İÇİNDEKİLER

1. VB 9.00 İLE PROGRAM YAZIMI

2. ALGORİTMA

Bir problemin çözümü için takip edilecek yolun belirlenmesine algoritma denir. Bir problemin çözümünde değişik kişiler değişik yollar izleyebilirler. Önemli olan gidilen yolun doğru ve istenilen sonuca ulaştırıcı olmasıdır. Bu da değişik kişilerin değişik algoritma hazırlayacağı anlamına gelir. Kısaca günlük yaşantımızda yaptığımız, yapacağımız bütün işlerin ister istemez kafamızda algoritmasını yaparız. Burada bir problemin çözümü için adım adım takip edilecek yolu belirlemeye ve hazırlanan bir çeşit taslağa gereksinmemiz olacaktır. Bu da algoritma diye tanımlanır.

2.1. Algoritmada Uyulması Gereken Kurallar

1. Basla komutu ile başlanmalıdır.
2. Verilen komutlar kısa, öz ve emir cümlecikleri şeklinde olmalıdır.
3. Her satıra bir satır numarası verilmelidir. Satır numaraları küçükten büyüğe doğru gitmelidir.
4. Kullanılan değişken adları kısa, net ve tanınabilir olmalıdır.
5. Algoritmada bazı satırların işlem görmeden atlanmasını ve işleme diğer satırdan başlamasını istiyorsak GİT komutu ile gideceği satır numarasına gönderilir ve işlem akışı o satırdan itibaren sürer.
6. Programın sonuna bittiğini belirtmek için DUR komutu verilmelidir.

2.2. ALGORİTMA ÖRNEKLERİ

2.2.1. Klavyeden döviz miktarı ve döviz kuru girildiğinde TL' ı olarak miktarını hesaplayan programın algoritması.

1.Adım: Başla
2.Adım: Döviz Miktarını Gir ⟶ DM
3.Adım: Günlük Kuru Gir ⟶ GK
4.Adım: TL ını hesapla ⟶ TL=DM*GK
5.Adım: Sonucu ekrana yaz ⟶ TL
6.Adım: Dur

2.2.2. Yıl içinde yapılan üç yazılı sınavın ortalamasını bulan programın algoritması.

1.Adım: Başla
2.Adım: 1. Yazılıyı Gir ⟶ Y1
3.Adım: 2. Yazılıyı Gir ⟶ Y2
4.Adım: 3. Yazılıyı Gir ⟶ Y3
5.Adım: Ortalamayı Hesapla ⟶ ORT=(Y1+Y2+Y3)/3
6.Adım: Ortalamayı Yaz ⟶ ORT
7.Adım: Dur

2.2.3. Faiz Hesabının Algoritması

1.Adım: Başla
2.Adım: Kapitali Gir ⟶ K
3.Adım: Süreyi Gir ⟶ N
4.Adım: Faiz fiyatını Gir ⟶ T
5.Adım: Faizi Hesapla ⟶ F=(K*N*T)/1200
6.Adım: Faizi Yaz ⟶ F
7.Adım: Dur

2.2.4. Yamuğun alanını hesaplayan programın algoritması.

1.Adım: Başla
2.Adım: Kısa Kenarı Gir ⟶ C
3.Adım: Uzun Kenarı Gir ⟶ A
4.Adım: Yüksekliği Gir ⟶ H
5.Adım: Alanı Hesapla ⟶ ALAN=(A+C)*H/2
6.Adım: Alanı Yaz ⟶ ALAN
7.Adım: Dur

2.2.5. Klavyeden M girilence Merhaba G girilince Günaydın, başka bir harf girilince de yanlış giriş yaptınız yazıp başa dönen programın algoritması.

1.Adım: Başla
2.Adım: Bir Harf Gir ⟶ HARF
3.Adım: Eğer HARF=M ise 6.satıra git
4.Adım: Eğer HARF=G ise 7.satıra git
5.Adım: YANLIŞ GİRİŞ YAPTINIZ yaz. Başa Dön
6.Adım: MERHABA yaz. Dur.
7.Adım: GÜNAYDIN yaz. Dur.

2.2.6. İkinci dereceden denklemin köklerini bulan programın algoritması.

1.Adım: Başla
2.Adım: a, b, c katsayılarını gir ⟶ A, B, C
3.Adım: Diskriminantı hesapla ⟶ D=B^2-4*A*C
4.Adım: Eğer D>0 ise iki kök vardır bunları hesapla ⟶ X1= (-b+D^(1/2))/2*a,
X2= (-b-D^(1/2))/2*a (sonucları yaz, dur)
5.Adım: Eğer D=0 ise tek kök vardır bunu hesapla ⟶ X= -b/2*a (sonucu yaz, dur)
6.Adım: Eğer D<0 ise reel kök yoktur yaz, dur

2.2.7. Bir iş başvurusunda işe alınacak adaylar için erkek olması, yaşının 30'dan küçük olması boyunun da en az 1.65 olması şartı gereklidir. Koşulu sağlayan adaylar için ekrana işe alındı yazan aksi durumda işe alınmadı yazan programın algoritması.

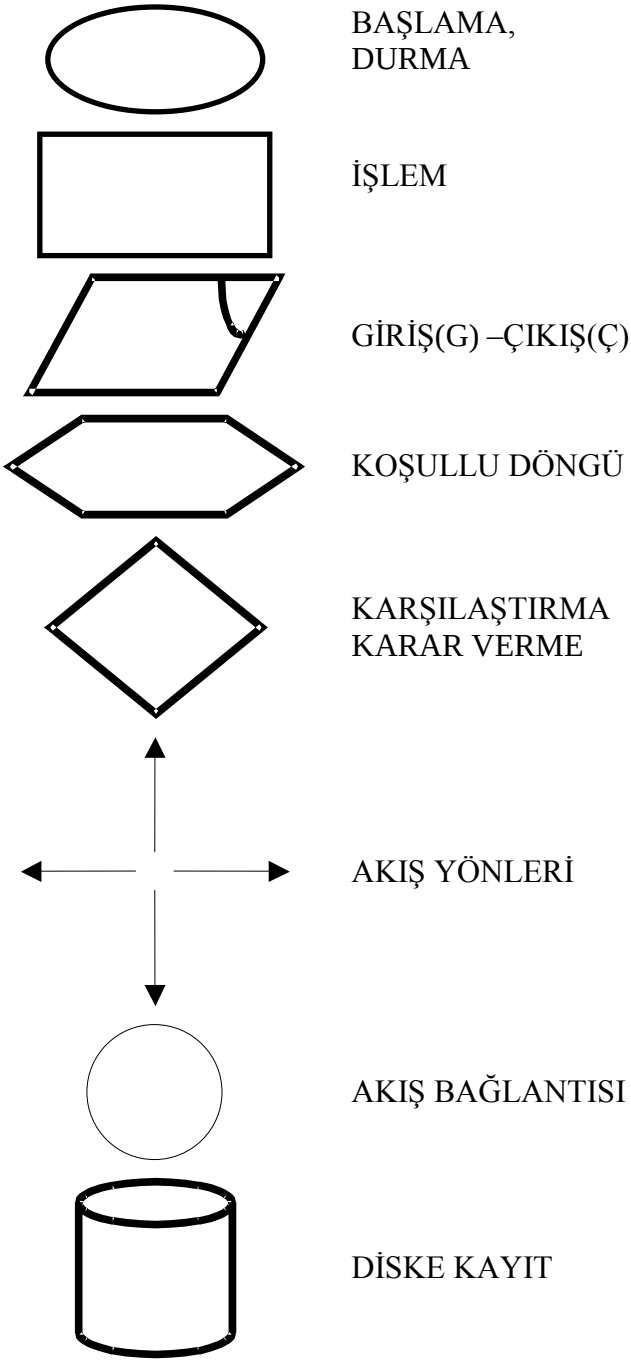
1.Adım: Başla
2.Adım: Klavyeden yaş, boy, cinsiyet, gir ⟶ Y, B, C
3.Adım: Eğer Y<30 ise 4. adıma değilse 7. adıma git
4.Adım: Eğer B>=1.65 ise 5. adıma değilse 7. adıma git
5.Adım: Eğer C=erkek ise 6. adıma değilse 7. adıma git
6.Adım: işe alındı yaz, dur
7.Adım: işe alınmadı yaz, dur

2.2.8. İki sayıyı karşılaştırarak en büyük olan sayıyı büyüktür diye yazdıran algoritma.

1.Adım: Başla
2.Adım: İki Sayı Gir ⟶ A, B
3.Adım: Eğer A>B ise A büyüktür yaz, dur.
4.Adım: Eğer B>A ise B büyüktür yaz, dur.
5.Adım: Eğer A=B ise sayılar eşittir yaz, dur.

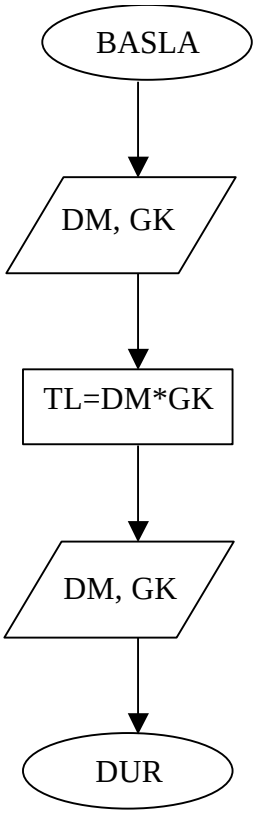
2.3. AKIŞ ŞEMALARI (DİYAGRAMLARI)

Problem çözümünde izlenecek yolun belirlenmesi için yapılacak algoritmanın şekillerle gösterilmesine akış şeması denir.
Akış şemalarını çizmek için aşağıdaki şekil ve semboller kullanılır:

