayarı yeni workbook açıldığında 3 adet sayfası olması yönündedir ama biz bu default ayarı değiştirmiş olabiliriz, mesela 1 veya 7 yapmış olabiliriz. O  yüzden ilk olarak bir If bloğu ile kontrol ederiz ve eğer 5 ten büyükse fazlalıkları sildirip, küçükse 5e tamamlayan bir kod yazarız. Sayfa sayımız zaten 5se kod hiçbir şey yapmaz.

Sub sayfaekleveyasil()

Application.DisplayAlerts = False 'sayfa silerken uyarmasın

If Sheets.Count < 5 Then

Do Until Sheets.Count = 5

Sheets.Add After: =Sheets(Sheets.Count)

Loop

ElseIf Sheets.Count > 5 Then

Do Until Sheets.Count = 5

Sheets(Sheets.Count).Delete

Loop

End If

End Sub

## Neler Öğrendik

Bu bölüm en önemli bölümlerden birisi. Zira burada kodlamanın temeli olan algoritma yapılarını gördük. Koşullu yapılarla belirli bir koşulda belirli kod parçalarının çalışmasını sağlarken, döngülerle aynı kodun birden çok çalışmasını sağladık. Makrolara(aslında tüm programlama dillerine) güç katan kısım da zaten bunlardır. Bu konuda kendinizi geliştirmenizi ne kadar tavsiye etsem azdır. Zira birçok problemin hap gibi bir cevabı bulunmayacak, bunlar için sizin bu yapıları kafanızda canlandırarak çözüm geliştirmeniz gerekecektir. Burda da üç aşağı beş yukarı belli sayıda düşünce kalıbı işinizi görecektir, bu kalıpların kafanızda yer etmesi için de bol miktarda probleme çözüm getirmeniz gerekmektedir.