

Korišćenje promenljivih

U POSLEDNJEM POGLAVLJU STE NAUČILI KAKO SE KREIRAJU PROCEDURE I KAKO SE PAMTE U MODULIMA. OVO POGLAVLJE NASTAVLJA SA OSNOVAMA ZA KREIRANJE VBA PROCEDURA, UČEĆI VAS KAKO DA KORISTITE PROMENLJIVE U PROCEDURAMA. KORIŠĆENJE PROMENLJIVIH JE KLJUČNI ELEMENT KREIRANJA PONOVO ISKORISTIVOG KODA.

KONCEPT PROMENLJIVE SE MOŽE JEDNOSTAVNO ISKAZATI: PROMENLJIVA PREDSTAVLJA LOKACIJU U MEMORIJI U KOJOJ SE ČUVA VREDNOST ILI REFERENCA ZA OBJEKAT. MEĐUTIM, IMPLEMENTACIJA KONCEPTA U ACCESS VBA JE KOMPLIKOVANIJA ZBOG OGROMNE FLEKSIBILNOSTI KOJA SE DOBIJA KORIŠĆENJEM PROMENLJIVIH. U OVOM POGLAVLJU ĆETE NAUČITI KAKO SE PROMENLJIVA KREIRA, KAKO SE DELI SA OSTALIM PROCEDURAMA, KAKO SE PROMENLJIVE PROSLEĐUJU PRILIKOM POZIVA PROCEDURE I KAKO SE PROMENLJIVA UNIŠTAVA KADA ZAVRŠITE SA RADOM. KAKO SE PROMENLJIVE ČUVAJU U MEMORIJI, RAZMOTRIĆEMO EFEKTE KOJE KORIŠĆENJE PROMENLJIVIH STVARA SA STANOVIŠTA ISKORIŠĆENOSTI MEMORIJE I PERFORMANSI. NAKON DISKUSIJE O PROMENLJIVIM, U OKVIRU OVOG POGLAVLJA JE RAZMOTRENO I KREIRANJE KONSTANTI, KAO I KORIŠĆENJE NIZOVA ZA ISTOVREMENO KORIŠĆENJE VIŠE PROMENLJIVIH. U POSLEDNJEM ODELJKU POGLAVLJA NAUČIĆETE KAKO DA KREIRATE SOPSTVENE TIPOVE PROMENLJIVIH.

Korišćenje promenljivih u procedurama

Sve procedure koje ste kreirali u poslednjem poglavlju su pisane na najjednostavniji mogući način:

- Naredbe koje ste pisali su bile trivijalne, jer ste samo učili kako se procedure kreiraju, pamte i pokreću (interesantnije i praktičnije naredbe smo ostavili za kasnija poglavlja).
- Upotreba promenljivih je bila izuzetno ograničena jer su procedure koristile promenljive samo kao argumente. Kada smo pokretali procedure sa argumentima, za argumente smo navodili vrednosti između navodnika (literale), kao što je `Concatenate ("Johnson", "Diane")` i `Function1(3)`.
- Nismo eksplicitno deklarirali tipove podataka, tako da su argumenti, tj. vrednosti, koje smo naveli i vraćene vrednosti funkcije, bili varijante. Kao što smo objasnili u prethodnom poglavlju, tip podataka `Variant` koristi dva puta više memorije u odnosu na osnovne tipove podataka, tako da daje slabije performanse u odnosu na slučaj kada se definiše tip podataka.

Svrha ovog odeljka je da Vas nauči kako da koristite promenljive u procedurama. Postoje dva bitna razloga za korišćenje promenljivih u procedurama:

- Za kreiranje ponovo iskoristivog koda. Ako proceduru možete i kasnije da iskoristite, izbegli ste pisanje dodatnog koda!
- Za kreiranje bržeg koda. Nikome nije potreban spor kod.

Ukoliko imate neke sumnje u vezi potrebe za korišćenjem promenljivih, sledeća dva jednostavna primera će Vas sigurno razuveriti.

N A P O M E N A

Da biste izvežbali koncepte predstavljene u ovom poglavlju, kreirajte kopiju baze podataka `Northwind`, pod nazivom `Northwind_Ch8`. Obavezno potvrdite opciju `Always Use Event Procedures`, na `Forms/Reports` kartici u dijalogu `Options` (prikazuje se biranjem komande `Tools`→`Options`). ■

Korišćenje promenljivih za kreiranje ponovo iskoristivog koda

Pretpostavimo da ste kreirali `Switchboard` obrazac (formu) sa komandnom dugmadi za otvaranje drugih obrazaca, a zatim ste sakrili `switchboard` obrazac. Prvi način bi podrazumevao da se najpre na novi `switchboard` obrazac postave komandna dugmad, a zatim se kreiraju procedure događaja koje prenose akcije. Primera radi, proći ćemo kroz korake za otvaranje obrasca `Customers`, u aplikaciji `Northwind`:

1. Kreirajte novi obrazac u `Northwind_Ch8` pod nazivom `frmSwitchboard`.
2. Postavite komandno dugme na obrazac i postavite njegovo svojstvo (osobinu) `Name` na `cmdCustomers`, a svojstvo `Caption` na `Customers`.

3. Kliknite na svojstvo `OnClick`, a zatim na dugme `Build`, koje se nalazi sa desne strane okvira svojstva. `Access` kreira i otvara modul obrasca i prikazuje kodni šablon za proceduru događaja.
4. Unesite dole prikazanu proceduru događaja:

```
Private Sub cmdCustomers_Click()  
    DoCmd.OpenForm "Customers"  
    Forms!frmSwitchboard.Visible = False  
End Sub
```

Procedura `cmdCustomers_Click` pokreće metod `OpenForm`, za objekat `DoCmd`, kojim se obrazac otvara. Procedura definiše argument naziva obrasca za metod `OpenForm`, kao vrednost između navodnika, "Customers". Da bi switchboard obrazac ostao skriven, procedura postavlja svojstvo obrasca `Visible` na `False`. Kako nema daljih koraka, procedura se završava.

5. Snimite obrazac, pređite u prikaz `Form` i kliknite na dugme. Obrazac `Customers` se otvara, a switchboard obrazac je skriven.

Sada ćemo malo poboljšati ovu proceduru.

Korišćenje svojstva `Me` za dobijanje boljih performansi

U proceduri `cmdCustomers_Click`, za pokretanje naredbe koja skriva switchboard obrazac, `Access` mora da koristi hijerarhijsku referencu za svojstvo `Visible`:

```
Forms!frmSwitchboard.Visible
```

Svaki nivo reference zahteva određeno vreme obrade, tj, svaki uskličnik i svaka tačka zahtevaju vreme izvršenja koje je neophodno za obradu reference. Hijerarhiju možete da izbegnete pomoću svojstva `Me`. Ovo svojstvo predstavlja specijalizovanu alatku za optimizaciju, koju `Access` obezbeđuje prilikom kreiranja procedura u modulu obrasca ili izveštaja. Korišćenje svojstva `Me` u modulu obrasca ili izveštaja predstavlja najbrži način za referenciranje obrasca ili izveštaja, jer `VBA` ne mora da troši vreme za obradu kompletne reference. Prvi korak za poboljšavanje ove procedure jeste zamena `Forms!frmSwitchboard` sa `Me`.

Korišćenje argumenta umesto literala

Obrazac `Switchboard` obično ima nekoliko dugmadi pomoću kojih se otvaraju različiti obrasci. Umesto da kreiramo posebne procedure događaja za svako dugme, naš cilj je da kreiramo jednu proceduru koju ćemo koristiti više puta. Da bi se postojeća procedura mogla ponovo koristiti, najpre moramo da utvrdimo šta je sprečava da bude ponovo iskoristiva. U proceduri `cmdCustomers_Click`, problem predstavlja literal "Customers" koji se koristi kao argument metoda `OpenForm`. Umesto literala, kao argument metoda `OpenForm` sada koristimo string promenljivu, koju ćemo nazvati `strFormName` (koristimo prefiks `str` kao naznaku za tip podataka koji se koristi), a, zatim, uređujemo prosleđivanje literala nazad u proceduru, u vidu argumenta procedure. Za deklarisanje tipa podataka argumenta, koristi se sledeća sintaksa:

```
procedurename (argumentname [As datatype])
```

Sa "unetim" argumentom, procedura dobija sledeći izgled:

```
Private Sub cmdCustomers_Click(strFormName As String)
    DoCmd.OpenForm strFormName
    Me.Visible = False
End Sub
```

Promena u funkcijsku proceduru

Modifikovana verzija naše procedure više ne predstavlja proceduru događaja, jer, po definiciji, procedura događaja za događaj Click ne može imati argumente. Ako pokušate da kompajlirate takvu proceduru, VBA generiše grešku prilikom kompajliranja. Postoje dve alternative. Jedna je da promenite ime potprocedure (sub procedure) tako da ne koristi sintaksu rezervisanu za proceduru događaja i pozovete potprocedure iz nove procedure događaja. Ipak, ovo nije baš dobro rešenje, jer se nameće problem prosleđivanja literala u novu proceduru događaja. Ako preimenujemo potprocedure u `OpenAForm` i pozovemo je iz procedure događaja, literal i dalje predstavlja deo VBA koda:

```
Private Sub cmdCustomers_Click()
    Call OpenAForm("Customers")
End Sub
```

Vrednost literala uneta u prozor Module se naziva *hard-coded vrednot* (čvrsto kodirana vrednot) i obično sprečava ponovno korišćenje koda.

Bolje rešenje je promena procedure u funkcijsku proceduru (funkciju) nazvanu `OpenAForm`, tako da se literal može uneti kao svojstvo obrasca, a ne kao hard-coded vrednost u modulu. Kada kreirate funkcijsku proceduru, VBA automatski rezerviše memorijsku lokaciju za pamćenje rezultata, bez obzira na to da li funkcija zaista vraća rezultat. Ako se ne definiše tip podataka za tu lokaciju, VBA pretpostavlja da se koristi tip podataka Variant. Da bi se izbeglo degradiranje performansi zbog velikih memorijskih zahteva podataka tipa Variant, definisaćemo vraćene vrednosti funkcije kao podatke tipa Integer. Tip podataka za vraćenu vrednost funkcijske procedure se definiše pomoću naredbe za deklarisanje koja ima sledeću sintaksu:

```
[Public|Private] Function functionname [(argumentlist)] [As type]
```

Dole prikazana funkcijska procedura se sada može koristiti i za ostalu komandnu dugmad sa istog obrasca:

```
Public Function OpenAForm(strFormname As String) As Integer
    DoCmd.OpenForm strFormname
    Me.Visible = False
End Sub
```

Funkcijsku proceduru dodeljujete događaju Click za komandno dugme pomoću sledeće sintakse:

```
=OpenAForm("formname")
```

Sledeći korak je kreiranje funkcijske procedure koja se može koristiti za bilo koji obrazac. Pošto se kompletno iskoristiva funkcijska procedura ne vezuje za određeni obrazac, funkcijsku proceduru smeštamo u standardni modul. Cena koju plaćate za prenos procedure u modul jeste činjenica da više ne možete da koristite referencu Me (Me možete da koristite samo za referenciranje obrasca ili izveštaja, u okviru procedure koja je zapamćena u modulu tog obrasca ili izveštaja). Da bi se procedura mogla ponovo koristiti, možete da koristite objekat Screen, za referenciranje obrasca koji sadrži komandno dugme i skrivanje obrasca pre otvaranja obrasca Customers. (Ako ne zamenite redosled naredbi, procedura otvara, a, zatim, zatvara navedeni obrazac, jer novootvoreni obrazac postaje aktivni obrazac.)

1. Kreirajte novi standardni modul pod nazivom basNavigation i unesite novu javnu funkciju pod nazivom OpenAForm. Unesite dole prikazanu proceduru:

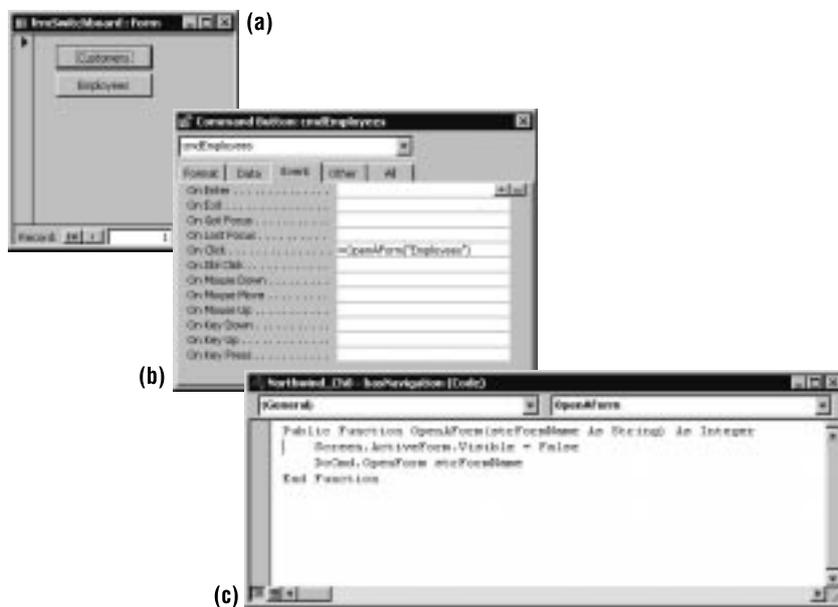
```
Public Function OpenAForm(strFormName As String) As Integer
    Screen.ActiveForm.Visible = False
    DoCmd.OpenForm strFormName
End Function
```

2. Otvorite Switchboard obrazac u prikazu Design. Kliknite na svojstvo OnClick dugmeta Customers i zamenite [Event Procedure] sa funkcijskom procedurom događaja, sa literalom kao argumentom, tako što ćete uneti =OpenAForm("Customers").
3. Postavite svojstvo HasModul obrasca na No. Kliknite na Yes kako bi se potvrdile opcije dijaloga. Snimite obrazac i pređite u prikaz Form.
4. Kliknite na dugme.VBA pokreće funkcijsku proceduru događaja kako bi se obrazac frmSwitchboard sakrio, a otvorio obrazac Customers. Iskopirajte dugme i preslikajte ga u drugi obrazac (videti sliku 8.1a).
5. Otvorite obrazac frmSwitchboard u prikazu Design. Iskopirajte i preslikajte dugme. Selektujte preslikano dugme, promenite njegovo svojstvo Name u cmdEmployees i promenite svojstvo Caption u Employees. Kliknite na svojstvo OnClick i promenite argument funkcije događaja u "Employees" (videti sliku 8.1b).
6. Snimite obrazac, pređite u prikaz Form i kliknite na dugme Employees. Access pokreće funkcijsku proceduru događaja (Slika 8.1c prikazuje proceduru u prozoru Module).

Na kraju, funkcijska procedura OpenAForm će biti upotrebljivija ako kao naslov komandnog dugmeta koristite naziv obrasca. U tom slučaju, procedura utvrđuje svojstvo Caption komandnog dugmeta i ne morate da prosledujete ime obrasca kao argument. Modifikovana funkcija izgleda ovako:

```
Public Function OpenAForm() As Integer
    Dim strFormName As String
    strFormName = Screen.ActiveControl.Caption
    Screen.ActiveForm.Visible = False
    DoCmd.OpenForm strFormName
End Sub
```

Svojstvo OnClick je postavljeno na =OpenAForm().



SLIKA 8.1 Komandna dugmad na obrascu (a) pokreću istu funkcijsku proceduru događaja. Dodelite funkciju kao izraz u postavci svojstva događaja (b). Klik na komandno dugme prosleđuje ime obrasca koji želite da otvorite funkcijom (c).

