Hedefler

Bu üniteyi çalıştıktan sonra;

- Veri Tiplerini öğrenmek,
- Değişkenleri öğrenmek,
- İfadeleri öğrenmek,
- Karar Yapılarını öğrenmek.

İçindekiler

Veri Tipleri, Değişkenler, İfadeler

Kod Açıklamaları

Tüm programlama dillerinde olduğu gibi Visual Basic.Net ile kodlama yaparken de kod satırları arasına, açıklayıcı bilgiler yazabilirsiniz. Böylece yazdığınız kodları okuyan başka bir kişinin de kolaylıkla ne yapıldığını anlamasını sağlamış olursunuz, yada bazen kendi kodlarınızı okurken bile açıklama satılarına ihtiyacınız olabilir.

Açıklama satırları eklerken, açıklama metninin sol tarafına ‘ işareti koymanız, yada **REM** deyimini yazmanız gerekir.

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

        ' Değişkenler tanımlanıyor
        Dim x1 As Integer
        Dim x2 As Integer
        Dim toplam As Integer

        ' Girilen sayılar okunuyor
        x1 = Textbox1.text
        x2 = TextBox2.Text

        REM Sonuç hesaplanıyor
        toplam = x1 + x2

        REM Sonuç yazdırılyor
        Label1.Text = toplam

End Sub
```

Açıklama satırları, uygulama derlendiğinde, kodun derlenmiş halinde yer almazlar ve uygulama çalışırken göz ardı edilirler. Yani sadece açıklama amaçlı kullanılırlar.

Değişkenler ve Veri Tipleri

Programlama içinde değerlerin tutulması için değişkenler kullanılır. Değişkenler bir bilginin bellekteki konumunu temsil eden sembolik isimlerdir. Bilgisayarda hemen hemen tüm işlemler bellekte yapılır. Program çalıştırıldığında değişken ve bu değişkenin türüne göre bellekte yer ayrılır. Program içerisinde veriler bu değişkenlere atanarak işlemler dinamik hale getirilir ve kolayca işlenebilir.

STANDART VERİ TİPLERİ		
Veri Tipi	**Byte**	**Aralık**
Short	2	-32,768 ... 32,767
Integer	4	-2,147,483,648 ... 2,147,483,647
Long	8	-9,223,372,036,854,775,808 ... 9,223,372,036,854,775,807
Single	4	-3.4028235E38 ... 3.4028235E38
Double	8	-1.79769313486231E308 ... 1.79769313486231E308
Decimal	16	-79,228 x 1024 ... 79,228 x 1024
Byte	1	0 ... 255
Char	2	0 ... 65,535
String	2	0 ... 2 milyar karakter
Boolean	2	True veya False (False durumunda 0 değeri döndürülür)
Date	8	1 Ocak 0001 ... 31 Aralık 9999
Object	4	Herhangi bir tip

Bir değişkenin tipi, onun için ne kadar önemliyse, değişkenin var olabilmesi için bir isime sahip olması da o kadar önemlidir. Yani bir değişkeni tanımlayabilmek için, bir değişken adına ve veri tipine ihtiyacımız vardır.

Değişken isimlendirilmesinde bazı kurallara uyulmalıdır;

- ✓ Değişken mutlaka bir harf ya da "_" (alt çizgi) işareti ile başlamalıdır.
- ✓ Değişken ismi 255 karakteri geçmemelidir (Aslında Visual Basic.NET' te 16383 karakter.
- ✓ Değişken ismi içinde boşluk bulundurulmamalıdır.
- ✓ Her ne kadar Visual Basic.NET değişken isimlerinde Türkçe karakterleri (ş,ı,ü,ö,ğ,ç...) desteklese de, değişken isimlerinde Türkçe karakterler kullanmamaya çalışın.
- ✓ Değişken isimlendirmede küçük-büyük harf ayrımı yoktur. Yani AD, ad, Ad, aD aynı değişkeni ifade eder.
- ✓ Visual Basic komutları ve fonksiyonları değişken ismi olarak kullanılmamalıdır.

Geçerli değişken isimleri : AD1, K127B, Birinci_Sayı, ADISOYADI, _128

Geçersiz değişken isimleri : 8ABC, SON SAYI, SUB

Değişkenlerin Tanımlanması

Visual Basic.NET' te değişken bildirim deyimi **Dim**' dir.

