POGLAVLJE 1

Osnove jezika Visual Basic .NET

U OVOM POGLAVLJU

U poslednjih deset godina Visual Basic je postao veoma popularan. Mnogi programeri smatraju da lakoća upotrebe Visual Basica predstavlja ključ njegovog uspeha. Drugi ističu da tehnika prevlačenja kontrola na obrazac omogućava da se brzo napravi korisnički interfejs programa, što je dovelo do široke popularnosti Visual Basica.

Velika grupa programera, od kojih su mnogi godinama koristili programske jezike kao što su C i C++, nije prihvatala činjenicu da se Visual Basic može koristiti za profesionalne namene. Tvrdili su da Visual Basic pruža jednostavan način da se napravi prototip, ali da bi ipak programeri morali kasnije ponovo da napišu kôd na jeziku kao što je C++ u cilju postizanja boljih performansi.

U okruženje .NET Microsoft je uključio dva nova programska jezika, Visual Basic .NET i C# (Microsoft je u okruženje .NET uključio i programski jezik Visual C++ .NET). Kao što ćete saznati, klase i rutine okruženja .NET ne zavise od programskog jezika, a koriste ih i C# i Visual Basic .NET. To znači da i programer koji koristi Visual Basic .NET i onaj koji koristi C# imaju na raspolaganju iste mogućnosti okruženja .NET.

Aplikacije napisane na Visual Basicu .NET radiće istom brzinom kao i one napisane na jeziku C#. Iako .NET okruženje omogućava C# programerima da prevlače kontrole na obrasce i tako naprave korisnički interfejs, veliki broj Visual Basic programera koji prelaze na platformu .NET, najčešće će koristiti jezik Visual Basic .NET.

Kroz osamnaest poglavlja u ovoj knjizi proučićete detaljno Visual Basic .NET i okruženje .NET. Ovo poglavlje je napisano da bi programeri koji su početnici u Visual Basicu upoznali osnove jezika, unapredili svoje znanje i shvatili mogućnosti koje im pružaju Visual Basic .NET i okruženje .NET. Ako ste iskusan Visual Basic programer, možete samo pregledati podnaslove u ovom poglavlju i odabrati teme za koje mislite da su nove, a zatim preći na zadnji deo ovog poglavlja u kom su navedene osnovne razlike između programskih jezikaVisual Basic i Visual Basic .NET.

Pravljenje prve konzolne aplikacije

U Visual Basicu .NET možete praviti razne vrste aplikacija, na primer, konzolne programe koji prikazuju rezultate u prozoru sličnom prozoru MS-DOS-a, kao što je prikazano na slici dole, Windows programe sa interfejsom u obliku obrazaca, ASP.NET stranice i još mnogo toga.

Budući da se konzolne aplikacije lako prave (možete brzo napraviti program koji samo prikazuje nekakav rezultat, bez postavljanja kontrola na obrazac), u mnogim vežbama u ovoj knjizi koriste se konzolne aplikacije.

PRIMER Da biste napravili konzolnu aplikaciju u Visual Studiju, uradite sledeće:

1. Izaberite File | New | Project. Prikazaće se okvir za dijalog New Project.
2. U okviru za dijalog New Project izaberite ikonicu Console Application. U polje Name upišite ime projekta koji razvijate, na primer DemoProgram, ali nemojte upisivati nastavak imena datoteke. Zatim u polje Location upišite ime direktorijuma u koji će Visual Studio snimiti direktorijum i datoteke vašeg projekta. Pritisnite OK. Visual Studio će prikazati prozor za kôd (slika 1-1).

U prozoru za kôd upišite sledeće naredbe u proceduru Main.

```
Sub Main()
   Console.WriteLine("Moj prvi VB.NET program!")
   Console.ReadLine()
End Sub
```

Slika 1-1 Prozor za kôd u Visual Studiju.

Da biste pokrenuli program, izaberite Debug | Start. Visual Studio će prikazati rezultate programa u obliku konzolnog prozora. Da biste zaustavili program i naredili Visual Studiju da zatvori prozor, pritisnite taster <ENTER>. Kada pišete programske naredbe u Visual Basicu .NET, pišite ih tačno onako kako su napisane u ovoj knjizi; proverite da li ste uneli znakove navoda, zareze, tačke i slično. U suprotnom, prekršićete neko pravilo sintakse Visual Basica .NET (pravila koja definišu strukturu jezika i format koji morate koristiti kada pišete programe). Kada se pojavi greška u sintaksi, Visual Studio će prikazati poruku koja opisuje grešku i broj reda gde se pojavila greška. Da bi Visual Studio mogao da prevede program, morate pronaći i ispraviti sintaksnu grešku.

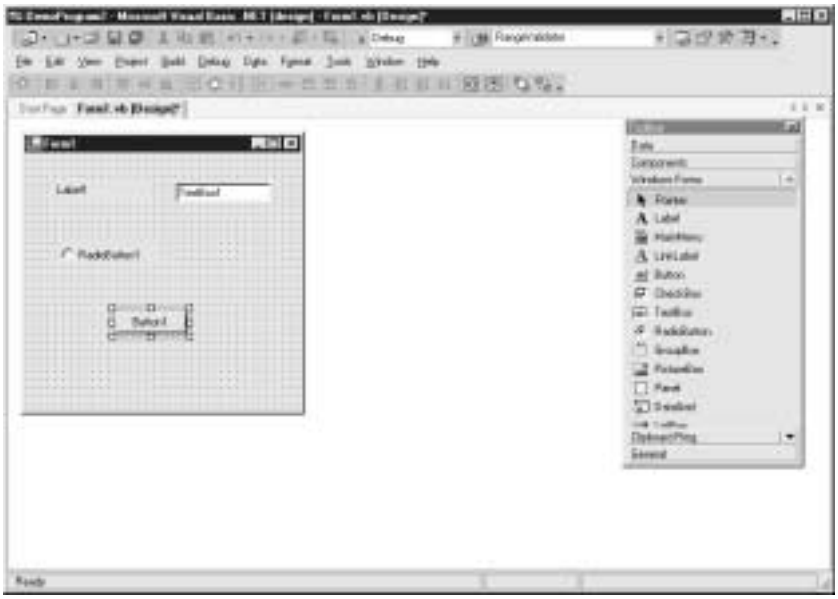
Kad god promenite programski kôd, naredite Visual Studiju da ponovo prevede program u izvršni oblik. Da biste ponovo preveli program, izaberite Build | Build Solution. Na primer, u naredbi iz prethodnog programa, promenite kôd tako da prikaže poruku „Hello, user“ – promenite metodu Console.WriteLine kao što sledi:

```
Console.WriteLine("Hello, user")
```

Zatim ponovo prevedite program i nakon toga izaberite Debug | Start da biste videli nov rezultat.

Pravljenje Windowsove aplikacije

U Visual Studiju možete praviti razne vrste aplikacija. Da bi napravili Windowsovu aplikaciju, programeri najčešće prevlače jednu ili više kontrola na obrazac, kao što je prikazano na slici 1-2.



Slika 1-2 Postavljanje kontrola u Visual Studiju pri pravljenju Windowsove aplikacije.

PRIMER U poglavlju 11 detaljno su opisane kontrole koje možete postaviti na obrazac. Da biste napravili jednostavnu Windowsovu aplikaciju, uradite sledeće:

1. U Visual Studiju izaberite File | New | Project. Visual Studio će prikazati okvir za dijalog New Project.
2. U okviru za dijalog New Project, pritisnite ikonicu Windows Application. U polje Name upišite ime projekta koje opisuje program, npr. DemoProgram (nemojte upisivati nastavak imena datoteke). Nakon toga, u polje Location upišite ime direktorijuma u koji Visual Studio treba da snimi direktorijum i datoteke projekta. Pritisnite OK. Visual Studio će prikazati prozor za rad sličan onom na slici 1-3 u kojem možete prevući kontrole na obrazac.
3. Da biste prikazali kutiju sa alatkama u kojoj se nalaze kontrole za obrazac, odaberite View | Toolbox i Visual Studio će otvoriti prozor Toolbox.
4. U prozoru Toolbox pronađite kontrolu Label i prevucite je na obrazac.
5. Kada postavite kontrolu na obrazac, pritisnite je desnim tasterom miša i izaberite Properties. Visual Studio će u prozoru Properties prikazati svojstva kontrole.
6. U okviru za dijalog Properties pronađite svojstvo Text i upišite **Hello, user**.

Da biste preveli program, izaberite Build | Build Solution, a zatim, da biste ga pokrenuli, izaberite Debug | Start.Visual Studio će prikazati obrazac programa u novom prozoru (slika 1-3).

Ako izmenite obrazac ili programski kôd, morate narediti Visual Studiju da ponovo prevede program da bi promene počele da važe.

Slika 1-3 Pravljenje i pokretanje jednostavne Windowsove aplikacije u Visual Studiju.

Biranje odgovarajućeg tipa podataka

Da bi sačuvao podatke koji mu trebaju tokom izvršavanja, program ih upisuje na lokacije koje imaju imena. Te lokacije programeri nazivaju promenljivama zato što se njihova vrednost može menjati tokom rada programa. Na primer, jedna promenljiva može predstavljati ime korisnika, druga adresu e-pošte i treća godinu rođenja.

Svaka promenljiva koju napravite u programu mora biti određenog tipa. Od tipa promenljive zavisi vrsta podataka koje možete čuvati u njoj. Tipovi podataka su npr. celobrojne vrednosti ili vrednosti u formatu pokretnog zareza (brojevi s decimalnim zarezom), ili alfanumerički znakovi (kao što su slova kojima se piše nečije ime). Tip promenljive takođe definiše skup operacija koje program izvršava s promenljivom. Na primer, program može da pomnoži dva broja, ali je množenje dva znakovna niza (kao što su imena, na primer) besmisleno. U tabeli 1-1 ukratko su opisani tipovi podataka u Visual Basicu. Za svaki tip, u tabeli je prikazan i opseg vrednosti koji se može čuvati u takvoj promenljivoj, kao i broj bajtova memorije koje Visual Basic mora zauzeti za takvu vrednost.

Tip podataka	Vrednosti	Veličina
Boolean (logički)	True (1) ili False (0)	2 bajta
Byte (bajt)	8-bitne vrednosti između 0 i 255	1 bajt
Char (znakovni)	16-bitni Unicode znakovi	2 bajta
DateTime	Datum i vreme	8 bajtova
Decimal (decimalni)	Brojevi sa 28 značajnih cifara u opsegu +/-79,228,162,514,264,337,593,543,950,335 bez decimalnog zareza do 7.9228162514264337593543950335 sa 28 mesta iza decimalnog zareza	12 bajtova

Tabela 1-1 Tipovi podataka u Visual Basicu .NET

Tip podataka	Vrednosti	Veličina
Double (dvostruka tačnost)	Brojevi u formatu pokretnog zareza sa 64 bita	8 bajtova
Integer (celobrojni)	Brojevi u opsegu -2,147,483,648 do 2,147,483,647	4 bajta
Long (dugački celobrojni)	Brojevi u opsegu od -9,223,372,036,036,854,775,808 do 9,223,372,036,036,854,775,807	8 bajtova
Short (kratki celobrojni)	Brojevi u opsegu od -32768 do 32767	2 bajta
Single (jednostruka tačnost)	Brojevi u formatu pokretnog zareza sa 32 bita	4 bajta

Tabela 1-1 Tipovi podataka u Visual Basicu .NET (nastavak)

PRIMER Odaberite tip podataka koji najbolje odgovara konkretnim podacima. Pretpostavimo da program radi sa vrednostima u opsegu od -10000 do 20000. Visual Basic .NET podržava tipove podataka Integer, Long i Short i svaki od njih može se upotrebiti za vrednosti u navedenom opsegu. Međutim, ako koristite tip podataka Short, program će trošiti manje memorije za čuvanje vrednosti – što znači da će brže snimati i čitati vrednosti nego sa ostala dva tipa podataka. Što je još važnije, ako izaberete tip Short, programer koji čita kôd vašeg programa znaće da se u promenljivoj čuvaju vrednosti u opsegu od -32768 do 32767. Visual Basic .NET više ne podržava tipove podataka Currency i Variant koji su postojali u prethodnim verzijama Visual Basica.

Deklarisanje promenljivih u programima

Promenljive postoje da bi programi mogli lako da skladište i čitaju podatke dok se izvršavaju. Programeri često promenljive opisuju kao „imenovano mesto u memoriji“ (skladište) koje sadrži vrednost. Veličina skladišta zavisi od tipa promenljive.

PRIMER Da biste deklarisali promenljivu u Visual Basicu .NET, koristićete naredbu Dim kojom se definiše ime i tip promenljive. Naredba Dim je tako nazvana zato što zadaje veličinu (dimenzije) skladišta za promenljivu. Sledeća naredba Dim deklariše promenljivu EmployeeNumber kao tip Integer:

```
Dim EmployeeNumber As Integer
```

U ovom slučaju, deklarisanjem promenljive tipa Integer, nalažete Visual Basicu .NET da zauzme četiri bajta da bi uskladištio vrednost promenljive celobrojnog tipa u opsegu -2,147,483,648 do 2,147,483,647. Programi često moraju da deklarišu nekoliko promenljivih istovremeno.

