T.C.
MİLLİ EĞİTİM BAKANLIĞI

MEGEP

(MESLEKİ EĞİTİM VE ÖĞRETİM SİSTEMİNİN
GÜÇLENDİRİLMESİ PROJESİ)

BİLİŞİM TEKNOLOJİLERİ

GÖRSEL PROGRAMLAMA
KOD PARÇALARI

ANKARA 2007

Milli Eğitim Bakanlığı tarafından geliştirilen modüller;

- Talim ve Terbiye Kurulu Başkanlığının 02.06.2006 tarih ve 269 sayılı Kararı ile onaylanan, Mesleki ve Teknik Eğitim Okul ve Kurumlarında kademeli olarak yaygınlaştırılan 42 alan ve 192 dala ait çerçeve öğretim programlarında amaçlanan mesleki yeterlikleri kazandırmaya yönelik geliştirilmiş öğretim materyalleridir (Ders Notlarıdır).

- Modüller, bireylere mesleki yeterlik kazandırmak ve bireysel öğrenmeye rehberlik etmek amacıyla öğrenme materyali olarak hazırlanmış, denenmek ve geliştirilmek üzere Mesleki ve Teknik Eğitim Okul ve Kurumlarında uygulanmaya başlanmıştır.

- Modüller teknolojik gelişmelere paralel olarak, amaçlanan yeterliği kazandırmak koşulu ile eğitim öğretim sırasında geliştirilebilir ve yapılması önerilen değişiklikler Bakanlıkta ilgili birime bildirilir.

- Örgün ve yaygın eğitim kurumları, işletmeler ve kendi kendine mesleki yeterlik kazanmak isteyen bireyler modüllere internet üzerinden ulaşılabilirler.

- Basılmış modüller, eğitim kurumlarında öğrencilere ücretsiz olarak dağıtılır.

- Modüller hiçbir şekilde ticari amaçla kullanılamaz ve ücret karşılığında satılamaz.

İÇİNDEKİLER

AÇIKLAMALAR

KOD	**482BK0068**
ALAN	**Bilişim Teknolojileri**
DAL/MESLEK	**Veri Tabanı Programcılığı**
MODÜLÜN ADI	**Görsel Programlama Kod Parçaları**
MODÜLÜN TANIMI	Kod Parçalarının tanıtımı ile ilgili öğrenme materyalidir.
SÜRE	40/32
ÖN KOŞUL	Görsel Programlamada Arayüz modülünü bitirmiş olmak
YETERLİK	Görsel programlamada kod kısımlarını yazmak
MODÜLÜN AMACI	**Genel Amaç** Gerekli ortam sağlandığında görsel programlamada kod yazabileceksiniz. **Amaçlar** 1. Kod okunaklığını sağlayabileceksiniz. 2. Değişkenler, sabitler ve listeler ile çalışabileceksiniz. 3. Diziler ile çalışabileceksiniz. 4. Operatörler ile çalışabileceksiniz. 5. Metin ve tarih veri türleri ile çalışabileceksiniz. 6. Şartlı deyimler ile çalışabileceksiniz.
EĞİTİM ÖĞRETİM ORTAMLARI VE DONANIMLARI	Bilgisayar laboratuvarı ve bu ortamda bulunan; görsel programlama için gerekli donanıma sahip bilgisayar, lisanslı işletim sistemi programı, kâğıt ve kalem hazır bulundurulmalıdır.
ÖLÇME VE DEĞERLENDİRME	Her faaliyet sonrasında o faaliyetle ilgili değerlendirme soruları ile kendi kendinizi değerlendireceksiniz. Modül içinde ve sonunda verilen öğretici sorularla edindiğiniz bilgileri pekiştirecek, uygulama örneklerini ve testleri gerekli süre içinde tamamlayarak etkili öğrenmeyi gerçekleştireceksiniz. Sırasıyla araştırma yaparak, grup çalışmalarına katılarak ve en son aşamada alan öğretmenlerine danışarak ölçme ve değerlendirme uygulamalarını gerçekleştiriniz.

GİRİŞ

Sevgili Öğrenci,

Bilgi dünyasının vazgeçilmezi olarak yerini alan bilgisayarlar ile ortaya çıkan yeni programlar günlük hayatta insanların işlerinde kolaylıklar sağlamaktadır. Büyük ticari kuruluşlardan küçük işletmelere kadar kullanılan paket programlar, bilgisayar destekli programlar ve pek çok alanda kullanılan birçok program, programlama dilleri kullanılarak yazılır.

Programlama dilleri zaman içerisinde gelişmiş, yerini yeni gelen versiyonlara devretmiş ve eski diller geçerliliğini yitirip yeni diller ortaya çıkmıştır. Bu yeni dillerden biri de Microsoft'un birkaç yıldır geliştirmekte olduğu "kişileri, kurumları ve sistemleri birbirine bağlayan yazılımlar" olarak tanımlanan .NET teknolojisidir.

Visual Studio.NET paketinin önemli bir parçasını oluşturan Visual Basic.NET, ilk çıktığı günden bu yana dünyanın dört bir yanında geniş kullanıma ulaştı ve .NET platformunun gözdesi haline geldi. .NET ile birlikte nesne yönelimli programlama desteğine de sahip olan Visual Basic, her kademedeki programcılar için her tür uygulamayı geliştirmede kullanılabilecek bir dil niteliği halini almıştır.

Bu modülün içerisinde ise Visual Basic.NET dilinin temel taşlarını oluşturan bilgileri bulacaksınız. .NET dilinde karşılaşılan hataları, bu hataları düzeltme yöntemlerini, değişkenler üzerinde işlem yapma ve bütün programlama dillerinde programcılığın önemli bir boyutunu oluşturan matematiksel ve mantıksal operatörleri bulacaksınız.

Bu modülde verilen bilgilerin programcılık hayatınızda yeni ufuklar açmasını temenni ederek derslerinizde başarılar dilerim.

ÖĞRENME FAALİYETİ–1

AMAÇ

Kod okunaklığını sağlayabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Visual Basic.NET programını, bir önceki sürümü olan Visual Basic 6.0 ile karşılaştırınız. Farklılıklarını rapor haline dönüştürerek arkadaşlarınızla paylaşınız.
- Visual Basic.NET programının kod editöründeki yenilikleri inceleyiniz.

1. KOD OKUNAKLIĞI

Hangi dili kullanırsanız kullanın (Pascal, C, Delphi ve Java…) tüm dillerde belli konularda aynı prensipler vardır. Bu temel konuları öğrenirseniz, çoğu dillerde rahatlıkla aynı işlemi yapabilirsiniz.

1.1. “Syntax Error” Yazım Hataları

Bu hatanın oluşma sebebi; kullandığımız programlama dilinin söz dizim (komut yazımı, noktalama, değişken deklarasyonu vb.) kurallarına uymayıp hatalı kod satırları yazışımızdır. Visual Studio editörü hataları bizim için yakalayıp, altını farklı bir renkle çizmek suretiyle bizi uyarmasına rağmen Visual Basic.NET’te sıkça yaptığımız söz dizimi hatalarından başlıcaları; declare (tanımlama) edilmemiş değişken kullanmak, uygunsuz tip atamaları yapmak ve bir prosedürün yanlış argümanlarla çağrılması gibi hatalardır.

1.2. Açıklama Satırları “ ‘ ” ve “REM” Deyimi

Bir cümleyi yazıp Enter tuşuna bastığımızda o komut cümlesi değerlendirilir ve yazım yanlışları saptanır. Büyük küçük harf ayrımı yapılmaz. Komutun ilk harfi otomatik olarak büyütülür. Komut olmayan açıklama satırları için tek tırnak, **Rem** ya da **Remark** kullanılır.

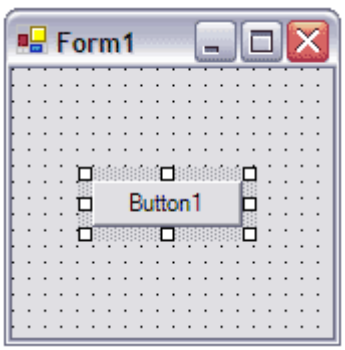
```
Rem yardım düğmesi
' Bu düğme tıklanınca çalışır.
' Satır arasına da açıklama koyabilirsiniz.
```

1.3. Kod Düzenleme Penceresi

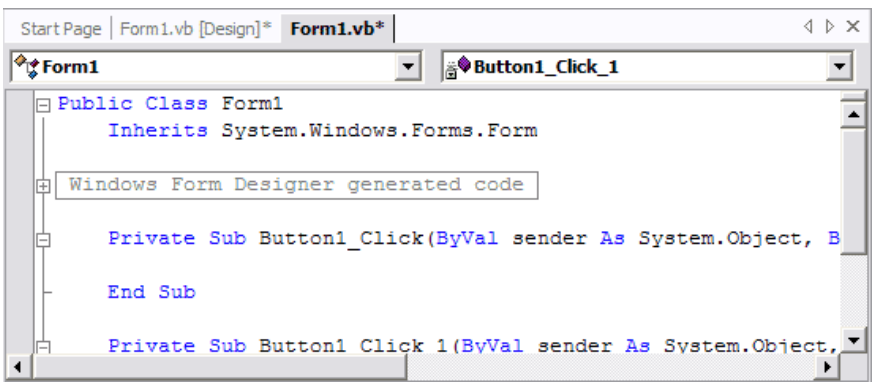
Nesneler ve prosedürlerle olayların ilişkilendirilmesi ve bu işlemin gerçekleşebilmesi için gerekli olan kod editörü kullanımına yönelik bilgiler ve örnekler sunulmuştur.

1.3.1.Olay Yöneticilerine Giriş

Modern programlamanın temel taşlarından olan Olaylar (Events); butonlar, formlar ve diğer birçok nesneyle (metin kutusu, açılan menü vb.) ilişkisel yapıdadır ve nesnelerin karakteristik özelliklerine göre şekillenir. Visual Basic.NET'te kodun olaylarla ilişkilendirilmesi görevini yürüten prosedürlere Olay Yöneticileri denir. Bu yöneticileri örnekleyecek olursak, formumuz üzerinde bir butonumuz mevcutsa ve bu butonun aktif olmasıyla gerçekleşecek birtakım işlemler tasarlamışsak bu butonun ButtonName_Click_Number olayını yönetmemiz gerekecektir. Bu işlem;



Resim 1.1: Forma Buton ekleme



Resim 1.2: Code Editörü

Button'nun çift tıklanması ile ulaşabileceğimiz **Button_Click** olayını, sağ tarafta bulunan kod penceresi yardımıyla yönetmemiz şeklinde gerçekleşir. Bu yönetimi gerçekleştirebilmek için **Code Editor** kullanımı ile ilgili önemli birkaç hususu bilmemiz gerekir.

1.3.2. Code Editor Kullanımı

Bir forma veya modüle kod eklemek için kullandığımız kod editörüne olaylarla ilgili kodlar eklerken dikkat etmemiz gereken önemli noktalarından biri de, kodun formla nasıl ilişkilendirildiğini bilmektir. Visual Basic.NET programlama dilinde Code Editörü çalıştırabilmek için etkin kullanılan 2 farklı yol vardır. Bunlar;

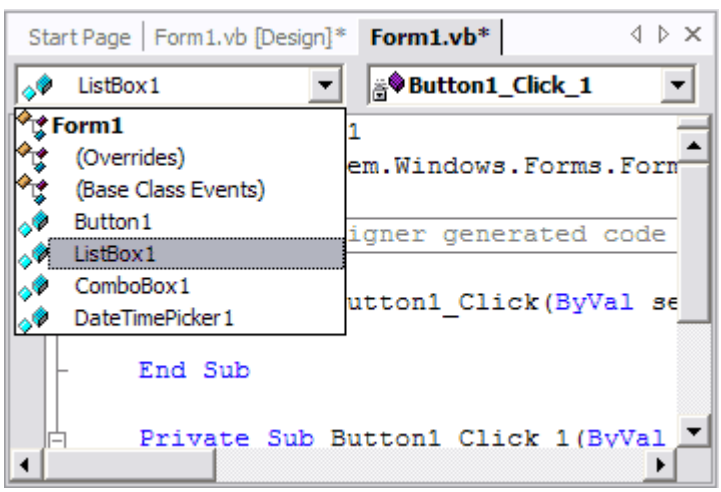
- Solution Explorer içinde formu seçip, sağ tuşa tıklayarak gelen menüde "View Code" öğesini seçmek.
- Code Editöründeki bölümüne ulaşmak istediğiniz nesne, aktif durumdayken klavyenin F7 tuşuna basmak.

Bu işlemlerden herhangi birinin ardından resim 1.2 gelecektir. Bu pencere üzerinde dikkat etmemiz gereken iki temel nokta sol ve sağda bulunan açılır listelerdir. Bunlardan sol tarafta bulunan "**Nesneler**" listesidir. Bu listeyi kullanarak üzerinde işlem yapmak istediğimiz nesneyi aktif hale geçirebiliriz. Sağ taraftaki açılır liste ise "**Prosedürler**" listesidir. Bu listeden de herhangi bir olayı veya prosedürü aktif olan nesne ile eşleştirip görev atamaları yapabiliriz.

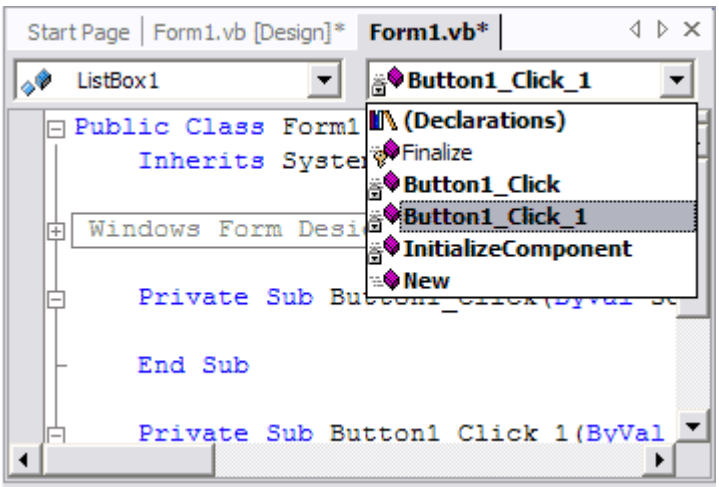
Windows formları diğer nesneler gibi gerçek nesnelerdir. Ayrıca buradaki gizli kodlar, Windows formunda yer alan nesneleri, nesneyi kaldırmak ve oluşturmak için kullanılan metotları ve bunlara ilaveten kritik önemi olan birçok nesne ve metodu barındırmaktadır. Burada önemli olan nokta ise, bu kodlara erişim yetkilendirmesinin güçlü bir güvenlik hiyerarşisine sahip oluşudur.

1.3.3. Olaya Kod Ekleme İşlemi

Bu işlemi gerçekleştirmek için öncellikle, resim 1.3'te görülen "**Nesneler**" listesinden kod eklemek istediğimiz nesneyi seçeriz. Ardından resim 1.4'te görülen "**Prosedürler**" listesinden de seçilen nesneye ait olayı belirledikten sonra, belirlenen olayın aktivasyonu ile birlikte gerçekleşmesini istediğimiz işlemleri, şekillerde de görülen kod bloğuna deklare ederiz.