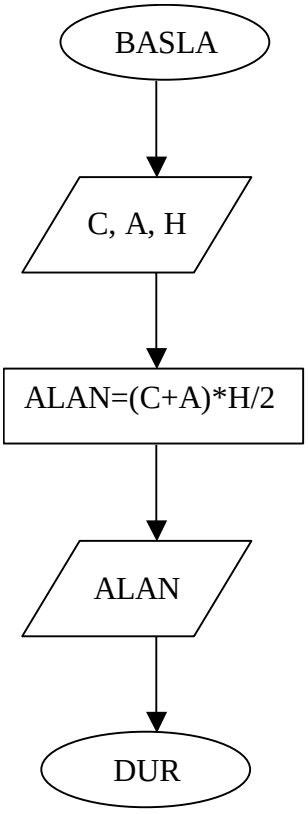


2.4. Akış Şeması Örnekleri:

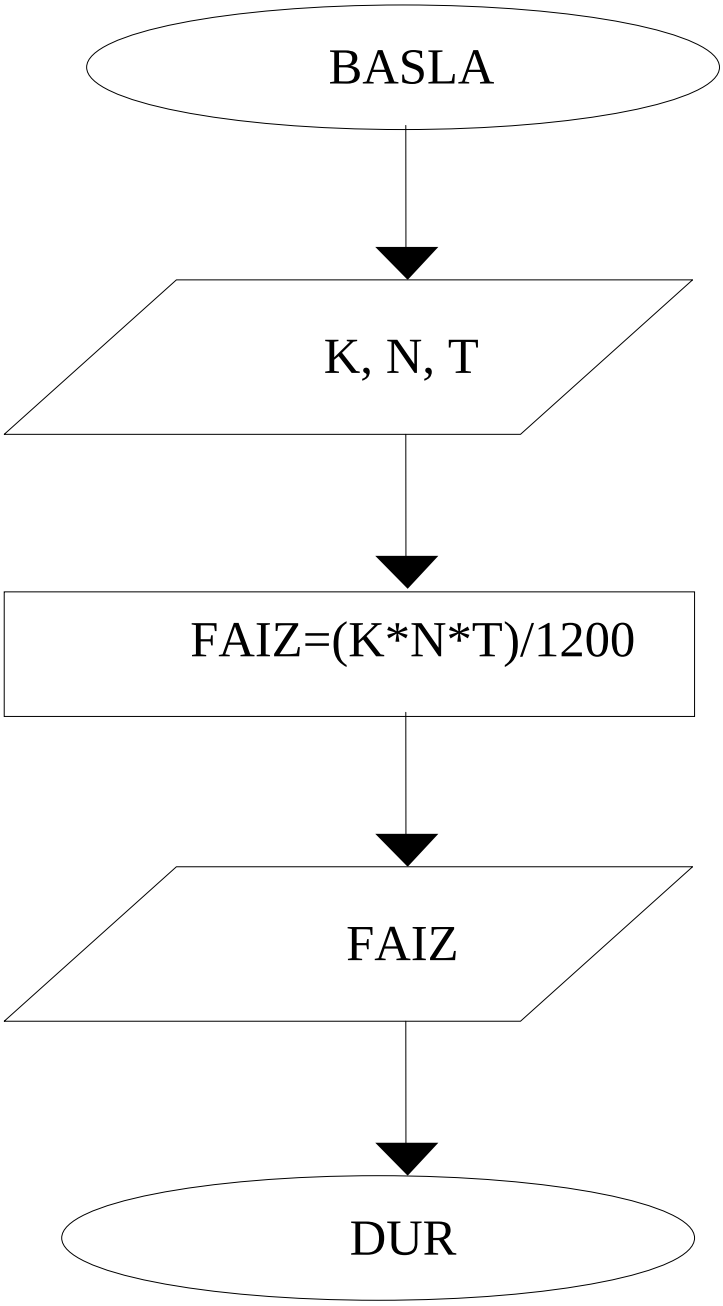
2.4.1. Klavyeden Döviz Miktarı ve Günlük Kur girildiğinde cebimizdeki dövizi TL' ye çeviren programın akış şeması:



2.4.2. Yamuğun alanını hesaplayan programın akış şeması

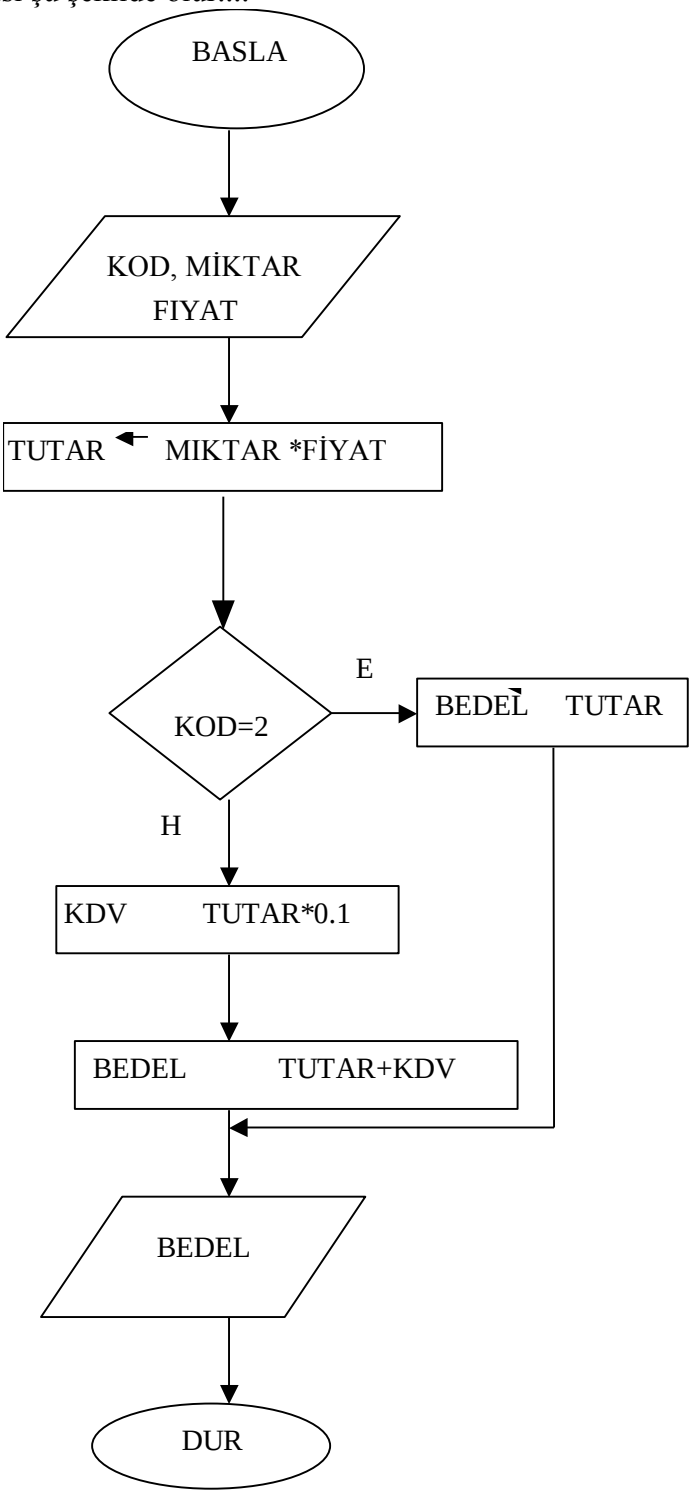


2.4.3. Faiz hesabının akış şeması:

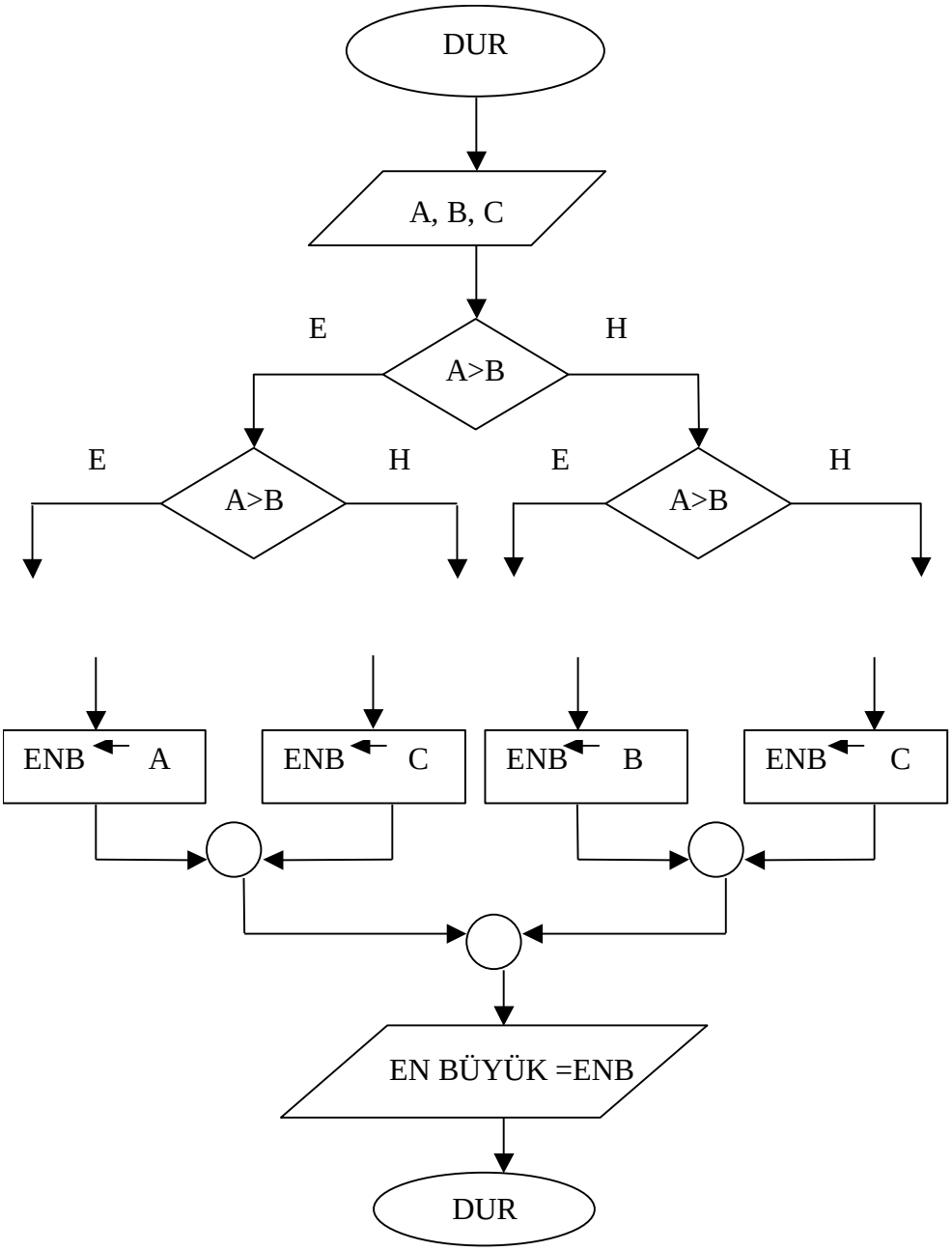


2.4.4. Bir işletmenin stok kodları (KOD) 1, 2 ve 3 olan üç tür mal sattığını kabul edelim. Bir ve üç kod numaralı malların KDV si olup iki kod numaralı malın KDV si bulunmasın. Kod, miktar ve fiyat verildiğinde toplam satış bedelinin bulunuşuna ilişkin bir akış şeması hazırlayalım.