SAVET

Prednost korišćenja funkcijske procedure događaja je u tome, što, kada kopirate i preslikavate objekat, preslikavate i sve dodele za funkcijske procedure događaja (a dodele za potprocedure događaja se odbacuju). ■

Korišćenje promenljivih za pisanje bržeg koda

Pretpostavimo da imate obrazac koji želite da koristite i za pregled i za unos podataka. U tipičnom modu za unos podataka, kontrole za podatke su omogućene i otključane, a imaju standardnu belu boju pozadine. U modu za pregled, onesposobljavate i zaključavate kontrolu kako bi se izbegle slučajne promene podataka i menjate boju pozadine tako da odgovara boji obrasca, čime se i vizuelno ukazuje na činjenicu da promena podataka nije moguća. Pogledaćemo primer procedure radi promene svojstava jedne kontrole.

1. Kreirajte novi obrazac pod nazivom frmDots. Postavite komandno dugme čije će svojstvo Name biti cmdChange, svojstvo Caption će biti Change. Postavite tekstualno polje pod nazivom txtChange. Neka komandno dugme, obavezno, bude prvo u redosledu za dobijanje fokusa, tako da ono ima fokus odmah po otvaranju obrasca. Postavite njegovo svojstvo TabIndex na 0, dok tekstualno polje u svojstvu TabIndex ima vrednost 1. U sledecem koraku kreiramo proceduru koja zabranjuje tekstualno polje txtChange, a kako ne možete da zabranite kontrolu koja ima fokus, kontrola txtChange ne može da dobije fokus kada se procedura pokrene.

2. U standardnom modulu basNavigation unesite dole prikazanu proceduru Change. Procedura Change menja svojstva tekstualnog polja txtChange:

```
Public Function Change()  
    Forms!frmDots!txtChange.Enabled = False  
    Forms!frmDots!txtChange.Locked = True  
    Forms!frmDots!txtChange.BackColor = 12632256  
End Function
```

3. Kliknite na svojstvo OnClick komandnog dugmeta i unesite =Change(). Snimite obrazac i pređite u prikaz Form (videti sliku 8.2).
4. Kliknite na dugme Change. Svojstva tekstualnog polja se menjaju.

Poboljšajmo proceduru.



SLIKA 8.2 Kliknite na dugme Change u prikazu Form

Eliminisanje uskličnika i tačaka

Kao što smo ranije objasnili, svaki uskličnik i tačka zahtevaju vreme izvršenja koje se troši na obradu reference. Procedura Change ima devet uskličnika i tačaka. Eliminisanjem bilo kog uskličnika, ili tačke dobijate brži kod. Za jednu proceduru broj uskličnika i tačaka i nije toliko bitan, ali zamislite koliko uskličnika i tačaka ima jedna kompletno automatizovana aplikacija. Znači, cilj je svesti broj tačaka na minimum. Kako da se rešite nepotrebnih uskličnika i tačaka? Pomoću promenljivih!

Procedura Change poziva objekat tekstualnog polja, tako da možete da kreirate novu objektu promenljivu koja će predstavljati tekstualno polje. Za kreiranje promenljive u proceduri možete da koristite naredbu sa sledećom sintaksom:

```
Dim variablename [As type]
```

Ova naredba se naziva *naredba deklaracije promenljive*, kojom se Accessu nalaže da rezerviše memorijsku lokaciju i dodeli joj naziv *variablename*. Količina rezervisane memorije je određena *As type* delom naredbe. Ovaj deo je opcioni i ukoliko ne navedete tip podataka, Access podrazumeva da se koristi tip Variant, koji je sa stanovišta potrebne memorije najzahtevniji. U našem primeru, objekat je tekstualno polje, pa koristimo tip podataka TextBox:

```
Dim ctl As TextBox
```

Ovom naredbom se `ctl` kreira kao objektna promenljiva tipa `TextBox`. Rezervisana je memorijska lokacija, ali se na njoj ništa ne nalazi (vrednost `Nothing`) sve dok objekat ne navedete u naredbi dodele:

```
Set ctl = Screen.ActiveForm.txtChange
```

Ovde koristimo objekat `Screen` kao referencu na aktivni obrazac, i izbegavamo referenciranje specifičnog obrasca. Sada možemo da zamenimo hijerarhijsku referencu na kontrolu sa kreiranom promenljivom. Sada procedura `Change` izgleda ovako:

```
Public Function Change()  
    Dim ctl As TextBox  
    Set ctl = Screen.ActiveForm.txtChange  
    ctl.Enabled = False  
    ctl.Locked = True  
    ctl.BackColor = 12632256  
End Sub
```

Prebrojite uskličnike i tačke - sada ih ima svega pet! Procedura će biti mnogo brža od one koja je imala devet uskličnika i tačaka.

Kreiranje sopstvenih konstanti

U finalnom podešavanju procedure se kreira konstanta, koja određuje broj koji Access koristi za sivu boju. Definisanjem konstante možete da izbegnete potrebu za ponovnim unosom tog broja. Konstantu možete da kreirate u okviru procedure pomoću sledeće naredbe za deklaraciju konstante:

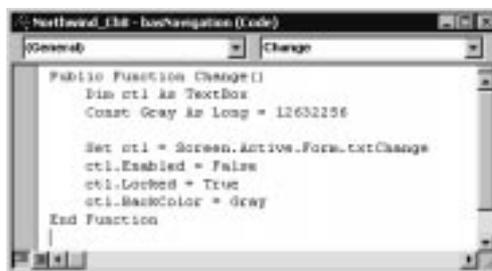
```
Const constantname [As type] = value
```

Kao i kod naredbe za deklaraciju promenljive, naredba za deklaraciju konstante rezerveše memorijsku lokaciju i dodeljuje joj navedeno ime. Deo `As type` u naredbi definiše tip podataka konstante, tako da Access zna koliko memorije treba dodeliti za konstantu. Razlika između naredbe za deklaraciju promenljive i konstante je u tome što se naredba za deklaraciju konstante koristi za dodeljivanje nepromenljive vrednosti. U našem primeru, broj je tipa `Integer`, ali premašuje granicu za tip `Integer`, tako da smo ga deklarirali kao `Long`, kao u sledećem primeru:

```
Const Gray As Long = 12632256
```

Dole je prikazana poboljšana procedura (i na slici 8.3). U poglavlju 9, "Kontrolisanje izvršenja", ćete naučiti dodatne tehnike (`With...End With`) za unapređivanje performansi ove procedure.

```
Public Function Change()  
    Dim ctl As TextBox  
    Const Gray As Long = 12632256  
  
    Set ctl = Screen.ActiveForm.txtChange  
    ctl.Enabled = False  
    ctl.Locked = True  
    ctl.BackColor = Gray  
End Function
```



SLIKA 8.3 Dugme Change na obrascu služi kao okidač za funkcijsku proceduru događaja

Kako procedure koriste promenljive

Primeri iz prethodnog odeljka su demonstrirali neke od načina na koje procedure koriste promenljive i konstante. Procedure mogu da koriste promenljive i konstante na sledeće načine:

- Moguće je kreirati promenljive i konstante za sopstvenu upotrebu. Promenljive i konstante koje se kreiraju u okviru procedure su raspoložive samo u okviru procedure u kojoj su kreirane, a nisu dostupne ostalim procedurama, čak ni procedurama koje se nalaze u okviru istog modula. Promenljive i konstante koje se kreiraju u okviru procedure se nazivaju *lokalne*, ili *promenljive i konstante na nivou procedure*.
- Promenljive se mogu koristiti kao argumenti. Procedura može zahtevati dodatne informacije u vidu argumenata kojima se definiše kako se procedura prenosi. Dodatne informacije u vidu promenljivih argumenata, takođe, mogu biti date u, ili *prosledene*, u proceduru kada je pozovete.
- Moguće je koristiti promenljive i konstante koje su kreirane na nekom drugom mestu. U odeljku Declarations modula možete da kreirate javne promenljive i konstante, tako da budu dostupne nekim, ili u svim procedurama u bazi podataka. Promenljive i konstante koje se kreiraju u odeljku modula Declarations se nazivaju promenljive i konstante na nivou modula.
- Moguće je izvršavanje naredbi koje menjaju vrednosti promenljivih, kojima procedura ima pristup.
- Moguće je vraćanje promenljive. Funkcijska procedura može da vraća vrednosti.

U sledećem odeljku su opisani različiti načini za kreiranje i uništavanje promenljivih i način na koji se promenljive prosleđuju iz jedne procedure u drugu.

Deklarisanje promenljivih

I konstante i promenljive možete da kreirate tako što ćete ih upotrebiti u naredbama. Kada prvi put koristite ime konstante ili promenljive, VBA automatski kreira privremenu lokaciju za smeštanje u memoriji sa imenom kojeg ste izabrali. Ovo se naziva *implicitna deklaracija*. Problem koji se javlja kod implicitne deklaracije je to što VBA ne prepoznaje tipografske greške i u slučaju da pogrešno otkucate ime promenljive jednostavno kreira novu promenljivu, bez

dodeljivanja dva imena. Nepotrebne greške možete da izbegnete dodavanjem naredbe `Option Explicit`, u odeljku `Declarations` svakog modula, tako da se zahteva eksplicitna deklaracija svih promenljivih i konstanti pomoću naredbi za deklaraciju. Tipične naredbe deklaracije su dve naredbe koje su postavljene na početku procedure `Change`:

```
Dim ctl As TextBox
Const Gray As Long = 12632256
```

NAPOMENA

U ovom poglavlju je pretpostavljeno da su sve promenljive i konstante eksplicitno deklarirane i da je u odeljku `Declarations` svakog modula korišćena naredba `Option Explicit`. VBA će automatski uneti naredbu `Option Explicit` ako ste potvrdili polje `Require Variable Declaration`, na jeziku Editor u Visual Basicovom dijalogu `Options`, kojeg možete dobiti biranjem `Tools Options` u Visual Basic Editoru (predefinisano, opcija nije selektovana). ■

Promenljive možete da kreirate samo na dva mesta:

- Za kreiranje *lokalnih*, tj. *promenljivih na nivou procedure*, naredbu deklaracije postavljate unutar procedure, ili u listi argumenata procedure.
- Za kreiranje *promenljivih na nivou modula*, naredbu deklaracije postavljate u odeljak `Declarations` modula.

Na slici 8.4 je prikazana promenljiva na nivou procedure i promenljiva na nivou modula.



SLIKA 8.4 Promenljiva na nivou procedure, *str*, i promenljiva na nivou modula, *strName*

Kada kreirate konstantu ili promenljivu, definišete sledeće četiri karakteristike:

- Ime;
- Tip podataka za konstantu, ili regularnu promenljivu, ili objektni tip podataka, ukoliko se radi o objektnoj promenljivoj;

- *Životni vek* promenljive (životni vek predstavlja vreme između kreiranja konstante ili promenljive i prestanka postojanja);
- *Domen* konstante, ili promenljive (domen definiše procedure koje mogu da "vide" i koriste konstantu, ili promenljivu - privatnu, ili javnu).

Imenovanje konstanti i promenljivih

Naredbe deklaracije se koriste za navođenje imena konstante, ili promenljive. VBA imena se ne razlikuju na osnovu rasporeda malih i velikih slova, što znači da je za VBA `frmcustomers` isto što i `frmCustomers`. Međutim, nakon eksplicitnog kreiranja promenljive, VBA automatski menja sve naredne pojave imena, tako da se podudaraju sa rasporedom velikih i malih slova koji ste naveli u naredbi deklaracije.

Imena konstanti i promenljivih moraju početi slovom; na dalje se mogu koristiti sva slova, brojevi i donja crtica (`_`); nije dozvoljeno navođenje ključnih reči; i nije dozvoljeno više od 255 karaktera. U poglavlju 2, "Upoznavanje sa objektima i događajima", je bilo reči o usvajanju konvencije imenovanja koja VBA kod čini čitljivijim i lakšim za razumevanje.

Navođenje tipa podataka

Frazu `As type` u naredbi deklaracije koristite za navođenje tipa podataka promenljive. Pošto je deklarisanje tipa podataka opciono, ukoliko ne navedete tip podataka, VBA dodeljuje tip `Variant`. Kao što je naznačeno u tabeli 7.1 (poglavljje 7, "Pisanje procedura"), tip `Variant` koristi dva puta više memorije od bilo kog drugog osnovnog tipa podataka. Kako je memorija uvek ograničena, što se više memorije utroši na smeštanje promenljive, to će manje memorije ostati na raspolaganju `Accessu` i drugim aplikacijama koje pokrećete. Sa manje memorije, `Access` se sporije pokreće, tako da treba izbegavati tip `Variant`. Međutim, postoje i dodatni razlozi u pogledu preformansi zbog kojih treba izbegavati tip `Variant`:

- Svaki put kada procedura dodeljuje vrednost promenljivoj tipa `Variant`, VBA mora da utroši određeno vreme na utvrđivanje vrste podataka koja se dodeljuje, a, zatim, menja tip promenljive tako da odgovara tipu dodeljenih podataka. Ovaj proces se naziva *suzbijanje* (coercing) variant promenljive.
- Kada koristite suzbijenu variant promenljivu u okviru nekog izračunavanja, VBA će, možda, morati da izvrši neka dodatna izračunavanja zbog konverzije tipa podataka prilikom izračunavanja. Ako VBA ne konvertuje tip podataka, generiše se run-time greška.
- Da biste izbegli run-time greške, često ćete morati da uključite naredbu za proveru tipa podataka promenljive. Na primer, ako izvodite numeričko izračunavanje, greške možete da izbegnete korišćenjem funkcije `IsNumeric`, kojom se pre izračunavanja utvrđuje da li je promenljiva numeričkog tipa podataka. Izvršavanje koda za proveru tipa podataka zahteva dodatno vreme izvršenja. Ako koristite određeni numerički tip podataka, VBA može da proveriti kompatibilnost za vreme kompajliranja koda.

Ukoliko iz određenih razloga niste prinuđeni da koristite tip Variant, uvek koristite promenljive sa eksplicitno navedenim tipom podataka. Kod regularnih promenljivih, ovo znači da za promenljive birate neki od tipova podataka sa najmanjim memorijskim zahtevima, koji su navedeni u tabeli 7.1.

Slično, kod objektnih promenljivih treba izbegavati generički tip podataka Object. Korišćenje tipa Object za analognu promenljivu je slično korišćenju tipa Variant kod regularne promenljive, jer procedura svaki put dodeljuje objekat i VBA mora da utroši vreme na utvrđivanje vrste dodeljenog objekta, a svaki put kada se procedura poziva na svojstvo ili metod, VBA mora da utroši vreme na proveru validnosti navedenog svojstva, ili metoda za objekat. Međutim, ako tačno navedete tip objekta, VBA proverava validnost svojstva, ili metoda samo za vreme kompajliranja koda. Navođenje tipa objekta kod objektnih promenljivih se naziva *hard typing*.

NAPOMENA

U cilju najefikasnijeg upravljanja memorijom i pisanja najbržeg koda, navodite tačno određene tipove podataka za regularne promenljive i najspecifičnije tipove objekata za objektno promenljive. ■

Razumevanje životnog ciklusa promenljive

Kada VBA čita naredbu deklaracije za promenljivu, poput

```
Dim intX As Integer
Dim txt As TextBox
```

kreira i imenuje privremenu memorijsku lokaciju, dodeljuje potrebnu količinu memorije za navedeni tip podataka i postavlja predefinisano (default) vrednost u zavisnosti od tipa podataka. Za promenljivu numeričkog tipa se postavlja vrednost nula, za stingove se postavlja string dužine nula (" "), za promenljivu tipa Variant se postavlja specijalna vrednost Empty, a za objektno promenljive se postavlja specijalna vrednost Nothing. Naredbe deklaracije prikazane na početku odeljka postavljaju promenljivu `intX` na nulu, a objektnu promenljivu `txt` na Nothing. Šta se dalje dešava, zavisi od toga da li je reč o regularnoj ili objektnoj promenljivoj.