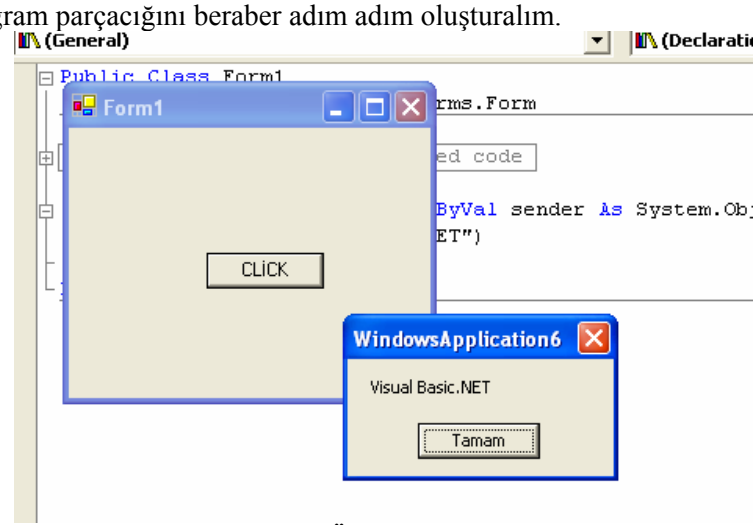


Resim 1.3: Code Editöründe Procedüre kısmından nesne seçme



Resim 1.4: Seçilen nesneye ait olayı seçme

Son olarak konuyla ilgili bir örnek verecek olursak; üzerine tek bir buton yerleştirilmiş bir formumuz ve bu butonunun Click olayına deklare edilmiş bir mesaj kutumuz olsun. Şimdi bu program parçacığını beraber adım adım oluşturalım.



Resim 1.5: Örnek Code Editörü

- File → New → Project seçeneğinden açılan diyalog kutusundan Visual Basic.NET Projects → Windows Application seçilir.
- Oluşan form üzerine Araç Kutusunu "**Toolbox**" kullanarak resim 1.3'teki gibi bir buton ekleyelim ve butonun properties penceresinden text özelliğini Click yapalım.
- Şimdi butona çift tıklamak suretiyle butonun Click olayına girelim.
- Şimdi açmış olduğumuz kod editörüne, MessageBox.Show("Visual Basic.NET") komut satırını yazalım.
- Son olarak F5 tuşunu kullanarak programımızı Run (Çalıştır) edelim.

Kod Editörü (Code Window) uygulama kodunun yazıldığı yerdir. Uygulamada yer alan her form ya da kod modülü, ayrı birer kod editörü gibi kullanılır.

1.4. Uzun Satır Sonlarında "_" Devam Karakteri

Visual Studio Code Editor içindeki program satırları 65.000 karakter uzunluğunda olabilse de 80 ya da daha az karakterli satırlarla çalışmak en kolayıdır. Uzun program ifadelerini satır hariç bir boşluk ve ardından bir satır devam karakteri (_) kullanarak birden çok satıra bölebiliriz. Ancak tırnak işaretleri arasındaki bir düzeyi satır devam karakteri kullanarak bölemezsiniz.

```
    Private Sub Form1_MouseMove(ByVal sender As Object, _
ByVal e As System.Windows.Forms.MouseEventArgs) _
Handles MyBase.MouseMove

    End Sub
```

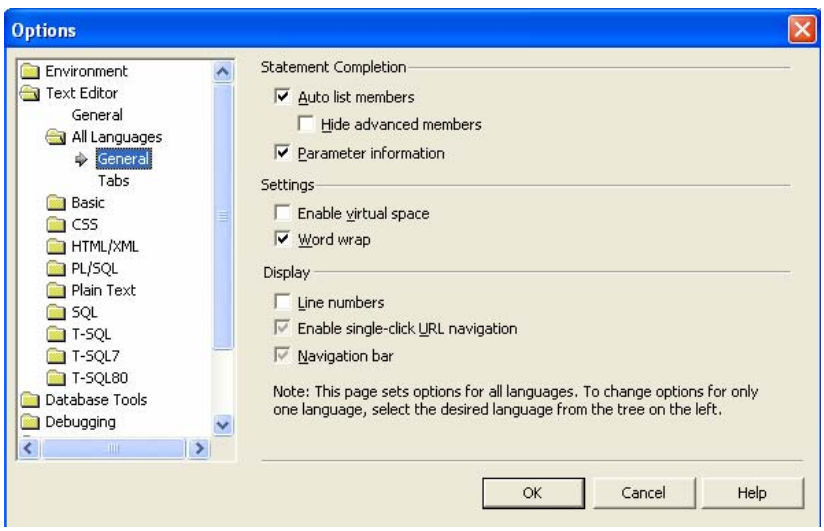
Resim 1.6: Uzun satır sonları işaretinin (_) kullanımı

Üstteki kod penceresinde de görüldüğü gibi uzun olan tek satır, üç kısa satıra bölündü. Bunu, satır devamı olan (_) karakteri koyarak yaptık. Satır devamı karakteri olmadan, uzun ilk satırı, üç kısa satıra bölmek derleyeci hatasına neden olacaktır. Bunun nedeni, Visual Basic'in, her bir kod satırına tamamlamış olarak kabul etmesidir. Satır devamı karakteri, Visual Basic.NET'te parçalara bölünmüş satırların beraber işleme gireceğini ifade eder.

1.5. Sözcük Kaydır Seçeneği (Edit*Advanced*Word Wrap)

Word Wrap özelliğinin aktif olması durumunda kod editörü içindeki uzun satırları alt satıra kaydırır. Bu özellik aktif değilse yazı Enter'e basıldığı noktadan sonra bir alt satıra geçer. Bu özelliği aktif hale getirmek için;

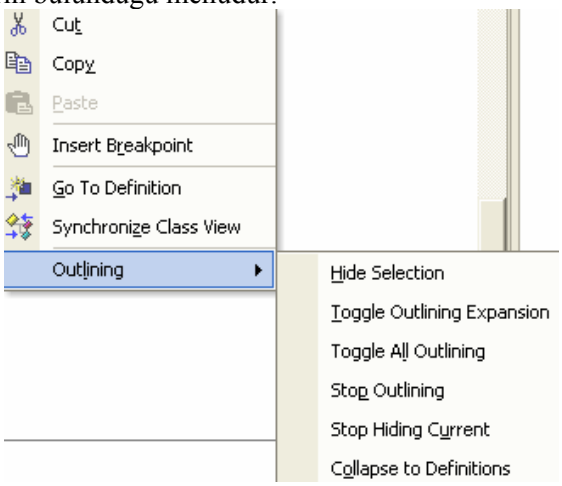
Tools menüsünden Options seçeneği seçilir. Gelen pencerede Text Editör dizininde yer alan All Languages seçeneğinde yer alan General sekmesinden ulaşılır.



Resim 1.7: Word Wrap seçeneğinin aktif hale getirilmesi

1.6. Kod Bloklarında "Outlining" Menü Seçenekleri (+ ve – simgeleri)

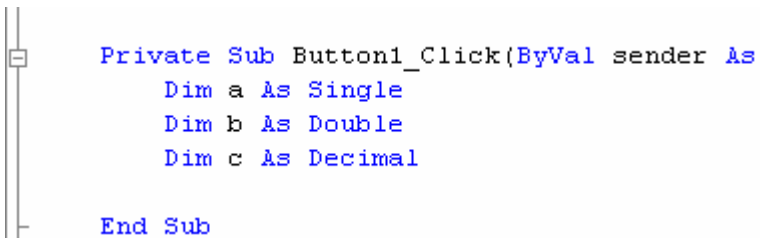
Kod editöründe bulunan prosedür, fonksiyon gibi alt programları saklayıp gösteren ve dış hat ile ilgili seçeneklerin bulunduğu menüdür.



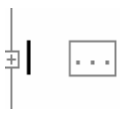
Resim 1.8: Outlining işlem menüsü

Resim 1.8'de görülen Outlining komutunun alt seçeneklerine bakacak olursak;

Hide Selection: Seçili metni gizler daha sonra bu metnin sol tarafındaki + ve – düğmesi ile açılıp saklanabilir.



Resim 1.9: Hide Selection özelliğinin (-) hali



Resim 1.10: Hide Selection özelliğinin (+) hali

Toggle Outlining Expansion: İmlecin bulunduğu alt programı gizler. Aynı şekilde + tuşuyla metin orijinal durumuna döndürülebilir.

Toggle All Outlining: Tüm alt programları gizler.

Stop Outlining: İmlecin bulunduğu alt programın dış hat ile bağlantısını keser. Bu şekilde metin gizlenmez.

Stop Hiding Current: Gizlenmiş metnin dış hat ile bağlantısını keser.

Collapse to Definitions: Tüm alt programları gizler.

1.7. “Text Editör” Araç Çubuğundaki “Bookmark – Kitap İzi” Seçenekleri

Visual Basic.NET programlama dilinde yazılan kod satırlarında yer alan önemli bir satır üzerine gidilmek istenebilir. Uzun yazılmış programlar içerisinde önem arz eden bu satıra ulaşmak için Bookmark-Kitap İzi özelliğinden faydalanılır. Bu işareti eklemek için imleç işaret bırakılmak istenen satır üzerinde iken araç çubuğunda yer alan Bookmark-Kitap İzi seçeneklerinden faydalanılır.

Resim 1.11: Bookmark-Kitap İzi Araç Çubuğu

1. Seçili olan satıra Bookmark-kitap izi ekler.

2. Bir sonraki Bookmark-kitap izi olan satıra gider.

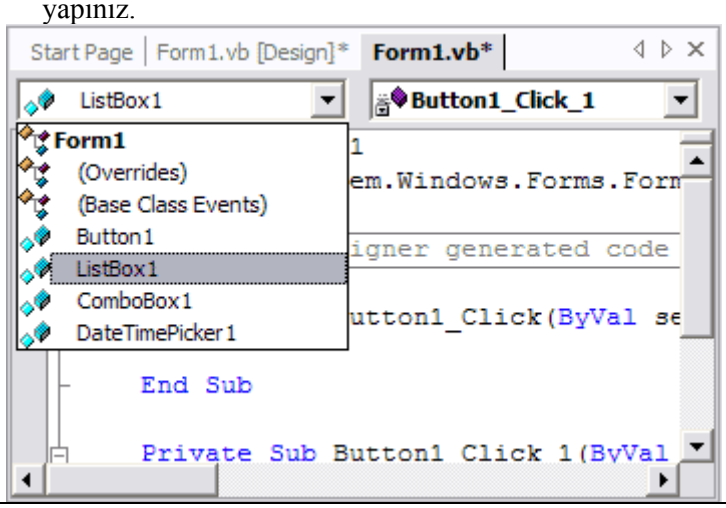
3. Bir önceki Bookmark-kitap izi olan satıra gider.

4. Bookmark – kitap izi eklenmiş bütün satırların işaretlerini kaldırır.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler

1. Program yazarken oluşan yazım hatalarını düzeltiniz.
`Public Class Form1`
`  Private Sub TextBox1_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChanged`
`  Ad = TextBox1Text`
`  End Sub`
`End Class`
	Programı yazarken imla kurallarını dikkatlice kontrol ediniz.

2. Kod yazarken gerekli yerlere prosedür tanımlamalarını yapınız.  	VB.NET'te yeni bir Project açınız. Karşınıza gelen forma bir tane buton ekleyiniz. Bu botuna ait bir prosedür oluşturunuz.
3. Değişken tanımlarını programın başlangıç kısmına yazınız.	

4. Dim komutları ile başlayan satırları seçip,
" Edit*Advanced*Comment Selection" ile açıklama satırı haline getiriniz.
`Public Class Form1`
`  Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click`
`  Dim a As Integer`
`  Dim ad As String`
`  End Sub`
`End Class`
	Yukarıda yapacağınız uygulamanın herhangi bir kod satırını seçiniz. Sonra Edit*Advanced*Comment Selection seçeneğini seçerek farkı inceleyiniz.

5. Uzun programlarda belli satırlara "Bookmark" kitap izi yapınız.	

ÖLÇME VE DEĞERLENDİRME

A- OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; ilk 7 soruda verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. Kod editöründe hatalı yazılan bir komuta "Syntax Error" hatası denir. ( )

2. Program içerisinde açıklama satırları çift tırnak (") işareti ile başlar.

3. Program işleme esnasında açıklama satırları derleyici tarafından dikkate alınmaz. ( )

4. Form ya da modüle kod eklemek için klavyeden F7 tuşuna basılır. ( )

5. Kod bloklarında seçili olan komutları gizlemek için seçili olan bölüm üzerinde sağ tuş yapıp, gelen menüden Outlining seçeneğinde yer alan Hide Selection komutundan yararlanılır. ( )

6. Kod pencresinde uzun satır sonlarını (#) işareti yardımıyla bölerek, satır devamı sağlanır. ( )

7. Uzun kod satırlı programlarda önemli olan satırları işaretlemek için Bookmark araç çubuğu kullanılır. ( )

8. Açıklama satırı yazmak için aşağıdaki işaretlerden hangisi kullanılır?
A) #
B) ^
C) ‘
D) “

9. Aşağıdaki kısayol tuşlarından hangisi kod düzenleme penceresini açar?
A) F1
B) F5
C) F6
D) F7

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konuları öğrenme faaliyete geri dönerek tekrar inceleyiniz.

ÖĞRENME FAALİYETİ–2

AMAÇ

Değişkenler, sabitler ve listeler ile çalışabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Bilgisayarda veri girişleri geçici veya kalıcı olarak nerelerde, hangi donanım parçaları içinde nasıl saklanıyor?
- Programlama dillerinde değişken tanımlarken uyulması gereken kuralları araştırınız?
- Daha önceden öğrenmiş olduğunuz programlama dillerinde yer alan veri türlerini inceleyerek Visual Basic.NET programında bulunan veri türleriyle kıyaslayınız.

2. DEĞİŞKENLER

Temel programlama işlevlerini gerçekleştirebilmek için "**değişkenler**" kullanılmaktadır. Bir değişken küçük bir depo alanıdır. İçinde sayılar, kelimeler, harfler saklanabilir. Değişkenler program içerisinde defalarca kullanılsa da, içinde aynı anda sadece bir bilgi tutar. Değişkenler geçici olarak kullanıldıkları için, program veya bilgisayar kapanınca silinir.

Visual Basic'in değişken tanımlamalarında bazı farklılıklar vardı fakat bu farklılıklar Visual Basic.NET'te çözülmüştür. Visual Basic 6.0'da farklı veri tipindeki değişkenlerin tek bir satırda tanımlanmasına izin veriliyordu.

Örneğin;

```
Dim baslik As String, sayac As Integer
```

Yukarıdaki kod Visual Basic.NET'te çalışmayacaktır. Çünkü VB.NET tek bir satırda sadece aynı veri tipine ait değişkenlerin tanımlanmasına izin vermektedir. Örneğin ;

```
Dim baslik, adSoyad As String
```

Yukarıdaki kod Visual Basic 6.0'da farklı bir şekilde çalışacaktır. Baslik değişkeni Variant olarak kabul edilecektir. Visual Basic.NET'te her ikisi de String kabul edilir. Değişkenlere tanımlama sırasında ilk değerleri verilebilir.

```
Dim kod As Integer = 23
Dim ad As String = "Dinçer GÜNDOĞDU"
Dim maas As Integer = birimucret * 30
```

2.1. “Dim” Anahtar Kelimesi, “Private, Public ve Static” Tanımlamalar

Bir değişkenin kullanılmadan önce tanımlanması gerekir. Bu tanımlama genellikle Dim deyimi ile yordamın başında yapılır.