Sledeće naredbe deklarišu promenljive u kojima se čuvaju ime zaposlenog, šifra, plata i telefonski broj.

```
Dim EmployeeName As String
Dim EmployeePhoneNumber As String
Dim EmployeeNumber As Integer
Dim EmployeeSalary As Double
```

U prethodnim deklaracijama promenljivih, prve dve naredbe deklarišu promenljive tipa String u kojima se mogu čuvati alfanumerički znakovi. Visual Basic .NET omogućava da više promenljivih istog tipa deklarišete u jednom redu:

```
Dim EmployeeName, EmployeePhoneNumber As String
```

Sledeća naredba je ekvivalentna prethodnoj; u oba primera, deklarišu se po dve promenljive tipa String.

```
 Dim EmployeeName As String, EmployeePhoneNumber As String
```

Savetujemo vam da deklarišete promenljive u zasebnim redovima, jer vam to omogućava da desno od deklaracije dodajete komentar koji opisuje promenljivu.

```
Dim EmployeeName As String        ' Ime i prezime zaposlenog
Dim EmployeePhoneNumber As String ' 10 cifara u formatu ###-###-####
```

Izaberite smisleno ime koje opisuje podatke što se čuvaju u promenljivoj. Tako drugi programer koji čita kôd programa može lakše shvatiti svrhu i način korišćenja promenljive na osnovu njenog imena. Ako pogledate sledeće deklaracije promenljivih, ime prve promenljive opisuje njenu namenu, dok se iz drugog imena ne može ništa zaključiti.

```
Dim ImeMesta As String
Dim X As String
```

Pošto deklarišete promenljivu u Visual Basicu .NET, možete koristiti operator dodeljivanja (znak jednako):

```
ImeMesta = "Valjevo"
ImeZaposlenog = "Branko"
ZaradaPoSatu = 60
```

Promenljivoj se često dodeljuje početna vrednost. Neki programeri to čine tako što deklarišu promenljivu u jednom redu, a zatim joj dodeljuju početnu vrednost u sledećem redu, ovako:

```
Dim ImeZaposlenog As String         ' Ime i prezime zaposlenog
ImeZaposlenog = "Bill Smith"

Dim TelBroj As String               ' 10 cifara u formatu ###-###-####
TelBroj = "281-555-1212"
```

Te dve operacije mogu se spojiti u jednu naredbu:

```
Dim ImeZaposlenog As String = "Bill Smith"

Dim TelBroj As String = "281-555-1212"
```

Visual Basic .NET ne razlikuje velika od malih slova, što znači da velika i mala slova znače isto u imenu promenljive. Svaka od sledećih naredbi dodeljuje vrednost 50000 promenljivoj koja se zove ZaradaPoSatu.

```
ZaradaPoSatu = 50000
zaradaposatu = 50000
ZARADAPOSATU = 50000
zArAdApOsAtU = 50000
```

Ako ne dodelite početne vrednosti promenljivama, Visual Basic .NET će ih inicijalizovati tokom prevođenja programa. Početne vrednosti zavise od tipa promenljive i prikazane su u tabeli 1-2.

Tip promenljive	Vrednost
Boolean	False
Date	12:00:00AM
Svi numerički tipovi	0
Object	Nothing

Tabela 1-2 Podrazumevane vrednosti koje Visual Basic .NET dodeljuje promenljivama

Program koji sledi, DeclareVariables.vb, deklariše i dodeljuje početne vrednosti za nekoliko promenljivih. Program nakon toga prikazuje vrednost svake promenljive pomoću metode Console.WriteLine:

```
Module Module1

    Sub Main()
        Dim EmployeeName As String
        Dim EmployeePhoneNumber As String
        Dim EmployeeSalary As Double
        Dim NumberOfEmployees As Integer

        EmployeeName = "Buddy Jamsa"
        EmployeePhoneNumber = "555-1212"
        EmployeeSalary = 45000.0
        NumberOfEmployees = 1

        Console.WriteLine("NumberOfEmployees: " & NumberOfEmployees)
        Console.WriteLine("EmployeeName: " & EmployeeName)
        Console.WriteLine("EmployeePhoneNumber: " & EmployeePhoneNumber)
        Console.WriteLine("EmployeeSalary: " & EmployeeSalary)

        Console.ReadLine() ' Pauza da bismo pogledali rezultate
    End Sub
End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu računara pojaviće se sledeći rezultati:

```
Number of employees: 1
Employee name: Buddy Jamsa
Employee phone number: 555-1212
Employee salary: 45000
```

▶ *NAPOMENA*

U prethodnim verzijama Visual Basica programeri su za dodeljivanje vrednosti promenljivoj koristili naredbu LET. Visual Basic .NET ne podržava naredbu LET.

Prikazivanje vrednosti na ekranu upotrebom metoda Console.Write i Console.WriteLine

Kada napravite konzolnu aplikaciju, program prikazuje rezultat na ekranu u DOS prozoru nalik onom na slici 1-4.

Slika 1-4 Prikaz rezultata konzolnog programa na ekranu.

Da biste prikazali rezultat konzolnog programa, upotrebite metode Console.Write i Console.WriteLine. Ove metode se razlikuju po tome što Console.WriteLine ispisuje i znakove za prelazak na početak novog reda, dok Console.Write to ne čini.

Da biste prikazali znakovni niz upotrebom metode Console.WriteLine, prosledite joj tekst između navodnika, kao u sledećem primeru:

```
Console.WriteLine("Hello, Visual Basic World!")
```

Slično tome, da biste prikazali broj ili vrednost promenljive, prosledite broj ili ime te promenljive (bez znakova navoda), kao u sledećem primeru:

```
Console.WriteLine(1001)
Console.WriteLine(ImeZaposlenog)
```

Programeri često ispred vrednosti prikazuju tekst kao što je „Godina rođenja korisnika je:“. Da biste prikazali takav rezultat, upotrebite operator nadovezivanja Visual Basica .NET da biste vrednost dodali na kraj znakovnog niza, kao u sledećem primeru:

```
Console.WriteLine("Godina rođenja korisnika je: " & GodinaRođenja)
```

PRIMER Sledeći program, OutputDemo.vb, prikazuje upotrebu metoda Console.Write i Console.WriteLine:

```
Module Module1

   Sub Main()
       Dim A As Integer = 100
       Dim B As Double = 0.123456789
       Dim Message As String  = "Zdravo!"

       Console.WriteLine(A)
       Console.WriteLine("Vrednost promenljive A je : " & A)
       Console.WriteLine(B)
```

```
        Console.WriteLine(B & " plus " & A & " = " & B + A)
        Console.WriteLine(Message)

        Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
100
Vrednost promenljive A je 100
0.123456789
0.123456789 plus 100 = 100.123456789
Zdravo!
```

Obratite pažnju na poslednju naredbu u programu, Console.ReadLine(). Ako u Visual Studiju pokrenete konzolnu aplikaciju, prozor će se zatvoriti čim se program završi. Kada se na kraju programa nalazi naredba Console.ReadLine(), program će čekati da korisnik pritisne taster <ENTER> da bi zatvorio prozor.

Formatiranje rezultata programa pomoću metode Console.WriteLine

U prethodnim primerima koristili ste funkcije Console.WriteLine i Write da biste prikazali poruke u konzolnom prozoru. U svakom primeru, aplikacije su samo ispisivale tekst.

```
Console.WriteLine("Hello, World!")
```

U prvi parametar funkcija Console.WriteLine i Write možete umetati pozicione oznake, u obliku {0}, {1} itd, i prosleđivati vrednosti za svaku pozicionu oznaku. Sledeća naredba prikazuje kako se koriste pozicione oznake u funkciji Console.WriteLine.

```
Console.WriteLine("Rezultat {0}", 3 + 7)
```

U ovom slučaju, funkcija WriteLine zamenjuje pozicionu oznaku {0} vrednošću 10. Sledeća naredba koristi tri pozicione oznake:

```
Console.WriteLine("Rezultat {0} + {1} = {2}", 3, 7, 3+7)
```

U ovom slučaju, funkcija WriteLine zamenjuje pozicionu oznaku {0} vrednošću 3, pozicionu oznaku {1} vrednošću 7 i pozicionu oznaku {2} vrednošću 10.

Obavezno morate zadati vrednosti za sve pozicione oznake. Na primer, ako definišete pozicione oznake {0} i {1} a prosledite samo jednu vrednost, nastaće izuzetak; ako program ne obrađuje izuzetke, prekinuće se.

Iza broja pozicione oznake možete upisati i simbol formata kao što je {1, d} ili {2, 7:f}. Simbol za format može po potrebi sadržati i vrednost za širinu iza koje sledi dvotačka i znak koji određuje tip vrednosti. U tabeli 1-3 ukratko su opisani simboli za tipove vrednosti.

Oznaka	Tip vrednosti
C ili c	Format za lokalne novčane jedinice.
D ili d	Decimalna vrednost.
E ili e	Eksponencijalni (naučni) zapis brojeva.
F ili f	Pokretni zarez.
G ili g	Bira eksponencijalni zapis brojeva ili zapis brojeva s decimalnom tačkom, u zavisnosti od toga koji je od ta dva načina kompaktniji.
N ili n	Formatiranje brojeva tako da se u velikim vrednostima grupe cifara razdvajaju posebnim znacima.
P ili p	Formatiranje za procentualne vrednosti
R ili r	Zove se još i „siguran“ format. Koristi se za brojeve u pokretnom zarezu da bi se garantovalo da se prilikom pretvaranja broja u znakovni niz i zatim ponovnog pretvaranja znakovnog niza u broj, dobija izvorna vrednost.
X ili x	Heksadecimalni formati.

Tabela 1-3 Simboli za format koje možete koristiti sa funkcijama Console.WriteLine i Console.Write

U prethodnom primeru videli ste kako se u funkcijama Console.WriteLine i Console.Write koristi nekoliko različitih simbola za format. U programima koji prikazuju vrednosti u formatu pokretnog zareza, kao i u onima koji prikazuju novčane vrednosti, često treba definisati broj cifara desno od decimalnog simbola koje funkcija prikazuje. Pretpostavimo da program treba da prikaže sadržaj promenljive Iznos, čija je tekuća vrednost 0.123456789. Da biste zadali broj cifara koji prikazuju funkcije WriteLine i Write, definišite pozicionu oznaku, širinu, simbol za format i broj cifara, kao u sledećem primeru:

```
Console.WriteLine("Pogledajte broj decimala {0, 12:f1}", _
   0.123456789)  '0.1
Console.WriteLine("Pogledajte broj decimala {0, 12:f9}", _
   0.123456789)  '0.123456789
```

Osim širine i simbola za format, u funkcijama možete koristiti i znak za povisilicu (tarabu) da biste formatirali podatke. Sledeća naredba zadaje funkciji Console.WriteLine da prikaže vrednost u pokretnom zarezu sa dve cifre desno od decimalnog simbola.

```
Console.WriteLine("Vrednost je {0, 0:###.##}", Vrednost)
```

Kada koristite znak povisilice da biste zadali format izlaznih podataka, funkcija WriteLine neće prikazati vodeće nule. Drugačije rečeno, rezultat za vrednost 0.123 biće samo .123. Ako želite da prikažete i vodeće nule, znak povisilice zamenite nulom kao u sledećem primeru:

```
Console.WriteLine("Vrednost je {0, 0:000.00}", Vrednost)
```

Program koji sledi, ConsoleWriteLineDemo.vb, prikazuje upotrebu različitih mogućnosti formatiranja:

```
Module Module1

    Sub Main()
        Dim A As Double = 1.23456789
        Console.WriteLine("{0} {1} {2}", 1, 2, 3)
        Console.WriteLine("{0, 1:D} {1, 2:D} {2, 3:D}", 1, 2, 3)
        Console.WriteLine("{0, 7:F1} {1, 7:F3} {2, 7:F5}", A, A, A)
        Console.WriteLine("{0, 0:#.#}", A)
        Console.WriteLine("{0, 0:#.###}", A)
        Console.WriteLine("{0, 0:#.#####}", A)

        Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
1 2 3
1  2   3
    1.2     1.235  1.23457
1.2
1.235
1.23457
```

Dodavanje znakova na kraj znakovnog niza

Tip promenljive String omogućava da radite sa alfanumeričkim znakovima (mala i velika slova abecede, cifre od 0 do 9 i interpunkcijski znakovi). Da biste dodelili vrednost promenljivoj tipa String, upišite tekst između navodnika:

```
Dim Name As String = "Jovan Jovanović"
```

Program UseStrings.vb dodeljuje vrednosti različitim promenljivama tipa String, a zatim ih prikazuje pomoću funkcije Console.WriteLine:

```
Module Module 1

   Sub Main()
       Dim Book As String= _
         "Visual Basic .NET kroz praktične primere"
       Dim Author As String = "Jamsa"
       Dim Publisher As String = "Mikro knjiga"
       Console.WriteLine(Book)
       Console.WriteLine(Author)
       Console.WriteLine(Publisher)

       Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
Visual Basic .NET kroz praktične primere
Jamsa
Mikro knjiga
```

Promenljive tipa String se najčešće koriste tako što se na njih dodaju znakovi. Dodavanje znakova promenljivoj tipa String programeri zovu nadovezivanje.

PRIMER Da biste nadovezali jedan znakovni niz na drugi, upotrebite znak ampersend (&) koji je u Visual Basicu .NET operator nadovezivanja. Sledeća naredba dodeljuje ime i prezime zaposlenog promenljivoj Zaposleni nadovezivanjem sadržaja promenljivih Ime i Prezime i dodeljivanjem rezultata promenljivoj Zaposleni.

```
Dim Ime As String = "Bill"
Dim Prezime As String = "Gates"
Dim Zaposleni As String
EmployeeName = FirstName & " " & LastName
```

Ime i prezime se razdvajaju znakom za razmak koji se nadovezuje na kraj sadržaja promenljive Ime. Tokom čitanja ove knjige nailazićete na naredbe Console.WriteLine u kojima se vrednost promenljive nadovezuje na znakovni niz.