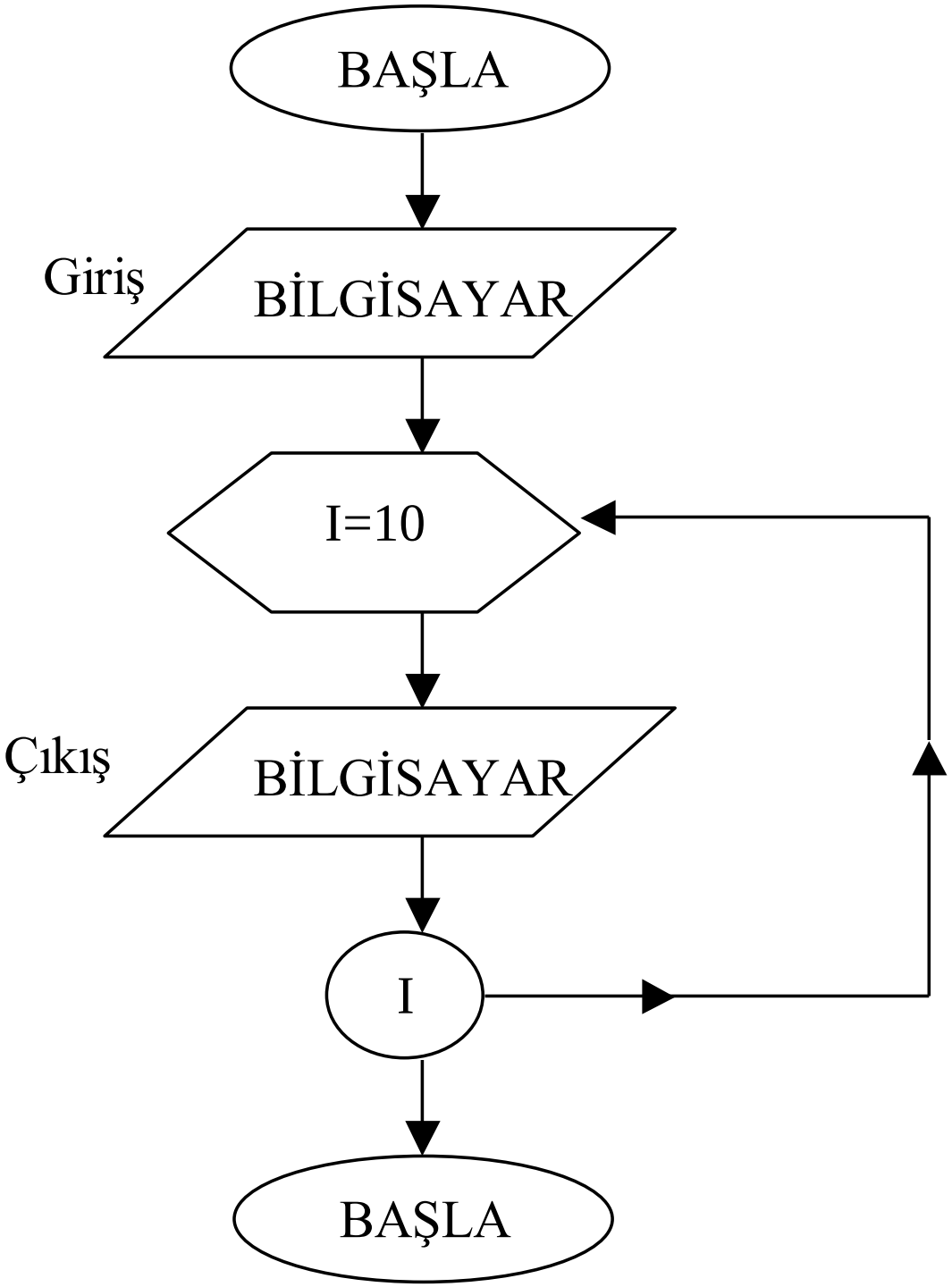
Örnek problemin giriş bilgileri malın kodu (KOD), satış miktarı (MIKTAR) ve malın fiyatı (FIYAT) tır. Çıkış bilgisi ise satış bedeli (BEDEL) olacaktır. BEDEL, MIKTAR, ve FIYAT çarpımının sonucuna, KOD değerine bağlı olarak gerekiyorsa %10 KDV eklenerek elde edilecektir. Giriş bilgilerinin klavyeden alınacağını çıkış bilgisinin ise ekrana verileceğini kabul ettiğimizde akış şeması şu şekilde olur....



2.4.5. A, B ve C'nin değerleri verildiğinde bunlardan en büyüğünü bulan programın akış şeması



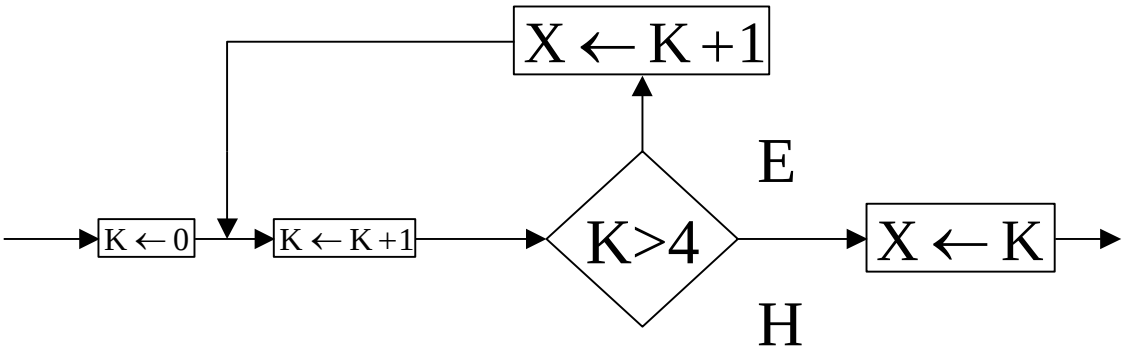
2.4.6. Ekrana 10 kez bilgisayar yazdıran programın akış şemasını hazırlayınız.



ÖDEV: 1. İki sayının (A ve B) çarpımı sonucu ortaya çıkan değerin işaretinin bulunmasına ilişkin karar tablosu aşağıda verilmiştir.

Çarpım İşareti Tablosu	Kural 1	Kural 2	Kural 3	Kural 4
Birinci sayı pozitif (A)	E	E	H	H
İkinci sayı pozitif (B)	E	H	E	H
Sonuç pozitif	X			X
Sonuç negatif		X	X	

2. Aşağıdaki akış şeması kesiminin çıkışında X'in değeri ne olacaktır?



3. Bir çalışanın ödeyeceği vergi, çalışanın yaşına göre belirlenmektedir. Bu oran 18 yaşından küçükler için %0, diğerleri %25 dir. Çalışanın ödeyeceği vergi miktarını hesaplayan programın algoritmasını ve akış şemasını hazırlayınız.

4. Bir işçinin haftalık ücreti normal olarak 40 saat üzerinden verilmektedir. Saat başına 2 birim ödeme yapılacaktır. 40 saatten fazla çalışılırsa saat ücreti 3 birim olacaktır. İşçinin haftalık çalışma saatini giren ve alacağı ücreti hesaplayan programın akış şemasını hazırlayınız.

5. Verilen Fahrenheit dereceyi (F) , Celcius derecesine (C) çeviren bir akış şeması hazırlayınız.

6. A, B ve C gibi üç sayı değerlerinden en küçüğünü bulan ve yazdıran bir programın akış şemasını hazırlayınız.

3. Variable Data Types

Veri türü	Kapladığı alan (Bayt)	Açıklama	
Bayt	1	8-bit işaretsiz tam sayı	0,1,2….255 (2^8)
Integer	4	32-bit işaretli tam sayı	32767(±2^15)
Long	8	64-bit işaretli tam sayı	± 2 147 483 647 (±2^31)
Double	8	64-bit ondalıklı sayı	A double-precision floating-point value with a range of – 1.79769313486232E308 to – 4.94065645841247E-324 for negative values, 4.94065645841247E-324 to 1.79769313486232E308 for positive values, and 0.
Single	4	32-bit ondalıklı sayı	A single-precision floating-point value with a range of – 3.402823E38 to – 1.401298E-45 for negative values, 1.401298E-45 to 3.402823E38 for positive values, and 0.
Date	8	Tarih bilgisi	
Boolean	2	Nümerik olmayan tip	TRUE/FALSE

4. Arithmetic operators

4.1. ^ Operator Example

This example uses the ^ operator to raise a number to the power of an exponent.

```
Dim MyValue
MyValue = 2 ^ 2   ' Returns 4.
MyValue = 3 ^ 3 ^ 3   ' Returns 19683.
MyValue = (-5) ^ 3   ' Returns -125.
```

4.2. * Operator Example

This example uses the ***** operator to multiply two numbers.

```
Dim MyValue
MyValue = 2 * 2   ' Returns 4.
MyValue = 459.35 * 334.90    ' Returns 153836.315.
```

4.3. / Operator Example

This example uses the / operator to perform floating-point division.

```
Dim MyValue
MyValue = 10 / 4   ' Returns 2.5.
```

```
MyValue = 10 / 3   ' Returns 3.333333.
```

4.4. \ Operator Example

This example uses the \ operator to perform integer division.

```
Dim MyValue
MyValue = 11 \ 4   ' Returns 2.
MyValue = 9 \ 3   ' Returns 3.
MyValue = 100 \ 3   ' Returns 33.
```

4.5. Mod Operator Example

This example uses the **Mod** operator to divide two numbers and return only the remainder. If either number is a floating-point number, it is first rounded to an integer.

```
Dim MyResult
MyResult = 10 Mod 5   ' Returns 0.
MyResult = 10 Mod 3   ' Returns 1.
MyResult = 12 Mod 4.3   ' Returns 0.
MyResult = 12.6 Mod 5   ' Returns 3.
```

4.6. + Operator Example

This example uses the **+** operator to sum numbers. The **+** operator can also be used to concatenate strings. However, to eliminate ambiguity, you should use the **&** operator instead. If the components of an expression created with the **+** operator include both strings and numerics, the arithmetic result is assigned. If the components are exclusively strings, the strings are concatenated.

```
Dim MyNumber, Var1, Var2
MyNumber = 2 + 2   ' Returns 4.
MyNumber = 4257.04 + 98112   ' Returns 102369.04.

Var1 = "34": Var2 = 6   ' Initialize mixed variables.
MyNumber = Var1 + Var2   ' Returns 40.

Var1 = "34": Var2 = "6"   ' Initialize variables with strings.
MyNumber = Var1 + Var2   ' Returns "346" (string concatenation).
```