Kullanımı:

Dim Değişken [As tipi]

Bir programda geçici olarak verilerin temsili ve değerlerin saklanması için değişkenler kullanılır. Değişkenlerin değerleri, program akışı içerisinde sürekli olarak değişebilir.

Örnek

```
x=18
y=23
x=x+3
y=y+x
```

şeklindeki atama ifadeleri ile, x değişkeninin değeri önce 18 ve y değişkeninin değeri 23 olarak atanmıştır. Sonra x’in değeri 21 ve y’nin değeri 44 olmuştur. Atama ifadesinde, “=” sembolünün sağ tarafındaki ifade hesaplanır ve bulunan sonuç sol taraftaki değişkenin bellekteki yerine atanır. Bu anlamda, bir değişkene yeni bir değer atanırsa, bu değişkenin önceki değeri silinecektir.

Değişken tanımlanmaz veya tanımlanırken tip belirtilmezse, Visual Basic.NET o değişkeni “**Object**” veri tipinde algılar.

Bilindiği gibi VB projelerine .frm uzantılı form dosyaları ve .bas uzantılı Module dosyaları eklenebiliyor. Buna göre değişkenler ya formlarda ya da BAS uzantılı modüllerde tanımlanabilir.

VB’deki hazır nesnelerin içerdiği kod penceresindeki her yordamda değişken tanımı yapılabilir. Ancak bu değişkenler sadece o yordam dahilinde kullanılabilir.

Object liste kutusunda bulunan (General) seçeneği seçildiğinde, Procedure liste kutusunda da (Declarations) aktif olur. Forma ait bütün yordamlarda kullanılmak istenen değişkenler General – Declarations kısmında tanımlanır. Bu değişkenler söz konusu form aktif olduğu süre içerisinde yaşar. Başka bir forma geçildiğinde bellekten silinir ve içeriği boşalır.

Özetlemek gerekirse Formlar söz konusu olduğunda değişkenler yordamların içinde ya da Forma ait (declaration) penceresinde tanımlanır. Yordam dahilinde tanımlanan değişkenler ancak tanımlandığı yordam içinde kullanılabilir. Söz konusu formun (declaration) penceresinde tanımlanan değişkenler Forma ait bütün yordamlarda kullanılabilir.

BAS uzantılı modüllerde de değişkenler tanımlanıp kullanılabilir. Projeye modül eklemek için Project – Add Module menüsünden faydalanılır.

Kod girişi yapılan bu pencere, formlara ait olan kod penceresinden farklı değildir. Fakat bu pencerede kod yoktur ama yazılabilir. Aynı formlarda olduğu gibi Sub ve End Sub arasında yordam olarak yazılmalıdır.

Formlarda olduğu gibi modüle de yordam eklemek için Tools menüsünden Add Procedure komutu kullanılır. Modül içinde tüm yordamlarda kullanılmak istenen değişken modülün (General) (Declarations) kısmında tanımlanır.

Projenin her yerinde yani Form ve modülerde kullanılmak istenen değişkenler BAS uzantılı bir modülün declaration penceresinde Global veya Public bildiri deyimi ile tanımlanır.

2.1.1. Private Sözcüğü

Yerel değişkenler tanımlamak için kullanılır. Bu deyim aynı zamanda yerel prosedürlerin ve Class Modüllerin tanımında da kullanılır.

Kullanımı:
Private değişken_adi [[indis]] [As tipi]

```
Private Sub Command1_Click()
        Dim x As Double
        x = x + 10
        MsgBox x
End Sub
```

2.1.2. Public Sözcüğü

Modül bazında Public ya da global değişken tanımlamak için kullanılır. Bir Public değişken hem proje bazında kullanılabilir hem de projenin tüm modüllerindeki prosedürler tarafından kullanılabilir. Eğer Public değişken Class modülde kullanılırsa proje dışında da değişkene erişebilir. Ayrıca Public olarak prosedürler ve Class modüller de tanımlanabilir.

Kullanımı:
Public değişken_adi [[indis]] [As [New] tipi]

2.1.3. Static Sözcüğü

Lokal değişkenleri, prosedürün her çağırılışında, bir önceki çağırılış sonunda hesaplanmış olan değerlerini saklar hale getirmek için, Static sözcüğü ile tanımlanmaları gereklidir.

```
Static x As Integer
        Private Sub Command1_Click()
        Static x As Double
        x = x + 10
        MsgBox x
End Sub
```

Değişkenlerin tanımlanmasını mecburi tutmak için Declarations kısmında Option Explicit deyimi yazılmalıdır.

2.2. İlkel Değişken Türleri (String, Char, Boolean, Date)

2.2.1. String

Karakter sınırı verilmezse 2 milyar karaktere kadar atama yapılabilen sayısal olmayan veri tipidir. Karakterlerden oluşan veriler String olarak tanımlanır. Örneğin öğrenci adı gibi bir bilgi String olarak tanımlanmalıdır. String veriler sayıları içerebilir ancak sayısal işleme giremez. String değişkenlere değer atamak için çift tırnak (“ ”)karakteri kullanılır. Veri tipini temsilen “ **$** ” karakteri de kullanılabilir. Bir String değişkene belli bir uzunluk da verilebilir. Bunun için asteriks (*) karakterinden sonra uzunluk birimi girilir.

```
Dim s As String
S = “ZEYNEP SAĞLAM”
Dim isim$= “ Dinçer”
Dim Adı As String * 20
```

Örnek :

1.

```
Show
Dim a As String
Dim b As String
Dim c As String

a = 168
b = 122
c = a + b
Print c
```

Sonuç = 168122 olur.

1.1.

```
Show

a$ = 168
b$ = 122

c$ = a + b
Print c
```

Sonuç = 168122 olur.

2.

```
Show
Dim a As Integer
Dim b As Integer
Dim c As Integer

a = 168
b = 122
c = a + b
Print c
```

Sonuç = 290 olur.

2.1.

```
Show

a% = 168
b% = 122

c% = a + b
Print c
```

Sonuç = 290 olur.

2.2.2. Char

İki bytelık bir karakter tipidir. İçinde sadece 1 karakter barındırabilir. Bu tipten değişkenlere atama yaparken sadece bir karakter ataması yapılabilir. Birden fazla atandığında sadece ilk karakter dikkate alınacaktır.

```
Dim C1,C2, C3
C1= "A"
C2= "ABC"
C3= Chr(65)
Msgbox(C1)                    'Sonuç=A
Msgbox(C2)                    'Sonuç=A
Msgbox(C3)                    'Sonuç=A olur.
```

2.2.3. Boolean

2 bytelık veri tipi olmasına rağmen True veya False değerleri alabilir. Yani daha çok iki durumlu değişkenlerde kullanılır. Bu tipten tanımlanan değişkenlere True, False değerleri atanabileceği gibi sayısal değer de atanabilir.

```
Dim sayi as Boolean
sayi=True
sayi=1
sayi=1999

Dim sayi as Boolean
sayi=False
sayi=0
```

Bu tip değişkenler üzerinde işlem yaparken ise True değeri -1 sayısına, False değeri ise 0 sayısına karşılık gelir.

2.2.4. Date

8 byte yer kaplayan bu değişkene 1/1/100 ile 31/12/9999 arasındaki tarih ve 0:00:00 ile 23:59:59 arasındaki saat atamaları yapılabilir.

2.3. Sayısal Değişken Türleri (Integer, Short, Long, Byte)

2.3.1. İnteger

Visual Basic'te tam sayı değişkenleri tanımlamak için kullanılır. Hafızada 2 byte yer kaplar. Alabileceği değer aralığı –32768 ile +32767 arasındadır. DefInt bildiri deyimi ile tanımlanabilir. Ayrıca bir değişkenin sonunda % karakteri bulunuyorsa bu değişken integer tipindedir.

Örnek

```
Private Sub Form_Load()
Dim Maas As Integer
DefInt A-C
Oran% = 100
A_sayı = 100
B_sayı = Oran*A_sayı
C_sayı = B_sayı + A_sayı – 1000
Maas= 32767
End Sub
```

Integer tipinde tanımlanan değişkenlere daha büyük sayılar atanırsa Overflow oluşur.

2.3.2. Short

İki bytelık işaretli tam sayı tipidir. -32768 ile 32767 arasında değer alabilir. Atama yapılacak değerin sonuna "S" karakteri kullanılarak short değer ataması yapılabilir.

Dim mesafe as Short=1200S

2.3.3. Long

Daha büyük bir aralıkta integer yani tam sayı tanımlamak için kullanabileceğimiz bir veri tipidir. Hafızada 4 byte yer kaplar. Kullanılabilecek uç degerler +2.147.483.647 ile -2.147.483.648 arasındadır. Long tipinde bir değişken tanımlamak için DefLng bildirimini veya değişken sonunda & karakterini kullanabiliriz.

Örnek

```
Private Sub Form_Load()
    DefLng A-B
    Bölüm=50000
    cıkan&=600000
    Kalan& = (cıkan / Bölüm ) * 10000
End Sub
```

2.3.4. Byte

1 bytelık işaretsiz tam sayı tipidir. 0 ile 255 arasında değer alabilir.

Örnek

```
Dim Not As Byte
Not=78
```

2.4. Sayıların Kararlılıkları (Single, Double)

2.4.1. Single

Tam sayı olmayan ondalıklı sayılar için kullanabileceğimiz bir veri tipidir. Kayan noktalı sayı olarak isimlendirilir. Single tipindeki veriler bellekte 4 byte yer kaplar. Negatif sayılar için alabileceği aralık -3.402823E38 ile –1.401298E-45, pozitif sayılar için alabileceği aralık 1.401298E-45 ile 3.402823E38 arasındadır. Single tipinde veri tanımlamak için DefSgn bildirimi veya değişken sonuna " ! " karakteri konur.

Örnek

```
Private Sub Form_Load()
        DefSgn A-B
        Bölüm=50000
        Cıkan!=600000
    End Sub
```

2.4.2. Double

Visual Basic'te kullanılabilecek en büyük sayısal değerlerin veri tipidir. Hafızada 8 byte yer kaplar. 16 haneye kadar hassastır. Maksimum alabileceği değerler pozitif sayılar için 4.94065645841247E-324 ile 1.797693134862232E308 , negatif sayılar için de -1.797693134862232E308 ile -4.94065645841247E-324 arasındadır. DefDbl bildirimi veya "#" sembolü ile double tipinde değişkenler tanımlanabilir.

Örnek

```
Private Sub Form_Load()
    DefDbl A-K
    Darı=50000
    Bugday=600000
    Arpa=340.56
End Sub
```

2.5. Türü Varsayılan Olan Değişkenler (Object)

Object: Önceki versiyonlarda Variant olarak tanımlanan veri tipi Visual Basic.NET'te Object olarak geçer. Bunun sebebi Visual Basic.NET'te bütün veri tiplerinin gerçekte birer nesne olmasındandır. Bu nesnelerin tümü Object sınıfında türemiştir.

```
Dim A           ‘ veya Dim A As Object olarak da tanımlanabilir.

A=5             ‘ Şu anda A Integer tipinde

A=A+200000      ‘ İşlem Integer sınırlarını aştığı için şu anda A Long

A=5.7           ‘ Sayı ondalıklı olduğu için şu anda A Single

A=A+5.3E+200    ‘ Sayı Single sınırını aştığı için şu anda A Double

A=’a1c2’        ‘ A’nın tipi String oldu

A= A+5          ‘ Hata, A son olarak String tipinde olduğu için işlem yapılamaz.
```

Görüldüğü gibi Object olarak tanımlanan değişkenlerin oldukça esnek bir yapısı vardır. Program esnasında herhangi bir tanıma gerek kalmadan tip değiştirebilmektedir. Bu avatajına rağmen Object olarak tanımlanmış tiplerle yapılan işlemler diğer tiplere göre çok daha yavaş çalışır. Çünkü, her işlemde, işleme giren değişkenlerin tipleri kontrol edilmek zorunda kalınacak ve işlemler yavaşlayacaktır.

Bu yüzden uzun süren döngülerde değişkenlerinizi gerçek tipleriyle tanımlamanız işlemleri hızlandıracaktır.

2.6. Veri Türlerinin Hafızadaki İlk Değerleri

Değişken tanımlama işlemi Visual Basic.NET programlama dilinde Dim bildiri deyimi ile yapılır. Değişkenin tanımlanması hafızada ayrılacak hafıza miktarının belirli olmasını sağlar. Eğer değişkenlerin tipini belirtmeden bir kullanım yaparsak bu değişkenlerin Object tipinde olduğu kabul edilir. Bu da hafızada gereksiz yer kaybına sebep olur.

2.7. Kayar Noktalı Değişkenlerin Bilimsel Gösterimi (E veya e)

Visual Basic programı belli sayıda veri tipi içerir. Integer, tam sayılar için kullanılır ve en yaygın olanıdır.

Tam sayı pozitif (55) veya negatif (- 55) veya sıfır olabilir. Bununla birlikte, integer veri tipi - .5, 0.5 veya 5.5 gibi ondalık noktasının sağında sayı olan kayan noktalı sayılar için kullanılmamalıdır. Bu veri tipi için integer anahtar sözcüğü, .Net Framework'ün systems.Int32 veri tipinin bir başka adıdır.

Visual Basic kayan noktalı sayılar için kullanılabilecek Double, Single ve Decimal gibi birkaç veri tipine sahiptir. Sayı büyüklüğünün önemli olduğu durumlarda Single veya Double, basamağın önemli olduğu durumlarda ise Decimal tipini kullanabiliriz. Single ve Double türü daha büyük sayılarla işlem yapabilmesine karşın basamak hassasiyetleri Decimal kadar iyi değildir.

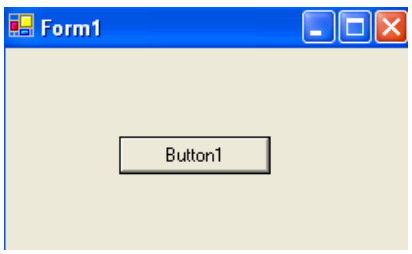
Single tipi sayıların sadece 7 basamağını tutabilmekte, gerisini üstel (10 üzeri) şekle çevirmektedir. Double türü ise 15 basamağa kadar tutmakta gerisini üstel hale getirmektedir. Decimal ise 29 basamağı tutabilmektedir. Sonuçta Decimal daha küçük sayılarla işlem yapar ama işlemdeki sayıların bütün basamaklarını korur.

Örnek

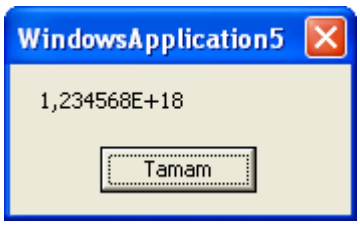
```
    Private Sub Button1_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button1.Click
        Dim a As Single
        Dim b As Double
        Dim c As Decimal

        a = 1234567890123456789
        a = a + 3
        MsgBox(a)
        b = 1234567890123456789
        b = b + 3
        MsgBox(b)
        c = 1234567890123456789
        c = c + 3
        MsgBox(c)

    End Sub
```

Resim 2.1: Kayan noktalı değişkenlerin bilimsel gösterimi

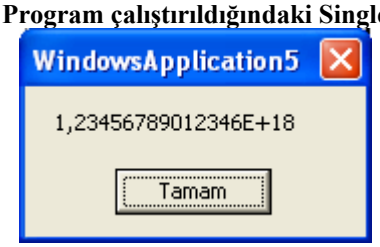
Program çalışıtırıldığında ;



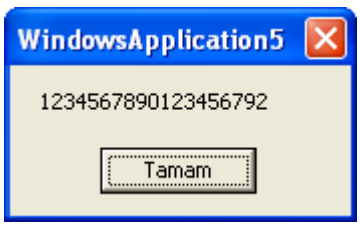
Resim 2.2: Program çalıştırıldığındaki ilk görüntü



Resim 2.3: Program çalıştırıldığındaki Single görünümü



Resim 2.4: Program çalıştırıldığındaki Double görünümü



Resim 2.5: Program çalıştırıldığındaki Decimal görünümü

2.8. Değişken İsimlendirme Kuralları

Değişkene verilecek ismin anlaşılır bir isim olması programın okunurluluğunu kolaylaştırır. Bir değişken tanımlanırken aşağıda verilen kuralların göz önünde bulundurulması gerekir.