```
Console.WriteLine("Rezultat je" & NekaPromenljiva)
```

Sledeći program, ConcatenateDemo.vb, prikazuje upotrebu operatora za nadovezivanje:

```
Module Module1

    Sub Main()
        Dim WebSite As String
        Dim Publisher As String = "Mikro knjiga"

        Console.WriteLine("Izdavač ove knjige je: " & Publisher)

        WebSite = "www.mikroknjiga.co.yu"
        Console.WriteLine("Pogledajte njihove knjige na : " & WebSite)

        Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
Izdavač ove knjige je: Mikro knjiga
Pogledajte njihove knjige na www.mikroknjiga.co.yu
```

Obavezno zadavanje tipa promenljive

Da biste izbegli greške koje nastaju usled pogrešno napisanih imena, trebalo bi da u programima zahtevate obavezno deklarisanje promenljivih. U deklaraciji promenljive zadaje se tip promenljive, čime se ograničavaju opseg vrednosti i vrste operacija nad promenljivom.

U sledećem programu, BadVariables.vb, nisu izričito deklarisane promenljive koje se koriste. Program dodeljuje vrednosti promenljivama koje se odnose na zaposlene a zatim prikazuje ime zaposlenog.

```
Option Explicit Off 'Dozvoljavamo da se u programu koriste promenljive
                    'koje nisu izričito deklarisane
Module Module 1

    Sub Main()
        EmployeeName = "Buddy Jamsa"
        EmployeePhoneNumber = "555-1212"
        EmployeeSalary = 45000.0
        NumberOfEmployees = 1

        Console.WriteLine("NumberOfEmployees: " & NumberOfEmployees)
        Console.WriteLine("EmployeeName: " & EmployeName)
        Console.WriteLine("EmployeePhoneNumber: " & EmployeePhoneNumber)
        Console.WriteLine("EmployeeSalary: " & EmployeeSalary)

        Console.ReadLine()'Pauza da bismo pogledali rezultate
    End Sub

End Module
```

Nažalost, kada izvršavate ovaj program, on neće prikazati ime zaposlenog. Umesto imena zaposlenog, program neće prikazati ništa:

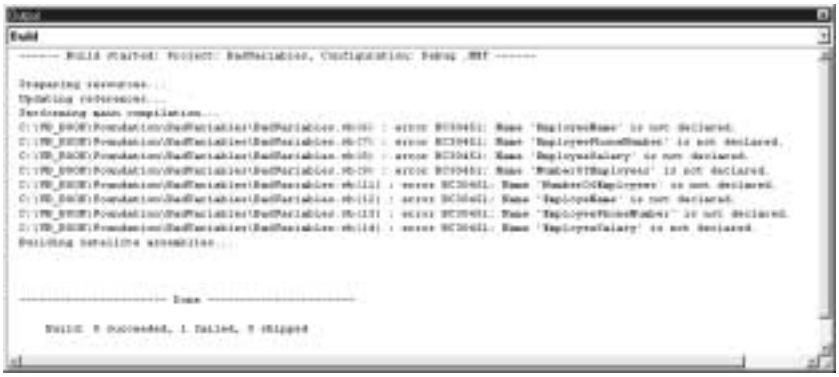
```
NumberOfEmployees: 1
EmployeeName:
EmployeePhoneNumber: 555-1212
EmployeeSalary: 45000
```

Ako pažljivo ispitate program, videćete da je u naredbi Console.WriteLine koja prikazuje ime zaposlenog, pogrešno napisano ime promenljive (izostavljeno je krajnje e u Employee). Visual Studio standardno zahteva da izričito deklarišete svaku promenljivu koju koristite u programu. U ovom slučaju, naredba Option Explicit Off nalaže prevodiocu da dozvoli programu korišćenje promenljive iako nije bila prethodno deklarisana (što ne bi trebalo da dozvolite).

PRIMER Da biste zahtevali obavezno deklarisanje promenljivih, napišite sledeću naredbu na početku programa:

```
Option Explicit On
```

Ako upišete ovu naredbu na početak programa Bad.Variables.vb, prevodilac će prijaviti sintaksnu grešku porukom nalik onoj na slici 1-5, koja saopštava da promenljive nisu deklarisane. Ako deklarišete promenljive da biste otklonili sintaksne greške, verovatno ćete otkriti pogrešno napisana imena promenljivih.



Slika 1-5 Poruke o sintaksnim greškama koje pokazuju da ima promenljivih koje nisu deklarisane.

Opseg i tačnost promenljivih

Kao što već znate, vrsta promenljive određuje opseg prihvatljivih vrednosti i skup dozvoljenih operacija nad promenljivom. Ako promenljivoj dodelite vrednost izvan prihvatljivog opsega, pojaviće se greška prekoračenja kapaciteta promenljive. Pretpostavimo da program koristi promenljivu tipa Short kojoj se mogu dodeliti vrednosti između -32,678 do 32,767, da promenljiva sadrži vrednost 32,767 i da joj se u programu dodaje vrednost 1, kako je prikazano ovde:

```
Dim Value As Short = 32767
Value = Value + 1
```

Vrednost 32,768 je izvan prihvatljivog opsega vrednosti za promenljivu. Kada se pojavi greška prekoračenja kapaciteta, nastaće izuzetak i prekinuće se program, uz prikazivanje poruke o grešci nalik onoj na slici 1-6. U poglavlju 9 opisani su izuzeci i kako bi programi trebalo da ih obrađuju.

Slika 1-6 Izuzetak (greška) koji se pojavljuje kada je vrednost koju program dodeljuje promenljivoj izvan prihvatljivog opsega.

PRIMER Kao što tip promenljive ograničava opseg vrednosti koje promenljiva može da prihvati, tako isto ograničava i tačnost promenljive. Promenljive tipa Single mogu biti tačne do 7 cifara desno od decimalnog zareza, dok su promenljive tipa Double tačne do 15 cifara. Sledeći program, ShowPrecision.vb, prikazuje tačnost promenljivih u jednostrukoj i dvostrukoj preciznosti.

```
Module Module1

    Sub Main()
        Dim A As Single = 0.123456789012345
        Dim B As Double = 0.123456789012345

        Console.WriteLine("Single: " & A)
        Console.WriteLine("Double: " & B)

        Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
Single : 0.1234568
Double : 0.123456789012345
```

Ograničena preciznost upotrebljenog tipa promenljive može dovesti do grešaka koje se teško otkrivaju. U sledećem programu, PrecissionError.vb, petlja For prolazi kroz vrednosti 0.01, 0.02, 0.03 do 0.1. Program treba da prikaže poruku kada promenljiva A ima vrednost 0.05.

```
Module Module1

    Sub Main()
        Dim A As Single

        For A = 0.01 To 0.1 Step 0.01
            If (A = 0.05) Then
                Console.WriteLine("Dostignuta je vrednost od 0.05")
            End If
         Next

         Console.WriteLine("Završena petlja")
         Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazan sledeći rezultat:

```
Završena petlja
```

Kao što vidite, program ne prikazuje poruku „Dostignuta je vrednost od 0.05“. Ovo se događa zato što je ograničena tačnost kojom računar prikazuje vrednost 0.05. Da biste napravili petlju koja će prikazati poruku, morate promeniti naredbu If, kako biste uzeli u obzir ograničenu tačnost i utvrdili da li je vrednost približna 0.05.

```
If (Math.Abs(A - 0.05) < 0.00001) Then
```

Numeričke operacije

Svaki koristan program izvršava operacije nad promenljivama. Na primer, program može pomnožiti promenljivu koja sadrži vrednost porudžbina sa stopom poreza na promet da bi izračunao porez koji korisnik treba da plati. Potom može dodati iznos poreza i troškove isporuke iznosu porudžbine kako bi izračunao ukupan iznos koji korisnik treba da plati.

```
Ukupno = 10.0 + 20.0
Porez = Ukupno * 0.05
ZaUplatu = Ukupno + Porez
```

Prilikom izvođenja ovakvih operacija programi koriste aritmetičke operatore. U tabeli 1-4 ukratko su opisani aritmetički operatori Visual Basica .NET.

Operator	Upotreba
+	Sabiranje
-	Oduzimanje
*	Množenje
/	Deljenje (standardno)
\	Deljenje (celobrojno)
Mod	Modulo (ostatak deljenja)
^	Stepenovanje

Tabela 1-4 Aritmetički operatori Visual Basica .NET

PRIMER Program koji sledi, OperatorDemo.vb, ilustruje upotrebu različitih aritmetičkih operatora u Visual Basicu .NET.

```
Module Module1

    Sub Main()
        Console.WriteLine("1 + 2 = " & 1 + 2)
        Console.WriteLine("3 - 4 = " & 3 - 4)
        Console.WriteLine("5 * 4 = " & 5 * 4)
        Console.WriteLine("25 / 4 = " & 25 / 4)
        Console.WriteLine("25 \ 4 = " & 25 \ 4)
        Console.WriteLine("25 Mod 4 = " & 25 Mod 4)
        Console.WriteLine("5 ^ 2 = " & 5 ^ 2)
```

```
        Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
1 + 2 = 3
3 - 4 = -1
5 * 4 = 20
25 / 4 = 6.25
25 \ 4 = 6
25 Mod 4 = 1
5 ^ 2 = 25
```

Svaki operator ima određeni prioritet. Prioritet određuje redosled izračunavanja operacija. Pretpostavimo da program treba da izračuna rezultat sledećeg izraza:

```
Result = 5 + 3 * 2
```

Kada bi Visual Basic .NET izvršavao operacije sleva nadesno, rezultat bi bio 16, što nije tačno. Međutim, prvo će izvršiti operaciju množenja, pošto operator množenja ima viši prioritet u odnosu na sabiranje.

```
Result = 5 + 3 * 2
Result = 5 + 6
Result = 11
```

U tabeli 1-5 prikazani su prioriteti aritmetičkih operatora Visual Basica .NET.

Prioritet operatora često nije dovoljan da potpuno opiše ispravan redosled izračunavanja. Pretpostavimo da program treba da izračuna porez na promet (koristeći stopu od 5% u ovom slučaju) za dva proizvoda čije su cene 10 i 20 dinara. Pogledajte sledeći izraz:

```
Porez = 10 + 20 * 0.05
```

Aritmetički operator	Upotreba
^	Stepenovanje
-	Negacija
*, /	Množenje i deljenje
\	Celobrojno deljenje
Mod	Ostatak deljenja
+,-	Sabiranje i oduzimanje. Zapamtite da, kada spajate dva znakovna niza operatorom +, ta operacija ima isti stepen prioriteta kao i sabiranje i oduzimanje, ali ako koristite operator &, onda spajanje ima niži prioritet.
And, Or, Not, Xor	Operatori za operacije sa bitovima

Tabela 1-5 Prioriteti aritmetičkih operatora u Visual Basicu .NET

Pošto operator množenja ima viši prioritet od operatora sabiranja, Visual Basic .NET će prvo izvršiti množenje, što će dati sledeći netačan rezultat za iznos poreza:

```
Porez = 10 + 20 * 0.05
      = 10 + 1
      = 11
```

PRIMER Da biste izričito zadali redosled izračunavanja aritmetičkih operacija, grupišite operacije između zagrada. Kada Visual Basic .NET obrađuje neki izraz, on prvo obrađuje delove izraza u zagradama. U prethodnom primeru s porezom na promet treba da upotrebite zagrade da bi program prvo sabrao prve dve stavke, pa zatim pomnožio zbir.

```
Porez = (10 + 20) * 0.05
```

U ovom slučaju bi program izračunao iznos poreza na sledeći način:

```
Porez = (10 + 20) * 0.05
      = (30) * 0.05
      = 1.50
```

Program koji sledi, PrecedenceDemo.vb, prikazuje kako upotreba zagrada radi zadavanja određenog redosleda operacija može menjati rezultat:

```
Module Module1

    Sub Main()
        Dim Expression1 As Double
        Dim Expression2 As Double
        Dim Expression3 As Double
        Dim Expression4 As Double

        Expression1 = 5 ^ 2 + 1 * 3 - 4
        Expression2 = 5 ^ (2 + 1) * 3 - 4
        Expression3 = 5 ^ (2 + 1) * (3 - 4)
        Expression4 = 5 ^ ((2 + 1) * (3 - 4))

        Console.WriteLine(Expression1)
        Console.WriteLine(Expression2)
        Console.WriteLine(Expression3)
        Console.WriteLine(Expression4)

        Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
24
371
-125
0.008
```

Pretvaranje jednog tipa podataka u drugi

Tip promenljive određuje opseg vrednosti koje promenljiva prihvata i skup operacija koje program može nad njom izvršiti. Ponekad ćete morati da dodelite vrednost promenljive jednog tipa promenljivoj drugog tipa. Programeri ovakve operacije zovu *pretvaranje tipa* (engl. *casting*) promenljive.

Dodela vrednosti „manjeg“ tipa, kao što je vrednost tipa Short, promenljivoj „većeg“ tipa, kao što je promenljiva tipa Integer, dozvoljena je pošto se manja promenljiva može smestiti u veću. Programeri ovakvo dodeljivanje koje pretvara vrednost promenljive manjeg tipa u veći tip nazivaju implicitno pretvaranje.

Nasuprot tome, ako dodeljujete vrednost promenljive većeg tipa promenljivoj manjeg tipa, Visual Basic .NET mora odbaciti deo bitova koji predstavljaju vrednost. Ako dodelite vrednost tipa Integer (za koju Visual Basic .NET koristi 32 bita) promenljivoj tipa Short (za koju Visual Basic .NET koristi 16 bitova), Visual Basic .NET će odbaciti vrednost gornjih 16 bitova, što očigledno može dovesti do pogrešnih rezultata.

PRIMER Ako vam ne smeta to što će Visual Basic .NET možda odbaciti deo vrednosti većeg tipa, morate zahtevati izričito (eksplicitno) pretvaranje. Da bi program izvršio izričito pretvaranje, upotrebite jednu od ugrađenih funkcija prikazanih u tabeli 1-6.