4.7. - Operator Example

This example uses the - operator to calculate the difference between two numbers.

```
Dim MyResult
MyResult = 4 - 2   ' Returns 2.
MyResult = 459.35 - 334.90   ' Returns 124.45.
```

5. Logical Operators

5.1. And Operator Example

This example uses the **And** operator to perform a logical conjunction on two expressions.

```
Dim A, B, C, D, MyCheck
A = 10: B = 8: C = 6: D = Null   ' Initialize variables.
MyCheck = A > B And B > C   ' Returns True.
MyCheck = B > A And B > C   ' Returns False.
MyCheck = A > B And B > D   ' Returns Null.
MyCheck = A And B   ' Returns 8 (bitwise comparison).
```

5.2. Or Operator Example

This example uses the **Or** operator to perform logical disjunction on two expressions.

```
Dim A, B, C, D, MyCheck
A = 10: B = 8: C = 6: D = Null   ' Initialize variables.
MyCheck = A > B Or B > C   ' Returns True.
MyCheck = B > A Or B > C   ' Returns True.
MyCheck = A > B Or B > D   ' Returns True.
MyCheck = B > D Or B > A   ' Returns Null.
MyCheck = A Or B   ' Returns 10 (bitwise comparison).
```

5.3. Not Operator Example

Used to perform logical negation on an expression. This example uses the **Not** operator to perform logical negation on an expression.

```
Dim A, B, C, D, MyCheck
A = 10: B = 8: C = 6: D = Null   ' Initialize variables.
MyCheck = Not(A > B)   ' Returns False.
MyCheck = Not(B > A)   ' Returns True.
MyCheck = Not(C > D)   ' Returns Null.
MyCheck = Not A   ' Returns -11 (bitwise comparison).
```

5.4. Eqv Operator Example

Used to perform a logical equivalence on two expressions. This example uses the **Eqv** operator to perform logical equivalence on two expressions.

```
Dim A, B, C, D, MyCheck
A = 10: B = 8: C = 6: D = Null   ' Initialize variables.
MyCheck = A > B Eqv B > C   ' Returns True.
MyCheck = B > A Eqv B > C   ' Returns False.
MyCheck = A > B Eqv B > D   ' Returns Null.
MyCheck = A Eqv B   ' Returns -3 (bitwise comparison).
```

5.5. Imp Operator Example

If bit in *expression1* is	**And bit in *expression2* is**	**The *result* is**	
0	0	1	AYNI
0	1	1	FARKLI
1	0	0	FARKLI
1	1	1	AYNI

Used to perform a logical implication on two expressions. This example uses the **Imp** operator to perform logical implication on two expressions.

```
Dim A, B, C, D, MyCheck
A = 10: B = 8: C = 6: D = Null   ' Initialize variables.
MyCheck = A > B Imp B > C   ' Returns True.
MyCheck = A > B Imp C > B   ' Returns False.
MyCheck = B > A Imp C > B   ' Returns True.
MyCheck = B > A Imp C > D   ' Returns True.
MyCheck = C > D Imp B > A   ' Returns Null.
MyCheck = B Imp A   ' Returns -1 (bitwise comparison).
```

5.6. Xor Operator Example

If expression1 is	And expression2 is	Then result is	
True	True	False	AYNI
True	False	True	FARKLI
False	True	True	FARKLI
False	False	False	AYNI

```
This example uses the Xor operator to perform logical exclusion on two
expressions.
Dim A, B, C, D, MyCheck
A = 10: B = 8: C = 6: D = Null   ' Initialize variables.
MyCheck = A > B Xor B > C   ' Returns False.
MyCheck = B > A Xor B > C   ' Returns True.
MyCheck = B > A Xor C > B   ' Returns False.
MyCheck = B > D Xor A > B   ' Returns Null.
MyCheck = A Xor B   ' Returns 2 (bitwise comparison).
```

6. Koşullu programlama

6.1. If...Then...Else Statement Example

This example shows both the block and single-line forms of the **If...Then...Else** statement. It also illustrates the use of **If TypeOf...Then...Else**.

```
Dim Number, Digits, MyString
Number = 53   ' Initialize variable.
If Number < 10 Then
   Digits = 1
ElseIf Number < 100 Then
' Condition evaluates to True so the next statement is executed.
   Digits = 2
Else
   Digits = 3
End If

' Assign a value using the single-line form of syntax.
If Digits = 1 Then MyString = "One" Else MyString = "More than one"
```

Use **If TypeOf** construct to determine whether the Control passed into a procedure is a text box.

```
Sub ControlProcessor(MyControl As Control)
   If TypeOf MyControl Is CommandButton Then
      Debug.Print "You passed in a " & TypeName(MyControl)
   ElseIf TypeOf MyControl Is CheckBox Then
      Debug.Print "You passed in a " & TypeName(MyControl)
   ElseIf TypeOf MyControl Is TextBox Then
      Debug.Print "You passed in a " & TypeName(MyControl)
   End If
End Sub
```

6.2. Choose Function Example

This example uses the **Choose** function to display a name in response to an index passed into the procedure in the `Ind` parameter.

```
Function GetChoice(Ind As Integer)
   GetChoice = Choose(Ind, "Speedy", "United", "Federal")
End Function
```

6.3. Select Case Statement Example

This example uses the **Select Case** statement to evaluate the value of a variable. The second **Case** clause contains the value of the variable being evaluated, and therefore only the statement associated with it is executed.

```
Dim Number
Number = 8   ' Initialize variable.
Select Case Number   ' Evaluate Number.
Case 1 To 5   ' Number between 1 and 5, inclusive.
   Debug.Print "Between 1 and 5"
' The following is the only Case clause that evaluates to True.
Case 6, 7, 8   ' Number between 6 and 8.
   Debug.Print "Between 6 and 8"
Case 9 To 10   ' Number is 9 or 10.
Debug.Print "Greater than 8"
Case Else   ' Other values.
   Debug.Print "Not between 1 and 10"
End Select
```

6.4. Switch Function Example

This example uses the **Switch** function to return the name of a language that matches the name of a city.

```
Function MatchUp (CityName As String)
   Matchup = Switch(CityName = "London", "English", CityName _
               = "Rome", "Italian", CityName = "Paris", "French")
End Function
```

6.5. IIf Function Example

This example uses the **IIf** function to evaluate the `TestMe` parameter of the `CheckIt` procedure and returns the word "Large" if the amount is greater than 1000; otherwise, it returns the word "Small".

```
Function CheckIt (TestMe As Integer)
   CheckIt = IIf(TestMe > 1000, "Large", "Small")
End Function
```

7. DÖNGÜLER

7.1. For...Next Statement Example

This example uses the **For...Next** statement to create a string that contains 10 instances of the numbers 0 through 9, each string separated from the other by a single space. The outer loop uses a loop counter variable that is decremented each time through the loop.

```
Dim Words, Chars, MyString
For Words = 10 To 1 Step -1    ' Set up 10 repetitions.
   For Chars = 0 To 9   ' Set up 10 repetitions.
      MyString = MyString & Chars   ' Append number to string.
   Next Chars   ' Increment counter
   MyString = MyString & " "   ' Append a space.
Next Words
```

7.2. For Each...Next Statement Example

This example uses the **For Each...Next** statement to search the **Text** property of all elements in a collection for the existence of the string "Hello". In the example, `MyObject` is a text-related object and is an element of the collection `MyCollection`. Both are generic names used for illustration purposes only.

```
Dim Found, MyObject, MyCollection
Found = False   ' Initialize variable.
For Each MyObject In MyCollection   ' Iterate through each element.
   If MyObject.Text = "Hello" Then   ' If Text equals "Hello".
      Found = True   ' Set Found to True.
      Exit For   ' Exit loop.
   End If
Next
```

7.3. Do...Loop Statement Example

This example shows how **Do...Loop** statements can be used. The inner **Do...Loop** statement loops 10 times, sets the value of the flag to **False**, and exits prematurely using the **Exit Do** statement. The outer loop exits immediately upon checking the value of the flag.

```
Dim Check, Counter
Check = True: Counter = 0   ' Initialize variables.
Do   ' Outer loop.
   Do While Counter < 20   ' Inner loop.
      Counter = Counter + 1   ' Increment Counter.
      If Counter = 10 Then   ' If condition is True.
         Check = False   ' Set value of flag to False.
         Exit Do   ' Exit inner loop.
      End If
   Loop
Loop Until Check = False   ' Exit outer loop immediately.
```

7.4. While...Wend Statement Example

This example uses the **While...Wend** statement to increment a counter variable. The statements in the loop are executed as long as the condition evaluates to **True**.

```
Dim Counter
Counter = 0   ' Initialize variable.
While Counter < 20   ' Test value of Counter.
   Counter = Counter + 1   ' Increment Counter.
Wend   ' End While loop when Counter > 19.
Debug.Print Counter   ' Prints 20 in the Immediate window.
```

7.5. Exit Statement Example

This example uses the **Exit** statement to exit a **For...Next** loop, a **Do...Loop**, and a **Sub** procedure.

```
Sub ExitStatementDemo()
Dim I, MyNum
   Do            ' Set up infinite loop.
      For I = 1 To 1000   ' Loop 1000 times.
         MyNum = Int(Rnd * 1000)   ' Generate random numbers.
         Select Case MyNum   ' Evaluate random number.
            Case 7: Exit For   ' If 7, exit For...Next.
            Case 29: Exit Do   ' If 29, exit Do...Loop.
            Case 54: Exit Sub   ' If 54, exit Sub procedure.
         End Select
      Next I
   Loop
End Sub
```

8. YAZILMIŞ FONKSİYONLAR

8.1. YTL Fonksiyonu

Bu fonksiyon visual basic programlama dili kullanılabilen bütün ortamlarda kullanılabilir. Buna örnek olarak Microsoft Excel verilebilir.