- Değişken ismi bir harf ile başlamalıdır. Bir rakam veya özel işaretle başlayamaz.
  Dim isim1, isim2 → doğru
  Dim 1isim, 2isim → yanlış
- Değişken isminde boşluk bulunmaz. Bunun yerine alt çizgi karakteri (_) kullanılabilir.
  Dim ad_soyad, Dogum_yeri → doğru
  Dim ad soyad, dogum yeri → yanlış
- Değişken isminde sadece harfler, rakamlar ve alt çizgi karakteri bulunabilir.
  Dim satis_tarihi, görev_yeri → doğru
  Dim satis – tarihi, görev.yeri → yanlış
- Değişkene verilecek isim Visual Basic komutlarından oluşmamalıdır.
  Dim not → yanlış
  Dim and → yanlış

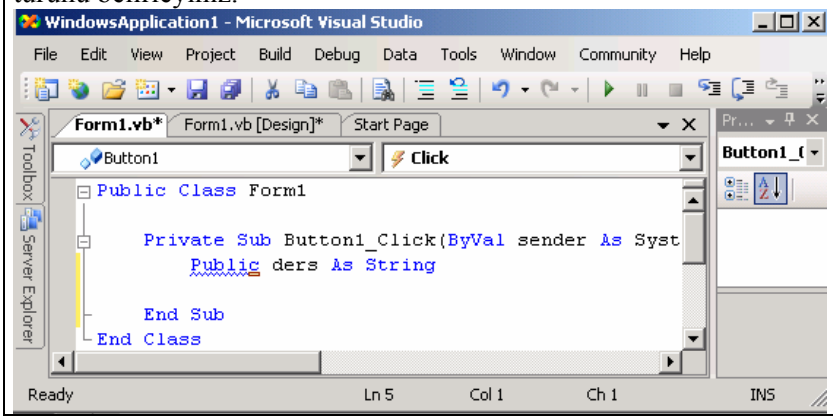
UYGULAMA FAALİYETİ

<table>
<tr><th>İşlem Basamakları</th><th>Öneriler</th></tr>
<tr><td>1. Dim komutu ile değişkeni tanımlayınız.
<pre>Public Class Form1

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

    End Sub
End Class</pre></td><td>Üç sayının aritmetik ortalamasını bulan programın değişkenlerini dim komutu ile tanımlayınız.</td></tr>
<tr><td>2. Değişkenin alabileceği minimum ve maksimum değere göre değişkenin türünü defterinize yazınız.</td><td></td></tr>
<tr><td>3. Değişkenin ilk değerini aktarınız.
<pre>Public Class Form1

    Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
 Dim ad As String = "DİNÇER"

    End Sub
End Class</pre></td><td>Ad değişkenine sizler de kendi isminizi atayınız.</td></tr>
<tr><td>4. Tüm programda aynı değere sahip olacak bir sabit tanımlayınız ve türünü belirleyiniz.
 
</td><td>Sizler de tanımlayacağınız ad, soyad ve yaş değişkenlerini tüm program içerisinde kullanacak şekilde tanımlayınız.</td></tr>
</table>

ÖLÇME VE DEĞERLENDİRME

A- OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; ilk 7 soruda verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. Bütün değişkenler kullanılmadan önce tanımlanmalıdır. ( )

2. Değişken isimlerini istediğimiz gibi verebiliriz. ( )

3. Integer tipinde tanımlanan bir değişkende ondalıklı bir sayı saklanabilir. ( )

4. Metin türü değişkenler üzerinde her türlü matematiksel işlemi yapabiliriz. ( )

5. Visual Basic'de sayi, Sayi ve SaYi değişkenlerinin tamamı farklıdır. ( )

6. Değişken tanımlama işlemi Dim bildirimi ile yapılır. ( )

7. Double veri tipinde tanımlanan bir değişken 7 basamaklı bir sayı tutabilir. ( )

8. Hangi değişken isimlendirmesi <u>hatalıdır</u>?
A) A2
B) _B3
C) ölçüt
D) AdAnA

9. String türündeki bir değişkene değer atamak için ifade hangi işaretler arasına yazılmalıdır?
A) *
B) +
C) "
D) '

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konuları öğrenme faaliyete geri dönerek tekrar inceleyiniz.

ÖĞRENME FAALİYETİ–3

AMAÇ

Uygun ortam sağlandığında dizilerle çalışabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Okulunuzda bulunan bütün öğrencilerin isimlerinin her biri, tek tek bir değişkende saklanmak istenseydi bu işlem nasıl gerçekleşirdi. Bu uzun işlemi düşünerek cevap bulmaya çalışınız.
- Tek boyutlu ve çok boyutlu dizi kavramlarını araştırarak arkadaşlarınızla örnekler üzerinde tartışınız.

3. DİZİLER

Aynı tipte ve birbirleriyle ilgili bilgilerin oluşturduğu bütüne **Dizi** adı verilir. Program içerisinde aynı anda aynı tür bilgiden çok sayıda mevcut olması ve bu bilgiler üzerinde toplu işlem yapılmasının gerekmesi durumunda diziler kullanılır. Bir dizi aynı tipte ve aynı adı paylaşan bir grup değişken demektir. Diziler birçok değişkene aynı adla ulaşmayı sağlayan bir grup veri yapısıdır. Bir indeks numarası ile dizi içindeki elemanlara ulaşılır. Dizi içindeki elemanlar aynı tipteki verilerdir. Örneğin haftanın günleri ya da iller gibi.

Dizilerin gerekliliğini iki örnek ile inceleyelim:

Örnek 1 100 adet isim ve telefon bilgisini saklamak için,
İsim : 100 adet
Telefon : 100 adet

Toplam 200 adet değişken kullanılmalıdır.

Örnek 2 Bir sınıfta okuyan 40 öğrencinin isim ve her öğrencinin 5'er farklı dersi ve bu derslerden aldıkları 3 farklı not bilgisini aynı anda bellekte tutmak için,
İsim : 40 adet
Dersler : 40 öğrenci x 5 ders = 200 adet
Notlar : 40 öğrenci x 5 ders x 3 not = 600 adet

Toplam 840 adet değişken kullanılmalıdır. Bu durum imkânsızdır.

Şimdi bu örnekleri dizilerle inceleyelim.

Örnek 1
İsim(x) →100 elemanlı
Telefon(x) →100 elemanlı

Örnek 2

40 elemanlı	İsim(I)	İsim dizisinin (I+1). elemanı
40,5 elemanlı	Dersler(I,J) (I+1).	Öğrencinin, (J+1). dersi
40,5,3 elemanlı	Notlar(I,J,K)(I+1).	Öğrencinin, (J+1). Dersinin, (K+1). notu

Dizi Tanımlama: Dim dizi değişkeni (eleman sayısı) As tipi

Dizilerin eleman sayıları sıfırdan başladığı için 5 elemanlı bir dizinin eleman sayısı 4 olarak belirtilir.

Örnek 3 Haftanın günleri

```
Private Sub Command1_Click()
  Dim gunler(6) As String
  Dim i As Integer
          gunler(0) = "Pazartesi"                    'dizinin 1. elemanı
          gunler(1) = "Salı"
          gunler(2) = "Çarşamba"
          gunler(3) = "Perşembe"
          gunler(4) = "Cuma"
          gunler(5) = "Cumartesi"
          gunler(6) = "Pazar"                        'dizinin 7. elemanı
                For i = 0 To 6
                        Print "Haftanın " & i + 1 & " . günü " & gunler(i)
                Next i
End Sub
```

Bir Microsoft Visual Basic.Net uygulamasında bilgi kullanmak önemli bir iştir ve programlarınız daha kapsamlı hale geldikçe verileri depolamak ve işlemek için ek araçlara gerek duyarsınız. Bu bölümde, değişkenleri ve diğer bilgileri diziler olarak adlandırılan kullanışlı kaplar biçiminde nasıl düzenleyeceğinizi öğreneceksiniz. Diziler denetlenecek birkaç düzine ya da daha çok öğe varsa veri denetleme işlemini düzenler. Ayrıca bir Visual Basic programında koleksiyonlar olarak adlandırılan nesne gruplarını kullanmayı ve özel For Each… Next döngüsünü kullanarak koleksiyonları işlemeyi de öğreneceksiniz. Diziler ve koleksiyonlar bir arada ele alındıklarında bir programda büyük miktarlarda bilgiyi denetlemek için kusursuz araçlardır.

Dizi bildirimleri ve dizilere veri atamaları artık aynı program ifadesi içinde yapılabilmektedir. Örneğin liste() adlı bir dizinin bildirimi ve bu diziye dört öğe eklemek için kullanılacak söz dizimi aşağıdaki gibidir.

Dim liste( ) as integer = (5.10.15.20)

3.1. Tek ve Çok Boyutlu Diziler

Diziler, bir program içinde veri depolamak için güçlü ve zaman sınamasından geçmiş oluşumlardır. Bir grup değere tek ad altında başvurmak ve bu değerleri tek tek ya da bir arada işlemek için basic, pascal, C ve diğer çok bilinen programlama dillerinin ilk sürümlerinde diziler kullanılmıştır.Tek boyutlu dizi tanımlaması şu şekildedir.

Kullanımı:

Dim DiziAdı(Dim1Index1) As Veri Türü

Dizi bildirim ifadesindeki bildiriler	Açıklama
Dizi adı	Diziyi program içerisinde tanımlamak için kullanacağız dizi adları genellikle değişken adlarındaki kurallara uyar.
Veri türü	Dizide depolayacağınız verilerin türü. Çoğu durumda bir dizinin tüm değişkenleri aynı türden olacaktır. Temel veri türlerinden birini dizide hangi veri türüne depolayacağınızdan emin değilseniz veya birden çok depolayacaksanız, object türünü belirleyebilirsiniz.
Boyut adedi	Dizinizin içereceği boyut adedi. Çoğu diziler tek boyutlu ya da iki boyutludur. Ancak, üç boyutlu bir şekil gibi karmaşık bir matematiksel modelle çalışıyorsanız ek boyutlar belirleyebilirsiniz.
Öğe adedi	Dizinizin içereceği öğe adedi. Dizinizdeki öğeler doğrudan dizi öğesinin konumuna karşılıktır. Visual Basic.NET'te ilk dizin her zaman sıfırdır.

Tablo 3.1: Dizi tanımlama öğeleri

Not : Belirlenmiş sayıda öğe içeren dizilere, sabit büyüklükte diziler denir. Değişen sayıda öğeler içeren diziler, (programın çalışması sırasında genişleyebilen diziler) **Dinamik Diziler** olarak adlandırılır.

Dim, dizi bildirimini yapan anahtar sözcüktür. Dizi standart bir modül içerisinde kullanılacaksa Dim yerine Public deyimi kullanılmalıdır. **Dizi adı** dizi değişkeninin adıdır. **Dim1Index** dizinin ilk boyutunun üst sınırıdır. Yani, öğe sayısının bir eksiğidir. Virgülle ayrılmaları koşuluyla ek boyutlar içerilebilir. **Veri türü**, dizide içerilecek veri türüne karşılık gelen bir anahtar sözcüktür.

Mesela, Personel isimli **tek boyutlu** ve 10 elemanı bulunan, içerisinde isimlerin saklanacağı bir dizi bildirimi yapmak için şöyle bir tanımlama gerekir.

Dim Personel (9) As String
Aynı bildirim bir standart modülde aşağıdaki gibi olacaktır.

Public Ders (9) As String

Not : Bir dizinin ilk öğesinin indisi varsayılan olarak 0'dır.

Diziyi oluşturduğunuzda Visual Basic bellekte dizi için yer ayırır. Aşağıdaki şekil dizinin kavramsal anlamda nasıl düzenlendiğini göstermektedir. 10 dizi öğesi 0'dan başlayarak 9'a kadar numaralandırılmaktadır. Çünkü dizi indisleri her zaman 0 ile başlar.

Personel

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Tablo 3.2: Personel dizisinin elemanları

Çok boyutlu bir dizi tanımı ise şu şekilde yapılmalıdır.

Kullanımı: Dim DiziAdı(Dim1Index, Dim2Index, ...) As Veri Türü

Matris adlı iki boyutlu, iki satır ve dokuz sütunlu kısa tam sayı verileri için yeri olan bir dize dizisinin bildirimini yapmak için olay yordamdan ya da formun üst tarafına aşağıdakileri yazmanız gerekir.

Dim Matris (1,8) As Short

Not : İki boyutlu dizilerde iki dizine gerek vardır. Visual Basic.NET'te en fazla 60 boyutlu dizi tanımlayabiliriz.

İki boyutlu bir dizinin bildirimini yaptığınızda Visual Basic bellekte dizi için yer ayırır. Daha sonra diziyi aşağıda gösterildiği gibi programınızda sanki bir değerler tablosu gibi kullanabilirsiniz.

Matris

	sütunlar	0	1	2	3	4	5	6	7	8
satırlar	0									
	1									

Tablo 3.3: Matris dizisi

3.2. Dizinin Üst Sınır Değeri (0. Eleman Olarak Başlangıç)

Diziler artık her zaman 0 tabanlı olduklarından bildirimler artık "TO" anahtar sözcüğünü belirli alt ve üst sınırlarla kullanarak yapılamamaktadır. 0 tabanlı dizilerin bir diğer yan etkisi, bir dizinin alt sınırı her zaman 0 olduğundan LBound ifadesininde her zaman sıfır değerine döndürmesindir (Ancak, UBound ifadesinin bir dizideki en yüksek dizinin [öğe sayısı -1] döndürmesi sürmektedir).

3.3. İndeks Değeri, Toplam Eleman Sayısı

Dim **Personel** (9) As String

Üstte verilen bir dizi tanımlaması yardımıyla bir dizinin indeks değeri ve toplam eleman sayısını açıklamak istersek **Personel** dizisinde eleman indisi maksimum 9, toplam eleman sayısı da 10 olur. Çünkü bir dizinin indeksinin alt sınırı 0 (sıfır)'dır.

3.4. "Redim" Dizi Yeniden Boyutlandırma Komutu, "Preserve" ile Eski Değerleri Koruma

Bir dinamik dizinin boyutlandırılması birkaç adımda olur, çünkü dizi büyüklüğü program çalışana kadar yapılmamış olmasına rağmen, dizi için tasarım sırasında bazı "rezervasyonlar" yapmanız gerekir. Bir dinamik dizi oluşturmak için aşağıdaki adımlar izlenir.

- Dizi adını ve türünü, tasarım sırasında belirtiniz, dizideki öğe sayısını boş bırakınız. Örneğin, Sıcaklık adlı bir dinamik dizi oluşturmak için aşağıdakini yazınız.