```
Dim Okul_Adi As String
```

Aynı anda birkaç değişken birlikte tanımlanabilir.

```
Dim x1, x2, sonuc As Double
```

visual Basic .NET' te tek satırda hem değişken tanımlaması hem de değer ataması yapmak mümkündür.

```
Dim yas As Integer = 40
```

Veri Tipleri ile İlgili Örnekler

Integer, Short, Byte ve Long Veri Tipi

Ondalık nokta içermeyen sayısal bilgileri bellekte tutmak için ***Integer*** tipinden yararlanılarak değişken tanımlanmaktadır. Visual Basic.NET projelerinde tam sayısal bilgiler için değişken tanımlamak için ***Integer*** Veri Tipinden başka ***Byte***, ***Short*** ve ***Long*** Veri Tipleri de kullanılmaktadır. ***Long*** değişkenler için bellekte 8 byte, ***Integer*** değişkenler 4 byte, ***Short*** değişkenler 2 byte, ***Byte*** değişkenler 1 byte yer kaplamaktadır.

Short değişkenlere -32,768 ile + 32,767 arasında değişen sayısal değerler aktarılabilirken, ***Byte*** değişkenlere 0 ile 255 arasında değişen değer aktarılabilir. ***Integer*** değişkenlere -2,147,483,648 ile + 2,147,483,647, ***Long*** tipindeki değişkenlere ise -9,223,372,036,854,775,808 ile +9,223,372,036,854,775,807 arasında değişen sayısal bilgiler aktarılabilir.

```
Dim Sayi1 As Byte
Dim Sayi2 As Short
Dim Sayi3 As Integer
Dim Sayi4 As Long

Sayi1 = 255
Sayi2 = 32767
Sayi3 = 2147483647
Sayi4 = 9223372036854775807
```

Single ve Double Veri Tipi

Ondalık nokta içeren sayısal bilgileri saklamak için ***Single*** yada ***Double*** veri tiplerinden yararlanarak değişken tanımlayabiliriz. Ondalık nokta içeren sayısal bilgi çok büyük ise ***Double,*** küçük ise ***Single*** veri tipi kullanılır. Her ***Single*** değişken belekte 4 byte yer kaplarken, ***Double*** değişkenler 8 byte yer işgal eder.

```
Dim x1 As Single
Dim x2 As Double

x1 = 12345.5678
x2 = 876.123456789
```

Date Veri Tipi

Tarihsel bilgileri bellekte saklamak için ***Date*** veri tipi ile değişken tanımlayabilirsiniz. ***Date*** tipindeki değişkenler bellekte 8 byte yer kaplamakta ve 1 Ocak 0001 ile 31 Ocak 9999 arasında kalan zaman bilgilerini saklayabilir. ***Date*** tipindeki değişkenlere, bir string gibi " " işaretleri arasında yada # # işaretleri arasında tarih bilgisi aktarılabilir.

```
Dim tarih As Date
Dim zaman As Date

tarih = "05.07.2010"
zaman = #2:12:00 AM#
```

Object Veri Tipi

Integer tipindeki değişkene kesirli bir sayı aktarmak mümkün olmadığı için, bu şekildeki bir proje çalıştırıldığında hata meydana gelecektir. Ancak ***Object*** veri tipi ile tanımlanan değişkenlere herhangi tipteki bilgiler aktarılabilir. Bu şekilde kullanıldığında hata meydana gelmeyecektir. ***Object*** tipteki

değişkene aktarılmış bilginin tipi öğrenmek istiyorsanız ***VarType()*** fonksiyonundan yararlanabilirsiniz. ***Object*** değişkenlerin tipi çalışma anında belirlendiği için performans düşmekte ve önerilmemektedir.