Funkcija	Namena
ToBoolean	Pretvara parametar u tip Boolean (True ili False).
ToByte	Pretvara parametar u 8-bitnu vrednost tipa Byte u opsegu 0-255.
ToChar	Pretvara parametar u 2-bajtni Unicode znak.
ToDateTime	Pretvara parametar u objekat tipa DateTime.
ToDecimal	Pretvara parametar u 12-bajtnu vrednost tipa decimal.
ToDouble	Pretvara parametar u 8-bajtnu vrednost tipa Double.
ToInt16	Pretvara parametar u 2- bajtnu vrednost tipa Short.
ToInt32	Pretvara parametar u 4-bajtnu vrednost tipa Integer.
ToInt64	Pretvara parametar u 8-bajtnu vrednost tipa Integer.
ToSByte	Pretvara parametar u 8-bitnu vrednost sa predznakom u opsegu -128 do 127.
ToSingle	Pretvara parametar u 4-bajtnu vrednost tipa Single.
ToString	Pretvara parametar u znakovni niz tipa String.
1ToUInt16	Pretvara parametar u 2-bajtnu vrednost tipa Short bez predznaka, u opsegu 0 do 65,535.
ToUInt32	Pretvara parametar u 4-bajtnu vrednost tipa Integer bez predznaka, u opsegu 0 do 4,294,967,295.
ToUInt64	Pretvara parametar u 8-bajtnu vrednost tipa Long Integer bez predznaka, u opsegu 0 do 18,446,744,073,709,551,615.

Tabela 1-6 Metode za pretvaranje tipa vrednosti koje se nalaze u imenskom prostoru System.Convert

Program koji sledi, CastDemo.vb, izvršava dve jednostavne operacije pretvaranja. Prva dodeljuje vrednost promenljive tipa Integer promenljivoj tipa Short, dok druga pretvara vrednost tipa Double u vrednost tipa Single.

```
Module Module1

    Sub Main()
        Dim BigInteger As Integer = 10000
        Dim LittleInteger As Short

        Dim BigFloat As Double = 0.123456789
        Dim LittleFloat As Single

        LittleInteger = BigInteger
        LittleFloat = BigFloat

        Console.WriteLine(LittleInteger)
        Console.WriteLine(LittleFloat)

        Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
10000
0.1234568
```

U slučaju promenljivih u formatu pokretnog zareza, došlo je do gubljenja značajnih cifara. Pretvaranje tipa Integer u tip Short bilo je uspešno jer je vrednost promenljive tipa Integer bila unutar opsega vrednosti koje može prihvatiti promenljiva tipa Short. Da ste, na primer, probali da u tip Short pretvorite vrednost 40000, nastao bi izuzetak prekoračenja kapaciteta. U zavisnosti od vrednosti koju sadrži promenljiva tipa Integer, ovaj program radi ili ne radi.

Da biste smanjili verovatnoću grešaka koje mogu nastati kada program dodeljuje vrednost šireg tipa promenljivoj užeg tipa, možete zahtevati od prevodioca da ne dozvoli takva dodeljivanja tako što ćete na početku programa upisati sledeću naredbu:

```
Option Strict On
```

Kada uključite striktno pretvaranje tipa, sledeća naredba dodeljivanja prouzrokovaće sintaksnu grešku:

```
LittleInteger = BigInteger
```

Grananje pomoću uslovnih operatora

Program je skup instrukcija koje procesor izvršava kako bi obavio određeni zadatak. Programi koji su do sada predstavljeni u ovom poglavlju, izvršavali su se od prve do poslednje naredbe redom, uključujući i nju.

U složenijim programima, često se izvršava jedan skup operacija ako je ispunjen neki uslov, a drugi skup operacija za drugi uslov. Na primer, korpa za kupovinu na Web lokaciji koristi jedne poreske stope za kupce iz Evrope, a druge za kupce iz Amerike.

Postupak kojim program utvrđuje koju naredbu treba da izvede, programeri nazivaju uslovnim grananjem. Za uslovno grananje koriste se naredbe If i If-Else.

Da biste upotrebili naredbu If, morate zadati logički uslov koji Visual Basic .NET svodi na vrednost True ili na False. Na primer, uslov može biti naredba If kojom se ispituje da li je ime korisnika „Smith“ ili da li je ukupna ocena studenta na ispitu veća ili jednaka 90:

```
If  (Username = "Smith") Then
    'Naredbe koje treba da se izvrše
End If

If  (TestScore > 90) Then
    'Naredbe koje treba da se izvrše
End If
```

Kada Visual Basic .NET naiđe na naredbu If, on ispituje odgovarajući uslov. Ako je vrednost uslova True, program će izvršiti naredbe koje se nalaze između naredaba If i End If. Ukoliko je vrednost uslova False, program neće izvršiti te naredbe, već će nastaviti sa izvršavanjem od prve naredbe iza naredbe End If.

Često se dešava da programi treba da izvrše jedan skup naredaba ako je uslov ispunjen, a drugi ako nije. U takvim slučajevima, u programima se može koristiti naredba If-Else. Sledeća naredba prikazuje poruku koja saopštava studentu da li je položio ispit (student je polažio ispit ako ima više od 70 bodova na ispitu):

```
If  (TestScore >= 70) Then
  Console.WriteLine("Položili ste")
Else
  Console.WriteLine("Pali ste")
End If
```

Da bi povećali čitljivost programa, programeri obično uvlače naredbe u upravljačkim strukturama, kao što su naredbe If i Else. Čitaoci već na prvi pogled mogu videti koje naredbe čine grupu. Da bi pojednostavio postupak uvlačenja, Visual Studio automatski uvlači naredbe koje upisujete u grani If.

Često se događa da program mora obraditi više uslova. Na primer, umesto da samo saopšti studentu da li je položio ili nije položio ispit, program treba da prikaže i ocenu koju je student dobio. Jedan od načina da program prikaže ocene studenta jeste upotreba nekoliko naredaba If, kao što je prikazano dole:

```
If  (TestScore >= 90) Then
  Console.WriteLine("Dobili ste 10")
End If

If  (TestScore >= 80) And (TestScore < 90) Then
  Console.WriteLine("Dobili ste 8")
End If
```

```
If  (TestScore >= 70) And (TestScore < 80) Then
  Console.WriteLine("Dobili ste 6")
End If

If  (TestScore < 70) Then
  Console.WriteLine("Pali ste")
End If
```

Prethodni program mora obraditi sve naredbe If bez obzira na rezultate ispita korisnika. Bolje rešenje je upotreba niza naredaba If-Else, kao što je prikazano dole:

```
If  (TestScore >= 90) Then
  Console.WriteLine("Dobili ste 10")
ElseIf  (TestScore >= 80) Then
  Console.WriteLine("Dobili ste 8")
ElseIf  (TestScore >= 70) Then
  Console.WriteLine("Dobili ste 6")
Else
  Console.WriteLine("Dobili ste 5")
End If
```

PRIMER Sledeći program, IfDemo.vb, prikazuje upotrebu nekoliko različitih naredaba If i If-Else:

```
Module Module1

    Sub Main()
        Dim TestScore As Integer = 80

        If  (TestScore >= 90) Then
              Console.WriteLine("Dobili ste 10")
        ElseIf  (TestScore >= 80) Then
              Console.WriteLine("Dobili ste 8")
        ElseIf  (TestScore >= 70) Then
              Console.WriteLine("Dobili ste 6")
        Else
              Console.WriteLine("Dobili ste 5")
        End If

        Dim Language As String = "Engleski"

        If  (Language = "Engleski") Then
             Console.WriteLine("Hello, world!")
        ElseIf  (Language = "Španski") Then
             Console.WriteLine("Hola, mundo!")
        End If

        If  (Now.Hour < 12 ) Then
             Console.WriteLine("Dobro jutro")
        ElseIf  (Now.Hour < 18) Then
             Console.WriteLine("Dobar dan")
        Else
```

```
            Console.WriteLine ("Dobro veče")
        End If
            Console.ReadLine()
        End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
Dobili ste 8
Hello, world!
Dobro jutro
```

Provedite malo vremena eksperimentišući sa ovim programom tako što ćete menjati vrednosti koje program dodeljuje promenljivama TestScore i Language.

Relacioni i logički operatori

U uslovima kao što su oni u naredbama If ili While, koriste se relacioni operatori za poređenje vrednosti. Pomoću relacionih operatora u naredbi If može se ispitivati da li je određena vrednost veća, jednaka ili manja od druge. Rezultat ispitivanja uslova uvek je vrednost tipa Boolean (može biti True ili False). U tabeli 1-7 ukratko su opisani relacioni operatori Visual Basica .NET.

Postoje mnogi slučajevi kada treba ispitati dve ili više mogućnosti. Naredba If može ispitivati da li je korisnik imao 21 godinu ili je stariji i da li živi u Sjedinjenim Državama. Za ispitivanje dva ili više uslova istovremeno koriste se logički operatori. Visual Basic .NET podržava logičke operatore And, Or, Xor i Not.

Operator	Odnos
>	Veće od
<	Manje od
>=	Veće ili jednako
<=	Manje ili jednako
=	Jednako
<>	Nije jednako (različito)
Like	Upoređuje da li se znakovni niz poklapa sa zadatim uzorkom

Tabela1-7 Relacioni operatori u Visual Basicu .NET

PRIMER Program ConditionDemo.vb prikazuje upotrebu logičkih operatora And, Or i Not.

```
Module Module1

    Sub Main()
        Dim ImaLjubimca As Boolean = False
        Dim ImaKucu As Boolean = True
        Dim ImaMacu As Boolean = True
```

```
        If  (Not ImaLjubimca) Then
             Console.WriteLine("Potreban vam je kućni ljubimac.")
        End If

        If  (ImaKucu  Or ImaMacu) Then
             Console.WriteLine("Psi i mačke su divni.")
        End If

        If  (ImaKucu And ImaMacu) Then
             Console.WriteLine("Kako se slažu vaš pas i mačka?")
        End If

        Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i izvršite ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
Potreban vam je kućni ljubimac.
Psi i mačke su divni.
Kako se slažu vaš pas i mačka?
```

Eksperimentišite malo sa ovim programom tako što ćete menjati vrednosti True i False koje program dodeljuje svakoj promenljivoj da biste videli kako promene utiču na svaki od uslova.

Obrada više uslova pomoću naredbe Select

PRIMER Više povezanih uslova se najčešće ispituje pomoću naredbe If-Else. Međutim, Visual Basic .NET nudi i upravljačku strukturu Select koja olakšava ispitivanje složenih uslova. Na prvi pogled, naredba Select izgleda veoma slično grupi naredaba If-Else. Na primer, sledeća naredba Select prikazuje tekst koji zavisi od dana u nedelji:

```
Dim Dan As Integer
DayOfWeek = Now.DayOfWeek

Select Case Dan
  Case 0
    Console.WriteLine("Nedelja")
  Case 1
    Console.WriteLine("Ponedeljak")
  Case 2
    Console.WriteLine("Utorak")
  Case 3
    Console.WriteLine("Sreda")
  Case 4
    Console.WriteLine("Četvrtak")
  Case 5
    Console.WriteLine("Petak")
  Case 6
    Console.WriteLine("Subota")
End Select
```

Možete zadati i grupu rezultata koji ispunjavaju uslov.

```
Dim TestScore As Integer
TestScore = 84

Select Case TestScore
   Case is >= 90
     Console.WriteLine("Dobili ste 10")
   Case is >= 80
     Console.WriteLine("Dobili ste 8")
   Case is >= 70
     Console.WriteLine("Dobili ste 6")
   Case Else
     Console.WriteLine("Dobili ste 5")
End Select
```

Kao što vidite, kada zadajete uslov u naredbi Select, morate upotrebiti rezervisanu reč Is. Zapamtite i to da naredba Select podržava i slučaj Else koji se izvršava kada nijedan od prethodno definisanih uslova nije ispunjen. U naredbi Select možete definisati i opseg vrednosti:

```
Dim TestScore As Integer
TestScore = 84

Select Case TestScore
   Case  90 To 100
     Console.WriteLine("Dobili ste 10")
   Case 80 To 89
      Console.WriteLine("Dobili ste 8")
   Case is 70 To 79
      Console.WriteLine("Dobili ste 6")
   Case Else
      Console.WriteLine("Dobili ste 5")
End Select
```

U prvom primeru naučili ste kako da koristite naredbu Select za poređenje sa jednom vrednošću. Ponekad će vam trebati da izvršite istu naredbu za više vrednosti. Sledeća naredba Select prikazuje poruke u zavisnosti od tekućeg dana u nedelji. Za više dana prikazuje se ista poruka.

```
Dim Dan As Integer
DayOfWeek = Now.DayOfWeek

Select Case Dan
   Case 0, 6
      Console.WriteLine("Uživajte u vikendu")
   Case 1, 2, 3
      Console.WriteLine("Još puno dana do vikenda")
   Case 4, 5
      Console.WriteLine("Vikend samo što nije")
End Select
```

Ponavljanje grupa naredaba

Program ponekad mora da ponovi jednu ili više naredaba dok je neki uslov ispunjen, ili da ponovi neku naredbu ili skup naredaba određeni broj puta. Programeri to ponavljanje nazivaju iteracijom. U Visual Basicu .NET možete koristiti četiri iterativne upravljačke strukture: petlje For, While, For Each i Do While.

```
Dim I As Integer

For I = 1 To 5
   Console.WriteLine(I)
Next
```

Kad se ova petlja izvrši, na ekranu će biti prikazani sledeći rezultati:

```
1
2
3
4
5
```

Petlja For se sastoji iz tri dela. Prvi deo naredbe inicijalizuje upravljačku promenljivu petlje (poznatu i kao brojač petlje). Drugi deo upoređuje upravljačku promenljivu sa uslovom petlje. Treći deo koji nije obavezan, određuje za koliko se upravljačka promenljiva povećava ili smanjuje (korak petlje). Sledeća petlja prikazuje brojeve 0, 10, 20... do 100 povećavajući upravljačku promenljivu za 10 u svakom ciklusu.

```
For I = 0 To 100 Step 10
   Console.WriteLine(I)
Next
```

Petlje For nisu ograničene samo na cele brojeve. Možete koristiti i promenljive tipa Single i Double kao upravljačke promenljive u petlji. Sledeća petlja For prikazuje vrednosti od 0.0 do 1.0 povećavajući upravljačku promenljivu petlje za 0.1 u svakom ciklusu.

```
Dim X As Double
For X = 0.0 To 1.0 Step 0.1
  Console.WriteLine(X)
Next
```

Zadavanjem negativne vrednosti za korak možete smanjivati vrednost brojača u petlji For. Sledeća petlja For ispisuje vrednosti od 100 do 0 smanjujući upravljačku promenljivu za 10 u svakom ciklusu.

```
For I = 100 To 0 Step -10
   Console.WriteLine(I)
Next
```

Nasuprot petlji For, koja ponavlja jednu ili više naredaba određen broj puta, petlja While ponavlja naredbe sve dok je određeni uslov ispunjen. Na primer, petlju While možete koristiti da biste učitali i prikazali redove neke datoteke. U tom slučaju, naredba u petlji While učitavaće sadržaj datoteke sve dok ne pročita celu datoteku. Ili, petlja

While može prikazivati meni i obrađivati izbore sve dok korisnik ne odabere opciju Završetak. Format petlje While je sledeći:

```
While (Uslov)
  'Naredbe koje se ponavljaju
End While
```

Zapamtite da, za razliku o prethodnih verzija Visual Basica u kojima je naredba Wend označavala kraj petlje, u Visual Basicu .NET se koristi End While.