  Dim Sicaklik( ) As Single

- Programın çalışması sırasında dizide olması gereken öğe sayısını belirlemek için kod ekleyiniz. Bir InputBox işlevi ya da metin kutusu nesnesi kullanarak bu bilgiyi kullanıcıdan isteyebilirsiniz ya da programın depolama gereksinimlerini özellikleri veya başka bir mantığı kullanarak hesaplayabilirsiniz. Örneğin, aşağıdaki ifadeler dizi büyüklüğünü kullanıcıdan alır ve günler adlı kısa tam sayı değişkenine atar.

  Dim Günler As Short
  Günler=InputBox ("Kaç gün?","Dizi Eleman değeri")

Diziyi yeniden boyutlandırmak için değişken, bir **ReDim** ifadesi içinde kullanılmalıdır. Diziler 0 tabanlı olduklarından 1 eksiği alınmalıdır. Örneğin, aşağıdaki ifade Sıcaklık dizisinin büyüklüğünü programın çalışması sırasında günler değişkenini kullanarak ayarlar.

ReDim Sicaklik (Günler-1)

ReDim için tek önemli nitelik daha önce bildirimi yapılan bir dizinin boyut sayısını değiştirmeye çalışmamanızdır. Üst sınırı belirlemek için UBound işlevini bir For...Next döngüsü içinde kullanınız ve gerekiyorsa dizi öğelerini işleyiniz:

```
For i=0 to UBound(Sıcaklık)
        Sıcaklık(i)=InputBox(Prompt, Title)
Next
```

Bir dizinin büyüklüğünü programın çalışması sırasında belirlemek için ReDim ifadesini kullanmıştık. Ancak, ReDim ifadesi ile ilişkili olası bir eksikliği bilmeniz gerekmektedir. İçinde veriler olan bir diziyi yeniden boyutlandırırsanız tüm veriler geri getirilemeyecek biçimde kaybedilecektir. ReDim ifadesi çalıştırıldıktan sonra bir dinamik dizinin içeriği 0 ya da Null olan varsayılan değere ayarlanır.

Bir dizinin boyutları değiştirilirken verileri korumakta kullandığınız **Preserve** anahtar sözcüğü kullanılır. **Preserve** anahtar sözcüğünün söz dizimi aşağıdaki gibidir.

ReDim **Preserve** DiziAdı(Dim1Öğeler, Dim2Öğeler....)

Böyle bir yeniden boyutlandırma ifadesinde dizinin aynı sayıda boyuta sahip olması ve aynı tür veriler içermesi gerekmektedir. Ek olarak, yalnız son boyutu değiştirebilmenizde ilgili özel bir uyarıya da gerek vardır. Örneğin, dizinizin iki ya da daha çok boyutu varsa yalnızca son boyutun büyüklüğünü değiştirebilir ve yine de dizinin içeriğini koruyabilirsiniz. Tek boyutlu diziler bu sınavı otomatik olarak geçer, böylece Preserve anahtar sözcüğünü kullanarak dinamik dizilerin büyüklüğünü özgürce artırabilirsiniz.

Aşağıdaki örnekler bir dinamik dizideki son boyutun büyüklüğünü dizide var olan verileri silmeden Preserve kullanarak nasıl artırabileceğinizi göstermektedir.

```
Dim Magaza( ) As string
```

Magaza adlı bir dizinin üstte verildiği şekilde bildirimini daha önce yaptıysanız aşağıdakine benzer bir kod kullanarak diziyi yeniden boyutlandırıp veri eklemesi yapabilirsiniz.

```
ReDim magaza ( 200 )
Magaza( 200 ) = “Erdal Mutlu”
```

Aşağıdaki söz dizimini kullanarak magaza dizisinin büyüklüğünü 301 öğeye (0-300) genişletebilir ve var olan içeriğini koruyabilirsiniz.

```
ReDim Preserve Magaza (300)
```

3.5. Diziye “{ }” ile İlk Değer Aktarma

Değişken tanımlamalarında olduğu gibi dizilerde de tanımlama sırasında ilk değerlerin verilmesi mümkündür. Örneğin aşağıdaki dizi tanımlamasında dizinin tüm değerleri tanımlama sırasında verilmektedir.

Dim isim(3) As String = { “Necati” , ” Kezban” , ”Zekeriya” , “Mahmut” }
isim dizisinin elemanlarına referans tanımlarken isim(0) ... isim(3) şeklinde olmalıdır.

3.6. İki Boyutlu Dizilerde Satır, Sütun kavramları (Tablo Yapmak)

Bir örnekle iki boyutlu dizilerde satır ve sütun kavramını açıklamak gerekirse aşağıdaki ifade matris (1,8) dizisi örneğinde 4 sayısını, 0.satır ve 2. sütuna yerleştirir.

matris(0,2) = 4

Bu da sonuclar diziminde aşağıdaki sonucu doğurur.

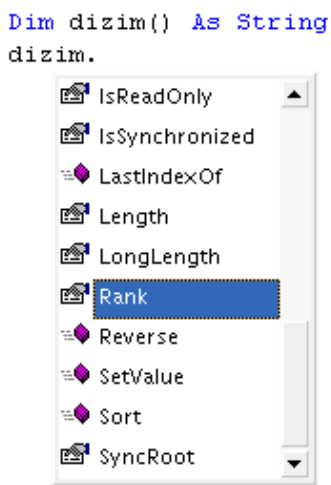
sütunlar

		0	1	2	3	4	5	6	7	8
satırlar	0			4						
	1									

Tablo 3.4: Matris dizisinin komuttan sonraki durumu

3.7. “Length, Rank, GetUpperBound, GetLength, BinarySearch, Sort” Komutları

Diziler array sınıfında tanımlanır, böylece array sınıfının tüm özellikleri diziler üzerinde kullanılabilir. Array sınıfının metotlarını kullanarak diziler üzerinde arama, sıralama gibi işlemleri gerçekleştirebiliriz. Bunları görmek için dizi adından sonra **noktaya “.”** basarak fonksiyonun özellikleri penceresine ulaşabilirsiniz.



Resim 3.1: Array fonksiyon özellikleri menüsü

3.7.1. Length

Dizinin eleman sayısını verir. Son elemana ulaşmak için dizim.length-1 biçiminde kullanılır.

3.7.2. Rank

Dizinin kaç boyutlu olduğunu verir.

3.7.3. GetUpperBound

Dizinin en büyük indeksli elemanının indeks bilgisini verir.

Örnek

Dim oyuncular(3) As String = { “Necati” , ” Kezban” , ”Zekeriya” , “Mahmut” }
```
Dim sayi As Integer = oyuncular.GetUpperBound(0)
```

3.7.4. GetLength

Dizinin belirtilen boyutundaki eleman sayısını verir.

3.7.5. BinarySearch

Dizide bir elemanı aramak için bu metodu kullanabiliriz. Aramanın yapılabilmesi için dizinin sıralanmış olması gerekir. Dizi sıralı değilse, önce **Sort** metodu ile sıralama yapabilir ve ardından BinarySearch metodu ile arama yapabiliriz.

Aranan eleman bulunursa, bulunduğu konum geri dönecektir. Mesela, **oyuncular** dizisinde **Mahmut** ismini aramak için;

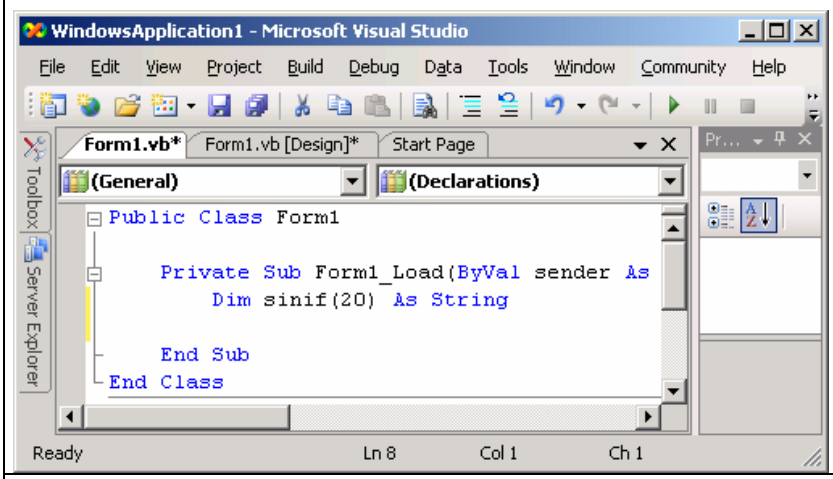
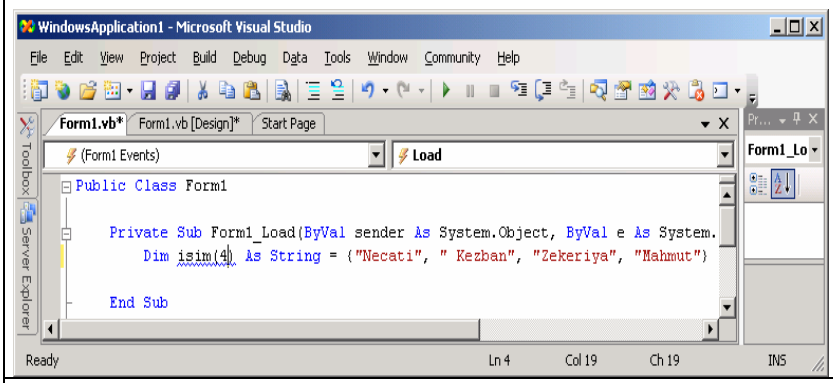
Konum= Array.BinarySearch(oyuncular, “kemal”) yazılır.

3.7.6. Sort

Diziyi sıralamak için bu metodu kullanırız. Örneğin **oyuncular** dizisini sıralamak için;

Array.Sort (oyuncular) yazılır.

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Diziyi oluşturmak için dizi Adi ( ) içine üst sınır sayısını yazınız. 	İçerisinde tam sayıların saklanacağı 50 elemanı bulunan **Rehber** dizisini tanımlayınız.
2. Dizinin türünü belirleyiniz.	
3. Diziye ilk değerlerini atayınız. 	Siz de 4 elemanı olan soyad dizinine kendinizin ve arkadaşlarınızın soyadlarını ilk değer olarak veriniz.
4. Diziye bir komutta birçok değer atayınız.	Sehirler={“Elazığ”, “Ankara”, “Malatya”, “İstanbul”} dizisinin ücüncü elemanı ile dizinin index değerini ve eleman sayısını ekrana yazdırınız.
5. Dizinin boyutlarını ReDim ile güncelleyiniz.	

ÖLÇME VE DEĞERLENDİRME

A- OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; ilk 8 soruda verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. Aynı tipte ve birbirleriyle ilgili bilgilerin oluşturduğu bütüne dizi denir. ( )

2. Değişen sayıda öğeler içeren diziler (programın çalışması sırasında genişleyebilen diziler) **Statik Diziler** olarak adlandırılır. ( )

3. Diziyi boyutlandırmak için değişken, bir **ReDim** ifadesi içinde kullanılır. ( )

4. Dizinin kaç boyutlu olduğunu bulmak için Sort komutu kullanılır. ( )

5. Diziye ilk değer aktarmak için " [ ] " işareti kullanılır. ( )

6. Dizi içerisindeki bir elemanı aramak için BinarySearch komutu kullanılır. ( )

7. Dim sinif (19) As String dizisinin toplam eleman sayısı 20'dir. ( )

8. Bir dizinin başlangıç index numarası 1'dir. ( )
Dim oyuncular(4) As String = { "Necati" , " Kezban" , "Zekeriya" , "Mahmut" }

9. Dim takim (4) As String ={"Fenerbahçe", "Galatasaray", "Beşiktaş", "Trabzon"} dizisinin takim(2) elemanı aşağıdakilerden hangisidir?
A) Trabzon
B) Fenerbahçe
C) Galatasaray
D) Beşiktaş

10. Dizinin kaç boyutlu olduğunu aşağıdaki komutlardan hangisi verir?
A) Sort
B) Rank
C) Getlength
D) BinarySearch

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konuları öğrenme faaliyete geri dönerek tekrar inceleyiniz.

ÖĞRENME FAALİYETİ–4

AMAÇ

Uygun ortam sağlandığında operatörler ile çalışabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Matematiksel işlemlerde kullanılan aritmetiksel operatörlerin işlem önceliklerini araştırınız.
- C programlama dilindeki birleştirme simgelerini (Unary Operator) araştırarak arkadaşlarınızla paylaşınız.

4. OPERATÖRLER

Visual Basic.NET programlama dilinde matematiksel ve lojik işlemleri yapmak için mevcut operatörler bulunur. Visual Basic.NET programında aritmetik işlemlerin bazıları operatörlerle yapılırken bazılarında ise fonksiyonlar kullanılır.

4.1. “=” ile Atama

Aditext= “ Meral ”

Üstteki kod satırında eşittir işareti (=)atama operatörü olarak kullanılır. Atama işlemi yapılırken rakamlar doğrudan fakat değerler ise “ ” arasında yapılmalıdır.

Atama operatörünün sağındaki sözcükler **çift tırnak** (**“ ”**) işareti arasındadır. Bu sözcükler bir karakter katarı (String) olarak adlandırılır. Bir karakter katarı iki ya da daha fazla karakter içerir. Karakterler bir harf, bir rakam, bir noktalama işareti veya bir boşluk karakteri içerebilir. Çift tırnak işareti bir karakter katarını belirtir.

Örnek

```
A=55
B=186
C=A+B
K=D/5
A$=”Kemal”
B$=”Ahmet”
C$=A$+B$
```

4.2. “+, -, *, /, \, MOD, ^” Aritmetik Simgeleri

4.2.1. “+” Operatörü

Bu operatör ile verilen iki veya daha fazla ifade toplanabilir. Genel yazılışı şöyledir.

Kullanımı:
Sonuc = Ifade1 + Ifade2

Burada Sonuc mutlaka sayısal bir değerdir.
Ifade1 çeşitli işlemlerden oluşmuş bir ifade veya bir sayıdır.
Ifade2 çeşitli işlemlerden oluşmuş bir ifade veya bir sayıdır.

Eğer Ifade1 ve Ifade2 string türünde veriler ise + operatörü kaynaştırma yapar. Yani Ifade2’ yi Ifade1’ın sonuna ekler.

Örnek

```
Sonuc=13+45                          'Sonuc=58
Sonuc=1378+56.78+435.908             'Sonuc=1870.688
A=89,B=3456
Deger=A+B                            ' Deger=3545
Ad= “Ebru”
Soyad= “Kayacı”
Dim Name As String
Name=Ad+Soyad                                  'Name=”Ebru Kayacı”
```

4.2.2. “ - ” Operatörü

Matematikte kullanılan çıkartma operatörüdür. Birinci ifadede verilen değerden ikinci ifadeyi çıkarır. Genel yazılışı şöyledir.

Kullanımı:
Sonuc = Ifade1 - Ifade2

Burada Sonuc mutlaka sayısal bir değerdir.
Ifade1 çeşitli işlemlerden oluşmuş bir ifade veya bir sayıdır.
Ifade2 çeşitli işlemlerden oluşmuş bir ifade veya bir sayıdır.

Örnek

```
Sonuc=3475.45-3445.90                ‘Sonuc=29.55
Deger=45-788-23                      ‘Deger=-766
Son=190, Ara=47
Son1=Son-Ara                         ‘Son=143
```

4.2.3. " * " Operatörü

Matematikteki çarpma operatörüdür. Verilen iki sayıyı çarpar. Genel yazılışı şöyledir.