```
Dim okul As Object

okul = 123.456
okul = "Sakarya Üniversitesi"
```

Tip Dönüşümleri

Her farklı veri tipindeki değişkenin verileri saklayış biçimleri farklıdır. Bellekte farklı şekilde dizilirler ve okunmaları ile yazılmaları farklılaşabilir.

Uygulamalarda değişkenler baştan belirli bir veri tipiyle tanımlansalar da bazen çalışma esnasında, bir veri tipinden diğerine dönüştürülmeye ihtiyaç duyabilir. Visual Basic.NET bu aktarma işlemi sırasında kullanabileceğiniz çok sayıda hazır fonksiyona sahiptir. Farklı tipteki değişkenlerin içeriklerini aktarırken, tip dönüşüm fonksiyonlarından yararlanabiliriz. Aşağıdaki tabloda bu tip dönüşüm fonksiyonları listelenmiştir.

Genel yazılış şekli :

Dönüşen_veri_tipi = **Tip_Dönüşüm_Fonksiyonu(**Dönüştürülecek veri tipi**)**

Fonksiyon İsmi	Tip Dönüştürücü	Açıklama
CBool	Boolean	String veya sayısal ifadeleri Boolean türüne dönüştürür. Sayısal bilgi sıfırdan farklı ise True, sıfıra eşitse False değeri döndürür. Dönüştürülecek string ifade yalnızca True ya da False olabilir.
CByte	Byte	Sayısal içerikli bir değeri Byte veri türüne dönüştürür. Sayının ondalık kısmı (varsa) tam sayıya yuvarlar.
CChar	Char	String ifadenin ilk karakterinin alarak Char veri türüne dönüştürür.
CDate	Date	Sayısal içerikli bir değeri Date veri türüne dönüştürür.
CDbl	Double	Sayısal içerikli bir değeri Double veri tipine dönüştürür.
CDec	Decimal	Sayısal içerikli bir değeri Decimal veri türüne dönüştürür.
CInt	Integer	Sayısal içerikli bir değeri Integer veri türüne dönüştürür. Sayının ondalık kısmı (varsa) tam sayıya yuvarlar.
CLng	Long	Sayısal içerikli bir değeri Long veri türüne dönüştürür. Sayının ondalık kısmı (varsa) tam sayıya yuvarlar.
CShort	Short	Sayısal içerikli bir değeri Short veri türüne dönüştürür. Sayının ondalık kısmı (varsa) tam sayıya yuvarlar.
CSng	Single	Sayısal içerikli bir değeri Single türüne dönüştürür.
CStr	String	Bir ifadeyi string türüne dönüştürür.