Function YTLOKU(R1 As Double)
rem *** YAZAN TELAT TÜRKYILMAZ 2007 ***
On Error Resume Next
Dim B(10), O(10), Y(10)
OKUNAN = ""

B(1) = "bir": B(2) = "iki": B(3) = "üç": B(4) = "dört": B(5) = "beş": B(6) = "altı"
B(7) = "yedi": B(8) = "sekiz": B(9) = "dokuz"

O(1) = "on": O(2) = "yirmi": O(3) = "otuz": O(4) = "kırk": O(5) = "elli": O(6) = "altmış"
O(7) = "yetmiş": O(8) = "seksen": O(9) = "doksan":

Y(1) = "yüz": Y(2) = "ikiyüz": Y(3) = "üçyüz": Y(4) = "dörtyüz": Y(5) = "beşyüz"
Y(6) = "altıyüz":Y(7) = "yediyüz": Y(8) = "sekizyüz": Y(9) = "dokuzyüz": Y(10) = "bin"

```
textmilyar = "milyar "
textmilyon = "milyon "
textbin = "bin "

D1 = CDbl(Format(R1, "#.##"))
D2 = Fix(D1)
KURUS = (D1 - D2) * 100
SAYI = CDbl(D2)

YMILYAR = Fix(SAYI / 10 ^ 11): kalan = SAYI - YMILYAR * 10 ^ 11
ONMILYAR = Fix(kalan / 10 ^ 10): kalan = kalan - ONMILYAR * 10 ^ 10
milyar = Fix(kalan / 10 ^ 9): kalan = kalan - milyar * 10 ^ 9
ymilyon = Fix(kalan / 10 ^ 8): kalan = kalan - ymilyon * 10 ^ 8
onmilyon = Fix(kalan / 10 ^ 7): kalan = kalan - onmilyon * 10 ^ 7
milyon = Fix(kalan / 10 ^ 6): kalan = kalan - milyon * 10 ^ 6
yuzbin = Fix(kalan / 10 ^ 5): kalan = kalan - yuzbin * 10 ^ 5
onbin = Fix(kalan / 10 ^ 4): kalan = kalan - onbin * 10 ^ 4
BIN = Fix(kalan / 10 ^ 3): kalan = kalan - BIN * 10 ^ 3
yuz = Fix(kalan / 10 ^ 2): kalan = kalan - yuz * 10 ^ 2
ONN = Fix(kalan / 10 ^ 1): kalan = kalan - ONN * 10 ^ 1
bir = Fix(kalan / 10 ^ 0): kalan = kalan - bir * 10 ^ 0

If YMILYAR > 0 Then OKUNAN = OKUNAN + Y(YMILYAR)
If ONMILYAR > 0 Then OKUNAN = OKUNAN + O(ONMILYAR)
If milyar > 0 Then OKUNAN = OKUNAN + B(milyar)
If YMILYAR > 0 Or ONMILYAR > 0 Or milyar > 0 Then OKUNAN = OKUNAN + textmilyar
If ymilyon > 0 Then OKUNAN = OKUNAN + Y(ymilyon)
If onmilyon > 0 Then OKUNAN = OKUNAN + O(onmilyon)
If milyon > 0 Then OKUNAN = OKUNAN + B(milyon)
If ymilyon > 0 Or onmilyon > 0 Or milyon > 0 Then OKUNAN = OKUNAN + textmilyon
If yuzbin > 0 Then OKUNAN = OKUNAN + Y(yuzbin)
If onbin > 0 Then OKUNAN = OKUNAN + O(onbin)
If yuzbin > 0 Or onbin > 0 Then OKUNAN = OKUNAN + B(BIN) + textbin + " "
If BIN > 1 And yuzbin < 1 And onbin < 1 Then OKUNAN = OKUNAN + B(BIN) & textbin
If BIN = 1 And yuzbin < 1 And onbin < 1 Then OKUNAN = OKUNAN + textbin

textytl = OKUNAN + Y(yuz) + O(ONN) + B(bir) + " YTL "

textkurus = ""
kalan = Format(KURUS, "#.##")
If IsNumeric(kalan) Then
ONN = Fix(kalan / 10 ^ 1): kalan = kalan - ONN * 10 ^ 1
bir = Fix(kalan / 10 ^ 0): kalan = kalan - bir * 10 ^ 0
textkurus = O(ONN) & B(bir) & " Y.kuruş"
End If

YTLOKU = textytl & textkurus
End Function
```

8.2. Klasör Listeleri

Aşağıda verilen program LISTE değişkenine DIZIN değişkeninde bulunan klasörleri yükler

```
Sub KLASORLER(LISTE, Optional DIZIN = "", Optional EKLE = 0)

If DIZIN = "" Then DIZIN = App.Path
If EKLE = 0 Then ReDim LISTE(0)
If EKLE <> 0 And TypeName(LISTE) = "Empty" Then ReDim LISTE(0)

   Set FS = CreateObject("Scripting.FileSystemObject")
   Set DC = FS.GETFOLDER(DIZIN)

 For Each D In DC.subfolders
   ReDim Preserve LISTE(UBound(LISTE) + 1)
   LISTE(UBound(LISTE)) = Split(D, "*")
 Next

End Sub
```

8.3. Dosya Listeleri

Aşağıda verilen program LISTE değişkenine DIZIN değişkeninde bulunan dosyaları yükler

```
Sub DOSYALAR(LISTE, Optional DIZIN = "", Optional EKLE = 0)

If DIZIN = "" Then DIZIN = App.Path
If EKLE = 0 Then ReDim LISTE(0)
If EKLE <> 0 And TypeName(LISTE) = "Empty" Then ReDim LISTE(0)

   Set FS = CreateObject("Scripting.FileSystemObject")
   Set DC = FS.GETFOLDER(DIZIN)

 For Each D In DC.Files
   ReDim Preserve LISTE(UBound(LISTE) + 1)
   LISTE(UBound(LISTE)) = Split(D, "*")
 Next

End Sub
```

8.4. Debug Ortamında Verilen Dizi İçeriğini Görüntülemek

```
Public Sub DEBUG_VAR_PRINT(L1)
For R = 1 To UBound(L1)
For C = 0 To UBound(L1(R))
If C <> UBound(L1(R)) Then Debug.Print L1(R)(C); "|";
If C = UBound(L1(R)) Then Debug.Print L1(R)(C)
Next
Next
End Sub
```

8.5. BASİT SIRALAMA

```
SUB SIRALA(LISTE)
```

```
For k = 1 To UBound(LISTE) - 1
For L = k + 1 To UBound(LISTE)
   If LISTE(L)(S1) < LISTE(k)(S1) Then
   TEMP = LISTE(k)
   LISTE(k) = LISTE(L)
   LISTE(L) = TEMP
   End If
Next L
Next k

END SUB
```

8.6. Dizileri Sıralama

Aşağıdaki program LISTE dizisini sıralamaktadır.

```
Sub SIRALA(LISTE, Optional SUTUN1 = -1, Optional ARTAN = 1, Optional SUTUN2 = -1,
Optional SUTUN3 = -1)

For K = 1 To UBound(LISTE) - 1
DEGISECEK = ""
If SUTUN1 <> -1 Then DEGISECEK = DEGISECEK & LISTE(K)(SUTUN1)
If SUTUN2 <> -1 Then DEGISECEK = DEGISECEK & LISTE(K)(SUTUN2)
If SUTUN3 <> -1 Then DEGISECEK = DEGISECEK & LISTE(K)(SUTUN3)

For L = K + 1 To UBound(LISTE)
OKUNAN = ""
If SUTUN1 <> -1 Then OKUNAN = OKUNAN & LISTE(L)(SUTUN1)
If SUTUN2 <> -1 Then OKUNAN = OKUNAN & LISTE(L)(SUTUN2)
If SUTUN3 <> -1 Then OKUNAN = OKUNAN & LISTE(L)(SUTUN3)

If ARTAN = 1 Then
   If OKUNAN < DEGISECEK Then
   TEMP = LISTE(K)
   LISTE(K) = LISTE(L)
   LISTE(L) = TEMP
   End If
End If
If ARTAN <> 1 Then
   If OKUNAN > DEGISECEK Then
   TEMP = LISTE(K)
   LISTE(K) = LISTE(L)
   LISTE(L) = TEMP
   End If
End If

Next L
Next K

End Sub
```

8.7. ARRAY EKLE

```
Sub ARRAY_EKLE(LISTE, EKLENECEK_LISTE)
   If TypeName(LISTE) = "Empty" Then ReDim LISTE(0)
   ReDim Preserve LISTE(UBound(LISTE) + 1)
   LISTE(UBound(LISTE)) = EKLENECEK_LISTE
End Sub
```

8.8. DOSYADAN DİZİ OLUŞTURMAK

```
Sub DOSYADAN_VARIANT(LISTE, DOSYAADI, Optional EKLE = 0)
On Error Resume Next

If EKLE = 0 Then ReDim LISTE(0)
If EKLE <> 0 And TypeName(LISTE) = "Empty" Then ReDim LISTE(0)

If Dir(DOSYADI) = "" Then MsgBox "DOSYA YOK": Exit Sub

      Open DOSYAADI For Input As #1
      Do While Not EOF(1)
      Line Input #1, SATIR
      SS = Split(SATIR, Chr(9))
      ReDim Preserve LISTE(UBound(LISTE) + 1)
      LISTE(UBound(LISTE)) = SS
      Set SS = Nothing
      SATIR = ""
      Loop
      Close

End Sub
```

9. FONKSİYONLAR

9.1. Abs Function Example

This example uses the **Abs** function to compute the absolute value of a number.

```
Dim MyNumber
MyNumber = Abs(50.3)    ' Returns 50.3.
MyNumber = Abs(-50.3)    ' Returns 50.3.
```

9.2. Array Function Example

This example uses the **Array** function to return a **Variant** containing an array.

```
Dim MyWeek, MyDay
MyWeek = Array("Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun")
' Return values assume lower bound set to 1 (using Option Base
' statement).
MyDay = MyWeek(2)    ' MyDay contains "Tue".
MyDay = MyWeek(4)    ' MyDay contains "Thu".
```

9.3. Asc Function Example

This example uses the **Asc** function to return a character code corresponding to the first letter in the string.

```
Dim MyNumber
MyNumber = Asc("A")    ' Returns 65.
MyNumber = Asc("a")    ' Returns 97.
MyNumber = Asc("Apple")    ' Returns 65.
```

9.4. Choose Function Example

This example uses the **Choose** function to display a name in response to an index passed into the procedure in the `Ind` parameter.

```
Function GetChoice(Ind As Integer)
   GetChoice = Choose(Ind, "Speedy", "United", "Federal")
End Function
```

9.5. Chr Function Example

This example uses the **Chr** function to return the character associated with the specified character code.

```
Dim MyChar
MyChar = Chr(65)    ' Returns A.
MyChar = Chr(97)    ' Returns a.
MyChar = Chr(62)    ' Returns >.
MyChar = Chr(37)    ' Returns %.
```

9.6. CurDir Function Example

This example uses the **CurDir** function to return the current path.

```
' Assume current path on C drive is "C:\WINDOWS\SYSTEM" .
' Assume current path on D drive is "D:\EXCEL".
' Assume C is the current drive.
Dim MyPath
MyPath = CurDir   ' Returns "C:\WINDOWS\SYSTEM".
MyPath = CurDir("C")   ' Returns "C:\WINDOWS\SYSTEM".
MyPath = CurDir("D")   ' Returns "D:\EXCEL".
```

9.7. Date Function Example

This example uses the **Date** function to return the current system date.

```
Dim MyDate
MyDate = Date   ' MyDate contains the current system date.
```

9.8. DateAdd Function Example

This example takes a date and, using the **DateAdd** function, displays a corresponding date a specified number of months in the future.

```
Dim FirstDate As Date   ' Declare variables.
Dim IntervalType As String
Dim Number As Integer
Dim Msg
IntervalType = "m"   ' "m" specifies months as interval.
FirstDate = InputBox("Enter a date")
Number = InputBox("Enter number of months to add")
Msg = "New date: " & DateAdd(IntervalType, Number, FirstDate)
MsgBox Msg
```

9.9. DateDiff Function Example

Returns a **Variant** (**Long**) specifying the number of time intervals between two specified dates.