Kullanımı:
Sonuc = Sayı1*Sayı2

Burada Sonuc mutlaka sayısal bir değerdir.
Sayı1 çeşitli işlemlerden oluşmuş bir ifade veya bir sayıdır.
Sayı2 çeşitli işlemlerden oluşmuş bir ifade veya bir sayıdır.

Örnek	
Sonuc= 34*9	'Sonuc=306
Deger= 87*12	'Deger=1044
A=5, B=56	
C=A*B	'C=280

4.2.4. " / " Operatörü

Matematikteki bölme operatörüdür. Verilen ilk sayıyı ikinci sayıya böler. Genel yazılışı şöyledir.

Kullanımı:
Sonuc = İfade1/İfade2

Burada Sonuc mutlaka sayısal bir değerdir.
İfade1 çeşitli işlemlerden oluşmuş bir ifade veya bir sayıdır.
İfade2 çeşitli işlemlerden oluşmuş bir ifade veya bir sayıdır. İfade2 sıfırdan farklı bir değer olmalıdır. Yoksa sıfıra bölme hatası oluşur.

Örnek:	
Sonuc= 34/9	'Sonuc=3.777778
Deger= 60/12	'Deger=5
A=5, B=56	
C=B/A	'C=11.2

4.2.5. "\ " Operatörü

Matematikteki bölme operatörüdür. Verilen ilk sayıyı ikinci sayıya böler. Ancak Sonuc mutlaka bir tam sayı değeridir. Bölüm ondalıklı ise sayının ondalık kısmı atılır. Genel yazılışı şöyledir.

Kullanımı:
Sonuc = İfade1\İfade2

Burada Sonuc mutlaka sayısal bir değerdir.

İfade1 çeşitli işlemlerden oluşmuş bir ifade veya bir sayıdır.
İfade2 çeşitli işlemlerden oluşmuş bir ifade veya bir sayıdır. İfade2 sıfırdan farklı bir değer olmalıdır. Yoksa sıfıra bölme hatası oluşur.

Örnek:

```
Sonuc= 34\9                    'Sonuc=3
Deger= 60\12                   'Deger=5
A=5, B=56
C=B\A                  'C=11
```

4.2.6. Mod Operatörü

Matematikteki mod alma operatörüdür. Verilen ilk sayının modunu ikinci sayıya göre alır. Genel yazım şekli aşağıdaki şekildeki gibidir.

Kullanımı:
Sonuc= İfade1 Mod İfade2

Burada Sonuc mutlaka sayısal bir değerdir.
İfade1 çeşitli işlemlerden oluşmuş bir ifade veya bir sayıdır.
İfade2 çeşitli işlemlerden oluşmuş bir ifade veya bir sayıdır.

Örnek

```
Sonuc= 34 Mod 9                'Sonuc=7
Deger= 60 Mod 2                'Deger=0
A=5.4 , B=57
C= B Mod A                     'C=2
D=57 Mod 5.5                   'D=3
E=90.5 Mod 6                   'E=0
E=90.2 Mod 6                   'E=0
E=90.7 Mod 6                   'E=1
```

4.2.7. "^ " Operatörü

Matematikteki üs alma operatörüdür. Verilen ilk sayının ikinci sayı kadar kuvvetini (üssünü) alır. Genel yazılışı şöyledir.

Kullanımı:
Sonuc= İfade1 ^ İfade2

Burada Sonuc mutlaka sayısal bir değerdir.
İfade1 çeşitli işlemlerden oluşmuş bir ifade veya bir sayıdır.
İfade2 çeşitli işlemlerden oluşmuş bir ifade veya bir sayıdır.

Örnek

```
Sonuc= 34 ^ 9                  'Sonuc=6.071699276646e+13
Deger= 60 ^ 2                  'Deger=3600
A=5 , B=5
C= B^A                         'C=3125
```

4.3. “( )” Kullanımı

Bir formülde öncelik sırasını belirlemek için **( )** operatörü kullanılır.

Sayı = (8-5*3)^2 atama işleminde, üs alma öncelik sırası çıkarma ve çarpmadan daha yüksek olmasına rağmen, üs almadan önce ayraçlar arasındaki değeri (-7) belirleyerek hesaplar. Bir formülde iç içe ayraçlar kullanarak işlem sıralarını değiştirebiliriz.

Sayı = ((8-5)*3)^2 ifadesinde Visual Basic’i ilk olarak içteki ayraç kümelerini hesaplamaya, sonra dıştaki ayraçlar içindeki işlemi yapmaya ve daha sonra da üs almaya yönlendirir. Yukarıdaki iki formülün sonuçları farklıdır. İlk formülün sonucu 49, ikincisinin ki 81’dir. Ayraçlar bir matematiksel işlemin sonucunu değiştirebildiği gibi kolay okunuş da sağlar.

4.4. “+=, -=, *=, /=, \=, ^=” Birleştirme Simgeleri (Unary Operatör)

Atama operatörü en temel operatördür denilebilir. Atama işlemi, bir değeri veya değişkenin içeriğini bir başka değişkene yerleştirmektir. Hemen hemen tüm programlama dillerinde atama operatörü olarak “=” simgesi kullanılır.

Visual Basic.NET programlama dilinde atama operatörleri de vardır. Bunlar atama operatörüyle diğer operatörlerden birinin birleştirilmesinden oluşur. Böylece kısa bir yazılımla hem aritmetik, öteleme gibi işlemler yaptırılır hem de atama yapılır. Yani, ifade yazımı kolaylaştırır. Örneğin, int tipinde olan toplam değişkenin değerini 1 artırmak için aşağıda ki gibi bir ifade kullanılabilir:

```
toplam = toplam + 1 ;
```

Bu ifade bitişik atama operatörüyle aşağıdaki gibi yazılabilir. Görüldüğü gibi değişken adı yukarıdaki yazımda 2, aşağıdaki yazımda ise 1 kez kullanılmıştır.

```
toplam += 1;
```

Tablo 4.1’de bitişik atama operatörlerinin listesi görülmektedir. Bu operatörler, özellikle uzun değişken kullanıldığı durumlarda yazım kolaylığı sağlar.

Operatör	Kullanım Şekli	Eşittir
+=	değişken1 += değişken2	değişken1 = değişken1 + değişken2
-=	değişken1-= değişken2	değişken1 = değişken1 – değişken2
*=	değişken1*= değişken2	değişken1 = değişken1 * değişken2
/=	değişken1/= değişken2	değişken1 = değişken1 / değişken2
\=	değişken1\= değişken2	değişken1 = değişken1 \ değişken2
^=	değişken1^= değişken2	değişken1 = değişken1 ^ değişken2

Tablo 4.1: Visual Basic.NET bitişik atama operatörleri

4.5. "Casting" Farklı Türlerin Birbirine Dönüştürülmesi

Visual Basic.NET'te zaman zaman herhangi bir veri tipinde saklanan değeri farklı bir veri tipine dönüştürme ihtiyacı duyarız. Bu işlemi yapan fonksiyonlara tip dönüşüm fonksiyonları adı verilir.

```
Private Sub Command1_Click()
    TextBox1.Text = 6
    TextBox2.Text = 10
    Label1.Caption = TextBox1.Text + TextBox2.Text
End Sub
```

Bu kodu çalıştırdığımızda Label1 içerisinde 610 değeri yazacaktır. Visual Basic.NET her iki metin kutusu içerisinde bulunan değerlerin birer metin olduğunu varsayarak iki metni de birleştirme işlemi yaptı. Eğer bu değerlerin toplanmasını istiyorsak tip dönüşümlerini kullanarak string tipindeki verileri integere çevirmemiz gerekirdi.

```
Private Sub Command1_Click()
    TextBox1.Text = 6
    TextBox2.Text = 10
    Label1.Caption = CInt(TextBox1.Text) + CInt(TextBox2.Text)
End Sub
```

Yukarıdaki kodda TextBox1 ve TextBox2 içeriği önce CInt adlı fonksiyonla tam sayıya çevrildi ve ardından toplama işlemi yapıldı. Label1'in içeriği de 16 olarak değişti.

Aşağıda Visual Basic'de kullanılan tip dönüşüm fonksiyonları verilmiştir.

Fonksiyon	Geri Dönen Değer	Yaptığı İşlem
CBool(Değer)	Boolean	Matematiksel ifadeyi Boolean türüne dönüştürür.
CByte(Değer)	Byte	Matematiksel ifadeyi Byte türüne dönüştürür.
CCur(Değer)	Currency	Matematiksel ifadeyi Currency türüne dönüştürür.
CDate(Değer)	Date	Matematiksel ifadeyi Date türüne dönüştürür.
CDbl(Değer)	Double	Matematiksel ifadeyi Double türüne dönüştürür.
CDec(Değer)	Decimal	Matematiksel ifadeyi Decimal sayıya dönüştürür.
CInt(Değer)	Integer	Matematiksel ifadeyi tam sayıya dönüştürür.
CLng(Değer)	Long	Matematiksel ifadeyi Long türüne dönüştürür.
CSng(Değer)	Single	Matematiksel ifadeyi Single türüne dönüştürür.
CVar(Değer)	Variant	Matematiksel ifadeyi Variant türüne dönüştürür.
CStr(Değer)	String	Matematiksel ifadeyi String türüne dönüştürür.

Tablo 4.2: Visual Basic.NET programında kullanılan tip dönüşüm fonksiyonları

Örnek

A=10 , B=5 , C=10 , D=0

```
Sonuc= CBool(A < B)          'Sonuc =False
Sonuc= CBool(A > B)          'Sonuc =True
Sonuc= CBool(A = C)          'Sonuc =True
```

Örnek

A=10 , B=5 , C=0

```
Sonuc= CByte(A < B)          'Sonuc =0
Sonuc= CByte(A > B)          'Sonuc =255
Sonuc= CByte(A = C)          'Sonuc =255
```

Örnek

A=1 , B=2 , C=36000 , D=36001

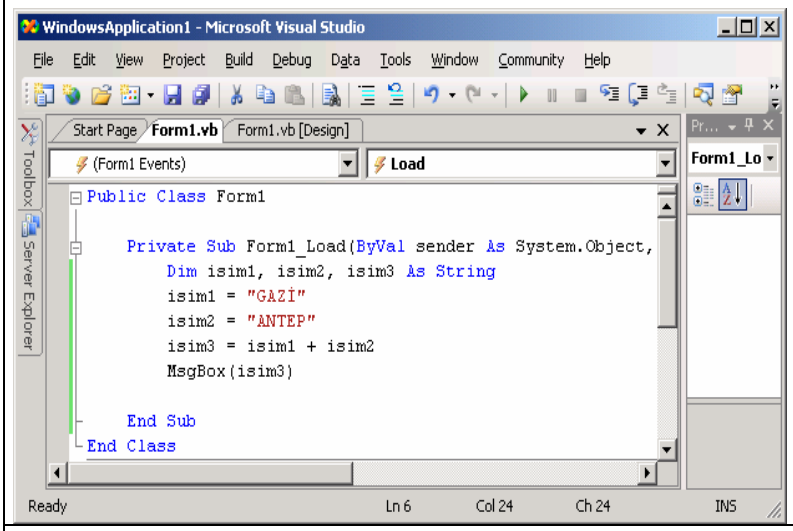
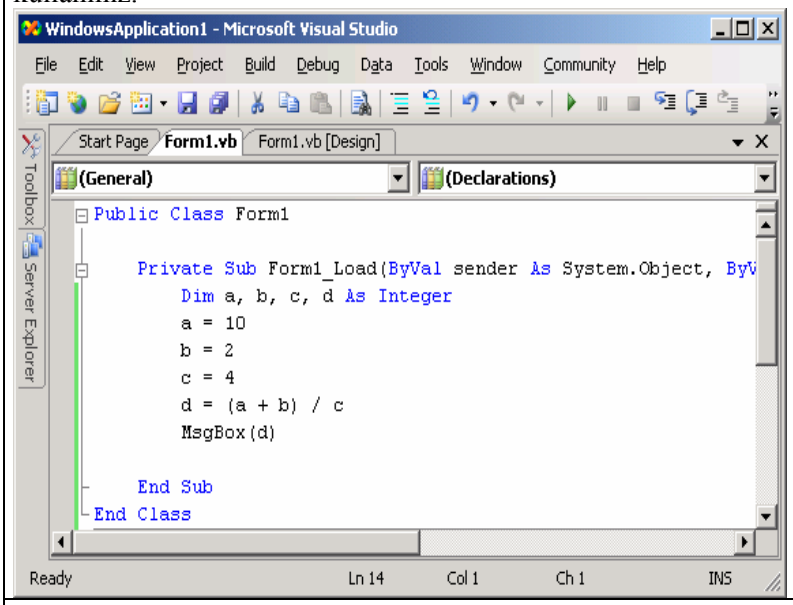
```
Sonuc= CDate(A)              'Sonuc =12/31/1899
Sonuc= CDate(B)              'Sonuc =1/1/1900
Sonuc= CDate(C)              'Sonuc =7/24/98
Sonuc= CDate(D)              'Sonuc =7/25/98
```

Örnek

A=2.4 , B=2.5 , C=2.6 , D=3.5

```
Sonuc= CInt(A)               'Sonuc =2
Sonuc= CInt(B)               'Sonuc =3
Sonuc= CInt(C)               'Sonuc =3
Sonuc= CInt(D)               'Sonuc =4
```

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Bir değişkenin üzerinde " = " kullanarak yeni bir değer atayınız.  	Ad ve soyad değişkenlerini tanımlayınız, kendi isim ve soy isimlerinizi bu değişkenlere atayarak, iki değişkeni toplayıp sonucu görüntüleyiniz.
2. Yeni değerin hesaplanmasında matematiksel işlem simgeleri kullanınız.  	Sayı1=20 Sayı2=80 Sayı3=5 Sayı4= (Sayı1+Sayı2)+Sayı3^2 işleminin sonucunu bulunuz.
3. Elde edilen yeni değerin ekran çıktısını gösteriniz.	

ÖLÇME VE DEĞERLENDİRME

A- OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; ilk 8 soruda verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. ^ operatörü bir sayının belli bir kuvvetini almayı sağlar. ( )

2. / operatörü ile \ operatörü arasında hiçbir fark yoktur. ( )

3. + operatörü hem aritmetik işlemlerde hem de karakter birleştirmelerde işlem görür. ( )

4. Matematiksel bir ifadede sonucu hesaplarken işlem önceliği yoksa soldan başlayıp sağa doğru devam edilir. ( )

5. Bölme işlemi sonucunda kalanı veren # operatörüdür. ( )

6. İşlemlerde aritmetiksel operatörler atama operatörlerinden daha önce yapılır. ( )

7. Matematiksel ifadeyi Boolean türüne dönüştürmek için "CBool" fonksiyonu kullanılır. ( )

8. # operatörü bir değişken birleştirme operatörüdür.( )

9. Varsayılan operatör önceliğini nasıl öncelikli kılarsınız?
A) +
B) #
C) ( )
D) Mod

10. Hangi mantıksal operatör iki yerine tek bir operand üzerinde işlem yapar?
A) OR
B) NOT
C) AND
D) ORELSE

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konuları öğrenme faaliyete geri dönerek tekrar inceleyiniz.

ÖĞRENME FAALİYETİ–5

AMAÇ

Metin ve tarih veri türleri ile çalışabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Karakter ve String karakterlerini öğrenerek arkadaşlarınızla paylaşınız.
- Birden fazla String ifadesine birleştirmek için programlamada kullanılan komut ve operatörleri diğer programlama dillerindeki operatörlerle karşılaştırınız.