Životni ciklus regularne promenljive

Pošto kreirate regularnu promenljivu, možete je koristiti u naredbama. Počinjete sa naredbom dodele kako bi se imenu promenljive dodelila vrednost, kao na primer `intX = 1`. Pošto se promenljiva dodeli, njeno ime koristite u drugim naredbama za promenu vrednosti, kao u naredbi `intX = intX + 1`, koja uvećava vrednost `intX` za jedan i rezultat dodeljuje u `intX`.

Naredbu dodele možete da koristite i za reinicijalizaciju promenljive postavljanjem promenljive na njenu predefinisano vrednost. Za regularne promenljive sa određenim tipom podataka, inicijalizacija promenljive podrazumeva jednostavno postavljanje njene vrednosti ili na nulu ili string dužine nula. Na primer, naredba `intX = 0` reinicijalizuje promenljivu `intX`.

Konačno, sama promenljiva se uništava. Uništavanje promenljive označava uništavanje privremene memorijske lokacije i oslobađanje memorije za ponovno korišćenje. Kako se promenljiva uništava? Odložimo odgovor za koji trenutak.

Životni ciklus objektne promenljive

Priča za objektnu promenljivu je nešto složenija, jer postoje dva elementa koje treba pratiti: objektna promenljiva i sam objekat. Pošto se objektna promenljiva deklarise, koristite naredbu dodele za ukazivanje objektne promenljive na objekat, kao u sledećem primeru:

```
Set txt = Forms!frmDots!txtChange
```

Objekat na koji promenljiva ukazuje možda već postoji u memoriji, ili se naredbom dodele stvarno kreira objekat u memoriji. Na primer, kada interaktivno otvorite obrazac, kao što ste otvorili frmDots iz prozora Database, Access kreira objekat Form u memoriji. Kada deklarirate objektnu promenljivu, kao što je Dim frm As Form i ukazete sa frm na obrazac pomoću naredbe Set frm = Forms!frmDots, Vi objektom promenljivom ukazujete na objekat koji već postoji u memoriji.

Kako postoje i objektna promenljiva i objekat, postoje dva memorijska zahteva: memorija koju troši objektna promenljiva i memorija koju zauzima objekat. Kao što je naznačeno u tabeli 7.1 (u poglavlju 7), objektna promenljiva zauzima 4 bajta bez obzira na to na šta ukazuje (4 bajta služe za smeštanje adrese, koja je obično tipa long integer), ali objekat na koji ukazuje može da zauzima stotine, ili hiljade bajtova memorije.

Nakon što se objektna promenljiva dodeli, koristite je u drugim naredbama za promenu objekta. Kada u VBA naredbi menjate objektnu promenljivu, u stvari menjate objekat. Na primer, u proceduri Change, tri naredbe koje menjaju svojstva objekta txt u stvari menjaju svojstva tekstualnog polja txtChange koji se nalazi na obrascu.

Naredbu dodele možete da koristite i za reinicijalizaciju objektne promenljive postavljanjem objektne promenljive na njenu predefinisanu vrednost. Kada se objektna promenljiva postavi na Nothing, uništavate link između objektne promenljive i samog objekta. Nakon uništavanja linka, objektna promenljiva i dalje postoji (i dalje zauzima 4 bajta u memoriji), dok objekat može, ali ne mora i dalje da postoji. Na primer, ako je objektna promenljiva frm ukazivala na obrazac frmDots i postavili ste frm = Nothing, objekat Form i dalje postoji u memoriji dok je obrazac otvoren. Postavljanjem frm na Nothing jedino raskidate link između objektne promenljive i objekta. S druge strane, ako je rst jedini objekat koji ukazuje na objekat Recordset, tada postavljanjem rst na Nothing raskidate link i uništavate objekat. U ovom slučaju, objekat se uništava zato što za objekat, koji ostaje u memoriji, morate imati ili dodeljenu objektnu promenljivu ili implicitnu referencu kreiranu u Accessu.

Kada otvorite obrazac, Access automatski kreira implicitnu referencu na objekat Form koji postoji dok je obrazac otvoren. Vaš kod ne može da utiče na implicitne reference obrasca dok je obrazac otvoren. S druge strane, kada otvorite skup zapisa (recordset), ne postoji implicitna referenca na objekat Recordset. Dok se u Vašem kodu nalazi barem jedna promenljiva koja ukazuje na objekat Recordset, objekat Recordset postoji u memoriji. Kada se raskinu sve reference na objekat Recordset, bilo postavljanjem objektne promenljive na Nothing, ili uništavanjem objektnih promenljivih, sam objekat Recordset se uništava i njegova memorija se oslobađa za druge namene.

Objekte u memoriji možete da uništite tako što ćete ih zatvoriti. Na primer, obrazac možete da zatvorite u okviru procedure pomoću metoda Close za objekat DoCmd, a skup zapisa možete da zatvorite pomoću njegovog metoda Close. Kada za objekat koristite jedan od metoda Close, objektna promenljiva, koja ukazuje na objekat, ostaje. Objekat na koji ukazuje, možda, više ne postoji, ali sama promenljiva i dalje postoji i može se dodeliti drugom objektu.

Uništavanje promenljivih

Pošto promenljive zauzimaju memoriju, sigurno ćete hteti da ih uništite kada prestanete da ih koristite. Ali, pitanje na koje još uvek nismo odgovorili jeste kako se promenljiva uništava? Interesantno, Access ne obezbeđuje naredbu za "uništavanje", tako da ne postoji način na koji možete eksplicitno uništiti promenljivu. Umesto toga, Access nudi dve alternative:

- Možete da deklarirate promenljivu u okviru procedure kao lokalnu, ili promenljivu na nivou procedure. Access će je automatski uništiti po završetku procedure.
- Promenljivu možete da deklarirate u odeljku Declarations modula, tj. kao promenljivu na nivou modula. Access će je uništiti kada zatvorite bazu podataka, ali ne pre toga.

U sledećem odeljku ćemo videti da postoje valjani razlozi za korišćenje promenljivih na nivou modula. Međutim, opšte pravilo je očigledno: Zbog količine memorije koja se zauzima, uvek kada je to moguće koristite lokalne, tj. promenljive na nivou procedure, umesto promenljivih na nivou modula.

SAVET

Za "kratak" rezime o naredbama deklaracije za procedure i promenljive, o navođenju tipova podataka za promenljive i vraćenim vrednosti funkcijskih procedura, pogledajte tabelu "Rezime deklaracija" na pratećem CD-u, u okviru Tables\Chapter8.pdf. ■

Korišćenje promenljivih na nivou procedure

Postoje dva načina za deklarisanje promenljive u proceduri: posebna naredba deklaracije unutar procedure i u listi argumenata koja se poziva iz druge procedure.

Deklarisanje promenljive u okviru procedure

Promenljiva na nivou procedure se može kreirati postavljanjem naredbe deklaracije u okviru procedure. Osnovna sintaksa izgleda ovako:

```
Dim variablename [As type]
```

Evo nekoliko primera:

```
Dim intCounter As Integer
Dim strLastName As String
Dim frmCustomers As Form
```

Moguće je deklarirati više promenljivih u okviru jedne naredbe deklaracije, ali morate da uključite tip podataka za svaku promenljivu ponaosob. Ova naredba deklarira `frm1` i `frm2` kao tip objekta `Form`:

```
Dim frm1 As Form, frm2 As Form
```

Ali, ova naredba deklarira `frm2` kao tip objekta `Form`, a `frm1` kao tip `Variant`:

```
Dim frm1, frm2 As Form
```

Naredba deklaracije kreira i imenuje promenljivu, dodeljuje količinu memorije u skladu sa frazom `As type` i inicijalizuje promenljivu. Naredba deklaracije ne dodeljuje vrednost. Za dodelu vrednosti morate da koristite naredbu `dodele`. Na primer, razmotrimo sledeću proceduru:

```
Public Sub SomeProcedure()  
    Dim cnn As ADODB.Connection  
    Set cnn = CurrentProject.Connection  
    ...  
End Sub
```

Ova naredba deklaracije kreira `cnn` kao objektnu promenljivu za tip objekta `ADO Connection`:

```
Dim cnn As ADODB.Connection
```

Naredba `dodele` ukazuje objektnom promenljivom na tekuću konekciju:

```
Set cnn = CurrentProject.Connection
```

NAPOMENA

Postoji još jedna verzija naredbe deklaracije koju možete da koristite za objektnu promenljivu. Prilikom deklaracije promenljive možete da koristite ključnu reč `New`, sa sintaksom `Dim objvar As New objecttype`. Kada uključite ključnu reč `New`, VBA automatski kreira novu instancu objekta i za neke tipove objekata ne morate da koristite naredbu `Set` prilikom dodele objektnoj promenljivoj. Za više informacija o ključnoj reči `New`, pogledajte poglavlje 14, "Kreiranje i modifikovanje objekata baze podataka pomoću Access VBA". ■

Deklarisanje promenljive u listi argumenata

Kada postavite argument u listu argumenata procedure, time kreirate promenljivu na nivou procedure. Za navođenje tipa svake promenljive argumenta možete da koristite frazu `As type`, na sledeći način:

```
argumentname As type
```

Evo i primera:

```
Public Function Concatenate (A As String, B As String)
```

Kada se promenljiva deklarira u listi argumenata, VBA automatski dodeljuje memoriju u skladu sa navedenim tipom promenljive i inicijalizuje promenljivu. Međutim, ne morate da odvajate naredbu `dodele`, jer VBA automatski dodeljuje vrednost za promenljivu u listi argumenata prilikom poziva procedure. Na primer, VBA izvršava sledeću naredbu:

```
MsgBox Concatenate ("Johnson", "Diane")
```

VBA ide na funkciju Concatenate i automatski promenljivoj A dodeljuje vrednost "Johnson", A "Diane" promenljivoj B.

```
Public Sub CallingProcedure()  
    MsgBox Concatenate ("Johnson", "Diane")  
End Sub  
Public Function Concatenate(LName As String, FName As String)  
    Concatenate = LName & ", " & FName  
End Sub
```

Razumevanje vidljivosti promenljivih na nivou procedure

Promenljiva na nivou procedure, kreirana bilo pomoću naredbe deklaracije ili u listi argumenata, je vidljiva samo u proceduri u kojoj je kreirana. Ostale procedure ne mogu da vide niti da koriste tu promenljivu. Jedini način, na koji druga procedura može da dobije pristup promenljivoj na nivou procedure, jeste da procedura, u kojoj je promenljiva kreirana, pozove tu drugu proceduru i promenljivu postavi kao argument te druge procedure. Promenljiva koja se u pozvanu proceduru prosleđuje kao argument može biti prosleđena, ako pozvana procedura zove još neku drugu proceduru. Lanac pozvanih procedura se naziva *stablo poziva* (call tree) procedure.

Pogledajmo jedan primer kako bi smo razjasnili te ideje:

1. Kreirajte novi modul pod nazivom `basVariables`. Unesite novu javnu potproceduru pod nazivom `LocalVariable`, kao što je dole navedeno. Procedura `LocalVariable` deklarise `strLocal` kao lokalnu, ili string promenljivu na nivou procedure.

```
Public Sub LocalVariable()  
    Dim strLocal As String  
    strLocal = "'Local variable in LocalVariable procedure'"  
    MsgBox strLocal  
End Sub
```

2. Pokrenite proceduru u prozoru Immediate unošenjem `Call LocalVariable` i pritisnite Enter. Jedina procedura koja može da koristi promenljivu `strLocal` je procedura `LocalVariable` i procedure kojima procedura `LocalVariable` prosledi ovu promenljivu kao argument.
3. Kreirajte dole prikazanu proceduru `GetLocal` u istom modulu (ili u drugom) i pokrenite je u prozoru Immediate.

```
Public Sub GetLocal()  
    MsgBox strLocal  
End Sub
```

Procedura nije uspešno izvršena jer procedura `GetLocal` ne vidi promenljivu `strLocal` u proceduri `LocalVariable`. Sa stanovišta procedure `GetLocal`, promenljiva `strLocal` još nije definisana.

4. Kliknite na dugme Reset u paleti sa alatkama. Unesite naredbu `Call GetLocal`, iza naredbe `MsgBox` u proceduri `LocalVariable`, a, zatim, pokrenite proceduru `LocalVariable` u prozoru Immediate.

Pokreće se procedura `LocalVariable`, štampa se vrednost promenljive `strLocal`, a, zatim, se generiše greška "Variable not defined" koja ukazuje na to da promenljiva nije definisana. Procedura se ne izvršava nezavisno od toga da li je pozivate nezavisno, ili iz procedure `LocalVariable`, jer procedura `GetLocal` ne može da vidi promenljivu `strLocal`. Jednostavnim pozivanjem druge procedure iz procedure u kojoj je lokalna promenljiva definisana ne znači da će promenljiva biti dostupna pozvanoj proceduri. Ako želite da pozvana procedura koristi promenljivu, morate je proslediti kao argument u pozvanu proceduru.

5. Modifikovaćemo obe procedure. Sa tim promenama, procedura `LocalVariable` poziva `GetLocal` i prosleđuje promenljivu `strLocal` kao argument, a procedura `GetLocal` prima promenljivu u vidu argumenta.

```
Public Sub LocalVariable()  
    Dim strLocal As String  
    strLocal = "'Local variable in LocalVariable procedure'"  
    MsgBox strLocal  
    Call GetLocal(strLocal)  
End Sub  
  
Public Sub GetLocal(strA As String)  
    MsgBox strA & " passed as an argument to GetLocal"  
End Sub
```

6. U prozoru `Immediate`, unesite `Call LocalVariable` i pritisnite `Enter`. Procedura `LocalVariable` definiše i prikazuje lokalnu promenljivu u okviru za poruku (message box) (videti sliku 8.5a), a, zatim, poziva proceduru `GetLocal`, prosleđujući promenljivu `strLocal` kao njen argument. Procedura `GetLocal` može da koristi prosleđenu promenljivu i prikazuje je u okviru za poruku (videti sliku 8.5b).



(a)



(b)

SLIKA 8.5 Procedura prikazuje svoju lokalnu promenljivu (a), a, zatim je prosleđuje u drugu proceduru, koja prikazuje prosleđenu promenljivu (b)

NAPOMENA

Dobra programerska praksa nalaže da se naredbe deklaracije za sve promenljive na nivou procedure postavljaju na početku procedure. ■

Promena životnog veka promenljivih na nivou procedure

Kada deklarirate promenljivu na nivou procedure i dodelite joj vrednost, njena vrednost se može kasnije menjati u drugim naredbama. Promenljiva zadržava novu vrednost sve dok se ponovo ne promeni, ili dok se procedura ne završi.

Po unapred definisanom podešavanju, promenljive na nivou procedure prestaju da postoje kada se procedura završi. Access poništava njihove vrednosti i oslobađa njihove lokacije u memoriji za dalju upotrebu. Međutim, postoji mogućnost produžetka životnog veka promenljive na nivou procedure i nakon završetka procedure. Prilikom kreiranja promenljive, umesto ključne reči `Dim` navedete ključnu reč `Static`. Pomoću ključne reči `Static` proširuje se životni vek promenljive za svo vreme za koje je pokrenut VBA kod u bilo kom modulu. Kada kreirate statičku promenljivu, VBA zadržava njenu vrednost i kada se procedura završi, a postavlja promenljivu na sačuvanu vrednost sledeći put kada se procedura pokrene.