```
Dim sayi1 As Integer
Dim sayi2 As Single = 2011


sayi1 = CInt(sayi2)
```

Tip dönüşümü için ayrıca ***Convert*** sınıfının metotları da kullanılabilir.

integer e = Convert.ToInt32(d)

şeklinde de yazabiliriz.

Dönüşüm metodu	Açıklama
Convert.ToBoolean (ifade)	Sayısal ifadeyi **boolean** tipine dönüştürür.
Convert.ToChar (ifade)	Tek karakteri **char** tipine dönüştürür.
Convert.DateTime (ifade)	Geçeri bir tarih veya zamanı **DateTime** tipine dönüştürür.
Convert.ToSingle (ifade)	Bir ifadeyi **single** tipine dönüştürür.
Convert.ToDouble (ifade)	İfadeyi **double** tipine dönüştürür.
Convert.ToInt16 (ifade)	Bir ifadeyi **short** tipine dönüştürür.
Convert.ToInt32 (ifade)	Bir ifadeyi **integer** tipine dönüştürür.

Değişkenin Kapsamı

Değişkenin kapsamı, değişkenin erişebildiği kod bölgesi anlamına gelir. Bir değişkenin kapsamı değişkenin nerede bildirildiğine bağlıdır :

- **Alt Program Düzeyi :** Local (Yerel) değişkenler olarak da bilinen bu değişkenler yalnızca bildirildikleri alt programın (olay programı) içinde geçerli olurlar.

- **Blok Düzeyi :** If…Then, Do … Loop veya For … Next bloğu içinde bildirilmiş değişkenler, kapsamı bildirildikleri blokla sınırlıdırlar.

- **Modül Düzeyi :** Modüllerde, sınıflarda ya da yapılarda bir alt (olay) programın dışında bildirilmiş değişkenlerin kapsamı, kullanılan sözcüğe karşı farklılık gösterir.

- **Public** sözcüğü ile bildirilen değişkenlere bildirildikleri modül ya da sınıf içindeki her yerden ve içinde bildirildiği uygulamaya başvuran diğer uygulamalar tarafından erişilebilir.
- **Friend** sözcüğü ile bildirilen değişkenlere bildirildikleri modül ya da sınıf içindeki her yerden erişilebilir.
- **Protected** sözcüğü ile bildirilen yalnızca bildirildikleri sınıf içinde ya da o sınıfı kalıtımla alan bir sınıf içinde erişilebilir.
- **Private** sözcüğü ile bildirilen yalnızca bildirildikleri modül, sınıf ya da yapı içinden erişilebilir.

Sabit Tanımlama

Programcılar bazen tanımladıkları değişkenin içeriğinin değiştirilmesini istemezler. Bu gibi durumlarda değişken tanımlamak yerine ***Const*** deyimi ile sabit tanımlayabilirsiniz. Sabitleri tanımlarken veri tipini belirtmenize gerek yoktur. Tanımladığınız sabitin içeriğini sonradan değiştirmek istemeniz halinde hata meydana gelir.

```
Const okul = "Adamyo"
Const pi = 3.14
Const tarih = "12/05/2011"
```

Diziler

Bazen bellekte bilgi saklamak için birden fazla yere gerek duyulur. Örneğin aynı sınıftaki öğrencilerin listesi gibi. Aynı özellik ve benzer işleve sahip değişkenleri tek tek tanımlamak pratik değildir. Bu ve benzeri durumlarda aynı tipteki değişkenleri ayrı ayrı tanımlamak yerine dizi değişkenlerinden yararlanılır. Aynı özelliğe sahip elemanların bir araya getirdiği gruba dizi denir. Diziler tek boyutlu, iki veya üç boyutlu olabilir (Matrislerde olduğu gibi). Diziler belirli kurallara uymak suretiyle tanımlanabilir. Visual Basic.NET' te dizilerin ilk elemanı 0. Eleman olarak kabul edilir. Tek boyutlu diziler, dizi ismi ve ardından dizinin eleman sayısı parantez içinde tanımlanır. Örneğin 50 elemanlı bir dizi X dizisi tanımlayacaksak, X(49) şeklinde tanımlamalıyız.

```
Dim liste(4) As Object


liste(0) = "Ahmet"
liste(1) = 1.8
liste(2) = 28
liste(3) = "Sakarya Üniversitesi"
liste(4) = "01.01.2011"
```

Aynı diziyi 3 sayısını belirtmeden de tanımlayabiliriz. Kendisi otomatik olarak değerlerden yararlanarak dizi boyutunu 3 olarak alır.

```
Dim aylar() As String = {"Ocak", "Şubat", "Mart", "Nisan"}
```

Structure (Yapı) Tanımlamak

Özellikle random dosyalara bilgi kaydedilirken değişik tipte birden fazla değişken bir araya getirilerek Structure (yapı) tanımlanır.