This example uses the **DateDiff** function to display the number of days between a given date and today.

```
Dim TheDate As Date   ' Declare variables.
Dim Msg
TheDate = InputBox("Enter a date")
Msg = "Days from today: " & DateDiff("d", Now, TheDate)
MsgBox Msg
```

9.10. DatePart Function Example

This example takes a date and, using the **DatePart** function, displays the quarter of the year in which it occurs.

```
Dim TheDate As Date   ' Declare variables.
Dim Msg
TheDate = InputBox("Enter a date:")
Msg = "Quarter: " & DatePart("q", TheDate)
MsgBox Msg
```

Setting	Description
yyyy	Year
q	Quarter
m	Month
y	Day of year
d	Day
w	Weekday
ww	Week
h	Hour
n	Minute
s	Second

The ***firstdayofweek*** argument has these settings:

Constant	Value	Description
vbUseSystem	0	Use the NLS API setting.
vbSunday	1	Sunday (default)
vbMonday	2	Monday
vbTuesday	3	Tuesday
vbWednesday	4	Wednesday
vbThursday	5	Thursday
vbFriday	6	Friday
vbSaturday	7	Saturday

9.11. Day Function Example

This example uses the **Day** function to obtain the day of the month from a specified date. In the development environment, the date literal is displayed in short format using the locale settings of your code.

```
Dim MyDate, MyDay
MyDate = #February 12, 1969#   ' Assign a date.
MyDay = Day(MyDate)   ' MyDay contains 12.
```

9.12. Dir Function Example

This example uses the **Dir** function to check if certain files and directories exist.

```
Dim MyFile, MyPath, MyName
' Returns "WIN.INI"  if it exists.
MyFile = Dir("C:\WINDOWS\WIN.INI")

' Returns filename with specified extension. If more than one *.ini
' file exists, the first file found is returned.
MyFile = Dir("C:\WINDOWS\*.INI")

' Call Dir again without arguments to return the next *.INI file in the
' same directory.
MyFile = Dir

' Return first *.TXT file with a set hidden attribute.
MyFile = Dir("*.TXT", vbHidden)

' Display the names in C:\ that represent directories.
MyPath = "c:\"   ' Set the path.
MyName = Dir(MyPath, vbDirectory)   ' Retrieve the first entry.
Do While MyName <> ""   ' Start the loop.
   ' Ignore the current directory and the encompassing directory.
   If MyName <> "." And MyName <> ".." Then
      ' Use bitwise comparison to make sure MyName is a directory.
      If (GetAttr(MyPath & MyName) And vbDirectory) = vbDirectory Then
         Debug.Print MyName   ' Display entry only if it
      End If   ' it represents a directory.
   End If
   MyName = Dir   ' Get next entry.
Loop
```

9.13. EOF Function Example

This example uses the **EOF** function to detect the end of a file. This example assumes that `MYFILE` is a text file with a few lines of text.

```
Dim InputData
Open "MYFILE" For Input As #1   ' Open file for input.
Do While Not EOF(1)   ' Check for end of file.
   Line Input #1, InputData   ' Read line of data.
   Debug.Print InputData   ' Print to the Immediate window.
Loop
Close #1   ' Close file.
```

9.14. Errors

This example uses the properties of the **Err** object in constructing an error-message dialog box. Note that if you use the **Clear** method first, when you generate a Visual Basic error with the **Raise** method, Visual Basic's default values become the properties of the **Err** object.

```
Dim Msg
' If an error occurs, construct an error message
On Error Resume Next   ' Defer error handling.
Err.Clear
Err.Raise 6   ' Generate an "Overflow" error.
' Check for error, then show message.
If Err.Number <> 0 Then
   Msg = "Error # " & Str(Err.Number) & " was generated by " _
         & Err.Source & Chr(13) & Err.Description
   MsgBox Msg, , "Error", Err.Helpfile, Err.HelpContext
End If
```

Trappable errors can occur while an application is running. Some trappable errors can also occur during development or compile time. You can test and respond to trappable errors using the **On Error** statement and the **Err** object. Unused error numbers in the range 1 – 1000 are reserved for future use by Visual Basic.

Code	Message
3	Return without GoSub
5	Invalid procedure call
6	Overflow
7	Out of memory
9	Subscript out of range
10	This array is fixed or temporarily locked
11	Division by zero
13	Type mismatch
14	Out of string space
16	Expression too complex
17	Can't perform requested operation
18	User interrupt occurred
20	Resume without error
28	Out of stack space
35	Sub, Function, or Property not defined
47	Too many DLL application clients
48	Error in loading DLL
49	Bad DLL calling convention
51	Internal error
52	Bad file name or number
53	File not found
54	Bad file mode
55	File already open
57	Device I/O error
58	File already exists
59	Bad record length

Code	Message
380	Invalid property value
381	Invalid property-array index
382	Property Set can't be executed at run time
383	Property Set can't be used with a read-only property
385	Need property-array index
387	Property Set not permitted
393	Property Get can't be executed at run time
394	Property Get can't be executed on write-only property
400	Form already displayed; can't show modally
402	Code must close topmost modal form first
419	Permission to use object denied
422	Property not found
423	Property or method not found
424	Object required
425	Invalid object use
429	Component can't create object or return reference to this object
430	Class doesn't support Automation
432	File name or class name not found during Automation operation
438	Object doesn't support this property or method
440	Automation error
442	Connection to type library or object library for remote process has been lost
443	Automation object doesn't have a default value
445	Object doesn't support this action
446	Object doesn't support named arguments
447	Object doesn't support current locale setting
448	Named argument not found

61	Disk full
62	Input past end of file
63	Bad record number
67	Too many files
68	Device unavailable
70	Permission denied
71	Disk not ready
74	Can't rename with different drive
75	Path/File access error
76	Path not found
91	Object variable or With block variable not set
92	For loop not initialized
93	Invalid pattern string
94	Invalid use of Null
97	Can't call Friend procedure on an object that is not an instance of the defining class
98	A property or method call cannot include a reference to a private object, either as an argument or as a return value
298	System DLL could not be loaded
320	Can't use character device names in specified file names
321	Invalid file format
322	Can’t create necessary temporary file
325	Invalid format in resource file
327	Data value named not found
328	Illegal parameter; can't write arrays
335	Could not access system registry
336	Component not correctly registered
337	Component not found
338	Component did not run correctly
360	Object already loaded
361	Can't load or unload this object
363	Control specified not found
364	Object was unloaded
365	Unable to unload within this context
368	The specified file is out of date. This program requires a later version
371	The specified object can't be used as an owner form for Show

449	Argument not optional or invalid property assignment
450	Wrong number of arguments or invalid property assignment
451	Object not a collection
452	Invalid ordinal
453	Specified not found
454	Code resource not found
455	Code resource lock error
457	This key is already associated with an element of this collection
458	Variable uses a type not supported in Visual Basic
459	This component doesn't support the set of events
460	Invalid Clipboard format
461	Method or data member not found
462	The remote server machine does not exist or is unavailable
463	Class not registered on local machine
480	Can't create AutoRedraw image
481	Invalid picture
482	Printer error
483	Printer driver does not support specified property
484	Problem getting printer information from the system. Make sure the printer is set up correctly
485	Invalid picture type
486	Can't print form image to this type of printer
520	Can't empty Clipboard
521	Can't open Clipboard
735	Can't save file to TEMP directory
744	Search text not found
746	Replacements too long
31001	Out of memory
31004	No object
31018	Class is not set
31027	Unable to activate object
31032	Unable to create embedded object
31036	Error saving to file
31037	Error loading from file

9.15. FileLen Function Example

This example uses the **FileLen** function to return the length of a file in bytes. For purposes of this example, assume that `TESTFILE` is a file containing some data.

```
Dim MySize
MySize = FileLen("TESTFILE")    ' Returns file length (bytes).
```

9.16. FreeFile Function Example

This example uses the **FreeFile** function to return the next available file number. Five files are opened for output within the loop, and some sample data is written to each.

```
Dim MyIndex, FileNumber
For MyIndex = 1 To 5    ' Loop 5 times.
   FileNumber = FreeFile    ' Get unused file
      ' number.
   Open "TEST" & MyIndex For Output As #FileNumber    ' Create file name.
   Write #FileNumber, "This is a sample."    ' Output text.
   Close #FileNumber    ' Close file.
Next MyIndex
```

9.17. Format Function Example

This example shows various uses of the **Format** function to format values using both named formats and user-defined formats. For the date separator (/), time separator (**:**), and AM/ PM literal, the actual formatted output displayed by your system depends on the locale settings on which the code is running. When times and dates are displayed in the development environment, the short time format and short date format of the code locale are used. When displayed by running code, the short time format and short date format of the system locale are used, which may differ from the code locale. For this example, English/U.S. is assumed.