Naredba For Each omogućava da se ponavlja jedna ili više naredaba za sve elemente niza. U sledećem na primeru, pomoću naredbe For Each prikazuju se imena datoteka koje se nalaze u tekućem direktorijumu.

```
Dim Files As String() = Directory.GetFiles(".")
Dim Filename As String

For Each Filename In Files
  Console.WriteLine(Filename)
Next
```

I najzad, petlja Do je slična petlji While po tome što i ona omogućava ponavljanje jedne ili više naredaba sve dok je određeni uslov ispunjen. Međutim, za razliku od petlje While, u kojoj se uslov ispituje na početku petlje, u petlji Do uslov se proverava na kraju petlje. To znači da će se naredbe unutar petlje Do uvek izvršiti bar jednom.

```
Do
  'Naredbe koje se ponavljaju
Loop While (Uslov)
```

PRIMER Program koji sledi, LoopDemo.vb, koristi petlje For, While i Do While da bi prolazio kroz opseg vrednosti. Zatim kôd koristi petlju For Each da bi prikazao imena datoteka u tekućem direktorijumu.

```
Imports System.IO

Module Module1

    Sub Main()
        Dim I As Integer

        For I = 0 To 10
            Console.Write(I & " ")
        Next

        Console.WriteLine()

        Dim X As Double = 0.0
        While (X < 100)
            Console.Write(X & " ")
            X = X + 25
        End While

        Console.WriteLine()
```

```
        Do
            Console.Write(X & " ")
            X = X - 10
        Loop While(X < 0)

        Console.WriteLine()

        Dim Files As String() = Directory.GetFiles(".")
        Dim Filename As String

        For Each Filename In Files
            Console.WriteLine(Filename)
        Next
Console.WriteLine(Filename)
    End Sub
End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
0 1 2 3 4 5 6 7 8 9 10
0 25 50 75
100
.\LoopDemo.exe
.\LoopDemo.pdb
```

Izbegavanje beskonačnih petlji

Petlje kao što su For i While omogućavaju ponavljanje bloka naredaba zadati broj puta ili sve dok je ispunjen određeni uslov. Može se dogoditi da se petlja nikad ne završava (najčešće zbog greške u programu). Petlje koje se ne završavaju programeri zovu beskonačne petlje. Ukoliko ne prekinete program zatvaranjem prozora, takva petlja će se izvršavati „večno“. Na primer, cilj sledeće petlje While je da prikaže vrednosti od 0 do 99. Međutim, ako pogledate petlje, videćete da se u njoj ne povećava vrednost promenljive I. Posledica toga je da se nikad neće ispuniti uslov za završetak petlje (I jednako 100) tako da će se petlja beskrajno ponavljati.

```
Dim I As Integer = 0

While I <100
  Console.WriteLine(I)
End While
```

PRIMER Da biste smanjili mogućnost nastajanja beskonačne petlje, trebalo bi ispitate svaku petlju i proverite da li su sledeća četiri koraka ispravna:

1. Inicijalizacija upravljačke promenljive (brojača).
2. Ispitivanje vrednosti upravljačke promenljive.
3. Izvršavanje naredbe unutar petlje.
4. Menjanje vrednosti upravljačke promenljive.

Pogledajte prethodnu petlju While. Kôd inicijalizuje promenljivu I kada je deklariše. Zatim se u prvoj naredbi petlje While ispituje vrednost upravljačke promenljive. Unutar petlje While izvršava se naredba Console.WriteLine. Međutim, ne menja se vrednost upravljačke promenljive, što dovodi do beskonačne petlje.

Izlazak iz petlje tokom izvršavanja

Petlji bi trebalo da bude pridružen uslov koji određuje da li će se izvršiti (a kasnije i ponavljati) naredbe unutar petlje. Ponavljanje petlje tipa For završava se kada vrednost upravljačke promenljive petlje postane veća od završne vrednosti petlje.

PRIMER Ponekad treba bezuslovno da se prekine petlja For (ili For Each) pre njenog normalnog završetka. U takvim slučajevima, upotrebite naredbu Exit For koja nalaže Visual Basicu .NET da izađe iz petlje i nastavi izvršavanje programa od naredbe neposredno iza petlje For (to je prva naredba posle naredbe Next). Slično tome, da biste izašli pre vremena iz petlje While, treba da izvršite naredbu Exit While.

▶ *NAPOMENA*

U nastavku ovog poglavlja opisane su procedure, koje omogućavaju da grupišete naredbe koje obavljaju određeni zadatak. Procedure obično izvršavaju naredbe redom, od prve ka poslednjoj. Međutim, može se dogoditi da morate da izađete iz procedure pre nego što se izvrše sve njene naredbe. U takvim slučajevima upotrebite naredbu Exit Sub. Međutim, kao i u slučaju petlji For i While, zbog bolje čitljivosti koda trebalo bi izbegavati naredbe Exit kad god je moguće.

Skraćeno ispitivanje uslova

Performanse programa možete popraviti promenom načina na koji Visual Basic .NET obrađuje uslove u kojima se koriste logički operatori And i Or. U naredbama kao što su If, Select i While, logički operator And se koristi za povezivanje dva uslova koja moraju imati vrednost True da bi ceo uslov bio zadovoljen (logička konjunkcija). Sledeća naredba koristi logički operator And da bi utvrdila li je zaposleni programer i da li zna da koristi Visual Basic .NET:

```
If (JesteProgramer) And (PoznajeVB) Then
  'Naredbe
End If
```

Visual Basic .NET pojedinačno ispituje sve delove uslova kako bi utvrdio da li je uslov ispunjen. Međutim, u slučaju operatora And, ako je vrednost prvog dela uslova False, onda je i vrednost celog uslova False. Slično tome, u slučaju operatora Or, ako je vrednost prvog dela izraza True, onda je i vrednost celog izraza True. Da biste poboljšali performanse, možete iskoristiti prednosti operatora AndAlso i OrElse. Operator AndAlso nalaže Visual Basicu .NET da ispita drugi deo izraza ako i samo ako je vrednost prvog dela True.

Operator OrElse nalaže Visual Basicu .NET da ispita drugi deo izraza ako i samo ako je vrednost prvog dela False.

PRIMER Program koji sledi, LazyEvaluation.vb, prikazuje upotrebu operatora AndAlso i OrElse:

```
Module Module1
    Sub Main()
        Dim ImaKucu As Boolean = True
        Dim ImaMacu As Boolean = False

    If  ImaKucu And Also ImaMacu Then
        Console.WriteLine("Ima i psa i mačku")
    End If

    If ImaKucu OrElse ImaMacu Then
        Console.WriteLine("Ima psa ili mačku ili možda oboje")
    End If

    Console.ReadLine()
    End Sub
End Module
```

Možda se pitate zašto Visual Basic .NET ne primenjuje uvek skraćeno ispitivanje. Razlog je to što ako se ne obradi drugi deo uslova, ponekad može doći do greške. Pogledajte sledeću naredbu If koja poziva funkciju u oba dela uslova:

```
If (NijeRadnoVreme() And VrataZatvorena()) Then
```

U zavisnosti od toga šta radi funkcija VrataZatvorena, možda nećete želeti da program izostavi pozivanje te funkcije samo zato što je funkcija NijeRadnoVreme vratila rezultat False (što bi se inače desilo prilikom skraćenog ispitivanja).

Prelamanje dugih naredaba

Kao što možete videti u primerima programa u ovoj knjizi, mnogo puta su duge naredbe prelomljene u dva ili više redova zbog ograničenih dimenzija stranica. Kada pišete program, nekad ćete želeti da podelite naredbu u dva ili tri reda da ne biste morali da pomerate horizontalno sadržaj prozora kako biste videli završetak naredbe.

PRIMER Da biste prelomili naredbu u više redova, upišite razmak a zatim znak za podvlačenje (_) na kraju reda, kao što je prikazano ovde:

```
NekoVeomaDugoImePromenljive = NekoDugoImeProcedure(ParametarJedan, _
  ParametarDva, ParametarCetiri, ParametarPet)
```

Programeri duge znakovne nizove često pišu u dva ili više redova. Da biste prelomili znakovni niz u dva ili više redova, podelite ga u više kraćih nizova koji se nalaze u zasebnim redovima. Između svakog znakovnog niza postavite operator nadovezivanja (&).

```
Console.WriteLine("Naziv knjige je : Visual Basic .NET kroz" & _
" praktične primere")
```

Kada napišete znakovni niz na ovakav način, nemojte zaboraviti da unesete znak za razmak na kraju jednog znakovnog niza ili na početak narednog ako je i početni znakovni niz sadržao razmake.

Upotreba operatora za dodeljivanje

Aritmetičke operacije se često koriste za promenu vrednosti promenljive. U sledećoj naredbi vrednost 1 se dodeljuje promenljivoj Counter.

```
Counter = Counter + 1
```

PRIMER Da bi pojednostavio operacije u kojima se u izrazu koristi tekuća vrednost promenljive i zatim se rezultat izraza dodeljuje istoj promenljivoj, Visual Basic .NET nudi skup operatora za dodeljivanje prikazanih u tabeli 1-8.

Operator	Namena
+=	Dodaje zadati izraz tekućoj vrednosti promenljive.
-=	Oduzima zadati izraz od tekuće vrednosti promenljive.
*=	Množi tekuću vrednost promenljive zadatim izrazom, a zatim rezultat dodeljuje promenljivoj.
/= i \=	Deli tekuću vrednost promenljive zadatim izrazom, a zatim rezultat dodeljuje promenljivoj.
^=	Stepenuje tekuću vrednost promenljive zadatim izrazom, a zatim rezultat dodeljuje promenljivoj.
&=	Spaja znakovni niz sa promenljivom, a zatim rezultat dodeljuje promenljivoj.

Tabela 1-8 Operatori za dodeljivanje u Visual Basicu .NET

U sledećoj naredbi prikazana je upotreba operatora koji povećava vrednost promenljive Counter za 1.

```
Counter += 1
```

U sledećem programu, AssignmentDemo.vb, prikazana je upotreba operatora za dodeljivanje Visual Basica .NET:

```
Module Module1

    Sub Main()
        Dim A As Integer

        A = 0
        A += 10
        Console.WriteLine("A += 10 daje " & A)

        A -= 5
        Console.WriteLine("A -= 5 daje " & A)

        A *= 3
        Console.WriteLine("A *= 3 daje " & A)
```

```
        A /= 5
        Console.WriteLine("A /= 5 daje " & A)

        A ^= 2
        Console.WriteLine("A ^= 2 daje " & A)

        Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
A += 10 daje 10
A -= 5 daje 5
A *= 3 daje 15
A /= 5 daje 3
A ^= 2 daje 9
```

Upisivanje komentara u kôd programa

Kada programirate, trebalo bi da u kôd dodajete komentare koji opisuju program i svrhu pojedinih promenljivih. Kada kasnije neko drugi bude čitao kôd, na osnovu komentara shvatiće šta program radi. Da biste napisali komentar u Visual Basicu .NET, upišite apostrof i zatim tekst komentara u istom redu. Visual Basic .NET će zanemariti tekst desno od apostrofa smatrajući ga za komentar.

PRIMER Komentare možete pisati u zasebnim redovima ili desno od naredbe:

```
' Sledeća procedura prikazuje tekući datum i vreme
Sub ShowDateTime()
  Console.WriteLine(Now())'Now daje tekući datum i vreme
End Sub
```

Kada testirate program, privremeno ćete isključivati neke naredbe. Umesto da ih brišete, samo upišite apostrof na početku naredaba. Kada Visual Basic .NET prevodi program on će ih zanemariti, ili tačnije rečeno, tumačiće ih kao komentare. Ako kasnije poželite da ponovo aktivirate komande, samo izbrišite apostrof.

```
Console.WriteLine("Ovaj red će se izvršiti")
'Console.WriteLine("Ovaj red neće")
```

Učitavanje znakova sa tastature pomoću metoda Console.Read i Console.ReadLine

Konzolna aplikacija učitava znakove koje korisnik upisuje pomoću tastature metodama Console.Read i Console.ReadLine. Razlika između te dve metode je u tome što Console.ReadLine vraća sve unete znakove do pritiska na taster <ENTER>, dok Console.Read

čita znakove do prvog razmaka, tj. pritisnutog tastera Tab ili razmaknice. Da biste vrednost koju je otkucao korisnik dodelili promenljivoj, upotrebite operator za dodeljivanje na sledeći način:

```
ImePromenljive = Console.ReadLine()
```

U mnogim primerima u ovoj knjizi naredba Console.ReadLine() nalazi se na kraju programa. Kada izvršavate konzolnu aplikaciju u Visual Studiju, konzolni prozor se zatvara čim program završi rad. Ako upišete naredbu Console.ReadLine() na kraju, program će se zaustaviti i čekati da korisnik pritisne taster <ENTER> da bi zatvorio prozor.