```
Structure ogrenci
    Dim ad As String
    Dim soyad As String
    Dim yas As Byte
    Dim sehir As String
End Structure
```

Yapıda yer alan değişkenler istenen tipte olabilir. Bu şekilde tanımlanan yapıyı kullanabilmek için yapıdan yararlanarak başka bir değişkenin tanımlanması gerekir.

Bu şekilde tanımlanan yapının elemanlarına istenildiği gibi bilgi aktarılabilir. Yapının adı ile elemanını birbirinden ayırmak için nokta(.) karakteri kullanılır.

```
Dim kayit As ogrenci

kayit.ad = "Ahmet"
kayit.soyad = "Yıldız"
kayit.yas = 19
kayit.sehir = "Ankara"
```

Enum İle Sıralı Sabitler Tanımlamak

Enum, sabitlerden meydana gelen sıralı tipleri tanımlamak için kullanılır.

```
Public Enum aylar
    ocak
    şubat
    mart
    nisan
    mayıs
End Enum
```

İfadeler

Her türlü sabit, değişken ve fonksiyonlardan meydana gelen ve program satırında eşitliğin sağ tarafında yer alan kısma ifade adı verilir. İfadelerin sol tarafında bir değişken ve onu izleyen atama (=) operatörü yer alır. Sağ taraftaki ifadede elde edilen değer, sol taraftaki değişkene (veya özelliğe) aktarılır.

Örnek – 1

```
Hiz = Yol / Zaman
Delta = B ^2 – 4*A*C
```

Örnek – 2

```
A = B * C : D = P * R ^ 2 : V = X * Y * Z
```

Dikkat

Normal olarak bir satırda bir eşitlik (ifade) yer alır. Birden fazla ifadeyi bir satırda yazmak için, ifadeler arasına "**:**" işareti konulmalıdır.

Örnek – 3

```
MATC = (X ^ 2) + (Y ^ 2) + Z * (D – (CX + CY) _
+ K ^ 3 – 8 * (ERX + ERY) ^ 3
```

Dikkat

İfade bir satırda bitmeyecek kadar uzunsa, bir sonraki satıra geçmeden önce alt çizgi "_" karakteri kullanılır.

Sayısal İfadeler : Çarpma, bölme, üs alma , toplama gibi aritmetik işlemlerin olduğu ifadelerdir. Bu işlemler aşağıdaki gibi sıralanabilir :

İşlem	Operatör	Örnek	Sayısal Örnek	Sonuç
Toplama	+	t + ftr	23 + 12	35
Çıkarma	-	R – K	23 – 12	11
Bölme	/	YOL / ZAMAN	5 / 2	2.5
Çarpma	*	X1 * X2	3 * 4	12
Tamsayı Bölme	\	F \ P	5 \ 2	2
Modüler Aritmetik	MOD	A MOD B	8 MOD 3	2
Üs Alma	^	X ^ 3	4 ^ 3	64

Matematiksel işlemlerin çalışmaları sırasında öncelikleri de önemlidir. Bir matematiksel işlem içerisinde, birden fazla işlem olduğunda, bilgisayarın hangi işlemi öncelikle çalıştıracağını bilmek önemlidir. Bu öncelik derecesi aşağıdaki gibi sıralanmaktadır ;

1. Parantez içi
2. Üs alma
3. Çarpma veya Bölme
4. Tamsayı Bölme
5. Modüler Aritmetik
6. Toplama veya çıkarma

Örnek

(-3 * (40 \ 3 * 1) + 41) ^ 2 = ?

(-3 * (40 \ 3) + 41) ^ 2

(-3 * 13 + 41) ^ 2

(-39 + 41) ^ 2

2 ^ 2

4

Visual Basic.NET' le sayısal ifadeler için gelen bir özelliği örnek ile açıklayalım :

X = X + 3 ifadesi, X değişkeninin önceki değeri ile 3 rakamının toplanarak X değişkeninin yeni değeri olarak atanması anlamına gelmektedir (Örneğin, X' in önceki değeri 5 ise, X' in yeni değeri X = 5 + 3 = 8 olacaktır). Bu ifade şu şekilde de yazılabilir : X += 3

Örnek

A \= C	→	A = A \ C
X /= 2	→	X = X / 2
SAYI += 5	→	SAYI = SAYI + 5
SONUC -= 600	→	SONUC = SONUC – 600

Karakter İfadeler : Karakter değişkenlerin sahip olduğu ifadelerdir. Karakter ifadeler üzerinde çarpma, bölme gibi aritmetik işlemler yapılamaz. Ancak karakter ifadelerin birbirine eklenmesini sağlayan toplama işlemi yapılabilir. Bu işlem için "**+**" veya "**&**" karakteri kullanılır.

Örnek