Primeru radi, kreiraćemo proceduru koja koristi statičku promenljivu. Unesite dole prikazanu proceduru `StaticVariable` u okviru modula `basVariables`.

```
Public Sub StaticVariable()
    Dim strNonStatic As String
    Static sstrStatic As String
    strNonStatic = strNonStatic & " nonstatic"
    sstrStatic = sstrStatic & " static"
    Debug.Print strNonStatic
    Debug.Print sstrStatic
End Sub
```

Procedura `StaticVariable` kreira promenljivu `strNonStatic` kao nestatičku promenljivu i `sstrStatic` kao statičku promenljivu. Obe promenljive su inicijalizovane na string dužine nula. (Koristite prefiks `s` kako biste naznačili da se radi o statičkoj promenljivoj.) Naredba dodele nadovezuje na vrednost `strNonStatic` reč `nonstatic`, a na vrednost `sstrStatic` reč `static`. Nadovezane vrednosti se zatim štampaju u prozoru `Immediate`.

Pokrenite proceduru u prozoru `Immediate` unošenjem `Call StaticVariable` i pritisnite `Enter`. Kada prvi put pokrenete proceduru, obe promenljive se inicijalizuju na stringove dužine nula, tako da `Debug` štampa samo po jednu reč u svakoj liniji. Kada se procedura završi, Access uništava promenljivu `strNonStatic`, ali zadržava promenljivu `sstrStatic` i njenu vrednost.

Sada ćemo pokrenuti proceduru po drugi put. Kada se procedura pokrene po drugi put, promenljiva `strNonStatic` je ponovo kreirana i reinicijalizovana na string dužine nula, a promenljiva `sstrStatic` ima staru zadržanu vrednost. Naredbom dodele se promenljivoj `strNonStatic` ponovo dodeljuje vrednost `nonstatic`, kao i pre, dok se za promenljivu `sstrStatic` na postojeći string dodaje reč `static`. Svaki sledeći put kada pokrenete proceduru, nestatička promenljiva se reinicijalizuje a zatim naredbom dodele postavlja na vrednost `nonstatic`, dok se vrednost promenljive `sstrStatic` zadržava, a naredbom dodele se na postojeći string nadovezuje reč `static`. Na slici 8.6a je prikazan prozor `Immediate` nakon trećeg izvršenja procedure.

Postavljanjem ključne reči `Static` u naredbama deklaracije možete da pretvorite sve lokalne promenljive u statičke. Isprobajte sledeći primer:

1. Selektujte sve naredbe procedure `StaticVariable` i pritisnite `Ctrl+C` kako bi se kreirala kopija. Postavite tačku umetanja iza poslednje procedure u modulu i pritisnite `Ctrl+V` za preslikavanje. (Obavezno budite u prikazu Full Module, u koji prelazite selektovanjem opcije Default To Full Module View u dijalogu Tools⇒Options.) Promenite naredbu deklaracije procedure umetanjem ključne reči `Static` i promenite naziv procedure kao u dole prikazanom primeru:

```
Public Static Sub StaticProcedure()
    Dim strNonStatic As String
    Static sstrStatic As String
    strNonStatic = strNonStatic & " nonstatic"
    sstrStatic = sstrStatic & " static"
    Debug.Print strNonStatic
    Debug.Print sstrStatic
End Sub
```

2. Kliknite u prozor Immediate, izaberite Edit⇒Select All i pritisnite Delete.
3. Pokrenite proceduru `StaticProcedure` u prozoru Immediate tri puta. Primitićete da su obe promenljive sada statičke. Upotreba ključne reči `Static` u definiciji procedure je višeg prioriteta od deklaracije sa ključnom reči `Dim` u okviru procedure (videti sliku 8.6b).



SLIKA 8.6 Statička promenljiva zadržava svoju vrednost između poziva procedure, a nestatička promenljiva se reinicijalizuje svaki put kada pozovete proceduru (a). Korišćenjem ključne reči `Static`, u deklaraciji procedure, se dodeljuje prostor za skladištenje za sve lokalne promenljive i zadržavaju se njihove vrednosti za svo vreme za koje je modul pokrenut, sve dok ne restartujete, ili resetujete modul (b).

4. Kliknite u modul `basVariables` i kliknite na dugme Reset u paleti sa alatcima. Kada resetujete modul, uništavaju se sve promenljive koje su deklarirane u okviru modula i oslobađa se prostor u memoriji koji su do tada zauzimali.

Kada unesete novu proceduru, sve promenljive možete da učinite statičkim tako što ćete selektovati **Insert** → **Procedure** i potvrditi opciju **All Local Variables As Statics** u dijalogu **Add Procedure** (videti sliku 8.7). Ako je ova opcija potvrđena, VBA automatski umeće ključnu reč **Static** u deklaraciji procedure.



SLIKA 8.7 Da bi se sačuvala vrednost svih lokalnih promenljivih deklariranih u proceduri, selektujte opciju *All Variables As Statics* u dijalogu *Add Procedure*

Prosleđivanje podataka u proceduru

Mogućnost pozivanja jedne procedure iz druge dozvoljava razdvajanje koda u manje, jednostavnije procedure. Kada iz jedne procedure pozivate drugu, možete da šaljete podatke u proceduru koja se poziva. Možete da šaljete kako literale, tako i promenljive.

Prosleđivanje literala

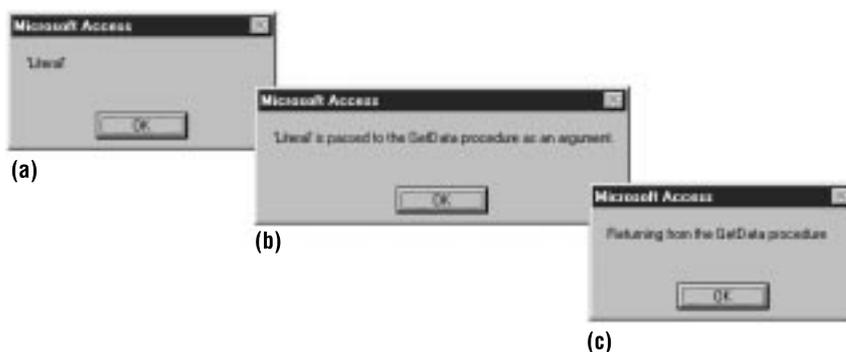
Najpre ćemo pogledati kako se u pozvanu proceduru prosleđuju literali. U modulu `basVariables` unesite procedure `PassingLiteral` i `GetData`:

```
Public Sub PassingLiteral()
    MsgBox "Literal"
    Call GetData("Literal")
    MsgBox "Returning from the GetData procedure"
End Sub

Public Sub GetData (A As String)
    MsgBox A & " is passed to the GetData procedure as an argument."
End Sub
```

Naredba `Call GetData("Literal")` u proceduri `PassingLiteral` poziva proceduru `GetData` i prosleđuje vrednost literala "Literal".

Pokrenite proceduru `PassingLiteral` u prozoru Immediate. Procedura `PassingLiteral` prikazuje svoju poruku (videti sliku 8.8a), a, zatim, poziva proceduru `GetData` prosleđujući joj vrednost literala. VBA u pozvanoj proceduri menja sve pojave slova A sa prosleđenom vrednošću. Procedura `GetData` prikazuje svoju poruku (videti sliku 8.8b) i zatim se završava. U sledećoj naredbi VBA se vraća u proceduru `PassingLiteral`, prikazuje poruku (videti sliku 8.8c) i zatim se završava.



SLIKA 8.8 Pozivajuća procedura prikazuje literal (a) i prosleđuje ga u pozvanu proceduru (b). Kada se pozvana procedura završi, kontrola se vraća u pozivnu proceduru (c).

Prosleđivanje promenljive po referenci, ili po vrednosti

Access obezbeđuje dva načina za slanje podataka promenljive: možete da pošaljete samu promenljivu, ili možete da pošaljete kopiju vrednosti promenljive.

Kada šaljete samu promenljivu, pozvana procedura može da manipuliše promenljivom i menja njenu vrednost. Slanje same promenljive se naziva *prosleđivanje po referenci* i predstavlja predefinisani metod za slanje promenljivih.

Kada šaljete kopiju vrednosti promenljive, VBA kreira kopiju na drugoj privremenoj memorijskoj lokaciji i šalje kopiju. Pozvana procedura može da koristi kopiju, ali i ako može da menja njenu vrednost, te promene nemaju uticaja na samu promenljivu. Slanje kopije promenljive se naziva *prosleđivanje promenljive po vrednosti*.

Da biste naznačili koji metod koristite, ispred imena argumenta, u listi argumenata pozvane procedure, navodite ključnu reč `ByRef` ako prosleđujete samu promenljivu, ili `ByVal` ako prosleđujete kopiju:

```
[ByRef|ByVal] argumentname [As type]
```

Da bismo ovo ispitali, kreiraćemo par procedura, proslediti promenljivu najpre po referenci a zatim po vrednosti. U modulu `basVariables` kreirajte procedure `PassingVariableByRef` i `GetVariableByRef`, kao u dole prikazanom primeru.

```
Public Sub PassingVariableByRef()  
    Dim strVariable As String  
    strVariable = "'Variable to be passed to another procedure'"
```

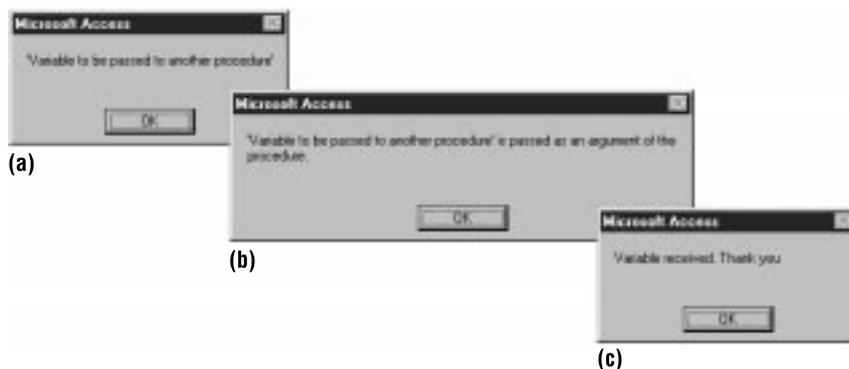
```

    MsgBox strVariable
    Call GetVariableByRef(strVariable)
    MsgBox strVariable
End Sub

Public Sub GetVariableByRef (ByRef strA As String)
    MsgBox strA & " is passed as an argument of the procedure."
    strA = "Variable received. Thank you"
End Sub

```

Procedura `PassingVariableByRef` deklarira promenljivu `strVariable`, dodeljuje joj vrednost a zatim je i prikazuje (videti sliku 8.9a). Procedura poziva proceduru `GetVariableByRef` i prosledjuje joj promenljivu. Procedura `GetVariableByRef` prima promenljivu, prikazuje je (videti sliku 8.9b) i menja njenu vrednost. Kada se procedura `GetVariableByRef` završi, VBA se vraća na sledeću naredbu u pozivnoj proceduri, prikazuje promenjenu promenljivu (videti sliku 8.9c) a zatim se završava.



SLIKA 8.9 Kada prosledujete promenljivu po referenci (a), pozvana procedura prima samu promenljivu (b) i može da menja njenu vrednost. Pozivna procedura koristi promenjenu vrednost (c).

Pratite sledeće korake kako biste videli kako funkcioniše prosledivanje promenljivih:

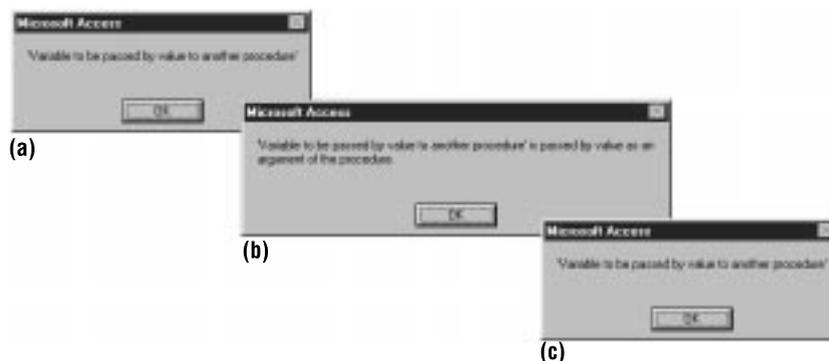
1. Pokrenite proceduru `PassingVariableByRef` u prozoru Immediate.
2. U prozoru Module, iskopirajte na Clipboard sav kod za obe procedure. Postavite tačku umetanja ispod poslednje procedure i preslikajte sadržaj Clipboard-a.
3. Selektujte preslikane procedure i izaberite `Edit` → `Replace`. Unesite **ByRef** u tekstualno polje Find What i **ByVal** u tekstualno polje Replace With. Proverite da li je selektovana opcija Selected Text među opcijama Search, a, zatim, kliknite na Replace All (videti sliku 8.10). Kliknite OK da potvrdite četiri zamene. Ponovo selektujte preslikane procedure. Unesite **passed** u tekstualno polje Find What i **passed by value** u tekstualno polje Replace With, a, zatim, kliknite na Replace All. Kliknite OK za potvrdu dve zamene i zatvorite dijalog Replace.



SLIKA 8.10 Koristite dijalog *Replace* za modifikovanje procedura zamenom vrednosti

4. Pokrenite proceduru `PassingVariableByRef` u prozoru *Immediate*.

Ovog puta se u pozvanu proceduru prosleđuje samo kopija vrednosti promenljive. Pozivna procedura prikazuje promenljivu (videti sliku 8.11a). Pozvana procedura prima kopiju promenljive i prikazuje svoju poruku (videti sliku 8.11b), zatim menja vrednost kao i ranije, ali se ovog puta samo menja vrednost kopije. Kada se pozvana procedura završi, VBA se vraća u pozivnu proceduru i prikazuje poslednju poruku sa nepromenjenom vrednošću promenljive (videti sliku 8.11c).



SLIKA 8.11 Kada prosleđujete promenljivu po vrednosti (a), pozvana procedura koristi kopiju promenljive (b). Iako pozvana procedura može da menja vrednost kopije, originalna promenljiva zadržava staru vrednost (c).

Iako se radi o različitim terminima, ova dva metoda su analogna poznatim metodima povezivanja i ugrađivanja dokumenata. Kada prosledite argument `ByRef`, "povezujete" proceduru i promenljivu, tako da se promene koje se izvedu u proceduri prenose i na promenljivu. Kada prosleđujete argument `ByVal`, "ugrađujete" kopiju promenljive i ne postoji link ka samoj promenljivoj, tako da se promene u proceduri ne prenose na promenljivu. Slanje promenljive po referenci je brže, jer VBA ne mora da troši vreme na kreiranje kopije.

NAPOMENA

Prosleđivanje promenljive po referenci umanjuje razliku između funkcijske procedure i potprocedure. Kada se promenljiva prosleđuje po referenci, funkcija ili potprocedure koja promenljive prima u vidu argumenata može da menja njihove vrednosti, ili objekte na koje ukazuju, a zatim vraća izmenjene promenljive. To znači da i funkcijske procedure i potprocedure mogu da vraćaju vrednosti pomoću argumenata koje prosleđuju po referenci. Razlika je samo u tome što funkcijska procedura može da vraća dodatne vrednosti, nezavisno od vrednosti koje se vraćaju preko argumenata. ■

Prosleđivanje argumenata u proceduru

Postoje dva moguća načina za prosleđivanje argumenata u proceduru: po redosledu i po nazivu.

Kada se argumenti prosleđuju po redosledu, navodite listu argumenata onim redosledom koji nalaže sintaksa procedure. Ako izostavite neki opcionni argument, morate da navedete zarez na mestu na kom bi se nalazio izostavljeni argument u listi.