PRIMER Sledeći program, OutputDemo.vb prikazuje upotrebu metoda Console.Read i Console.ReadLine:

```
Module Module1

    Sub Main()
        Dim Age As Integer
        Dim FirstName As String
        Dim Salary As Double

        Console.Write("Godine: ")
        Age = Console.ReadLine()

        Console.Write("Ime: ")
        FirstName = Console.ReadLine()

        Console.Write("Plata: ")
        Salary = Console.ReadLine()
        Console.WriteLine(Godine & " " Ime & " " Plata)
        Console.Write("Unesite godine, ime i platu: ")
        Age = Console.Read()
        FirstName = Console.Read()
        Salary = Console.ReadLine()
        Console.WriteLine(Age & " " & Firstname & " " & Salary)

    End Sub
EndModule
```

Provedite neko vreme eksperimentišući sa ovim programom. Ukoliko otkucate vrednost koja ne odgovara očekivanom tipu podataka (na primer, ako program zahteva da unesete godine, a vi otkucate ime umesto godina), nastaće izuzetak i program će biti prekinut. U poglavlju 9 detaljno su objašnjeni izuzeci i obrada izuzetaka. Da bi se izbegle takve greške, mnogi programi učitavaju sve znakove sa tastature u jedan znakovni niz, koji zatim pretvaraju u promenljivu tipa Integer ili Double po potrebi.

Prikazivanje teksta u prozorima za poruke

Konzolne aplikacije prikazuju izlazne podatke u konzolnom prozoru pomoću metoda Console.Write i Console.WriteLine. Slično tome, Windowsovi programi prikazuju rezultate pomoću jedne ili više kontrola postavljenih na obrascu. Međutim, obe vrste programa

(i konzolni i Windowsovi programi) mogu koristiti prozore za poruke, da bi prikazali obaveštenje i tražili odgovor korisnika u obliku pritiska na dugme (kao što su OK ili Cancel).

Programeri na Visual Basicu koriste funkciju MsgBox da bi prikazali prozor sa porukom korisniku:

```
MsgBox("Hello, User")
```

Iako Visual Basic .NET podržava funkciju MsgBox, preporučljivo je da u novim Windowsovim programima koristite metodu Show klase MessageBox kako biste prikazali prozor sa porukom (u stvari funkcija MsgBox poziva metodu MessageBox.Show).

```
MessageBox.Show("Hello, User")
```

Da biste utvrdili koje je dugme u prozoru za poruku korisnik pritisnuo, dodelite rezultat metode MessageBox.Show promenljivoj, kao što je prikazano ovde:

```
ImePromenljive = MessageBox.Show("Message", "Title", _
   MessageBoxButtons.OKCancel)
```

PRIMER Program koji sledi, MsgBoxDemo.vb, prikazuje upotrebu funkcije MsgBox u konzolnoj aplikaciji:

```
Module Module1
  Sub Main()
     MsgBox("Message")
     MsgBox("Message", MsgBoxStyle.AbortRetryIgnore, "Title")
  End Sub
End Module
```

Program koji sledi, MsgBoxDemo.vb prikazuje upotrebu različitih tipova prozora za poruke:

```
Public Class Form1
    Inherits Systems.Windows.Forms.Form

#Region " Windows Form Designer generated code "
   'Kod nije prikazan
#End Region

  Private Sub Form1_Load(ByVal sender As System.Object, _
   ByVal e As System.EventArgs) Handles MyBase.Load
      MessageBox.Show("Message")
      MessageBox.Show("Message", "Title")
      MessageBox.Show("Message", "Title" MessageBoxButtons.OKCancel)
      MessageBox.Show("Message", "Title",_
        MessageBoxButtons.AbortRetryIgnore, MessageBoxIcon.Warning)
      Me.Close()
  End Sub
End Class
```

Prihvatanje podataka od korisnika pomoću funkcije Input Box

Konzolne aplikacije preuzimaju od korisnika znakove koje korisnik kuca na tastaturi upotrebom metode Console.ReadLine. Slično tome, Windowsove aplikacije čitaju podatke od korisnika pomoću jedne ili više kontrola postavljenih na obrascima. To znači da obe vrste aplikacija mogu koristiti funkciju InputBox da bi zahtevale podatke od korisnika, kao što je prikazano na slici 1-7. Tekst koji je uneo korisnik možete upisati u promenljivu, kao što je prikazano dole:

```
VariableName = InputBox("Enter name")
```

Slika 1-7 Upotreba funkcije InputBox koja korisniku omogućava da unese podatak.

PRIMER Funkciji InputBox možete proslediti parametre pomoću kojih zadajete tekst koji se prikazuje u prozoru, naslov prozora, horizontalno i vertikalno rastojanje prozora (od gornjeg levog ugla ekrana). Program koji sledi, InputBoxDemo.vb, koristi funkciju InputBox:

```
Module Module1

    Sub Main()
        Dim Name As String
        Dim Age As Integer
        Dim Salary As Double

        Name = InputBox("Unesite ime")
        Age = Input Box("Unesite godine")
        Salary = Input Box("Unesite platu")

        Console.WriteLine(Name)
        Console.WriteLine(Age)
        Console.WriteLine(Salary)
    End Sub

End Module
```

Kao i u prethodnom primeru, korisnik mora uneti vrednost koja po formatu odgovara tipu promenljive kojoj dodeljujete rezultat funkcije. Ako, na primer, dodeljujete rezultat promenljivoj tipa Integer a korisnik ne unese numerički podatak, nastaće izuzetak. U poglavlju 9 detaljno su objašnjeni izuzeci i obrada izuzetaka. Da biste izbegli takve greške, dodelite vrednost funkcije InputBox promenljivoj tipa String, a zatim je pretvorite u odgovarajući tip.

Deljenje poslova

Kada je program obiman i složeniji, programeri ga često razbijaju na manje blokove koda koji obavljaju određene zadatke, pošto se manjim blokovima lakše upravlja. Da bi organizovali programe po zadacima, programeri koriste procedure i funkcije.

Razlika između procedura i funkcija jeste u tome što funkcija vraća rezultat a procedura to ne čini. Na primer, funkcija MessageBox.Show prikazuje okvir za dijalog i nakon toga vraća vrednost u zavisnosti od toga koje je dugme korisnik pritisnuo. Da biste u kodu upotrebili vrednost funkcije, dodelite rezultat promenljivoj, kao što je prikazano ovde:

```
Promenljiva = ImeFunkcije(OpcioniParametri)
```

Da biste napravili proceduru, navedite rezervisanu reč Sub, zatim jedinstveno ime procedure (trebalo bi da opisuje namenu procedure), zatim zagrade u kojima se mogu nalaziti promenljive koje omogućavaju prosleđivanje vrednosti proceduri (programeri ih nazivaju parametrima). Ako procedura nema parametre, napišite prazne zagrade iza imena procedure. Zatim upišite naredbe procedure, i na kraju, upišite naredbu End Sub, kao što je prikazano ovde:

```
Sub ImeProcedure ()
  ' Ovde upisati naredbe
End Sub
```

Sledeće naredbe čine proceduru GreetUser, koja prikazuje poruku korisniku pomoću metode Console.WriteLine.

```
Sub GreetUser()
  Console.WriteLine("Zdravo")
  Console.WriteLine("Tekući datum i vreme je: " & Now())
  Console.WriteLine("Želim vam prijatan dan.")
End Sub
```

Da biste pozvali proceduru, upišite njeno ime a zatim zagrade, kao što je prikazano ovde:

```
GreetUser()
```

Postupak upotrebe procedura programeri zovu „pozivanje procedura“. Kada program naiđe na poziv procedure, on prelazi na prvu naredbu te procedure. Kada završi izvršavanje procedure, program nastavlja da se izvršava od prve naredbe koja sledi poziv procedure.

▶ *NAPOMENA*

Visual Basic .NET ne podržava naredbu GoSub koju su programeri nekada veoma često koristili za pozivanje procedura.

Što je program obimniji, imaće veći broj procedura. Program za obradu teksta, na primer, može pozvati jednu proceduru da bi proverio pravopis nekog dokumenta, drugu da bi snimio dokument a treći da bi ga štampao. U slučaju konzolnih aplikacija, izvršavanje

glavnog programa uvek počinje od prve naredbe u proceduri Main. (Proceduru Main možete shvatiti kao kôd koji sadrži glavne ili početne naredbe). Iz procedure Main program može pozivati ostale procedure.

```
Module Module 1

    Sub ShowBookInformation()
        Console.WriteLine ("Naslov: VisualBasic .NET kroz "& _
           praktične primere")
        Console.WriteLine("Autor: Jamsa")
        Console.WriteLine("Izdavač: Mikro knjiga")
        Console.WriteLine("Cena: 864.00")
    End Sub

    Sub GreetInEnglish()
        Console.WriteLine("Hello, world")
    End Sub

    Sub GreetInSpanish()
        Console.WriteLine("Hola, mundo")
    End Sub

    Sub ShowTime()
        Console.WriteLine("Trenutno je: " & Now)
    End Sub

    Sub Main()
        ShowTime
        GreetInEnglish
        GreetInSpanish
        ShowBookInformation
        Console.ReadLine()
    End Sub

End Module
```

Kao što vidite, u programu su definisane četiri procedure, čija se tela nalaze između naredaba Sub i End Sub. Program će početi da se izvršava od procedure Main. Kada program naiđe na poziv procedure ShowTime, on prelazi na prvu naredbu u toj proceduri – u ovom slučaju to je naredba Console.WriteLine koja ispisuje tekući datum i vreme. Kada se procedura završi, program će nastaviti da se izvršava od prve naredbe koja sledi iza poziva procedure ShowTime, a to je poziv procedure GreetInEnglish. Program će onda početi da izvršava naredbe te procedure. Program će nastaviti da poziva ostale procedure i da se vraća u proceduru Main sve dok ne izvrši sve naredbe u proceduri Main.

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
Trenutno je 4/9/2002 10:46:47 AM
Hello, world
Hola, mundo
Naslov: Visual Basic .NET kroz praktične primere
```

```
Autor: Jamsa
Izdavač: Mikro knjiga
Cena: 49.99
```

Kao što je već napomenuto, funkcije se razlikuju od procedura jer vraćaju rezultat koji se može dodeljivati promenljivama ili koristiti u izrazima. U poglavlju 5 opisane su aritmetičke funkcije koje se nalaze u klasi Math. Sledeće naredbe prikazuju upotrebu nekih funkcija klase Math:

```
Dim NekaVrednost, NekiUgao As Double

NekaVrednost = Math.Sqrt(100)
NekiUgao = Math.Acos(2.225)
```

Kada program naiđe na poziv funkcije Math.Sqrt, izvršiće se odgovarajuće naredbe funkcije. Kada se ta funkcija završi, rezultat se dodeljuje promenljivoj NekaVrednost. Zatim će program nastaviti da se izvršava od sledeće naredbe, što je u ovom slučaju funkcija Acos klase Math.

Da biste napravili funkciju, napišite rezervisanu reč Function iza koje sledi jedinstveno ime funkcije i par zagrada u kojima možete deklarisati parametre. Morate upisati i tip (na primer, Integer ili String) vrednosti koju funkcija vraća. Naredbe koje čine kôd funkcije upišite između zaglavlja funkcije (koje programeri zovu i potpis funkcije) i naredbe End Function, kao što je prikazano dole:

```
Function ImeFunkcije(OpcioniParametri) As_
TipPovratneVrednostiFunkcije
    'Naredbe koje čine kôd funkcije dolaze ovde
End Function
```

Sledeće naredbe čine funkciju GetBookPrice koja vraća rezultat tipa Double:

```
Function GetBookPrice() As Double
   'Naredbe koje čine kôd funkcije dolaze ovde
End Function
```

Za prosleđivanje povratne vrednosti funkcije može se koristiti naredba Return ili se povratna vrednost funkcije može dodeliti imenu funkcije. U funkciji GetBookPrice, sledeće naredbe vraćaju vrednost 864,00:

```
Return 864.00

GetBookPrice = 864.00
```

Da biste pozvali funkciju upišite ime funkcije praćeno zagradama i parametrima (ako ih funkcija ima). Rezultat funkcije se obično dodeljuje nekoj promenljivoj, kao što je prikazano ovde:

```
Dim Price As Double

Price = GetBookPrice()
```

Vrednost koju funkcija vraća možete koristiti u svakom izrazu, npr. A = B + NekaFunkcija() ili u pozivu metode Console.WriteLine, kao što je prikazano ovde:

```
Console.WriteLine("Cena je: " & GetBookPrice())
```

Program FunctionDemo.vb sadrži dve funkcije, jednu koja vraća vrednost tipa Double i drugu koja vraća vrednost tipa String. Program najpre poziva obe funkcije dodeljujući rezultate promenljivama. Zatim poziva svaku funkciju u metodi Console.WriteLine kako bi prikazao vrednost funkcija. Na kraju, funkcija GetBookPrice koristi se u naredbi If.