```
X = "SAKARYA"
Y = "ÜNİVERSİTESİ"
Z = X + " " + Y
M = X & Y
```

ise

```
Z = "SAKARYA ÜNİVERSİTESİ"
```

M = "SAKARYAÜNİVERSİTESİ" olacaktır.

İlişki İfadeleri : İki değeri karşılaştırmak amacıyla kullanılan ifadelerdir. Karşılaştırılan değerler sayısal veya karakter şeklinde olabilir. Aşağıdaki tabloda bu ifadeler örnekleriyle gösterilmiştir.

İfade	Anlamı	Örnek
<	Küçük	A < B
<=	Küçük Eşit	C <= 87
>	Büyük	X^2 > Y * 56
>=	Büyük Eşit	T1 >= 3 * T2
=	Eşit	G = R2
<>	Eşit Değil	YT <> 4.8

Mantıksal İfadeler : Bu ifadeler iki veya daha fazla ifade arasında kullanılırlar. Mantıksal operatörlerle iki veya daha fazla ilişkiyi birbirine bağlarlar, sonunda Doğru ya da Yanlış değerini hesaplarlar. Bunlardan en önemlileri NOT, AND ve OR 'dur. Aşağıdaki tabloda bu operatörlerin doğruluk değerleri gösterilmiştir. Visual Basic 'te Doğru ifadesi olarak True kelimesi, Yanlış ifadesi olarak False kelimesi kullanılır. (**D**: Doğru, **Y**: Yanlış)

A İfadesi	B İfadesi	NOT A	A AND B	A OR B
D	D	Y	D	D
D	Y	Y	Y	D
Y	D	D	Y	D
Y	Y	D	Y	Y

Bunun dışında 3 mantıksal ifade **ANDALSO**, **ORELSE** ve **XOR** daha vardır.

AndAlso ifadesinde 1. sorgunun yanlış olması durumunda, 2. sorguya bakılmadan ifadenin yanlış olduğuna karar verilir. Örneğin ; **A AndAlso B . A** ifadesi yanlışsa **B**'ye bakmaya gerek yoktur. Sonuç **Yanlış**'tır.

Karar Yapıları

If... Then ...Else

Visual Studio.NET' te karar yapıları oluşturmak için ***IF*** deyimi kullanılır. ***IF*** deyimi, programın akışını verilen koşula bağlı olarak oluşturulan program bloğuna geçilmesini veya söz konusu program bloğunun işletilmeyip atlatılmasını sağlar. Komutların yerine getirilmesi, belli ifadelerin doğru ya da yanlış olmasına bağlı olduğu durumlarda kullanılan bir deyimdir.

5 farklı formu vardır :

IF <İfade> THEN <İşlem>	Eğer **<İfade>** doğru ise THEN'i izleyen deyim işlem görür. Aksi halde programın akışı IF deyimini izleyen satıra geçer.

IF <İfade> THEN
  <İşlem>
END IF
	Eğer **<İfade>** doğru ise THEN'i izleyen deyim işlem görür. Aksi halde programın akışı END IF deyimini izleyen satıra geçer. Bu formun bir önceki formdan farkı, ifadenin doğru olması durumunda birden fazla satır işlem görebilmektedir.

IF <İfade> THEN <İşlem> ELSE <İşlem>	Eğer **<İfade>** doğru ise THEN'i izleyen deyim işlem görür. Aksi halde ELSE deyimini izleyen deyim icra edilecektir.

IF <İfade> THEN
  <İşlemler>
ELSE
  <İşlemler>
END IF
	Eğer **<İfade>** doğru ise THEN'i izleyen deyimler işlem görür. <İfade> yanlış ise ELSE deyimini izleyen komutlar çalıştırılacaktır. Bu formun bir önceki formdan farkı, ifadenin doğru ya da yanlış olması durumunda birden fazla satır işlem görebilmektedir.

IF <İfade-1> THEN
  <İşlemler>
ELSEIF <İfade-2> THEN
  <İşlemler>
ELSEIF <İfade-3> THEN
  <İşlemler>
  .....
  .....
ELSE
  <İşlemler>
END IF
	Eğer **<İfade-1>** doğru ise 1. THEN deyimini izleyen deyimler bloğu işlem görecek, eğer yanlış ise bu kez ELSEIF deyimini izleyen **<İfade-2>**'nin doğruluğu araştırılacaktır. Eğer doğru ise 2. THEN deyimini izleyen deyimler işlem görecektir. İşlemler bu şekilde devam edecektir. Bütün koşulların yanlış olması durumunda ELSE deyimini izleyen deyimler çalıştırılacaktır

Örnek : **Öğrencinin not ortalamasını kontrol eden kod satırı.**