5. METİN VE TARİH VERİ TÜRLERİ

5.1. "String" Veri Tipi

Metin türü bilgileri saklamak için kullanılabilecek veri türüdür. Hafızada içerisinde bulunan karakter kadar yer kaplar. $ işareti ile de tanımlanabilir.

```
Dim Ad As String                Dim Ad$
Ad= "Uğur ŞAHİN"                Ad= "Uğur ŞAHİN"
```

String türü değişkenlere sabit bir uzunlukta yer ayırmak istersek aşağıdaki şekilde bir tanımlama yapmalıyız.

Kullanımı:

```
Dim Ad As String *12
```

5.2. "" Kullanarak Atama Yapma

String türündeki karakter değişkenlerine atama yapmak için kullanılır. Öreneğin;

```
Dim sehir As String
sehir= "ELAZIĞ"
```

5.3. “Date” Türüne “#” Karakterleri Arasında Değer Aktarma

Bu tipteki değişkenlere atama bir string gibi “# #” karakterleri arasında da yapılabilir. Aslında bu değişken ondalık sayı tipinden tanımlandığı için bu tipte tanımlanmış tarihler arasında işlem yapılabilir. Örneğin iki tarih arasındaki gün sayısını bulmak için çıkarma işlemi yapılabilir.

```
Dim tarih As date
tarih = “17/11/2006”
```

Object olarak tanımlanmış bir değişkene “#” karakterleri arasında değer aktarılabilir.

```
Dim DogumGunu
DogumGunu = #21-4-1974#
```

5.4. Tarih ve Saat Biçimleri

8 byte kaplayan bu değişkene 1/1/100 ile 31/12/9999 arasındaki tarih ve 0:00:00 ile 23:59:59 arasındaki saat atamaları yapılabilir.

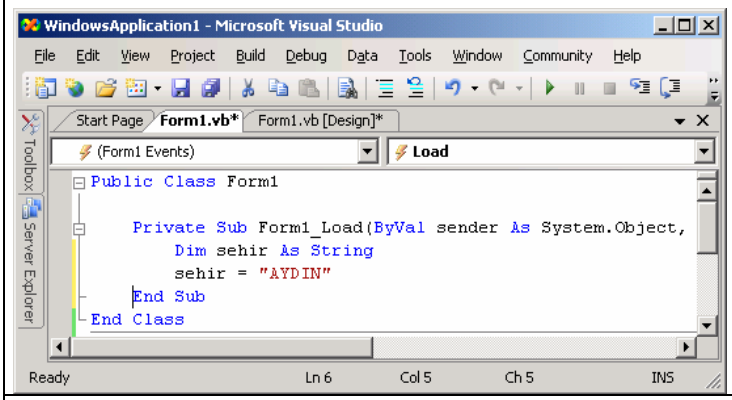
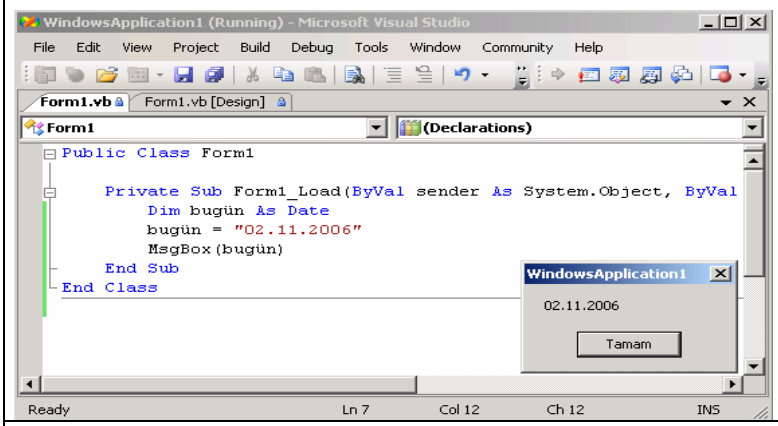
5.5. Metinleri “&” veya “&=” ile Birleştirme

String operatörleri birleştirmek için kullanılır.

Kullanımı:
Str3=Str1&Str2
& operatöründe birleştirilecek ifadenin String olması şart değildir.

```
Dim Str1,Str2 As String
Dim Sayi As Integer
Str1= “Özlem”
Str2= “Banu”
Sayi= 5
MsgBox(Str1& “ ”&Str2&Sayi)                    ‘Sonuç Özlem Banu5
```

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Metin türünde değişken tanımlayınız. 	Siz de yaşadığınız ile sehir değişkenine atayınız.
2. Metin değişkene değer aktarınız.	
3. Metin veya tarih türündeki değişkenler üzerinde işlemler yapınız. 	Siz de çalıştığınız günün tarihini bugün değişkenine atayarak çalıştırınız.
4. A$= “Bugün”, B$= “hava”, C$= “çok”, A$= “güzel” değişkenlerini birleştirerek sonucu ekrana yazdırınız.	

ÖLÇME VE DEĞERLENDİRME

A- OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; ilk 5 soruda verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. & işareti ile bir string tanımlanır. ( )

2. Ders string değişkenine matematik ifadesi aktarmak için ders=matematik yazılır. ( )

3. + operatörü kullanarak string değişkenlerini birleştirebiliriz. ( )

4. Tarih biçimindeki değişkenler 1/1/100 ile 31/12/9999 arasında değer alır. ( )

5. Object olarak tanımlanmış bir değişkene "#" karakterleri arasında değer aktarılabilir. ( )

6. Aşağıdakilerden hangisi Stringleri birleştirme operatörlerindendir?
A) *
B) %
C) =
D) &

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konuları öğrenme faaliyete geri dönerek tekrar inceleyiniz.

ÖĞRENME FAALİYETİ–6

AMAÇ

Uygun ortam sağlandığında şartlı deyimler ile çalışabileceksiniz.

ARAŞTIRMA

Bu faaliyet öncesinde hazırlık amaçlı aşağıda belirtilen araştırma faaliyetlerini yapmalısınız.

- Visual Basic 6.0 programalama dilinde kullanılan şart deyimlerini araştırınız.
- Birden fazla şart verilmek istendiğinde yapılması gereken işlemleri araştırınız.
- Programlama dillerinde kullanılan operatörleri araştırınız.

6. ŞARTLI DEYİMLER

Şart cümleleri basit anlamıyla bir ya da daha fazla değerin ilişkisel ve lojik operasyonlara tabi tutulduğu ifadelerdir. Belirli bir şartın doğru olup olmadığını araştıran bu ifadelerin sonucu 1 (TRUE) veya 0 (FALSE) değerlerini alabilir. 1 (TRUE) sonucu araştırılan şartın doğru olduğunu, 0 (FALSE) sonucu ise araştırılan şartın yanlış olduğunu gösterir. Şart cümlelerini bir nevi operand olarak kullanan komut cümleleri (IF ve SELECT CASE) elde edilen sonucun 1 (TRUE) veya 0 (FALSE) olmasına bağlı olarak belirgin program parçalarının icra edilip edilmemesine sebep olur.

6.1. "Şart, True ve False" Deyimleri

Bazı ifade veya ifadelerin yerine getirilebilmesi belirli şart veya şartlar grubuna bağlı olabilir. Bu durumda, program içerisinde bu tür ifadeleri işlemeden önce gerekli şartların kontrol edilmesi zorunludur. İlişkilendirmeler ya da mantıksal operatörler kullanılarak şartın DOĞRU veya YANLIŞ olup olmadığı kontrol edilir.

Bir olay yordamında bilgi işlemek için en kullanışlı araçlardan biri koşullu deyimdir. Bir koşullu deyim program kodu içine bir özellik, bir değişken ya da bir başka veri parçası hakkında True veya False sorusu soran tam bir program ifadesi parçasıdır. Örneğin;

Sayi < 100

Koşullu deyimi, Sayi değişkeni 100'den küçük bir değer içeriyorsa True, 100'e eşit ya da ondan büyük bir değer içeriyorsa False olarak değerlendirilir. Bu sonuca göre ifadeler icra edilir veya edilmez. VB.NET'te temel olarak iki çeşit şart kontrol ifadesi kullanılır. Bunlar;

1. If yapıları
2. Select-Case yapısı

6.2. If ve Select-Case Komutları

Kontrol komutları programcılar tarafından sıkça kullanılan belirli ifadeleri kontrol etmek veya bazı şartların gerçekleşip gerçekleşmediğini denetlemek amacıyla kullanılan komutlardır.

6.2.1. IF Yapısı

Programın akışı IF deyimi ile birlikte verilen koşula bağlı olarak çalışır. Program akışı, bu komutlar sayesinde ELSE, ELSEIF veya END IF deyimleri ile oluşturulan işlem bloğunun çalışması veya söz konusu program bloğunun işletilmeyip atlatılması sağlanır.

Kullanımı:

```
IF Şart THEN
        Komutlar
ELSE
        Komutlar
END IF
```

Şartın gerçekleşmesi durumunda THEN deyiminden sonraki satır işletilir. Gerçekleşmemesi durumunda ise ELSE deyiminden sonraki satırlar işletilir. Tek satırda şart yazılırsa END IF deyimi kullanılmaz.

Kullanımı:

```
IF Şart Komutlar THEN Komutlar
```

Örnek Girilen üç notun ortalamasını alıp, bu notların ortalamasına göre öğrencinin geçtiğini veya kaldığını yazan programı yazalım.

```
Private Sub Form_Load()
Dim n1,n2,n3,ort
n1=val(InputBox("1.Sınav Notunu Giriniz:", "Sınav");
n2=val(InputBox("2.Sınav Notunu Giriniz:", "Sınav");
n3=val(InputBox("3.Sınav Notunu Giriniz:", "Sınav");
ort=(n1+n2+n3)/3
IF (ort<50) Then
MsgBox("Kaldınız" &ort)
Else
MsgBox("Geçtiniz" &ort)
END IF
End Sub
```

6.2.2. Select Case Yapısı

İşlev bakımından IF deyimine çok benzemektedir. Çok sayıda IF yapısı içi içe kullanıldığı zaman programın okunurluğu azalır ve programı izlemek zorlaşır. Bu gibi durumlarda Select Case yapısı kullanılır.

Kullanımı:

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

Genel yazılımdan anlaşılacağı gibi bloğu başlatan Select Case deyiminden sonra yapılacak karşılaştırmalarda kullanılacak bir kontrol değişkeni bulunmaktadır. Eğer kontrol değişkeninin içeriği "ifade1" olarak verilen değerle aynı ise, birinci ifadenin içeriğini araştıran Case deyiminden bir sonraki Case deyimine kadar olan program satırları işletilir ve program akışı End Select deyiminden sonraki satıra geçer.

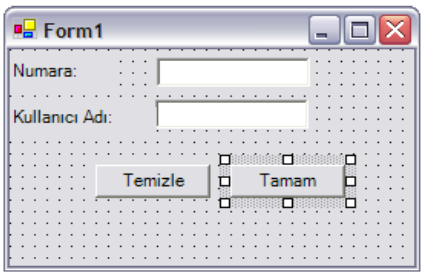
Örnek Yukarıdaki örneği Select Case yapısını kullanarak yapalım.

```
Private Sub Form_Load()
Dim n1,n2,n3,ort
  n1=val(InputBox("1.Sınav Notunu Giriniz:", "Sınav");
  n2=val(InputBox("2.Sınav Notunu Giriniz:", "Sınav");
  n3=val(InputBox("3.Sınav Notunu Giriniz:", "Sınav");
  ort=(n1+n2+n3)/3
        Select Case ort
Case  ort<50
                MsgBox("Kaldınız" &ort)
 Case ort>50
                MsgBox("Geçtiniz" &ort)
  End Select
End Sub
```

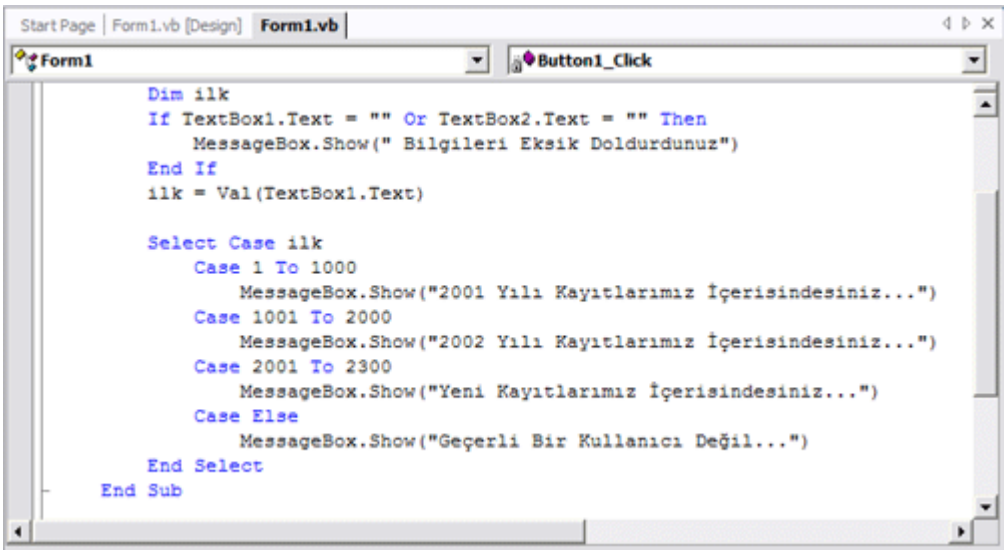
Örnek: 1-3 arasında girilen sayıyı bulan ve ekrana mesaj yazan programı yapalım.

```
Private Sub Form_Load()
Dim sayi
sayi=val(InputBox("1 ile 3 arasında bir sayı giriniz.");
Select Case sayi
        Case 1
                MsgBox("Girdiğiniz Sayı 1")
        Case 2
                MsgBox("Girdiğiniz Sayı 2" )
        Case 3
                MsgBox("Girdiğiniz Sayı 3" )
End Select
End Sub
```

Örnek Bu örnekte formdan girilen iki textbox'ın içinde veri olup olmadığını kontrol eden (if deyimi kullanarak) eğer varsa verinin hangi aralıkta bulunduğuna göre işlemler dizisini (kullanıcıya mesaj verme işlemi) aktif eden bir select case yapısını anlatmaktayız.



Resim 6.1: Form hazırlama ekranı



Resim 6.2: Kod yazma penceresi

6.3. =, <>, <=, >=, < ve > İlişkisel Operatörleri

Bu operatörler ile verilen ifadeler arasında karşılaştırmalar yapılır. Genel karşılaştırma operatörleri şunlardır.

6.3.1. "=" Operatörü

Bu operatör verilen iki ifadenin eşit olup olmadığını anlamak için kullanılır.

Kullanımı:
Sonuc=(İfade1 = İfade2)

Burada Sonuc Boolean tipinde bir değerdir.
İfade1 bir sayı veya bir metindir.
İfade2 bir sayı veya bir metindir.

Örnek

```
Sonuc=(100=345)                    'Sonuc=False
Sonuc=(100=100)                    'Sonuc=True
A=12,B=45
Sonuc=(A=B)                        'Sonuc=False
C=23,D=23
Sonuc=(A=D)                        'Sonuc=True
```

6.3.2. "< >" Operatörü

Bu operatör verilen iki ifadenin farklı olup olmadığını anlamak için kullanılır.