`MyTime` and `MyDate` are displayed in the development environment using current system short time setting and short date setting.

```
Dim MyTime, MyDate, MyStr
MyTime = #17:04:23#
MyDate = #January 27, 1993#

' Returns current system time in the system-defined long time format.
MyStr = Format(Time, "Long Time")

' Returns current system date in the system-defined long date format.
MyStr = Format(Date, "Long Date")

MyStr = Format(MyTime, "h:m:s")    ' Returns "17:4:23".
MyStr = Format(MyTime, "hh:mm:ss AMPM")    ' Returns "05:04:23 PM".
MyStr = Format(MyDate, "dddd, mmm d yyyy")    ' Returns "Wednesday,
   ' Jan 27 1993".
' If format is not supplied, a string is returned.
MyStr = Format(23)    ' Returns "23".

' User-defined formats.
MyStr = Format(5459.4, "##,##0.00")    ' Returns "5,459.40".
MyStr = Format(334.9, "###0.00")    ' Returns "334.90".
MyStr = Format(5, "0.00%")    ' Returns "500.00%".
```

```
MyStr = Format("HELLO", "<")    ' Returns "hello".
MyStr = Format("This is it", ">")    ' Returns "THIS IS IT".
```

9.18. GetAttr Function Example

This example uses the **GetAttr** function to determine the attributes of a file and directory or folder.

```
Dim MyAttr
' Assume file TESTFILE has hidden attribute set.
MyAttr = GetAttr("TESTFILE")    ' Returns 2.

' Returns nonzero if hidden attribute is set on TESTFILE.
Debug.Print MyAttr And vbHidden

' Assume file TESTFILE has hidden and read-only attributes set.
MyAttr = GetAttr("TESTFILE")    ' Returns 3.

' Returns nonzero if hidden attribute is set on TESTFILE.
Debug.Print MyAttr And (vbHidden + vbReadOnly)

' Assume MYDIR is a directory or folder.
MyAttr = GetAttr("MYDIR")    ' Returns 16.
```

Constant	Value	Description
vbNormal	0	Normal.
vbReadOnly	1	Read-only.
vbHidden	2	Hidden.
vbSystem	4	System file.
vbDirectory	16	Directory or folder.
vbArchive	32	File has changed since last backup.

9.19. Hour Function Example

This example uses the **Hour** function to obtain the hour from a specified time. In the development environment, the time literal is displayed in short time format using the locale settings of your code.

```
Dim MyTime, MyHour
MyTime = #4:35:17 PM#    ' Assign a time.
MyHour = Hour(MyTime)    ' MyHour contains 16.
```

9.20. IIf Function Example

This example uses the **IIf** function to evaluate the `TestMe` parameter of the `CheckIt` procedure and returns the word "Large" if the amount is greater than 1000; otherwise, it returns the word "Small".

```
Function CheckIt (TestMe As Integer)
   CheckIt = IIf(TestMe > 1000, "Large", "Small")
End Function
```

9.21. InStr Function Example

This example uses the **InStr** function to return the position of the first occurrence of one string within another.

```
Dim SearchString, SearchChar, MyPos
SearchString ="XXpXXpXXPXXP"   ' String to search in.
SearchChar = "P"   ' Search for "P".

' A textual comparison starting at position 4. Returns 6.
MyPos = Instr(4, SearchString, SearchChar, 1)

' A binary comparison starting at position 1. Returns 9.
MyPos = Instr(1, SearchString, SearchChar, 0)

' Comparison is binary by default (last argument is omitted).
MyPos = Instr(SearchString, SearchChar)   ' Returns 9.

MyPos = Instr(1, SearchString, "W")   ' Returns 0.
```

9.22. Int Function, Fix Function Example

This example illustrates how the **Int** and **Fix** functions return integer portions of numbers. In the case of a negative number argument, the **Int** function returns the first negative integer less than or equal to the number; the **Fix** function returns the first negative integer greater than or equal to the number.

```
Dim MyNumber
MyNumber = Int(99.8)   ' Returns 99.
MyNumber = Fix(99.2)   ' Returns 99.

MyNumber = Int(-99.8)   ' Returns -100.
MyNumber = Fix(-99.8)   ' Returns -99.

MyNumber = Int(-99.2)   ' Returns -100.
MyNumber = Fix(-99.2)   ' Returns -99.
```

9.23. IsDate Function Example

This example uses the **IsDate** function to determine if an expression can be converted to a date.

```
Dim MyDate, YourDate, NoDate, MyCheck
MyDate = "February 12, 1969": YourDate = #2/12/69#: NoDate = "Hello"
MyCheck = IsDate(MyDate)   ' Returns True.
MyCheck = IsDate(YourDate)   ' Returns True.
MyCheck = IsDate(NoDate)   ' Returns False.
```

9.24. IsArray Function Example

This example uses the **IsArray** function to check if a variable is an array.

```
Dim MyArray(1 To 5) As Integer, YourArray, MyCheck   ' Declare array
variables.
YourArray = Array(1, 2, 3)   ' Use Array function.
MyCheck = IsArray(MyArray)   ' Returns True.
MyCheck = IsArray(YourArray)   ' Returns True.
```

9.25. IsEmpty Function Example

This example uses the **IsEmpty** function to determine whether a variable has been initialized.

```
Dim MyVar, MyCheck
MyCheck = IsEmpty(MyVar)    ' Returns True.

MyVar = Null    ' Assign Null.
MyCheck = IsEmpty(MyVar)    ' Returns False.

MyVar = Empty    ' Assign Empty.
MyCheck = IsEmpty(MyVar)    ' Returns True.
```

9.26. IsError Function Example

This example uses the **IsError** function to check if a numeric expression is an error value. The **CVErr** function is used to return an **Error Variant** from a user-defined function. Assume `UserFunction` is a user-defined function procedure that returns an error value; for example, a return value assigned with the statement `UserFunction = CVErr(32767)`, where 32767 is a user-defined number.

```
Dim ReturnVal, MyCheck
ReturnVal = UserFunction()
MyCheck = IsError(ReturnVal)    ' Returns True.
```

9.27. IsMissing Function Example

This example uses the **IsMissing** function to check if an optional argument has been passed to a user-defined procedure. Note that **Optional** arguments can now have default values and types other than **Variant**.

```
Dim ReturnValue
' The following statements call the user-defined function procedure.
ReturnValue = ReturnTwice()    ' Returns Null.
ReturnValue = ReturnTwice(2)    ' Returns 4.

' Function procedure definition.
Function ReturnTwice(Optional A)
   If IsMissing(A) Then
      ' If argument is missing, return a Null.
      ReturnTwice = Null
   Else
      ' If argument is present, return twice the value.
      ReturnTwice = A * 2
   End If
End Function
```

9.28. IsNull Function Example

This example uses the **IsNull** function to determine if a variable contains a **Null**.

```
Dim MyVar, MyCheck
MyCheck = IsNull(MyVar)    ' Returns False.

MyVar = ""
MyCheck = IsNull(MyVar)    ' Returns False.

MyVar = Null
MyCheck = IsNull(MyVar)    ' Returns True.
```

9.29. IsObject Function Example

This example uses the **IsObject** function to determine if an identifier represents an object variable. `MyObject` and `YourObject` are object variables of the same type. They are generic names used for illustration purposes only.

```
Dim MyInt As Integer, YourObject, MyCheck   ' Declare variables.
Dim MyObject As Object
Set YourObject = MyObject   ' Assign an object reference.
MyCheck = IsObject(YourObject)   ' Returns True.
MyCheck = IsObject(MyInt)   ' Returns False.
```

9.30. LBound Function Example

This example uses the **LBound** function to determine the smallest available subscript for the indicated dimension of an array. Use the **Option Base** statement to override the default base array subscript value of 0.

```
Dim Lower
Dim MyArray(1 To 10, 5 To 15, 10 To 20)    ' Declare array variables.
Dim AnyArray(10)
Lower = Lbound(MyArray, 1)    ' Returns 1.
Lower = Lbound(MyArray, 3)   ' Returns 10.
Lower = Lbound(AnyArray)   ' Returns 0 or 1, depending on
   ' setting of Option Base.
```

9.31. LCase Function Example

This example uses the **LCase** function to return a lowercase version of a string.

```
Dim UpperCase, LowerCase
Uppercase = "Hello World 1234"   ' String to convert.
Lowercase = Lcase(UpperCase)   ' Returns "hello world 1234".
```

9.32. Left Function Example

This example uses the **Left** function to return a specified number of characters from the left side of a string.

```
Dim AnyString, MyStr
AnyString = "Hello World"   ' Define string.
MyStr = Left(AnyString, 1)   ' Returns "H".
MyStr = Left(AnyString, 7)   ' Returns "Hello W".
MyStr = Left(AnyString, 20)   ' Returns "Hello World".
```

9.33. Len Function Example

The first example uses **Len** to return the number of characters in a string or the number of bytes required to store a variable. The **Type...End Type** block defining `CustomerRecord` must be preceded by the keyword **Private** if it appears in a class module. In a standard module, a **Type** statement can be **Public**.

```
Type CustomerRecord   ' Define user-defined type.
   ID As Integer   ' Place this definition in a
   Name As String * 10   ' standard module.
   Address As String * 30
End Type

Dim Customer As CustomerRecord   ' Declare variables.
Dim MyInt As Integer, MyCur As Currency
Dim MyString, MyLen
MyString = "Hello World"   ' Initialize variable.
MyLen = Len(MyInt)   ' Returns 2.
MyLen = Len(Customer)   ' Returns 42.
MyLen = Len(MyString)   ' Returns 11.
MyLen = Len(MyCur)   ' Returns 8.
```

The second example uses **LenB** and a user-defined function (**LenMbcs**) to return the number of byte characters in a string if ANSI is used to represent the string.

```
Function LenMbcs (ByVal str as String)
   LenMbcs = LenB(StrConv(str, vbFromUnicode))
End Function

Dim MyString, MyLen
MyString = "ABc"
' Where "A" and "B" are DBCS and "c" is SBCS.
MyLen = Len(MyString)
' Returns 3 - 3 characters in the string.
MyLen = LenB(MyString)
' Returns 6 - 6 bytes used for Unicode.
MyLen = LenMbcs(MyString)
' Returns 5 - 5 bytes used for ANSI.
```

9.34. StrConv Function Example

This example uses the **StrConv** function to convert a Unicode string to an ANSI string.

```
Dim i As Long
Dim x() As Byte
x = StrConv("ABCDEFG", vbFromUnicode)   ' Convert string.
For i = 0 To UBound(x)
   Debug.Print x(i)
Next
```

9.35. LoadPicture Function Example

This example uses the **LoadPicture** function to load a picture into a **PictureBox** control and to clear the picture from the control. To try this example, add a **PictureBox** control to a **Form** object, paste the code into the Declarations section of the **Form**, and then run the example and click the **Form**.

```
Private Sub Form_Click ()
   Dim Msg as String   ' Declare variables.
   On Error Resume Next   ' Set up error handling.
   Height = 3990
   Width = 4890   ' Set height and width.
   Picture1.Picture = LoadPicture("PAPER.CUR", vbLPCustom, vbLPColor, 32, 32)
' Load cursor.
   If Err Then
      Msg = "Couldn't find the .cur file."
      MsgBox Msg   ' Display error message.
      Exit Sub   ' Quit if error occurs.
   End If
   Msg = "Choose OK to clear the bitmap from the picturebox."
   MsgBox Msg
   Picture1.Picture = LoadPicture()   'Clear the picturebox.
End Sub
```