```
Module Module1
    Function GetBookPrice () As Double
        GetBookPrice = 864.00
    End Function

    Function GetBookTitle() As String
        GetBookTitle = "Programiranje u Visual Basicu"_
        &" .NET kroz praktične primere"
    End Function

    Sub Main()
        Dim Price As Double
        Dim Title As String
        Price = GetBook Price()
        Title = GetBookTitle()

        Console.WriteLine(Price)
        Console.WriteLine(Title)

        Console.WriteLine(GetBookPrice())
        Console.WriteLine(GetBookTitle())

        If  (GetBookPrice() = 864.00) Then
             Console.WriteLine("Cena knjige je 864.00")
        End If

        Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
864.00
Visual Basic .NET kroz praktične primere
864.00
Visual Basic .NET kroz praktične primere
Cena knjige je 864.00
```

NAPOMENA

Kada čitate knjige o Visual Basicu .NET, često ćete zapaziti da se u tim knjigama reči funkcija, procedura i metoda koriste kao sinonimi. Ne zaboravite da se funkcija razlikuje od procedure po tome što funkcija vraća rezultat. Reč metoda je opšti izraz koji se odnosi i na funkcije i na procedure koje postoje u klasama. Klase se detaljno razmatraju u poglavlju 3.

Prosleđivanje parametara procedurama i funkcijama

Procedure i funkcije služe za organizovanje naredaba koje obavljaju određeni zadatak. Vrednosti koje se prosleđuju procedurama zovu se parametri. Na primer, u prethodnom delu ovog poglavlja prosledili ste vrednosti metodi Console.WriteLine da bi ih ona prikazala:

```
Console.WriteLine("Vrednosti su {0}, {1}, {3}", 100, 200, 300)
Console.WriteLine("Cena je: " & GetBookPrice())
```

Na sličan način ste prosleđivali vrednosti i metodi MessageBox.Show:

```
MessageBox.Show("Hello, World")
```

Da biste prosledili vrednosti proceduri ili funkciji morate ih upisati između zagrada iza imena procedure i razdvojiti vrednosti zarezima. Da bi procedura ili funkcija mogla da koristi prosleđene vrednosti, treba deklarisati promenljive u koje će one biti smeštene. Promenljive deklarišite u zagradama iza imena procedure. Naredbe koje slede čine proceduru Pozdrav. Program joj prosleđuje poruku (parametar) koju procedura prikazuje:

```
Sub Pozdrav(ByVal Poruka As String)
  Console.WriteLine("Poštovani korisniče,")
  Console.WriteLine("Današnja poruka je: " & Poruka)
End Sub
```

U prethodnom primeru procedure deklarisana je promenljiva tipa String koja se zove Poruka. Za sada zanemarite rezervisanu reč ByVal koja će biti objašnjena u narednom primeru. Procedura Pozdrav poziva se na sledeći način:

```
Pozdrav("Želimo vam prijatan dan")
```

Procedurama i funkcijama mora se proslediti tačan broj i tip parametara. Ako funkcija očekuje parametar tipa Double, a program joj prosledi vrednost tipa String, doći će do greške.

Program koji sledi, ThreeSubs.vb, sastoji se od nekoliko procedura s različitim brojem i tipovima parametara:

```
Module Module1

    Sub OneValue(ByVal Name As String)
        Console.WriteLine("Hello, " & Name)
    End Sub

    Sub TwoValues(ByVal Age As Integer, ByVal Name As String)
        Console.WriteLine("Godine: " & Age)
        Console.WriteLine("Ime: " & Name)
    End Sub

    Sub ThreeValues(ByVal Name As String, ByVal Age As Integer, _
          ByVal Salary As Double)
        Console.WriteLine("Godine: " & Age)
        Console.WriteLine("Ime: " & Name)
        Console.WriteLine("Plata: " & Salary)
    End Sub
```

```
    Sub Main()
        OneValue("Mr Gates")
        Console.WriteLine()
        TwoValues(50, "Mr Gates")
        Console.WriteLine()
        ThreeValues("Mr Gates", 50, 250000.0)
        Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
Hello, Mr Gates

Godine: 50
Ime: Mr Gates

Ime: Mr Gates
Godine: 50
Plata: 250000
```

Deklarisanje lokalnih promenljivih u procedurama

Iza zaglavlja procedure možete deklarisati promenljive, koje programeri zovu lokalnim promenljivama, kao što je prikazano u sledećem primeru:

```
Sub NekaFunkcija()
  Dim I As Integer
  Dim Zbir As Double

  ' Ovde dolaze naredbe
End Sub
```

Programeri zovu promenljive deklarisane u procedurama lokalnim zato što su postojanje promenljive i njena vrednost poznati samo proceduri. Ostali deo koda ne zna da u proceduri postoje lokalne promenljive, niti ih može koristiti. Pošto su promenljive lokalne, imena koja im date neće izazvati dvosmislenost čak i ako u drugim procedurama postoje istoimene promenljive.

Sledeći program, LocalVarDemo.vb, definiše lokalne promenljive u procedurama. Iako promenljive imaju ista imena u svakoj proceduri, svaka lokalna promenljiva je zasebna i vrednosti u jednoj proceduri ne utiču na vrednosti promenljivih u drugoj.

```
Module Module1

    Sub YahooInfo()
        Dim Name As String = "Yahoo"
        Dim Price As Double = 17.45
        Dim I As Integer = 1001
```

```
        Console.WriteLine("U proceduri YahooInfo")
        Console.WriteLine("Ime: " & Name)
        Console.WriteLine("Cena: " & Price)
        Console.WriteLine("I: " & I)
    End Sub

    Sub BookInfo()
        Dim Name As String = "C # kroz praktične primere"
        Dim Price As Double = 49.99
        Dim I As Integer = 0

        Console.WriteLine("U proceduri BookInfo")
        Console.WriteLine("Ime: " & Name)
        Console.WriteLine("Cena: " & Price)
        Console.WriteLine("I: " & I)
    End Sub

    Sub Main()
        YahooInfo()
        Console.WriteLine()
        BookInfo()
        Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
U proceduri YahooInfo
Ime: Yahoo
Cena: 17.45
I: 1001

U proceduri BookInfo
Ime: C# kroz praktične primere
Cena: 49.99
I: 0
```

Menjanje vrednosti parametara unutar procedure

U odeljku „Prosleđivanje parametara procedurama i funkcijama“ naučili ste kako da prosledite parametar proceduri ili funkciji. U svim primerima koji su dosad bili prikazani, procedure su imale parametre ali nisu menjale njihove vrednosti. Ako pogledate i procedure koje su prikazane ranije, videćete da su u svakoj od njih parametri deklarisani sa rezervisanom reči ByVal.

```
Sub NekaProcedura(ByVal A As Integer)
```

Rezervisana reč ByVal znači da se parametri funkciji prosleđuju po vrednosti. Kada program pozove proceduru, Visual Basic .NET će napraviti kopiju vrednosti parametra. Zatim će proceduri proslediti tu kopiju, a ne izvornu promenljivu. Kada prosleđujete

parametre po vrednosti, procedura ne može izmeniti vrednost parametra. Pogledajte sledeći program, ByValueDemo.vb, u kojem se proceduri prosleđuje vrednost numeričkog tipa. Nakon što se izvrši procedura, promenljiva i dalje ima vrednost 100. Parametar se prosleđuje po vrednosti, a lokalna vrednost važi samo dok se procedura ili funkcija izvršava.

```
Module Module1

    Sub NoChangeToParameter(ByVal A As Integer)
        A = 1001
        Console.WriteLine("Vrednost A u proceduri: " & A)
    End Sub

    Sub Main()
        Dim Number As Integer = 100

        Console.WriteLine("Broj pre poziva funkcije: " & Number)
        NoChangeToParameter(Number)
        Console.WriteLine("Broj posle poziva funkcije: " & Number)
        Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
Vrednost pre pozivanja procedure: 100
Vrednost A u proceduri: 1001
Vrednost posle pozivanja procedure: 100
```

PRIMER Da bi procedura mogla da promeni vrednost prosleđene promenljive, procedura mora znati njenu memorijsku lokaciju. Da bi procedura znala adresu parametra, morate joj proslediti parametar po referenci. Ispred imena promenljive upišite rezervisanu reč ByRef. Sledeći program, ByRefDemo.vb, prosleđuje parametar po referenci proceduri ParameterChange. Unutar procedure menja se, a zatim prikazuje vrednost parametra. Pošto se vrednost prosleđuje po referenci, promena vrednosti važi i posle završetka procedure.

```
Module Module1

    Sub ParameterChange(ByRef A As Integer)
        A = 1001
        Console.WriteLine("Vrednost A u proceduri: " & A)
    End Sub

    Sub Main()
        Dim Number As Integer = 100

        Console.WriteLine("Vrednost pre pozivanja procedure: "
        & Number)
        ParameterChange(Number)
```

```
        Console.WriteLine("Vrednost posle pozivanja procedure: "
        & Number)
        Console.ReadLine()
    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
Vrednost pre pozivanja procedure: 100
Vrednost A u proceduri: 1001
Vrednost posle pozivanja procedure: 1001
```

Zadavanje dosega da bi se odredile lokacije u programu gde promenljiva ima značenje

Promenljive možete deklarisati u procedurama kao lokalne, kao parametre procedura i u upravljačkim strukturama kao što su naredbe If ili While. U zavisnosti od mesta gde deklarišete promenljivu, menjaće se i mesto u programu gde promenljiva ima značenje (drugačije rečeno, naredbe u programu koje mogu koristiti promenljivu). Delove programa gde promenljiva ima značenje programeri nazivaju doseg promenljive. Pretpostavimo da imate dve procedure i da se u obe koristi promenljiva Counter, kao što je prikazano:

```
Sub BigLoop()
  Dim Counter As Integer

  For Counter = 1000 To 10000
    Console.WriteLine(Counter)
  Next
End Sub

Sub LittleLoop()
  Dim Counter As Integer

  For Counter = 0 To 5
    Console.WriteLine(Counter)
  Next
End Sub
```

Kada deklarišete lokalnu promenljivu u proceduri, doseg takve promenljive (mesta gde promenljiva ima značenje) ograničen je na samu proceduru. Izvan svake od ove dve procedure program ne zna da postoji promenljiva Counter. U ovom slučaju postoje dve promenljive Counter i imaju različit doseg. Usled različitog dosega obe promenljive upotreba istog imena promenljive u različitim procedurama ne izaziva dvosmislenost.

Kada pravite konzolnu aplikaciju, promenljivu možete deklarisati izvan procedura. U sledećem programu deklarisana je promenljiva Counter izvan obe procedure.

```
Dim Counter As Integer

Sub BigLoop()
  For Counter = 1000 To 10000
    Console.WriteLine(Counter)
  Next
End Sub

Sub LittleLoop()
  For Counter = 0 To 5
    Console.WriteLine(Counter)
  Next
End Sub
```

Kada deklarišete promenljivu izvan procedure, promenljiva ima globalni doseg, odnosno vidljiva je u celom programskom kodu. Svaka procedura u programu može promeniti vrednost globalne promenljive (što može dovesti do grešaka koje se vrlo teško otkrivaju kada ne očekujete da procedura menja vrednost promenljive). Zato ne bi trebalo da koristite globalne promenljive (ili bi bar trebalo ozbiljno ograničiti njihovu upotrebu).

Kada lokalna promenljiva ima isto ime kao globalna, koristi se lokalna promenljiva (i zanemaruje globalna). U našem primeru, promena vrednosti lokalne promenljive Counter neće uticati na globalnu promenljivu Counter, i obrnuto. Međutim, pošto takve situacije mogu zbuniti programera koji čita kôd, trebalo bi izbegavati upotrebu globalnih promenljivih. Ako procedura mora promeniti vrednost promenljive, treba joj proslediti parametar po referenci (upotrebom rezervisane reči ByRef u deklaraciji parametara procedure). Na taj način, lakše se vidi da procedura menja vrednost parametra.

PRIMER Program koji sledi, ScopeDemo.vb, prikazuje kako doseg utiče na promenljive. U programu se koriste globalna promenljiva Counter, lokalna promenljiva u proceduri koja se takođe zove Counter i promenljiva Counter čiji je doseg naredba If.

```
Module Module1

    Dim Counter As Integer

    Sub BigLoop()
        For Counter = 1000 To 1005 'koristi globalnu promenljivu Counter
            Console.Write(Counter & " ")
        Next
            Console.WriteLine()
    End Sub

    Sub LittleLoop()
        Dim Counter As Integer

        For Counter = 0 To 5 'koristi lokalnu promenljivu Counter
            Console.Write(Counter & " ")
        Next
        Console.WriteLine()
    End Sub
```

```
    Sub Main()
        Counter = 100

        Console.WriteLine("Counter na početku: " & Counter)
        BigLoop()
        Console.WriteLine("Counter posle procedure BigLoop: " &_
        Counter)
        LittleLoop()
        Console.WriteLine("Counter posle procedure LittleLoop: " &_
        Counter)

        If (Counter > 1000) Then
            Dim Counter As Integer = 0

            Console.WriteLine("Counter u naredbi If: " & Counter)
        End If

        Console.WriteLine("Counter na kraju: " & Counter)

        Console.ReadLine()

    End Sub

End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
Counter na početku: 100
1000 1001 1002 1003 1004 1005
Counter posle procedure BigLoop: 1006
0 1 2 3 4 5
Counter posle procedure LittleLoop: 1006
Counter u naredbi If: 0
Counter na kraju : 1006
```

Čuvanje više vrednosti istog tipa u jednoj promenljivoj

Promenljive omogućavaju da programi skladište i učitavaju vrednosti dok se program izvršava. Promenljive obično sadrže po jednu vrednost u datom trenutku. Međutim, mnogi programi rade sa više povezanih vrednosti istog tipa. Na primer, takav je program koji treba da izračuna prosečnu vrednost 50 rezultata na ispitu, da promeni cene stotinak artikala ili da izračuna prostor na disku koji je zauzet datotekama u tekućem direktorijumu. Da biste uskladištili više vrednosti istog tipa (npr. 50 vrednosti tipa Integer) u jednoj promenljivoj, možete upotrebiti strukturu podataka u obliku memorijskog niza.

Da biste napravili niz prvo deklarišite promenljivu odgovarajućeg tipa, a zatim broj elemenata koji ćete u njoj čuvati. U sledećeoj naredbi deklariše se niz Ocena u kome možete čuvati 50 vrednosti tipa Integer (prvi element niza označen je sa 0, a pedeseti sa 49).