Kada se argumenti prenose po nazivu, navodite ime argumenta iza kog sledi operator dodele (:=), a iza njega sledi vrednost argumenta. Argumente možete da navodite u proizvoljnom redosledu, a možete izostavljati opcione argumente.

Primer radi, deklaracija argumenata procedure Concatenate izgleda ovako:

```
Public Function Concatenate(A,B)
```

Svaka od sledećih naredbi poziva funkciju:

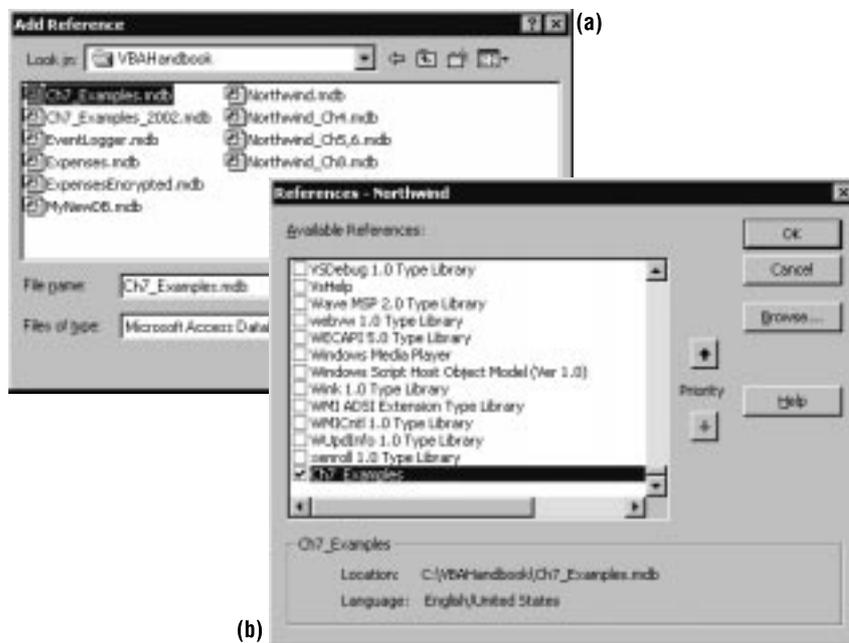
```
Call Concatenate("Johnson", "Diane")
Call Concatenate(B:="Diane", A:="Johnson")
```

U prethodnom poglavlju smo kreirali funkciju Concatenate, tako da se nalazi u drugoj bazi podataka. Funkcija će biti dostupna i tekućoj bazi podataka ako postavite referencu na bazu podataka Ch7_Examples.

1. Izaberite Tools → References i kliknite na dugme Browse u dijalogu References.
2. U dijalogu Add Reference, izaberite Microsoft Access Databases iz kombinovane liste Files Of Type, pronađite bazu podataka Ch7_Examples (videti sliku 8.12a) i kliknite OK. U listu se dodaje referenca na bazu podataka (videti sliku 8.12b).
3. U prozoru Immediate unesite sledeće linije i pritisnite Enter.

```
? Concatenate("Johnson", "Diane")
? Concatenate(B:="Diane", A:="Johnson")
```

Ovaj primer prikazuje da kada prosledite argumente po imenu, možete da navodite argumente po bilo kom redosledu.



SLIKA 8.12 Dodavanje reference na drugu bazu

Korišćenje variant argumenata

Pomoću variant argumenata je moguće kreirati proceduru koja prihvata više različitih tipova podataka. Na primer, funkcija `Multiply` množi svoje argumente i kao rezultat vraća proizvod.

```
Public Function Multiply (X,Y)
    Multiply = X*Y
End Function
```

Prilikom poziva ove funkcije možete da prosledite dve vrednosti bilo kog tipa podataka. Ako VBA može da konvertuje tipove podataka u kompatibilne tipove, proizvod se može izračunati.

Procedura `GetMultiply` deklarise promenljive sa nekoliko različitih tipova podataka i pokušava da izračuna proizvod prosledivanjem promenljivih u funkciju `Multiply`.

```
Public Sub GetMultiply()
    Dim intX As Integer, sngY As Single
    Dim dtmX As Date
    Dim strX As String, strY As String
    intX = 2
    sngY = 3
    dtmX = #5/11/2001#
    strX = "2"
    strY = "Three"
```

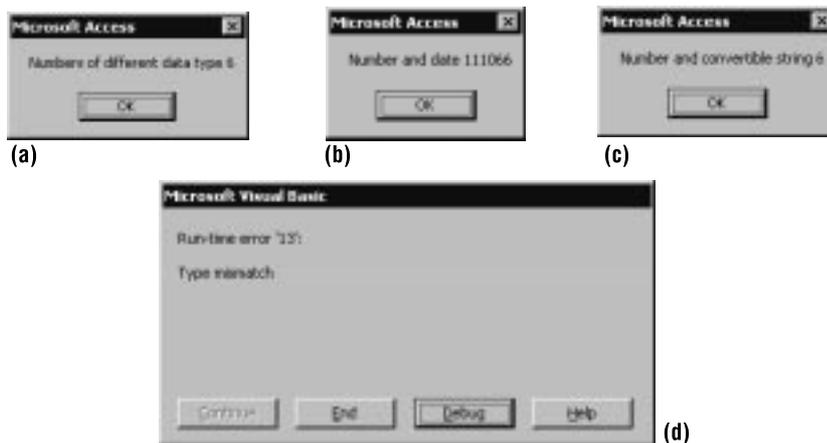
```

MsgBox "Numbers of different data type " & Multiply(intX, sngY)
MsgBox "Number and date " & Multiply(dtmX, sngY)
MsgBox "Number and convertible string " & Multiply(strX, sngY)
MsgBox "Number and non-convertible string " & Multiply(intX, strY)
End Sub

```

Unesite javnu funkcijsku proceduru `Multiply` i javnu potproceduru `GetMultiply` u okviru modula `basVariables`. Zatim u prozoru Immediate pokrenite proceduru `GetMultiply`.

Kada prosledite dva broja različitih numeričkih tipova podataka, VBA uspešno konvertuje tipove podataka (videti sliku 8.13a). Kada prosledite broj i datum, VBA konvertuje datum u njegov numerički ekvivalent i izračunava proizvod (videti sliku 8.13b). Kada prosledite broj kao string, npr. "2", VBA konvertuje string u numerički tip podataka i izračunava proizvod (videti sliku 8.13c). Međutim, kada prosleđujete string, npr. "Three", VBA ne može da konvertuje tip podataka i prikazuje poruku greške "Type mismatch" (videti poruku 8.13c).



SLIKA 8.13 Procedura uspešno konvertuje brojeve različitih tipova podataka (a), datum (b) i string kog je moguće konvertovati (c), ali generiše grešku kada VBA ne može da konvertuje string koji se ne može prevesti u broj (d)

Run-time greške možete da izbegnete pomoću variant argumenata, tako što ćete sami proveriti tip podataka pre nego što procedura pokuša da izvede izračunavanje. Za ispitivanje podataka možete da koristite ugrađene funkcije iz tabele 8.1. Primere procedura koje ispituju tipove podataka pre izračunavanja možete pronaći u Poglavlju 9.

Tabela 8.1: Ugrađene funkcije za ispitivanje podataka

Funkcija	Opis
IsArray(varname)	Vraća True ako je promenljiva niz.
IsDate(expression)	Vraća True ako se izraz može konvertovati u datum, ili False za ostale slučajeve. Izraz može biti datum ili string.
IsEmpty(expression)	Ako je izraz jedna variant promenljiva, vraća True ako promenljivoj još uvek nije dodeljena vrednost, ili ako je postavljena na vrednost Empty; u ostalim slučajevima vraća False.
IsErr(expression)	Vraća True ako je numerički izraz konvertovan u vrednost greške pomoću funkcije CVer, a False ako nije.
IsMissing(argname)	Vraća True ako se argument funkcije nije preneo iz druge funkcije, a False ako jeste.
IsNull(expression)	Vraća True ako izraz ne sadrži nikakve podatke. Izraz može biti numerički, ili tipa string.
IsNumeric(expression)	Vraća True ako se celi izraz može posmatrati kao broj, a False u ostalim slučajevima. Vraća False ako je izraz datum. Izraz može biti numerički, ili tipa string.
IsObject(expression)	Vraća True ako je izraz promenljiva tipa Variant ili Object, a False u ostalim slučajevima.
TypeName(varname)	Ako je varname naziv promenljive sa osnovnim tipom podataka vraća se string koji obezbeđuje informacije o promenljivoj.
VarType(varname)	Ako je varname naziv promenljive sa osnovnim tipom podataka, vraća se vrednost koja ukazuje na tip podataka.

Prosleđivanje objekata kao argumenata

I objekte možete da prosleđujete kao argumente procedure. Primera radi, kreiraćemo funkcijsku proceduru koja menja svojstvo Caption objekta. Koristićemo tip objekta Object za argument, tako da možemo da prosledimo bilo koji tip objekta koji ima svojstvo Caption. Zatim ćemo pozvati funkciju i proslediti objekat čiji natpis želimo da promenimo.

1. Kreirajte novi obrazac pod nazivom frmCaption. Postavite labelu na obrazac, unesite **Label** za njeno svojstvo Caption i postavite svojstvo Name na lblCaption.

2. Unesite, dole prikazanu, funkciju ChangeCaption u okviru standardnog modula basVariables.

```
Public Function ChangeCaption(objectname As Object)
    objectname.Caption = "My Caption"
End Function
```

3. Postavite komandno dugme pod nazivom cmdLabel na obrazac i postavite njegovo svojstvo Caption na Change Label Caption. Kliknite na svojstvo OnClick i kliknite na dugme Build sa desne strane tekstualnog polja. Unesite, dole prikazani, kod u šablonu za kod:

```
Private Sub cmdLabel_Click()
    Call ChangeCaption(lblCaption)
End Sub
```

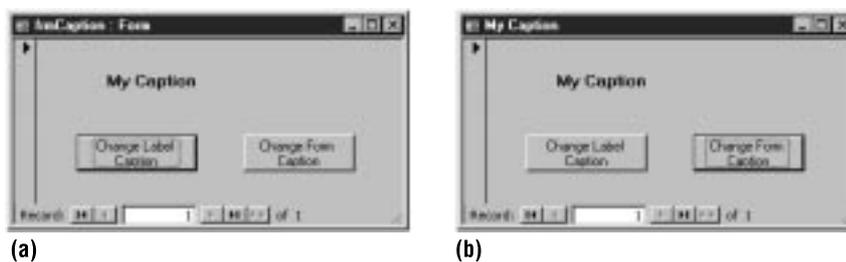
Ova procedura poziva funkciju `ChangeCaption` i prosleđuje labelu kao argument pomoću skraćene sintakse kojom se poziva na kontrolu. Kada kliknete na dugme, naslov labela se menja u `My Caption`.

- Postavite komandno dugme pod nazivom `cmdForm` na obrazac i postavite njegovo svojstvo `Caption` na `Change Form Caption`. Kliknite na modul obrasca i unesite dole prikazanu proceduru događaja (VBA automatski dodeljuje proceduru `OnClick` svojstvu dugmeta).

```
Private Sub cmdForm_Click()
    Call ChangeCaption(Forms!frmCaption)
End Sub
```

Ova procedura poziva funkciju `ChangeCaption` i prosleđuje obrazac kao argument pomoću sintakse sa usključnikom, koja ukazuje na obrazac preko njegovog imena. Kada kliknete na dugme, naslov obrasca se menja u `My Caption`.

- Uklonite obrazac i pređite u prikaz `Form`. Kliknite na dugme `Change Label Caption` (videti sliku 8.14a). Kliknite na dugme `Change Form Caption` (videti sliku 8.14b).



SLIKA 8.14 Prosleđivanje labela (a) i objekta `Form` (b) kao argumenata procedure

Prosleđivanje obrasca kao argumenta

Kada pokrećete proceduru smeštenu u modulu obrasca i želite da pozovete drugu proceduru i pri tome prosledite referencu obrasca, možete da koristite svojstvo `Me` za pozivanje obrasca. (Kada je procedura pokrenuta u modulu obrasca, svojstvo `Me` ukazuje na obrazac.) Na primer, kliknite na obrazac `frmCaption` i pređite u prikaz `Design`. Kliknite na modul obrasca i promenite proceduru događaja `cmdForm_Click()` na sledeći način:

```
Private Sub cmdForm_Click()
    Call ChangeCaption(Me)
End Sub
```

Snimite obrazac, pređite u prikaz `Form` i kliknite na dugme `Change Form Caption`. Naslov obrasca se menja kao i pre.

Za referenciranje obrasca možete da koristite i svojstvo Form. Na primer, promenite proceduru događaja cmdForm_Click na sledeći način, a zatim testirajte dugme.

```
Private Sub cmdForm_Click()  
    Call ChangeCaption(Form)  
End Sub
```

Korišćenje svojstva Form za prosleđivanje obrasca kao argumenta je veoma korisno ako koristite funkcijsku proceduru događaja. Na primer, možete da pozovete funkciju ChangeCaption za promenu naslova obrasca kada se obrazac otvori. Da biste videli kako ovo radi, pređite u prikaz Design i unesite =ChangeCaption(Form) za svojstvo događaja OnOpen obrasca. Snimite obrazac i pređite u prikaz Form. Funkcijska procedura se pokreće i menja naslov obrasca.

U postavci svojstva događaja, ne možete da koristite svojstvo Me za referenciranje obrasca. Ako biste uneli izraz =ChangeCaption(Me) za svojstvo događaja OnOpen obrasca, obrazac se ne bi mogao otvoriti u prikazu Form. Obrazac se vraća u prikaz Design.

Korišćenje neodređenog broja argumenata

Po unapred definisanom podešavanju, neophodno je navesti argumente u listi argumenata procedure. Ako ne pošaljete vrednosti svih argumenata, generiše se run-time greška. Ponekad je zgodno kreirati proceduru sa neodređenim brojem argumenata. Postoje tri načina za obezbeđivanje ove fleksibilnosti:

- Navođenjem da su argumenti opcioni.
- Korišćenjem argumenata sa korisnički definisanim tipovima podataka.
- Korišćenjem niza argumenata.

Ovaj odeljak prikazuje kako da navedete opcione argumente. Za više informacija o metodima za kreiranje procedura sa neodređenim brojem argumenata pogledajte odeljke "Korišćenje nizova" i "Kreiranje korisničkih tipova podataka".

Argument se označava opcionim tako što u listi argumenata ispred njegovog naziva postavite ključnu reč Optional. Opcioni argumenti mogu biti bilo kog tipa podataka. Po pravilu, opcioni argumenti se moraju navesti na kraju liste argumenata. Kada naznačite koji je argument opcioni, svi argumenti koji slede iza njega, takođe, moraju biti opcioni.

Da bismo ilustrovali te koncepte, pretpostavimo da želite da kreirate funkciju koja izračunava proizvod dva ili tri broja. Funkcijska procedura Product množi tri broja kada se proslede tri broja, ali se ne može uspešno izvršiti ako nije poslat opcioni argument. U okviru modula basVariables unesite proceduru Product.

```
Public Function Product (X, Y, Optional Z)  
    Product = X*Y  
    Product = Product*Z  
End Function
```

U prozoru Immediate unesite ?Product(2,3,4) i pritisnite Enter. Zatim unesite ?Product(2,3) i pritisnite Enter. Prvi proizvod se izračunava, ali drugi izaziva generisanje greške.