9.36. LOF Function Example

This example uses the **LOF** function to determine the size of an open file. This example assumes that TESTFILE is a text file containing sample data.

```
Dim FileLength
Open "TESTFILE" For Input As #1   ' Open file.
FileLength = LOF(1)   ' Get length of file.
Close #1   ' Close file.
```

9.37. LTrim, RTrim, and Trim Functions Example

This example uses the **LTrim** function to strip leading spaces and the **RTrim** function to strip trailing spaces from a string variable. It uses the **Trim** function to strip both types of spaces.

```
Dim MyString, TrimString
MyString = "  <-Trim->  "   ' Initialize string.
TrimString = LTrim(MyString)   ' TrimString = "<-Trim->  ".
TrimString = RTrim(MyString)   ' TrimString = "  <-Trim->".
TrimString = LTrim(RTrim(MyString))   ' TrimString = "<-Trim->".
' Using the Trim function alone achieves the same result.
TrimString = Trim(MyString)   ' TrimString = "<-Trim->".
```

9.38. Mid Function Example

The first example uses the **Mid** function to return a specified number of characters from a string.

```
Dim MyString, FirstWord, LastWord, MidWords
MyString = "Mid Function Demo"   ' Create text string.
FirstWord = Mid(MyString, 1, 3)   ' Returns "Mid".
LastWord = Mid(MyString, 14, 4)   ' Returns "Demo".
MidWords = Mid(MyString, 5)   ' Returns "Function Demo".
```

The second example use **MidB** and a user-defined function (**MidMbcs**) to also return characters from string. The difference here is that the input string is ANSI and the length is in bytes.

```
Function MidMbcs(ByVal str as String, start, length)
   MidMbcs = StrConv(MidB(StrConv(str, vbFromUnicode), start, length),
vbUnicode)
End Function

Dim MyString
MyString = "AbCdEfG"
' Where "A", "C", "E", and "G" are DBCS and "b", "d",
' and "f" are SBCS.
MyNewString = Mid(MyString, 3, 4)
' Returns ""CdEf"
MyNewString = MidB(MyString, 3, 4)
' Returns ""bC"
MyNewString = MidMbcs(MyString, 3, 4)
' Returns "bCd"
```

9.39. Minute Function Example

This example uses the **Minute** function to obtain the minute of the hour from a specified time. In the development environment, the time literal is displayed in short time format using the locale settings of your code.

```
Dim MyTime, MyMinute
MyTime = #4:35:17 PM#    ' Assign a time.
MyMinute = Minute(MyTime)    ' MyMinute contains 35.
```

9.40. Month Function Example

This example uses the **Month** function to obtain the month from a specified date. In the development environment, the date literal is displayed in short date format using the locale settings of your code.

```
Dim MyDate, MyMonth
MyDate = #February 12, 1969#    ' Assign a date.
MyMonth = Month(MyDate)    ' MyMonth contains 2.
```

9.41. Now Function Example

This example uses the **Now** function to return the current system date and time.

```
Dim Today
Today = Now    ' Assign current system date and time.
```

9.42. QBColor Function Example

This example uses the **QBColor** function to change the **BackColor** property of the form passed in as `MyForm` to the color indicated by `ColorCode`. **QBColor** accepts integer values between 0 and 15.

```
Sub ChangeBackColor (ColorCode As Integer, MyForm As Form)
   MyForm.BackColor = QBColor(ColorCode)
End Sub
```

Number	Color	Number	Color
0	Black	8	Gray
1	Blue	9	Light Blue
2	Green	10	Light Green
3	Cyan	11	Light Cyan
4	Red	12	Light Red
5	Magenta	13	Light Magenta
6	Yellow	14	Light Yellow
7	White	15	Bright White

9.43. RGB Function Example

This example shows how the **RGB** function is used to return a whole number representing an **RGB** color value. It is used for those application methods and properties that accept a color specification. The object `MyObject` and its property are used for illustration purposes only. If `MyObject` does not exist, or if it does not have a **Color** property, an error occurs.

```
Dim RED, I, RGBValue, MyObject
Red = RGB(255, 0, 0)   ' Return the value for Red.
I = 75   ' Initialize offset.
RGBValue = RGB(I, 64 + I, 128 + I)     ' Same as RGB(75, 139, 203).
MyObject.Color = RGB(255, 0, 0)   ' Set the Color property of
   ' MyObject to Red.
```

Part	Description
red	Required; **Variant** (**Integer**). Number in the range 0–255, inclusive, that represents the red component of the color.
green	Required; **Variant** (**Integer**). Number in the range 0–255, inclusive, that represents the green component of the color.
blue	Required; **Variant** (**Integer**). Number in the range 0–255, inclusive, that represents the blue component of the color.

9.44. Right Function Example

This example uses the **Right** function to return a specified number of characters from the right side of a string.

```
Dim AnyString, MyStr
AnyString = "Hello World"   ' Define string.
MyStr = Right(AnyString, 1)   ' Returns "d".
MyStr = Right(AnyString, 6)   ' Returns " World".
MyStr = Right(AnyString, 20)   ' Returns "Hello World".
```

9.45. Rnd Function Example

This example uses the **Rnd** function to generate a random integer value from 1 to 6.

```
Dim MyValue
MyValue = Int((6 * Rnd) + 1)    ' Generate random value between 1 and 6.
```

9.46. Round Function

Returns a number rounded to a specified number of decimal places.

Round(*expression* [**,***numdecimalplaces*]**)**

The **Round** function syntax has these parts:

Part	Description
expression	Required. Numeric expression being rounded.
numdecimalplaces	Optional. Number indicating how many places to the right of the decimal are included in the rounding. If omitted, integers are returned by the **Round** function.

9.47. Second Function Example

This example uses the **Second** function to obtain the second of the minute from a specified time. In the development environment, the time literal is displayed in short time format using the locale settings of your code.

```
Dim MyTime, MySecond
MyTime = #4:35:17 PM#    ' Assign a time.
MySecond = Second(MyTime)    ' MySecond contains 17.
```

9.48. Shell Function Example

This example uses the **Shell** function to run an application specified by the user.

```
' Specifying 1 as the second argument opens the application in
' normal size and gives it the focus.
Dim RetVal
RetVal = Shell("C:\WINDOWS\CALC.EXE", 1)    ' Run Calculator.
```

9.49. Space Function Example

This example uses the **Space** function to return a string consisting of a specified number of spaces.

```
Dim MyString
' Returns a string with 10 spaces.
MyString = Space(10)

' Insert 10 spaces between two strings.
MyString = "Hello" & Space(10) & "World"
```

9.50. Sqr Function Example

This example uses the **Sqr** function to calculate the square root of a number.

```
Dim MySqr
MySqr = Sqr(4)    ' Returns 2.
MySqr = Sqr(23)    ' Returns 4.79583152331272.
MySqr = Sqr(0)    ' Returns 0.
MySqr = Sqr(-4)    ' Generates a run-time error.
```

9.51. Str Function Example

This example uses the **Str** function to return a string representation of a number. When a number is converted to a string, a leading space is always reserved for its sign.

```
Dim MyString
MyString = Str(459)    ' Returns " 459".
MyString = Str(-459.65)    ' Returns "-459.65".
MyString = Str(459.001)    ' Returns " 459.001".
```

9.52. StrComp Function Example

This example uses the **StrComp** function to return the results of a string comparison. If the third argument is 1, a textual comparison is performed; if the third argument is 0 or omitted, a binary comparison is performed.

```
Dim MyStr1, MyStr2, MyComp
MyStr1 = "ABCD": MyStr2 = "abcd"    ' Define variables.
MyComp = StrComp(MyStr1, MyStr2, 1)    ' Returns 0.
MyComp = StrComp(MyStr1, MyStr2, 0)    ' Returns -1.
MyComp = StrComp(MyStr2, MyStr1)    ' Returns 1.
```

9.53. String Function Example

This example uses the **String** function to return repeating character strings of the length specified.

```
Dim MyString
MyString = String(5, "*")    ' Returns "*****".
MyString = String(5, 42)    ' Returns "*****".
MyString = String(10, "ABC")    ' Returns "AAAAAAAAAA".
```

9.54. Time Function Example

This example uses the **Time** function to return the current system time.

```
Dim MyTime
MyTime = Time    ' Return current system time.
```

9.55. UBound Function Example

This example uses the **UBound** function to determine the largest available subscript for the indicated dimension of an array.

```
Dim Upper
Dim MyArray(1 To 10, 5 To 15, 10 To 20)   ' Declare array variables.
Dim AnyArray(10)
Upper = UBound(MyArray, 1)    ' Returns 10.
Upper = UBound(MyArray, 3)    ' Returns 20.
Upper = UBound(AnyArray)    ' Returns 10.
```

9.56. UCase Function Example

This example uses the **UCase** function to return an uppercase version of a string.

```
Dim LowerCase, UpperCase
LowerCase = "Hello World 1234"    ' String to convert.
UpperCase = UCase(LowerCase)    ' Returns "HELLO WORLD 1234".
```

9.57. Val Function Example

This example uses the **Val** function to return the numbers contained in a string.

```
Dim MyValue
MyValue = Val("2457")    ' Returns 2457.
MyValue = Val(" 2 45 7")    ' Returns 2457.
MyValue = Val("24 and 57")    ' Returns 24.
```

9.58. Weekday Function Example

This example uses the **Weekday** function to obtain the day of the week from a specified date.

```
Dim MyDate, MyWeekDay
MyDate = #February 12, 1969#    ' Assign a date.
MyWeekDay = Weekday(MyDate)    ' MyWeekDay contains 4 because
   ' MyDate represents a Wednesday.
```

Constant	Value	Description
vbSunday	1	Sunday
vbMonday	2	Monday
vbTuesday	3	Tuesday
vbWednesday	4	Wednesday
vbThursday	5	Thursday
vbFriday	6	Friday
vbSaturday	7	Saturday

9.59. Year Function Example

This example uses the **Year** function to obtain the year from a specified date. In the development environment, the date literal is displayed in short date format using the locale settings of your code.

```
Dim MyDate, MyYear
MyDate = #February 12, 1969#   ' Assign a date.
MyYear = Year(MyDate)   ' MyYear contains 1969.
```