```
Dim Ocena(49) As Integer
```

Vrednostima u nizu pristupa se zadavanjem indeksa elementa niza (zadavanjem lokacije tražene vrednosti). Prva vrednost u nizu nalazi se na lokaciji 0. U sledećoj naredbi dodeljuje se ocena na ispitu prvom elementu u nizu:

```
Ocene(0) = 91
```

U sledećim naredbama dodeljuju se vrednosti za prvih pet elemenata u nizu Ocene:

```
Ocene(0) = 91
Ocene(1) = 44
Ocene(2) = 66
Ocene(3) = 95
Ocene(4) = 77
```

Elementima niza može da se pristupa pomoću petlje For čiji brojač zadaje lokaciju u nizu. Sledeća petlja For prikazuje vrednosti prvih pet elemenata niza.

```
For I = 0 To 4
  Console.WriteLine(Ocene(I))
Next
```

Kao što je rečeno, prvi element niza nalazi se na lokaciji 0. U slučaju niza od 50 elemenata kao što je Ocene, poslednji element bio bi Ocene(49). Ako program pokuša da dodeli vrednost izvan opsega niza, nastaće izuzetak. U poglavlju 9 detaljno su objašnjeni izuzeci i obrada izuzetaka.

Kada dodeljuje vrednosti elementima niza, program može, na primer, učitavati vrednosti iz datoteke, zahtevati da ih korisnik unese ili računati vrednosti po određenoj formuli. Osim toga, program može popuniti niz početnim vrednostima koje su upisivane između vitičastih zagrada, kao što je prikazano ovde (kada inicijalizujete niz na ovaj način, ne zadajete veličinu niza u zagradama iza imena niza):

```
Dim Values() As Integer = {100, 200, 300, 400, 500}
Dim Prices() As Integer = {25.5, 4.95, 33.4}
```

Tokom rada programa dešavaće se da ponestane mesta u nizu za sve potrebne vrednosti. U takvim slučajevima možete koristiti naredbu ReDim koja povećava ili smanjuje dimenzije niza.

Sledeći program, ArrayDemo.vb, pravi niz u koji upisuje deset vrednosti. Nakon toga, pomoću petlje For prikazuje sadržaj niza. Niz u Visual Basicu .NET je objekat u kojem se čuvaju podaci o nizu (kao što je broj elemenata koji niz sadrži). Takav objekat ima i metode koje možete koristiti za rad sa elementima niza. U ovom programu prikazano je nekoliko svojstava i metoda te klase:

```
Module Module1

    Sub Main()
        Dim Values() As Integer = {100, 200, 300, 400, 500}
        Dim MyValues(5) As Integer
        Dim Prices() As Double = {25.5, 4.95, 33.4}

        Dim I As Integer
```

```
        For I = 0 To 4
            Console.Write(Values(I) & " ")
        Next
        Console.WriteLine()

        ' Kopiranje jednog niza u drugi
        Values.CopyTo(MyValues, 0)
        For I = 0 To 4
            Console.Write(MyValues(I) & " ")
        Next
        Console.WriteLine()

        Values.Reverse(Values)
        For I = 0 To 4
            Console.Write(Values(I) & " ")
        Next
        Console.WriteLine()

        Console.WriteLine("Veličina niza: " & Values.Length)
        Console.WriteLine("Donja granica niza: " & _
          Values.GetLowerBound(0))
        Console.WriteLine("Gornja granica niza: " & _
          Values.GetUpperBound(0))

        For I = 0 To Prices.GetUpperBound(0)
            Console.Write(Prices(I) & " ")
        Next
        Console.WriteLine()

        Console.ReadLine()
    End Sub
End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
100 200 300 400 500
100 200 300 400 500
500 400 300 200 100
Veličina niza: 5
Donja granica niza: 0
Gornja granica niza: 4
25.5 4.95 33.4
```

Grupisanje vrednosti u strukturu

U odeljku „Čuvanje više vrednosti istog tipa u jednoj promenljivoj“, naučili ste kako da grupišete više vrednosti istog tipa u niz. U nizove možete smeštati samo vrednosti istog tipa, na primer, samo vrednosti tipa Integer, tipa Double itd. Za grupisanje srodnih podataka koriste se namenski tipovi podataka, koje zovemo strukture. Za razliku od niza koji može sadržati samo podatke istog tipa, strukture mogu sadržati više podataka različitog tipa.

Pretpostavimo da program treba da obrađuje podatke o knjizi kao što su naslov, autor, izdavač i cena. Za tu namenu program deklariše sledeće promenljive:

```
Dim Title As String 'Naslov
Dim Author As String 'Autor
Dim Publisher As String 'Izdavač
Dim Price As Double 'Cena
```

Promenljive možete proslediti proceduri kao parametre:

```
ShowBook(Title, Author, Publisher, Price)
```

Da biste naknadno uključili i podatke o broju stranica i broju poglavlja u knjizi, možete deklarisati dve nove promenljive za podatke o poglavljima i stranicama i da zatim promenite parametre svih procedura:

```
Dim PageCount As Integer 'broj stranica
Dim ChapterCount As Integer 'broj poglavlja

ShowBook(Title, Author, Publisher, Price, PageCount, ChapterCount)
```

Druga mogućnost je da definišete strukturu u kojoj ćete grupisati podatke o knjigama:

```
Structure Book
  Dim Title As String
  Dim Author As String
  Dim Publisher As String
  Dim Price As Double
End Structure
```

Struktura je tip podataka koji omogućava grupisanje srodnih podataka. Nakon što ste definisali strukturu Book, u programu možete deklarisati promenljivu tipa Book.

```
Dim BookInfo As Book
```

Da biste dodelili vrednosti svakom članu promenljive, koristite operator tačka (.) (kojim se razdvaja ime promenljive od imena člana).

```
BookInfo.Title = "Visual Basic .NET kroz praktične primere"
BookInfo.Author = "Jamsa"
BookInfo.Publisher= "Mikro knjiga"
BookInfo.Price = 864.00
```

Potom, umesto da prosleđujete pojedinačne promenljive, proceduri možete proslediti jednu promenljivu tipa struktura:

```
ShowBook(BookInfo)
```

Ako u program naknadno treba dodati i datum izdavanja i težinu knjige, dovoljno je da dodate nove promenljive definiciji strukture Book i ne morate menjati sintaksu poziva procedura.

```
Structure Book
  Dim Title As String
  Dim Author As String
  Dim Publisher As String
  Dim Price As Double
```

```
    Dim Copyright As String
    Dim Weight As Double
  End Structure
```

PRIMER U sledećem programu, StructureDemo.vb, prvo se definiše struktura Book, a zatim se na osnovu te strukture pravi promenljiva BookInfo. Pomoću operatora tačka program dodeljuje vrednosti svakom članu strukture. I na kraju, kôd prosleđuje strukturnu promenljivu proceduri koja prikazuje podatke o knjizi.

```
Module Module1

     Structure Book
     Dim Title As String
     Dim Author As String
     Dim Publisher As String
     Dim Price As Double
  End Structure

  Sub ShowBook(ByVal SomeBook As Book)
    Console.WriteLine(SomeBook.Title)
    Console.WriteLine(SomeBook.Author)
    Console.WriteLine(SomeBook.Publisher)
    Console.WriteLine(SomeBook.Price)
  End Sub

  Sub Main()
    BookInfo.Title = "Visual Basic .NET kroz praktične primere"
    BookInfo.Author = "Jamsa"
    BookInfo.Publisher= "Mikro knjiga"
    Book.Price = 864.00

    ShowBook(BookInfo)
    Console.ReadLine()
  End Sub
End Module
```

Nakon što prevedete i pokrenete ovaj program, na ekranu će biti prikazani sledeći rezultati:

```
Visual Basic .NET kroz praktične primere
Jamsa
Mikro knjiga
864.00
```

Poboljšavanje čitljivosti koda pomoću konstanti

U programima se često koriste numeričke vrednosti. Na primer, numerička vrednost upravlja petljom For, kao što je prikazano ovde:

```
For I = 0 To 50
  Console.WriteLine(I)
Next
```

Numeričku vrednost možete koristiti i u naredbi If za poređenje:

```
If (I = 50) Then
  Console.WriteLine("Obrađuje poslednju vrednost")
End If
```

Dalje, možete upotrebiti numeričku konstantu da biste definisali veličinu niza:

```
Dim Studenti(50) As Integer
```

U sledećem programu, UseNumbers.vb, koriste se numeričke konstante. Ako pažljivije pregledate naredbe u programu, videćete da se često koristi vrednost 50:

```
Module Module1

    Sub Main()
        Dim Studenti(50) As Integer
        Dim I As Integer

        For I = 0 To 50
            Studenti(I) = I
        Next

        For I = 0 To 50
            Console.WriteLine(Studenti(I))
        Next

        Console.ReadLine()
        'Zaustavlja program da bismo videli rezultate
    End Sub

End Module
```

Umesto da koristite brojčane vrednosti na ovaj način, bolje im dodelite imena koja nešto znače i tako iskoristite prednosti imenovanih konstanti.

PRIMER Da biste napravili konstantu, upišite naredbu Const kao u sledećem primeru:

```
Const BrojStudenata As Integer = 50
```

Upotrebite ime konstante gdegod biste inače koristili numeričku konstantu. Na primer, deklarišite sledeći niz:

```
Dim Studenti(BrojStudenata) As Integer
```

Potom u petlji For upotrebite konstantu da biste zadali krajnju vrednost petlje, kao u sledećem primeru:

```
For I = 0 To BrojStudenata
  Studenti(I) = I
Next
```

Ako uporedite prethodne dve petlje For, videćete da konstante olakšavaju čitanje koda i razumevanje programa. Dovoljan je jedan pogled na petlju da programer shvati da će u petlji biti obrađen svaki student.

Upotrebom konstanti pojednostavljujete program kada se broj studenata promeni od 50 na 100. U prvom programu trebalo bi da promenite svako pojavljivanje vrednosti 50 na vrednost 100. Svaki put kada menjate program, možete napraviti grešku (na primer, da upišete 10 umesto 100). Ako u kodu koristite konstante, promenite samo sledeću naredbu:

```
Const BrojStudenata As Integer = 100
```

Prevodilac Visual Basica .NET zameniće konstante umesto vas.

Kratak opis razlika između Visual Basica i Visual Basica .NET

Ako ste iskusan programer na Visual Basicu, možda ste preskočili mnoge jednostavne primere u odeljcima ovog poglavlja. U ovom odeljku sažećemo najvažnije razlike između Visual Basica i Visual Basica .NET.

- Visual Basic .NET ne podržava tip podataka Variant. U poglavlju 4 videćete da sve klase .NET nasleđuju tip System.Object.
- Visual Basic .NET ne podržava tip podataka Currency. Umesto njega u programima treba koristiti tip Decimal.
- Visual Basic .NET ne podržava upotrebu naredbe LET za dodeljivanje vrednosti promenljivoj. Umesto toga, koristite operator za dodeljivanje.
- Visual Basic .NET ne podržava naredbu DefType kojom se u prethodnim verzijama Visual Basica zadavao podrazumevan tip promenljive. Trebalo bi da steknete naviku da u svojim programima uvek izričito deklarišete promenljive.
- Visual Basic .NET ne podržava tipove podataka koje definiše korisnik. Koristite umesto toga strukture ili klase da biste grupisali srodne podatke.
- Visual Basic .NET više ne podržava funkciju IsMissing. Umesto nje, u programima bi trebalo da koristite funkciju IsNothing da biste utvrdili da li određeni objekat ima vrednost.
- Visual Basic .NET ne podržava upotrebu naredbe GoSub za pozivanje procedura. Umesto toga, pozovite proceduru navođenjem njenog imena iza koga stoje zagrade, u kojima se mogu po potrebi navesti parametri.
- Visual Basic .NET ne podržava statične procedure. Ako lokalna promenljiva u proceduri mora zadržati svoju vrednost između dva poziva, u proceduri deklarišite statičnu promenljivu.
- Visual Basic .NET ne podržava deklarisanje promenljivih tipa String fiksne dužine.
- U Visual Basicu .NET tip podataka Integer je 32-bitan, što omogućava da se u promenljivama tipa Integer čuvaju vrednosti u opsegu od -2,147,483,648 do -2,147,483,647.
- Visual Basic .NET koristi tip podataka Short da predstavi 16 bitne vrednosti, koje mogu biti u opsegu od -32,768 do 32,767.

- U Visual Basicu .NET donja granica memorijskih nizova je 0 (a ne 1). Programi pisani na Visual Basicu .NET ne mogu koristiti naredbu OptionBase za definisanje podrazumevane vrednosti donje granice niza. Visual Basic .NET ne dozvoljava zadavanje donje i gornje granice kada deklarišete niz. Svi nizovi koje napravite u Visual Basicu .NET imaće donju granicu 0.
- U Visual Basicu .NET koriste se metode klase Math za aritmetičke i trigonometrijske operacije, kao što su računanje kvadratnog korena ili sinusa i kosinusa datog ugla.
- Kada u Visual Basicu .NET pozivate proceduru, morate pisati zagrade čak i kada procedura nema parametre.
- Visual Basic .NET standardno prosleđuje promenljive procedurama po vrednosti (pomoću rezervisane reči ByVal), što znači da procedure ne mogu menjati izvornu vrednost promenljive. Ako procedura mora promeniti vrednost promenljive, treba da prosledite promenljivu po referenci (upotrebom rezervisane reči ByRef).
- Iako Visual Basic .NET podržava funkciju MsgBox, preporučljivo je da koristite metodu Show klase MessageBox.
- U Visual Basicu .NET naredba Wend, koja označava kraj petlje While, zamenjena je naredbom End While.
- U Visual Basicu .NET naredba Debug.Print zamenjena je naredbom Debug.WriteLine.
- Visual Basic .NET omogućava da deklarišete promenljive u upravljačkim strukturama, kao što je petlja While ili naredba If. Doseg takve promenljive ograničen je na strukturu.
- Visual Basic .NET koristi vrednost Nothing da označi da neki objekat ne sadrži vrednost. Visual Basic .NET ne podržava vrednost Null niti Empty.