Na mestima na kojima bi kod procedure mogao izazvati run-time grešku u slučaju da argument nije prosleđen, možete da koristite funkciju `IsMissing` kojom utvrđujete da li je opcioni argument prosleđen. Funkcija `IsMissing(argname)` vraća `True` ako nije prosleđena vrednost za argument *argname*, a `False` u suprotnom. U ovom primeru možete da modifikujete funkciju `Product` tako da koristi funkciju `IsMissing`.

```
Public Function Product (X, Y, Optional Z)
    Product = X*Y
    If IsMissing(Z) Then Exit Function
    Product = Product*Z
End Function
```

Modifikovana procedura ispituje da li je u funkciju prosleđen treći argument. Ako treći argument nije prosleđen, procedura se završava bez izvršenja naredbe `Product = Product*Z`. Za više informacija o donošenju odluka u okviru procedure pogledajte Poglavlje 9.

Prosleđivanje podataka u proceduru događaja

Access automatski navodi i nazive i argumente potprocedura događaja. Ne možete da menjate listu argumenata na bilo koji način (osim na dole navedeni način). Svrha predefinisane liste argumenata za događaj je u obezbeđivanju komunikacije između Vašeg koda i Accessa. Primera radi, procedura događaja za događaj `NotInList` kombinovane liste ima sledeću sintaksu:

```
Private Sub controlName_NotInList(NewData As String, Response As Integer)
```

NewData je (read-only) string kojeg Access koristi za prosleđivanje teksta kojeg korisnik unese u tekstualni okvir kombinovane liste. Access prosleđuje tekst u proceduru događaja. *Response* je konstanta koja se koristi za naznačavanje da li se predefinisana poruka prikazuje ili ne, kao i za dodavanje vrednosti *NewData* u listu kombinovane liste. Procedura događaja vraća konstantu nazad u Access.

NAPOMENA

Jedina promena, koju možete da izvedete u listi argumenata procedure događaja, je to što možete da koristite standardne prefikse za VBA promenljive. Na primer, možete da zamenite argumente procedure događaja `NotInList` sa `strNewData` i `intResponse`. ■

Većina događaja nema unapred dodeljene argumente. Na primer, događaji `Click`, `Load`, `Activate`, `AfterUpdate`, `Initialize` i `Terminate` nemaju argumente. Događaj kod kog možete da otkazete predefinisano ponašanje nakon događaja ima argument pod nazivom `Cancel`, sa tipom podataka `Integer`. Na primer, događaji `DbClick`, `Unload` i `BeforeUpdate` imaju argument `Cancel As Integer`. Za otkazivanje predefinisano ponašanja nakon događaja postavljate argument `Cancel` na `True`, u naredbi dodele u proceduri događaja. U narednim poglavljima ćete videti primere prosleđivanja argumenata u procedure događaja.

NAPOMENA

Listu argumenata procedure događaja možete videti u tabeli "Argumenti za procedure događaja" na pratećem CD-u, u okviru `Tables\Chapter8.pdf`. ■

Korišćenje promenljivih na nivou modula

Promenljivu na nivou modula možete da kreirate postavljanjem naredbe deklaracije u odeljku Declarations modula, pomoću sledeće sintakse:

```
[Private|Public] variablename As type
```

Ključna reč koja se koristi u naredbi deklaracije određuje domen promenljive, tj. koje procedure mogu da vide promenljivu. Kao i procedure, promenljive na nivou modula mogu biti privatne i javne. Pomoću ključne reči `Private` kreirate privatne promenljive na nivou modula, a njih mogu videti samo procedure u okviru tog modula. Pomoću ključne reči `Public` kreirate javne promenljive na nivou modula, koje mogu da vide sve procedure u svim modulima baze podataka, ili projekta. Javna promenljiva na nivou modula se još naziva *globalna promenljiva*, a za nju možete da koristite prefiks `g` kao oznaku da se radi o javnoj promenljivoj na nivou modula.

Da biste izučili promenljive na nivou modula, izvedite naredne korake:

1. Deklarišite promenljive na nivou modula u odeljku Declarations modula `basVariable`, kao što je prikazano u donjem primeru.

```
Private strPrivate As String  
Public gstrPublic As String
```

2. Unesite dole prikazane procedure `PublicScoping` i `SamePublic` u okviru `basVariables` modula. Procedura `PublicScoping` dodeljuje vrednost `'Public'` javnoj promenljivoj na nivou modula `gstrPublic` i poziva proceduru `SamePublic` u istom modulu, a, zatim, proceduru `OtherPublic` koja se nalazi u drugom modulu.

```
Public Sub PublicScoping()  
    gstrPublic = "Public"  
    Call SamePublic  
    Call OtherPublic  
End Sub
```

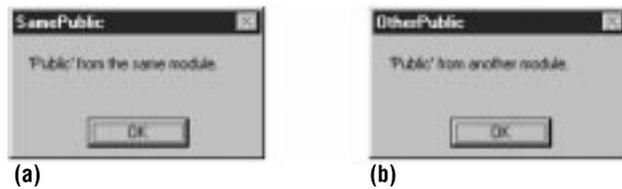
```
Public Sub SamePublic()  
    MsgBox gstrPublic & " from the same module.", , "SamePublic"  
End Sub
```

3. Kreirajte novi modul pod nazivom `basOther` i unesite proceduru `OtherPublic`:

```
Public Sub OtherPublic()  
    MsgBox gstrPublic & " from another module.", , "OtherPublic"  
End Sub
```

4. U prozoru Immediate unesite `PublicScoping` i pritisnite Enter.

Nakon dodele vrednosti javnoj promenljivoj, procedura `PublicScoping` poziva drugu proceduru iz istog modula, koja prikazuje promenljivu (videti sliku 8.15a). Zatim poziva sledeću proceduru koja se nalazi u drugom modulu, a koja, takođe, prikazuje vrednost promenljive (videti sliku 8.15b). Obadve pozvane procedure vide javnu promenljivu na nivou modula.



SLIKA 8.15 Javnu promenljivu na nivou modula vide sve procedure istog modula (a) i procedure bilo kog drugog modula u projektu (b)

- Unesite dole prikazane procedure `PrivateScoping` i `SamePrivate` u okviru `basVariables` modula. Procedura `PrivateScoping` dodeljuje vrednost `'Private'` privatnoj promenljivoj na nivou modula `strPrivate` i poziva procedure iz istog i drugog modula.

```
Public Sub PrivateScoping()
    strPrivate = "'Private'"
    Call SamePrivate
    Call OtherPrivate
End Sub
```

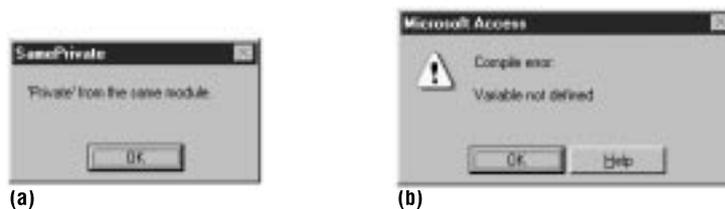
```
Public Sub SamePrivate()
    MsgBox strPrivate & " from the same module.", , "SamePrivate"
End Sub
```

- Unesite dole prikazanu proceduru `OtherPrivate` u okviru `basOther` modula:

```
Public Sub OtherPrivate()
    MsgBox strPrivate & " from another module. ", , "OtherPrivate"
End Sub
```

- U prozoru Immediate unesite `PrivateScoping` i pritisnite Enter.

Nakon dodele vrednosti privatnoj promenljivoj, procedura `PrivateScoping` poziva drugu proceduru iz istog modula, koja prikazuje promenljivu (videti sliku 8.16a). Procedura `Private-Scoping` poziva još jednu proceduru, iz drugog modula. Ovog puta, VBA generiše run-time grešku (videti sliku 8.16b). Pošto je promenljiva `strPrivate` privatna za procedure iz `basVariables` modula, nije vidljiva u proceduri `OtherPrivate` koja se nalazi u modulu `basOther`.



SLIKA 8.16 Privatnu promenljivu na nivou modula vide samo procedure istog modula (a), a ne i procedure drugih modula (b)

8. Kliknite na dugme Reset.

NAPOMENA

Za deklaraciju privatne promenljive na nivou modula možete da koristite i ključnu reč `Dim` umesto `Private`, ali ključna reč `Private` čini kod razumljivijim. ■

Javne promenljive na nivou modula deklarisanе u standardnom, ili nezavisnom class modulu (ali ne u modulu obrasca ili izveštaja) mogu biti korišćene i u drugim bazama podataka, kao reference na bazu podataka u kojoj su javne promenljive deklarisanе. Javne promenljive na nivou modula možete da ograničite na tekuću bazu podataka uključivanjem naredbe `Option Private Module` u odeljku `Declarations` standardnog ili nezavisnog modula klase u kom su deklarisanе. Nemojte da mešate ovo sa ključnom reč `Private`. Kada se koristi u naredbi `Option Private Module`, ključna reč `Private` znači privatno za datu bazu podataka, dok kada se koristi u naredbi deklaracije promenljive, ili procedure, znači privatno za modul.

Razumevanje vidljivosti promenljivih na nivou modula

Pravila vidljivosti (domena) se razlikuju za promenljive na nivou modula, koje su kreirane u standardnim modulima i promenljive kreirane u nezavisnim modulima klase, i promenljive kreirane u modulima obrazaca i izveštaja. Osnovna namena modula obrazaca ili izveštaja jeste smeštanje konstanti, promenljivih i procedura neophodnih za obrazac ili izveštaj. Zbog toga je vidljivost konstanti i promenljivih u modulu obrasca ili izveštaja ograničena. U ovim modulima ne postoji mogućnost kreiranja javnih konstanti, ali se neke promenljive mogu učiniti javnim. Evo nekih specifičnosti za promenljive na nivou modula kreirane u modulima za obrasce ili izveštaje.

- Javnu promenljivu ne možete da deklarirate kao string fiksne dužine.
- Ne možete da deklarirate javni niz.
- Javne promenljive deklarisanе u modulu obrasca ili izveštaja dostupne su samo modulima iste baze podataka, a ne i ostalim bazama podataka.

Procedure koje se smeštaju u bibliotekama za dinamičko povezivanje (DLL procedure), a koje se deklariraju u odeljku `Declarations` modula, mogu biti i javne i privatne. DLL procedure deklarisanе naredbom `Public Declare` se mogu pozivati iz bilo koje procedure bilo kog modula, dok DLL procedure deklarisanе naredbom `Private Declare` mogu da pozivaju samo procedure modula u kom je DLL procedura deklarisanа. U okviru standardnog ili nezavisnog modula klase možete da deklarirate i javne i privatne DLL procedure (ako ne koristite nijednu ključnu reč, DLL procedura se predefinisano deklarira kao javna). S druge strane, u modulima obrazaca ili izveštaja možete da deklarirate samo privatne DLL procedure.

Životni vek promenljivih na nivou modula

Kada otvorite Access bazu podataka, Access najpre otvara module koji su mu neophodni za početak, a kasnije otvara ostale module u skladu sa procedurama koje se pozivaju. Na primer, ako obrazac ima modul obrasca, taj modul se učitava u memoriju kada se obrazac prvi put otvori. Standardni ili modul klase se učitava kada prvi put pozovete neku proceduru koja se nalazi u tom modulu. Kada je modul učitav u memoriju, VBA dodeljuje potreban prostor za smeštanje svih promenljivih i konstanti smeštenih u odeljku Declarations. Kada se modul jednom učita u memoriju, tamo ostaje za svo vreme za koje je aplikacija pokrenuta. Ovo važi i za module obrazaca i izveštaja - zatvaranje obrasca ili izveštaja ne uklanja modul iz memorije. Posledica je to što promenljive na nivou modula (i statičke promenljive na nivou procedure) zauzimaju mnogo memorije i zadržavaju svoje vrednosti sve dok ne zatvorite bazu podataka. Zato je preporučljivo koristiti promenljive na nivou modula samo ako je to zaista neophodno.

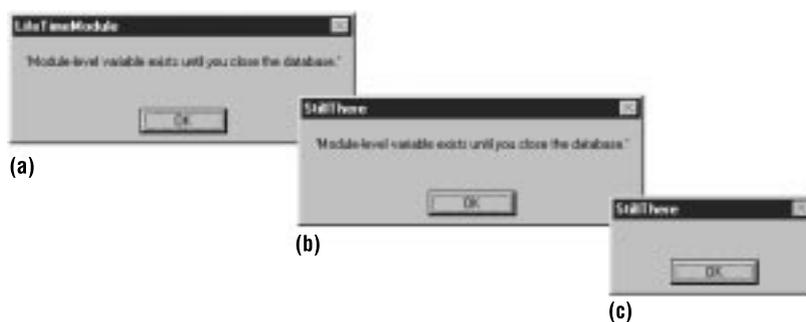
Da biste bolje ispitali promenljive na nivou modula sledite naredne korake.

1. U odeljku Declarations modula `basVariables`, deklarirate javnu promenljivu na nivou modula unošenjem **Public strModule As String**. (Ključna reč `Public` čini promenljivu vidljivom i u prozoru `Immediate`.)
2. U okviru modula `basVariables`, unesite dole prikazanu proceduru `LifeTimeModule`.

```
Public Sub LifeTimeModule()
    strModule = "'Module-level variable exists until you close '"
    & "the database.'"
    MsgBox strModule, , "LifeTimeModule"
End Sub
```

Procedura `LifeTimeModule` postavlja vrednost promenljive `strModule`, prikazuje poruku a zatim se završava.

3. U prozoru `Immediate` unesite `LifeTimeModule` i pritisnite `Enter`. Procedura dodeljuje i prikazuje vrednost (videti sliku 8.17a).



SLIKA 8.17 Procedura `LifeTimeModule` dodeljuje vrednost javnoj promenljivoj na nivou modula (a). Procedura `StillThere` prikazuje tekuću vrednost promenljive (b). Nakon resetovanja modula, promenljiva se inicijalizuje (c).

Promenljiva `strModul` zadržava vrednost postavljenu u ovoj proceduri sve dok ne promenite vrednost, zatvorite bazu podataka, ili kliknete na dugme Reset.

4. U okviru modula `basVariables` unesite dole prikazanu proceduru `StillThere`. Procedura `StillThere` prikazuje tekuću vrednost promenljive `strModule`.

```
Public Sub StillThere()  
    MsgBox strModule, , "StillThere"  
End Sub
```

5. U prozoru Immediate unesite `StillThere` i pritisnite Enter. Prikazuje se poruka sa tekućom vrednošću promenljive (videti sliku 8.17b).
6. Kliknite na dugme Reset u paleti sa alatima. VBA briše sve javne i privatne promenljive u modulu i oslabada prostor u memoriji koji su zauzimala.
7. Pokrenite proceduru `StillThere` u prozoru Immediate. Poruka je prazna jer je promenljiva inicijalizovana na string dužine nula (videti sliku 8.17c).

Korišćenje konstanti

Ako se u Vašem kodu stalno ponavljaju neke konstantne vrednosti, kod će biti čitljiviji ako koristite *konstante*. Konstanta predstavlja ime za određeni broj ili string koji se ne menja. Obično operativni sistemi i aplikacije koriste skupove unutrašnjih konstanti, mada možete da kreirate i sopstvene korisnički-definisane konstante. Korišćenje konstanti ubrzava kod, jer VBA upisuje vrednost konstante u kompajliranu verziju za vreme kompajliranja, tako da ne mora da proverava vrednosti za vreme izvršenja aplikacije.

Korišćenje unutrašnjih konstanti

U okviru Access VBA možete da koristite sistemski definisane konstante, Accessove unutrašnje konstante, VBA unutrašnje konstante i ADO unutrašnje konstante. *Sistemski definisane konstante* su konstante koje obezbeđuje operativni sistem. *Unutrašnja konstanta* predstavlja naziv kojim aplikacija označava broj koji se ne menja.

Korišćenje sistemski definisanih konstanti

Access ima četiri sistemski definisane konstante koje možete da koristite u svim objektima baze podataka, osim u modulima: `Yes`, `No`, `On` i `Off`. Access, takođe, obezbeđuje još tri sistemski definisane konstante koje možete da koristite u svim objektima baze podataka, uključujući i module: `True`, `False` i `Null`.