```
.....
If ortalama < 1.6 Then MsgBox("Öğrenci üst dönemden ders alamaz.")
```

veya

```
.....
If ortalama < 1.6 Then
        MsgBox("Öğrenci üst dönemden ders alamaz.")
End If
```

Örnek : **Öğrencinin not durumuna göre "KALDI" yada "GEÇTİ" mesajı veren kod satırları.**

```
.....
If ort < 50 Then
   MsgBox("KALDI")
Else
   MsgBox("GEÇTİ")
End If
```

Örnek : **Okuldaki Bölümlerin isimlerine göre bir "kod" değeri oluşturan satırlar.**

```
.....
If bolum = "BİLGİSAYAR PROGRAMCILIĞI" Then
     kod = "32"
ElseIf bolum = "MEKATRONİK" Then
     kod = "36"
ElseIf bolum = "ELEKTRONİK TEKNOLOJİSİ" Then
     kod = "38"
Else
     kod = "40"
End If
```

Select Case – End Select

SELECT CASE –END SELECT deyimi işlev bakımından ***IF...ELSE IF...*** deyimine benzemektedir. Özellikle çok sayıda iç içe ***IF...ELSE IF...*** blokları kullanıldığı zaman programın okunurluğu azalır ve programı işlemek zorlaşır. Bu gibi durumlarda ***SELECT CASE-END SELECT*** deyimini kullanabiliriz.

```
Select Case Kontrol Değişkeni
   Case <ifade 1>
        ........
   Case <ifade 2>
        ........
   Case Else
        ........
End Select
```

Blok ***Select Case*** ile başlar ve ***End Select*** ile biter. ***Select Case*** deyiminden sonra yapılacak karşılaştırmalarda kullanılacak bir ***kontrol değişkeni*** bulunmaktadır. ***Kontrol değişkeninin*** içeriği ***ifade 1*** olarak verilen değerle aynı ise ***ifade 1***' in bulunduğu ***Case*** satırından sonraki satırlar, bir sonraki ***Case*** deyimine kadar olan satırlar işlenir ve programın akışı ***End Select*** deyimini izleyen satıra geçer.

Örnek : **Okunan Bölüm kodu (kod) için Bölümün adını (bolum) belirleyen satırlar.**

```
Select Case kod
    Case "32"
        bolum = "BİLGİSAYAR PROGRAMCILIĞI"
    Case "36"
        bolum = "MEKATRONİK"
    Case "38"
        bolum = "ELEKTRONİK TEKNOLOJİSİ"
    Case Else
        bolum = "İNTERNET VE AĞ TEKNOLOJİLERİ"
End Select
```