Kullanımı:
Sonuc=(İfade1 <>İfade2)

Burada Sonuç Boolean tipinde bir değerdir.
İfade1 bir sayı veya bir metindir.
İfade2 bir sayı veya bir metindir.

Örnek

```
Sonuc=(100<>345)                   'Sonuc=True
Sonuc=(100<>100)                   'Sonuc=False
A=12,B=45
Sonuc=(A<>B)                       'Sonuc=True
C=23,D=23
Sonuc=(A<>D)                       'Sonuc=False
```

6.3.3. "<" Operatörü

Bu operatör verilen birinci ifadenin ikinci ifadeden küçük olup olmadığını anlamak için kullanılır.

Kullanımı:
Sonuc=(İfade1 < İfade2)

Burada Sonuc Boolean tipinde bir değerdir.
İfade1 bir sayı veya bir metindir.
İfade2 bir sayı veya bir metindir.

Örnek

```
Sonuc=(100<345)                    'Sonuc=True
Sonuc=(100<100)                    'Sonuc=False
Sonuc=(100<130)                    'Sonuc=False
A=12,B=45
Sonuc=(A<B)                        'Sonuc=True
C=23,D=23
Sonuc=(A<D)                        'Sonuc=False
```

6.3.4. “>” Operatörü

Bu operatör verilen birinci ifadenin ikinci ifadeden büyük olup olmadığını anlamak için kullanılır.

Kullanımı:

Sonuc=(İfade1 > İfade2)

Burada Sonuc Boolean tipinde bir değerdir.
İfade1 bir sayı veya bir metindir.
İfade2 bir sayı veya bir metindir.

Örnek

```
Sonuc=(800>345)          'Sonuc=True
Sonuc=(100>100)          'Sonuc=False
Sonuc=(100>130)          'Sonuc=False
A=90,B=45
Sonuc=(A>B)              'Sonuc=True
C=23,D=23
Sonuc=(A>D)              'Sonuc=False
```

6.3.5. “>=” Operatörü

Bu operatör verilen birinci ifadenin ikinci ifadeden büyük veya eşit olup olmadığını anlamak için kullanılır.

Kullanımı:

Sonuc=(İfade1 >= İfade2)
Burada Sonuc Boolean tipinde bir değerdir.
İfade1 bir sayı veya bir metindir.
İfade2 bir sayı veya bir metindir.

Örnek :

```
Sonuc=(800>=345)         'Sonuc=True
Sonuc=(100>=100)         'Sonuc=True
Sonuc=(100>=130)         'Sonuc=False
A=90,B=45
Sonuc=(A>=B)             'Sonuc=True
C=23,D=23
Sonuc=(A>=D)             'Sonuc=True
```

6.3.6. “<=” Operatörü

Bu operatör verilen birinci ifadenin ikinci ifadeden küçük veya eşit olup olmadığını anlamak için kullanılır.

Kullanımı:
Sonuc=(İfade1 <= İfade2)

Burada Sonuc Boolean tipinde bir değerdir.
İfade1 bir sayı veya bir metindir.
İfade2 bir sayı veya bir metindir.

```
Örnek
Sonuc=(800<=345)                    'Sonuc=False
Sonuc=(100<=100)                    'Sonuc=True
Sonuc=(100<=130)                    'Sonuc=True
A=90,B=45
Sonuc=(A<=B)                        'Sonuc=False
C=23,D=23
Sonuc=(A<=D)                        'Sonuc=True
```

6.4. “And, Or, Not, Xor, AndAlso, OrElse” İlişkisel Operatörleri

6.4.1. “And ” Operatörü

AND operatörü tüm şartların doğru olduğu anlarda -1 değerini döndürür, diğer durumlarda ise 0 değerini döndürür.

```
sayı1 = 4
sayı2 = 5
Sonuc = (sayı1 = 4) AND (sayı2 = 5)      ‘Sonuc = -1
Sonuc = (sayı1 = 4) AND (sayı2 = 8)      ‘Sonuc = 0
```

6.4.2. “Or ” Operatörü

OR operatörü, tüm şartların yanlış olduğu anlarda 0 değerini döndürür, diğer durumlarda ise -1 değerini döndürür.

```
sayı1 = 4
sayı2 = 5
Sonuc = (sayı1 = 2) OR  (sayı2 = 5)      ‘Sonuc = -1
Sonuc = (sayı1 = 4) OR (sayı2 = 5)       ‘Sonuc = -1
Sonuc = (sayı1 = 2) OR (sayı2 = 4)       ‘Sonuc = 0
```

6.4.3. “Xor ” Operatörü

XOR operatörü, şartlardan sadece birinin doğru olduğu anlarda -1 değerini döndürür, diğer durumlarda ise 0 değerini döndürür.

```
sayı1 = 4
sayı2 = 5
Sonuc = (sayı1 = 2) XOR (sayı2 = 5)        ‘Sonuc = -1
Sonuc = (sayı1 = 4) XOR (sayı2 = 5)        ‘Sonuc = 0
```

6.4.4. “ Not ” Operatörü

NOT operatörü çıkan sonucu tam tersine çevirir. Sonuç –1 çıkarsa 0’a, 0 çıkarsa –1’e çevirir.

```
sayı1 = 4
sayı2 = 5
Sonuc = sayi1=sayi2                        ‘Sonuc = 0
Sonuc = NOT(sayi1=sayi2)                   ‘Sonuc = 0
```

6.4.5. “AndAlso ” Operatörü

And deyimi ile bir kontrol işleminde iki tanımlamanın da sağlanıp sağlanmadığını kontrol ederiz. Mesela adı Ali, soyadı Veli olan kişi ile ilgili işlemler yapmamız gerekirse bunu bir if kontrolü ile sorgulayabiliriz.

```
If ad = "Ali" And soyad = "Veli" Then
' işlemler
End If
```

Burada Visual Basic önce ad değişkeninin Ali olup olmadığını kontrol eder. Eşitliğin doğrulandığına veya doğrulanmadığına aldırmadan soyadı değişkeninde doğru olup olmadığını kontrol eder ama eğer and yerine andalso kullansaydık, ad değişkeninin değerinin Ali olup olmadığına bakar. Olmadığına göre ikinci bölümü kontrol bile etmez!

AndAlso yapısını iç içe yazılacak iki if kontrolüne benzetebilirsiniz. Bu sadece iki if kontrollü yapının kısayolu sayılabilir.

```
If ad = "Ali" AndAlso soyad = "Veli" Then
'işlemler
End If
```

6.4.6. “OrElse ” Operatörü

Or deyiminin birinci veya ikinci kontrollerden en azından birisinin doğru değerini dönmesini gerektirdiğini biliyoruz. OrElse ise AndAlso’nun çalışmasına benzemektedir. OrElse deyimi şu şekilde işler. Eğer birinci kontrol bölümü zaten doğru dönüyor ise ikinciyi kontrol etmez, çünkü en azından birisinin doğrulanması gerekmektedir.

6.5. Mantıksal İşlemlerin Çalışma Yönü

Lojik iki ifadenin karşılaştırılması esnasında kullanılan mantıksal operatörler arasında da işlem önceliği söz konusudur. Mantıksal operatörlerin dereceleri karşılaştırma operatörlerinin derecelerinden daha düşüktür. Tablo 6.1'de, en yüksekten en düşüğe doğru mantıksal operatörler arasında öncelik sırası listelenmektedir.

Öncelik	Operatör	Açıklama
1	Not	Negatif Değer
2	And, AndAlso	Birleşme
3	Or, OrElse	Ayrılma
4	XOR	Dışlama

Tablo 6.1: Mantıksal operatörlerin işlem öncelik sırası

Eşit öncelikli mantıksal operatörler aynı ifade içerisinde yer alırsa, aralarındaki öncelik soldan sağa doğru işlenir.

6.6. "Boolean" Değişkenlere Atama Yapma

True ya da False olarak değerlendirilebilen deyimler Boolean deyimler olarak da bilinir. True veya False değerleri bir Boolean değişkenine atanabilir.

Kullanımı:
Dim a as Boolean

6.7. "Is ve To" Anahtar Kelimeleri

Select Case şartlı cümlelerde şart kriterlerini belirtmek için Is ve To komutlarından faydalanarak şart kriterleri esnekleştirilir.

Is : Karşılştırma yaparken
To : Bir aralık belirtirken

```
Örnek
Dim sayi as integer
        select case sayi

            case is >10                 'ondan büyükse
            case is <5                  'beşten küçükse
            case 15 to 20               '15 ile 20 arasıysa
            case else                   'geriye kalan diğer durumlar

        end select
```

UYGULAMA FAALİYETİ

İşlem Basamakları	Öneriler
1. Matematik dersinde almış olduğunuz 3 yazılı ortalamasını hesaplayan ekrana "GEÇTİ", "KALDI" yazdırınız. Programı kaydederek öğretmeninize teslim ediniz.	Bu programı yazarken IF kalıbını kullanınız.
2. Üstte yaptığınız programı Fizik dersi için de yapınız.	
3. Not ifadesi 0-24 arasında ise F, 25-44 arasında ise E, 45-54 arasında ise D, 55-69 arasında ise C, 70-84 arasında ise B, 85-100 arasında ise A yazdırınız.	
4. IF komutu çok satırlı ise "End If" ile bitiriniz.	
5. 3. uygulamayı Select Case yapısı kullanarak hazırlayınız.	Bu programı yazarken SELECT CASE kalıbını kullanınız.

ÖLÇME VE DEĞERLENDİRME

A- OBJEKTİF TESTLER (ÖLÇME SORULARI)

Aşağıdaki sorulardan; ilk 6 soruda verilen ifadeye göre parantez içine doğru ise "D", yanlış ise "Y" yazınız. Diğer sorular için uygun şıkkı işaretleyiniz.

1. Verilen bir şartın doğru olması durumunda değişken TRUE değerini alır. ( )

2. IF komutu çok satırlı ise "End" ile bitirilir. ( )

3. İki ifadenin eşit olup olmadığını anlamak için (=) operatörü kullanılır. ( )

4. Deger=(45<32) işleminin sonucu TRUE değerini alır. ( )

5. Birleştirme operatörü olarak da bilinen operatör OR'dur . ( )

6. Select Case yapısında bir aralık belirtirken TO ifadesi kullanılır. ( )

7. Girilen bütün şartların doğru olması durumunda ancak sonucun doğru olduğu operatör aşağıdakilerden hangisidir?
A) AND
B) XOR
C) OR
D) ORELSE

8. Verilen iki ifadenin farklı olup olmadığını denetleyen operatör hangisidir?
A) <
B) >
C) <>
D) =

DEĞERLENDİRME

Cevaplarınızı cevap anahtarı ile karşılaştırınız. Doğru cevap sayınızı belirleyerek kendinizi değerlendiriniz. Yanlış cevap verdiğiniz ya da cevap verirken tereddüt yaşadığınız sorularla ilgili konuları öğrenme faaliyete geri dönerek tekrar inceleyiniz.

MODÜL DEĞERLENDİRME

PERFORMANS TESTİ (YETERLİK ÖLÇME)

Modül ile kazandığınız yeterliği aşağıdaki kriterlere göre değerlendiriniz.

DEĞERLENDİRME ÖLÇÜTLERİ	Evet	Hayır
Program yazarken oluşan yazım hatalarını düzelttiniz mi?		
Benzer kod bloklar ile diğer kodlar arasında boşluk bıraktınız mı?		
Değişken tanımlarını programın başlangıç kısmına yazdınız mı?		
Uzun programlarda belli satırlara “Bookmark” kitap izi bırakdınız mı?		
Dim komutu ile değişken tanımladınız mı?		
Değişkenin alabileceği minimum ve maksimum değere göre değişkenin türünü belirttiniz mi?		
Tüm programda aynı değere sahip olacak bir sabit tanımlama ve türünü belirlediniz mi?		
Dizinin türünü belirlediniz mi?		
Diziye ilk değerlerini aktardınız mı?		
Dizinin boyutlarını ReDim ile güncellediniz mi?		
Bir değişkenin üzerinde “ = ” kullanarak yeni bir değer aktardınız mı?		
Yeni değerin hesaplanmasında matematiksel işlem simgeleri kullandınız mı?		
Metin türünde değişken tanımladınzı mı?		
Metin değişkene değer aktardınız mı?		
Metin veya tarih türündeki değişkenler üzerinde işlemler yaptınız mı?		
IF komutunu kullanarak belli şartlar altında programın akışını değiştirdiniz mi?		
IF komutunda “ Else ” ile şartın olumsuz olması halindeki kodları yazdınız mı?		
Peşpeşe IF’lerde “ElseIf” ile komutları birleştirdiniz mi?		
IF komutu çok satırlı ise “End If” ile bitirdiniz mi?		
Tam sayı aralıklarında ve çok ElseIf’lerde IF komut yerine “ Select Case” kullandınız mı?		

DEĞERLENDİRME

Yaptığınız değerlendirme sonucunda eksikleriniz varsa öğrenme faaliyetlerini tekrarlayınız.

CEVAP ANAHTARLARI

ÖĞRENME FAALİYETİ-1 CEVAP ANAHTARI

1	D
2	Y
3	D
4	D
5	D
6	Y
7	D
8	C
9	D

ÖĞRENME FAALİYETİ-2 CEVAP ANAHTARI

1	D
2	Y
3	Y
4	Y
5	Y
6	D
7	Y
8	C
9	C

ÖĞRENME FAALİYETİ-3 CEVAP ANAHTARI

1	D
2	Y
3	D
4	Y
5	Y
6	D
7	D
8	D
9	D
10	B

ÖĞRENME FAALİYETİ-4 CEVAP ANAHTARI

1	D
2	Y
3	D
4	D
5	Y
6	D
7	D
8	Y
9	C
10	B

ÖĞRENME FAALİYETİ-5 CEVAP ANAHTARI

1	D
2	Y
3	D
4	D
5	Y
6	D

ÖĞRENME FAALİYETİ-6 CEVAP ANAHTARI

1	D
2	Y
3	D
4	Y
5	Y
6	D
7	A
8	C

KAYNAKÇA

- HOCAOĞLU Özgür, **Visual Basic .NET**, Pusula Yayıncılık, 2005.
- HALVORSON Michael, **Microsoft Visual Basic .NET Step By Step**, Microsoft Pres, A Divicion of Microsoft Corporation One Microsoft Way Redmond, 2002.
- PALA Zeydin, **Microsoft Visual Basic.NET**, Türkmen Kitabevi, 2003.
- KARAGÜLLE İhsan, **Visual Basic.NET Başlangıç Rehberi,** Türkmen Yayınevi, 2003.
- GÜLEÇ Hakan (Çeviren), **Visual Basic 2005**, Alfa Yayınları, 2006.
- YANIK Memik, **Visual Basic 5.0,** Beta Basım Yayım Dağıtım A.Ş, 1997.
- ÇÖMLEKÇİ Mehmet (Çeviren), **Visual Basic 6 Temel Kullanım Klavuzu**, Alfa Basım Yayım Dağıtım San. Ve Tic. Ltd. Şti, 1999.
- http://www.msakademik.net/makaleler.aspx?grup=VBN
- http://www.vbturk.net/
- http://www.bmsoftware.net/programlama/vbnet/vbnet02.aspx
- http://www.hazirkod.com/default.asp?fform_kategori_id=20&fform_kategori=VISUAL%20BASIC.NET
- http://www.vbasicmaster.com
- Mastering, Visual Basic .NET
  Evongelos Petroutsos, SYBEX, Inc., Alameda, CA, 2002