Korišćenje Accessovih unutrašnjih konstanti

Access VBA poseduje skup unutrašnjih konstanti koje možete da koristite za predstavljanje argumenata različitih Accessovih metoda, funkcija i svojstava. Ove konstante imaju prefiks `ac`. To su na primer `acCmdRefresh`, `acForm`, `acPrevious` i `acPreview`. Dobra karakteristika je mogućnost grupisanja unutrašnjih konstanti. Kada se radi o argumentu za metod, funkciju ili svojstvo koji zahteva unutrašnju konstantu, navodi se naziv skupa konstanti za argument, nazvan *skupom nabrojanih konstanti* za argument. Na primer, argument zapis (record) za metod `GoToRecord` ima skup unutrašnjih konstanti pod nazivom `AcRecord` koji uključuje konstante: `acFirst`, `acGoTo`, `acLast`, `acNewRec`, `acNext` i `acPrevious`.

Korišćenje VBA unutrašnjih konstanti

VBA poseduje i skup unutrašnjih konstanti koje možete da koristite prvenstveno za navođenje postavki svojstava i argumenata u VBA kodu. Te konstante imaju prefiks `vb`. Primeri su `vbAbort`, `vbRetry` i `vbCancel`, koje možete da koristite za navođenje argumenata za funkciju `MsgBox`. VBA grupiše unutrašnje konstante po kategorijama uključujući `ColorConstants`, `Constants`, `KeyCodeConstants` i `SystemColorConstants`. VBA takođe grupiše unutrašnje konstante u skupove nabrojivih konstanti, uključujući prefiks `vb` u nazivu grupe. Na primer, `VbDayOfWeek` je ime skupa nabrojanih konstanti `vbFriday`, `vbMonday`, `vbSaturday`, `vbSubday`, `vbThursday`, `vbTuesday`, `vbUseSystemDayOfWeek` i `vbWednesday`.

Korišćenje ADO unutrašnjih konstanti

ADO objektni modul poseduje skup unutrašnjih konstanti koje se koriste prvenstveno za navođenje postavki svojstava za objekte za pristup podacima i za argumente metoda. Te konstante imaju prefiks `ad`. Primeri su `adEditNone`, `adEditInProgres`, `adEditAdd` i `adEditDelete` koje koristite za navođenje svojstva `EditMode` za objekat `Recordset`. ADO, takođe, grupiše unutrašnje konstante u skupove nabrojivih konstanti i svaku grupu imenuje opisnim nazivom sa sufiksom `Enum`. Na primer, `LockTypeEnum` je skup nabrojivih konstanti koje predstavljaju raspoložive tipove zaključavanja zapisa, uključujući `adLockBatchOptimistic`, `adLockOptimistic`, `adLockPesimistic` i `adLockReadOnly`.

Prikazivanje unutrašnjih konstanti

Object Browser možete da koristite za prikazivanje unutrašnjih konstanti aplikacije. Za otvaranje Object Browsera iz prikaza Module, kliknite na dugme Object Browser u paleti sa alatkama, pritisnite F2 ili izaberite View → Object Browser.

U dijalogu Object Browser, u okviru kombinovane liste Project/Library nalazi se tekući projekat i sve reference koje su ranije postavljene u Ch7_Examples projektu (videti sliku 8.18a). Kombinovana lista uključuje i reference na sedam objektnih biblioteka koje su kreirane u direktorijumu System u fascikli Windows, još prilikom instalacije Accessa: Access (Microsoft Access 10.0 Object Library), ADODB (Microsoft ActiveX Data Objects 2.6 Library), ADOX (Microsoft ADO Ext 2.6 for DLL and Security), DAO (Microsoft DAO 3.6 Object Library), Office (Microsoft Office 10.0 Library), stdole (OLE Automation) i VBA (Visual Basic for Applications). Ove objektno biblioteke sadrže reference na konstante obezbeđene ovim aplikacijama osim referentnih informacija o objektima i komandama. Izvedite naredne korake kako biste prikazali neke od unutrašnjih konstanti:

1. Selektujte Access kao objektnu biblioteku. U listi Classes, u donjem levom uglu dijaloga, selektujte stavku AcFormView. Lista Members prikazuje skup nabrojivih konstanti (videti sliku 8.18b). AcFormView je skup nabrojivih konstanti za prikaz argumenta metoda OpenForm. Lista Classes uključuje imena svih skupova nabrojivih konstanti za metode, funkcije i svojstva, sa prefiksom Ac. Nabrojive konstante svakog skupa su prikazane u listi Members i imaju prefiks ac.



SLIKA 8.18 Lista kombinovane liste Project/Library navodi projekte i objektnu biblioteku za koje su u tekućoj bazi podataka postavljene reference (a). Object Browser možete da koristite za proučavanje skupova nabrojivih konstanti za argument metoda, funkciju ili svojstvo u Accessu (b), ili ADO (c).

2. Selektujte Constants iz liste Classes u donjem levom uglu dijaloga. U listi Members sa desne strane se prikazuju unutrašnje konstante. Iz liste Members izaberite acPrompt. U delu ispod liste se prikazuje broj koji odgovara konstanti i neke dodatne informacije o konstanti. Kliknite na dugme Help ukoliko je dozvoljeno, kako biste imali pristup online helpu.
3. Selektujte ADODB kao objektnu biblioteku. Iz liste Classes selektujte stavku DataTypeEnum. U listi Members se prikazuje skup nabrojivih konstanti (videti sliku 8.18c). Lista Classes uključuje imena svih skupova nabrojivih konstanti, s tim da se koriste opisna imena sa sufiksom Enum. Na primer, DataTypeEnum je skup unutrašnjih konstanti koje predstavljaju tipove podataka koje ADO koristi.
4. Iz liste Members izaberite adBoolean kako bi se prikazale dodatne informacije o konstanti.

NAPOMENA

U budućim verzijama Accessa možda će doći do promene numeričkih vrednosti koje predstavljaju unutrašnje konstante. Zato je u kodu poželjno koristiti unutrašnje konstante a ne njihove vrednosti. ■

Kreiranje sopstvenih konstanti

Sopstvene konstante možete da kreirate pomoću naredbi deklaracije. Kreiranje konstanti je slično kreiranju promenljivih: možete da kreirate konstante i na nivou procedure i na nivou modula. Razlika je u tome što se u naredbi deklaracije za konstantu uključuje i dodela konstantne vrednosti, kao u sledećem primeru:

```
Const constantname As type = constantvalue
```

Kada VBA čita naredbu deklaracije za konstantu, kreira i imenuje privremenu memorijsku lokaciju, dodeljuje potreban prostor za smeštanje navedenog tipa podataka i smešta vrednost. Deo As type je opcion; ako ne navedete tip podataka, VBA bira najefikasniji tip za smeštanje unete vrednosti.

Kreiranje konstanti na nivou procedure

Konstantu na nivou procedure kreirate i dodeljujete joj vrednost pomoću naredbe deklaracije u okviru procedure, na sledeći način:

```
Const constantname As type = constantvalue
```

Na primer, ova naredba postavlja konstantu intMax na Integer vrednost 144:

```
Const intMax As Integer = 144
```

Konstanta na nivou procedure nije vidljiva izvan procedure.

Kreiranje konstanti na nivou modula

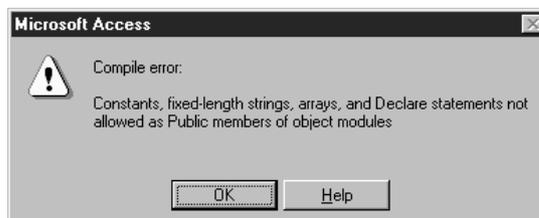
Konstante na nivou modula možete da kreirate postavljanjem naredbe deklaracije u Declarations odeljak modula. Za standardni modul možete da kreirate javne i privatne konstante na nivou modula pomoću sledeće sintakse:

```
[Public|Private] Const constantname [As type] = constantvalue
```

ko izostavite ključnu reč `Public` ili `Private`, konstanta se po unapred definisanom podešavanju (default) kreira kao privatna i dostupna je samo procedurama iz modula u kom je kreirana. Na primer, sledeće linije kreiraju `gsngInterestRate` kao javnu konstantu, koja je vidljiva i upotrebljiva u svim procedurama u projektu i `sngTaxRate` konstantu, koja je vidljiva i upotrebljiva samo u procedurama modula u kom je kreirana, dok je za procedure ostalih modula nevidljiva:

```
Public Const gsngInterestRate As Single = 7.75  
Private Const sngTaxRate As Single = 32.5
```

Za modul obrasca ili izveštaja možete da kreirate samo privatne konstante. To znači da unošenje naredbe, kao što je `Public Const intMyConstant As Integer = 2`, u okviru Declarations odeljka modula obrasca ili izveštaja dovodi do greške (videti sliku 8.19).



SLIKA 8.19 Poruka greške koja se prikazuje nakon pokušaja da se u modulu obrasca ili izveštaja deklarise javna konstanta

Korišćenje nizova

VBA obezbeđuje *nizove* kao zgodan način za efikasan rad sa nekoliko promenljivih koje imaju isti tip podataka. Na primer, možete da kreirate niz kontrola tekstualnih polja na obrascu, ili niz otvorenih obrazaca u bazi podataka.

Niz predstavlja seriju promenljivih na koje se pozivate pomoću istog imena i pomoću broja, koji se naziva *indeks*, a kojim se promenljive razdvajaju. Promenljive u jednom nizu moraju biti istog tipa podataka. Međutim, ako niz ima tip podataka `Variant`, individualni elementi niza mogu sadržati različite vrste podataka, kao što su brojevi, stringovi ili objekti. Možete da kreirate niz sa osnovnim tipom podataka, ili sa korisnički definisanim tipom. Kolekcije objekata tipa `Applications` i objekata za pristup podacima koje su razmatrane u poglavlju 5, "Osnove VBA programiranja", i 6, "Razumevanje ADO objektnog modela", predstavljaju nizove objekata. Na primer, `Forms` je niz otvorenih obrazaca, a `Tables` je niz tabela u bazi podataka.

Glavna namena nizova je da pojednostave kod i učine ga što efikasnijim. Kada želite da obradite skup elemenata niza, možete ih lako postaviti u petlju prolazeći kroz svaki element pomoću indeksa, broja kojim se prati broj ponavljanja. Poglavlje 9 objašnjava formiranje petlji preko nizova.

Možete da kreirate niz sa fiksnim brojem elemenata. (Niz fiksne veličine se još naziva *ordinarni niz*.) Takođe, možete da kreirate niz bez navođenja broja elemenata, a, zatim, postavite veličinu niza za vreme izvršenja procedure. Niz koji se deklarise bez navedenog broja elemenata se

naziva *dinamički niz*. Pretpostavimo da treba pratiti vrednosti u kontrolama određenog obrasca. Pošto znate koliko se kontrola nalazi na obrascu, za čuvanje vrednosti možete da koristite niz fiksne veličine. Međutim, ako želite da koristite proceduru za više obrazaca, koristite dinamički niz jer se broj kontrola može razlikovati za svaki obrazac.

U okviru procedure možete da kreirate *lokalne nizove* i *deljive nizove* u okviru Declarations odeljka modula. VBA standardno koristi nulu kao prvi indeks niza (donja granica). U ovom slučaju najveći indeks niza (*gornja granica*) je za jedan manji od broja elemenata. Nizovi mogu biti višedimenzionalni i mogu imati do 60 dimenzija.

NAPOMENA

Osnovni memorijski zahtev niza bilo kog tipa podataka iznosi 20 bajtova plus 4 bajta za svaku dimenziju niza, plus broj bajtova samog podatka. Memorija za podatke predstavlja proizvod broja elemenata i veličinu svakog elementa. Promenljiva tipa Variant zahteva 12 bajtova uz dodatak memorije neophodne za niz. ■

Kreiranje nizova fiksne veličine

Navođenjem naredbe `Dim` ili `Static`, u okviru procedure ili liste argumenata procedure, možete da kreirate niz fiksne veličine na nivou procedure. Za deklarisanje niza, iza naziva niza u okviru zagrada postavljate granice indeksa niza, kao u sledećim primerima. (Ukoliko prilikom imenovanja koristite mađarsku notaciju, možete da koristite prefiks `a` u nazivu promenljive kako biste ukazali da se radi o nizu.)

Naredba

```
Dim astrNames(20) As String
```

deklariše niz pod nazivom `astrNames` od 21 elementa tipa `String`, čiji se indeksi kreću od 0 do 20. Kada je donja granica niza 0, gornju granicu postavljate u okviru zagrada u naredbi deklaracije.

Naredba

```
Static asngTaxRates(1 To 10) As Single
```

deklariše statički niz pod nazivom `asngTaxRates` od 10 elemenata, tipa `Single` sa indeksima od 1 do 10. Ako je donja granica veća od nule, u naredbi deklaracije se u zagradama navode obe granice sa ključnom reči `To` između njih.

Naredba

```
Public Function Addresses(astrNames(20) As String)
```

deklariše niz pod nazivom `astrNames` od 21 elementa tipa `String` koji predstavlja argument funkcije `Addresses`.

Deklarisanjem niza u Declarations odeljku modula kreirate niz fiksne veličine na nivou modula. Za deljenje niza između svih procedura u svim modulima u projektu koristite ključnu reč `Public`, dok za deljenje, samo između procedura u modulu u kom je kreiran, koristite ključnu reč `Private`. Izuzetak je to što možete da kreirate javni niz i u modulu obrasca ili izveštaja. Evo nekih primera deklarisanja deljivih nizova.

Naredba

```
Public aintMatrix(9,9) As Integer
```

deklariše dvodimenzionalni javni niz pod nazivom `aintMatrix` sa 100 elemenata tipa `Integer`, sa parovima indeksa od (0,0) do (9,9).

Naredba

```
Private atxtAmounts(1 To 6) As TextBox
```

deklariše privatni niz pod nazivom `atxtAmounts` čije su elementi polja za tekst sa indeksima od 1 do 6.

Reinicijalizacija elemenata niza

Pomoću naredbe `Erase` možete da reinicijalizujete elemente niza fiksne veličine. Da biste bolje proučili nizove, kreirajte novi modul pod nazivom `basArrays` u koji ćete smeštati procedure koje ćemo kreirati u ovom odeljku. Zatim kreirajte dole prikazanu proceduru `CreateArray`.

```
Public Sub CreateArray()  
    Dim astrArray(3) As String  
    astrArray(0) = "Margaret"  
    astrArray(1) = "Peacock"  
    astrArray(2) = "Sales Representative"  
    MsgBox astrArray(1) & ", " & astrArray(0) & " is a " & astrArray(2)  
    Erase astrArray  
    MsgBox astrArray(1) & ", " & astrArray(0) & " is a " & astrArray(2)  
End Sub
```

Pokrenite proceduru u prozoru `Immediate`. Procedura kreira niz fiksne dužine sa 3 string elementa, dodeljuje im string vrednosti i prikazuje poruku sa njihovim vrednostima (videti sliku 8.20a). Pošto otkazete poruku, procedura koristi naredbu `Erase` za reinicijalizaciju elemenata na stringove dužine nula i prikazuje novu poruku sa reinicijalizovanim vrednostima (videti sliku 8.20b).



SLIKA 8.20 Kreiranje niza fiksne dužine (a) i korišćenje naredbe `Erase` za reinicijalizaciju elemenata niza (b)

NAPOMENA

Naredba `Erase` se razlikuje za nizove fiksne veličine i za dinamičke nizove. Kada koristite naredbu `Erase` za dinamički niz, prostor za skladištenje, koji je do tada dinamički niz zauzimao, se oslobađa za dalju upotrebu. Međutim, kada koristite naredbu `Erase` za niz fiksne veličine, elementi se reinicijalizuju, a memorija se zadržava. Ako morate da koristite niz samo za određeno vreme, deklarirate ga kao dinamički niz, a po završetku rada ga povratite pomoću naredbe `Erase`. ■

Kreiranje niza na osnovu vrednosti u listi

Pomoću funkcije `Array` možete da kreirate niz na osnovu vrednosti u listi, korišćenjem sledeće sintakse:

```
Array(arglist)
```

Funkcija `Array` kreira niz pomoću vrednosti iz liste argumenata *arglist* i vraća niz koji sadrži tip `Variant`. Vrednosti morate da razdvojite zarezima; ako u listi nema vrednosti, kreira se niz veličine nula.

Funkcije `LBound` i `UBound` vraćaju najmanji i najveći indeks u navedenim dimenzijama niza, respektivno:

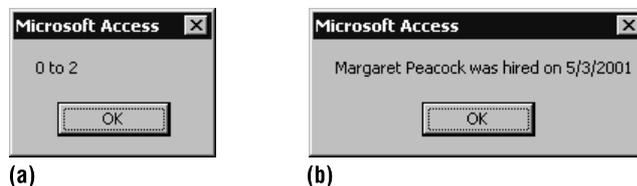
```
LBound(arrayname,dimension)  
UBound(arrayname,dimension)
```

Opcioni argument `dimension` predstavlja broj koji ukazuje u kojoj se dimenziji nalazi vraćena donja ili gornja granica, s tim da se sa 1 označava prva dimenzija, sa 2 druga, i tako redom. Ako izostavite dimenziju, pretpostavlja se da želite vrednosti za prvu dimenziju.

Da biste bolje ispitali ove funkcije, kreirajte dole prikazanu proceduru `ArrayFunction` u okviru `basArray` modula.

```
Public Sub ArrayFunction()  
    Dim varData As Variant  
    varData = Array("Margaret", "Peacock", #5/3/2001#)  
    MsgBox LBound(varData) & " to " & UBound(varData)  
    MsgBox varData(0) & " " & varData(1) & " was hired on " & _  
        varData(2)  
  
End Sub
```

Pokrenite proceduru u prozoru `Immediate`. Procedura kreira variant promenljivu, koristi funkciju `Array` za kreiranje niza od tri vrednosti iz liste, a rezultat dodeljuje variant promenljivoj. (Promenljiva `varData` mora biti tipa `Variant` tako da se u njoj mogu pamtit različiti tipovi podataka.) Poruka prikazuje donju i gornju granicu niza (videti sliku 8.21a), a u drugoj poruci su prikazane vrednosti (videti sliku 8.21b). U ovoj proceduri, promenljiva `varData` sadrži niz.



SLIKA 8.21 Funkcije `LBound` i `UBound` možete da koristite za utvrđivanje veličine niza kreiranog funkcijom `Array` (a). Poruka prikazuje tri vrednosti niza (b).

Kreiranje dinamičkih nizova

Dinamičke nizove koristite kada unapred ne znate koliko će niz imati elemenata. Dinamički niz se kreira pomoću dve naredbe: jednom naredbom deklarirate niz, a drugom naredbom navodite veličinu niza. Prilikom deklaracije niza koristi se ista sintaksa kao i kod niza fiksne veličine, osim što ne navodite broj elemenata niza (ne navodite ništa između zagrada), a, zatim, koristite naredbu `ReDim` za određivanje veličine i dodelu potrebnog prostora za skladištenje navedenog tipa podataka.

Kreiranje dinamičkog niza na nivou procedure

Dinamički niz na nivou procedure možete da kreirate uključivanjem naredbe deklaracije u okviru procedure sa sledećom sintaksom:

```
(Dim|Static) arrayname() [As type]
```

a, zatim, u proceduri navedete naredbu `ReDim` kojom određujete veličinu niza. Na primer, ova naredba kreira lokalni dinamički niz:

```
Dim astrNames() As String
```

Ova naredba postavlja 5 elemenata sa indeksima od 0 do 4:

```
ReDim astrNames(4)
```

Kreiranje dinamičkog niza na nivou modula

Dinamički niz na nivou modula kreirate postavljanjem naredbe deklaracije u `Declarations` odeljak modula, pomoću sledeće sintakse:

```
[Public|Private] arrayname() [As type]
```

a, zatim, uključujete naredbu `ReDim` u bilo kojoj proceduri u kojoj se niz poziva, kako bi se odredila i njegova veličina. Na primer, postavljanjem sledeće naredbe u `Declarations` odeljak modula kreirate dinamički niz na nivou modula:

```
Public asngTaxRates() As Single
```

Postavljanjem sledeće naredbe u okviru procedure postavlјate 5 elemenata niza sa indeksima od 1 do 5:

```
ReDim asngTaxRates(1 To 5)
```

Postavljanje granica dinamičkog niza

Granice dinamičkog niza se mogu postaviti pomoću promenljivih celobrojnog tipa, kao u sledećem primeru:

```
ReDim asngTaxRates(intlower To intupper)
```

Naredba `ReDim` određuje veličinu niza i inicijalizuje njegove elemente u skladu sa njihovim tipom podataka. To znači da `ReDim` inicijalizuje dve vrednosti na nulu, ako je reč o numeričkom nizu, ili na stringove dužine nula, ako je reč o nizu stringova, na vrednost `Empty` za niz tipa `Variant` i `Nothing` za niz objekata. Naredbu `ReDim` možete da koristite i za promenu veličine niza kada se za to ukaže potreba. Međutim, `ReDim` uništava sve trenutno smeštene vrednosti u elementima niza, osim ako niste koristili ključnu reč `Preserve`.

Ključnu reč `Preserve` možete da koristite kada želite da zadržite vrednosti elemenata niza, a istovremeno želite da promenite veličinu niza. Možete da promenite jedino poslednju dimenziju niza. Ako smanjite veličinu poslednje dimenzije niza, podaci u eliminisanim elementima se uništavaju. Kada koristite ključnu reč `Preserve`, ne možete da menjate broj dimenzija, već samo veličinu poslednje dimenzije niza.

Da biste bolje razmotrili ove ideje, u okviru `basArray` modula kreirajte dole prikazanu proceduru `DynamicArray`.

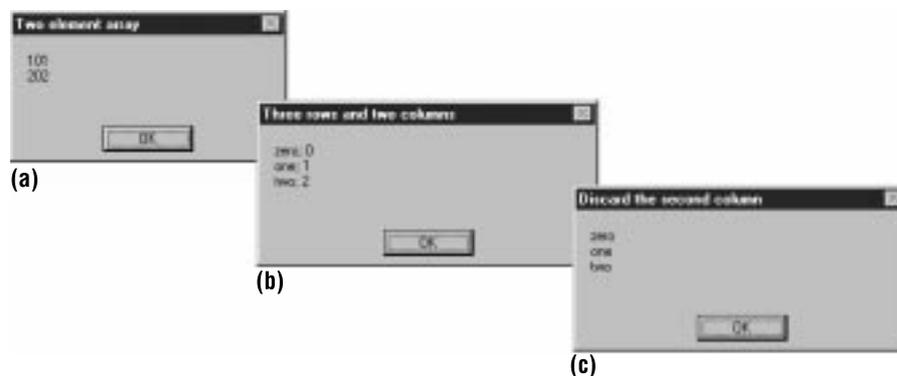
```
Public Sub DynamicArray()
    Dim avar() As Variant, str As String
    ReDim avar(2)
    avar(0) = 101
    avar(1) = 202
    MsgBox avar(0) & vbCrLf & avar(1), , "Two element array"
    ReDim avar(3,2)
    avar(0,0) = "zero": avar(0,1) = 0
    avar(1,0) = "one": avar(1,1) = 1
    avar(2,0) = "two": avar(2,1) = 2
    str = avar(0,0) & "; " & avar(0,1) & vbCrLf
    str = str & avar(1,0) & "; " & avar(1,1) & vbCrLf
    str = str & avar(2,0) & "; " & avar(2,1)
    MsgBox str, , "Three rows and two columns"
    ReDim Preserve avar(3,1)
    str = avar(0,0) & vbCrLf & avar(1,0) & vbCrLf & avar(2,0)
    MsgBox str, , "Discard the second column"
End Sub
```

Procedura kreira niz `avar` kao dinamički niz sa variant elementima, a za određivanje veličine niza koristi niz naredbi `ReDim`. Prva naredba `ReDim` definiše niz kao niz sa dva elementa. Druga naredba `ReDim` poništava dodeljene vrednosti i postavlja dvodimenzionalni niz sa tri vrste i dve kolone. Svaka od sledeće tri linije uključuje par naredbi dodele, s tim da se naredbe razdvajaju dvotačkom. Treća naredba `ReDim` uključuje ključnu reč `Preserve` kako bi se snimile vrednosti nakon određivanja veličine poslednje dimenzije. Sada je niz definisan kao niz od tri elementa sa tri vrste i jednom kolonom, a vrednosti iz druge kolone su uništene.

Pokrenite proceduru u prozoru Immediate. Tok procedure možete da pratite pomoću poruka koje se javljaju na ekranu (videti sliku 8.22).

NAPOMENA

Unutrašnja VBA konstanta `vbCrLf` omogućava pozicioniranje na početak novog reda (carriage return). Ovu konstantu možete da koristite kada u poruci koja se prikazuje na ekranu želite ispisivanje teksta u novom redu. ■



SLIKA 8.22 Procedura koristi dinamički niz i naredbu `ReDim` za predstavljanje niza od dva elementa (a) i dvodimenzionalni niz sa tri vrste i dve kolone (b). Procedura koristi naredbu `ReDim Preserve` za odbacivanje druge kolone (c).

Korišćenje nizova kao argumenata

Ako ne želite da navedete broj argumenata koje pozvana procedura može da primi, možete da koristite ključnu reč `ParamArray` kako biste mogli da određujete broj argumenata u zavisnosti od poziva procedure i taj broj postavljate u niz variant elemenata. Argument `ParamArray` mora biti poslednji argument u listi argumenata. Na primer, ako kreirate generičku proceduru, za izračunavanje prosečne u skupu numeričkih vrednosti, možete da koristite ključnu reč `ParamArray` u argumentima, na sledeći način:

```
Public Function Average(ParamArray aArgs())
```

Argument `aArgs` predstavlja naziv niza. Kada pozovete funkciju `Average`, možete da pošaljete proizvoljan broj argumenata, kao u sledećoj naredbi:

```
Call Average(2,5,46,23,1)
```

Kada koristite ključnu reč `ParamArray`, podrazumevano, argumenti su tipa `Variant` i ne možete da postavite neki drugi tip podataka. Zajedno sa `ParamArray` ne možete da koristite ključne reči `ByRef`, `ByVal`, ili `Optional`.

Kreiranje korisničkih tipova podataka

Pomoću osnovnih tipova podataka, navedenih u tabeli 7.1 u poglavlju 7, možete da kreirate i sopstvene tipove podataka. Možete da kreirate sopstveni tip podataka za jednu promenljivu u kojoj se čuva nekoliko elemenata sa nekoliko različitih tipova podataka. Na primer, možete da koristite jednu promenljivu kao referencu na ime kupca (tip String), datum porudžbine (tip Date) i količinu naručene robe (tip Currency). Korisnički tipovi podataka se mogu kreirati samo u Declarations odeljku modula.

Korisnički tip podataka se kreira pomoću naredbi Type i End Type. Nakon definisanja korisničkog tipa podataka, možete da deklarirate promenljivu korisničkog tipa, bilo kao promenljivu na nivou procedure, ili na nivou modula. Kada se promenljiva deklarise sa korisničkim tipom podataka, element niza možete da pozovete pomoću sledeće sintakse:

```
variablename.elementname
```

Elementima niza dodeljujete vrednosti pomoću regularnih naredbi dodele.

Da biste bolje proučili ove koncepte, kreirajte dole prikazani korisnički tip podataka pod nazivom OrderInfo u okviru odeljka Declarations modula basVariables, i dole prikazanu proceduru ViewOrder, takođe, u modulu basVariables.

```
Public Type OrderInfo
    Customer As String
    OrderDate As Date
    Amount As Currency
End Type

Public Sub ViewOrder()
    Dim ord As OrderInfo
    ord.Customer = "Alfreds Futterkiste"
    ord.OrderDate = #5/10/2001#
    ord.Amount = 3124.98
    MsgBox ord.Customer & " placed an order on " & ord.OrderDate & _
        " for $" & ord.Amount
End Sub
```

U prozoru Immediate pokrenite proceduru ViewOrder. Procedura deklarise promenljivu sa novim tipom OrderInfo i dodeljuje vrednosti za svaki element. Prikazuje se poruka sa elementima promenljive ord (videti sliku 8.23).



SLIKA 8.23 Promenljiva deklarirana korisničkim tipom podataka u kojoj se čuva ime kupca, datum porudžbine i količina

Ako koristite ključnu reč `Public`, korisnički tip podataka je dostupan (vidljiv) u svim procedurama i svim modulima projekta. S druge strane, ako koristite ključnu reč `Private`, ograničavate vidljivost samo na modul u kom je korisnički tip podataka definisan. Ako se korisnički tip podataka kreira u standardnom modulu, tip je po unapred definisanom podešavanju javni. Ako se kreira u modulu obrasca ili izveštaja, korisnički tip podataka može biti samo privatni i ne možete da promenite njegovu vidljivost ni pomoću ključne reči `Public`.

Korisnički tipovi podataka mogu sadržati objekte, nizove fiksne veličine, i dinamičke nizove, kao što je prikazano u sledećem primeru:

```
Private Type Orders
    frmInput As Form
    rptOutput As Report
    dbOrders As Database
End Type

Public Type CustomerInfo
    Customer As String
    Address(2) As String
    ' A fixed-size array with two elements
    Phone() As String
    ' A dynamic array
    FirstOrder As Date
End Type
```

Jedan korisnički tip podataka možete da koristite za prosleđivanje više argumenata u proceduru pomoću jedne promenljive. Na primer, četiri promenljive tipa `CustomerInfo` možete da prosledite u proceduru `CustomerDataEntry` u vidu jedne promenljive na sledeći način:

```
Public Sub CustomerDataEntry(CurrentCustomer As CustomerInfo)
```

Zaključak

Ovo poglavlje je dalo uvod za korišćenje promenljivih u procedurama. Glavni razlog za korišćenje promenljivih je to što se dobija brži kod i što navođenjem tipova podataka možete da kreirate ponovo upotrebljive procedure, jer se umesto navođenja imena određenih objekata navode promenljive. Najznačajnije tačke koje smo istakli u ovom poglavlju su:

- Možete da kontrolišete koje će procedure moći da koriste promenljivu i koliko će promenljiva "živeti", deklarisanjem promenljive na nivou procedure ili na nivou modula, sa odgovarajućim ključnim rečima.
- Da bi kod bio brži, treba uvek koristiti najspecifičniji tip podataka.
- Promenljive na nivou procedure možete da deklarirate na dva mesta: u posebnom odeljku za deklarisanje u okviru procedure, ili u listi argumenata procedure koja se poziva iz druge procedure.

- Promenljivu na nivou procedure deklarirate u okviru procedure koristeći sintaksu `Dim [Static] variablename As type`, ili u listi argumenata koristeći sintaksu `argumentname As type`.
- Promenljiva na nivou procedure nije direktno vidljiva iz bilo koje druge procedure. Jedini način na koji se ovakva promenljiva može koristiti u drugoj proceduri jeste da joj se prosledi kao lokalna promenljiva u vidu argumenta.
- Kada se procedura završi, Access uništava sve promenljive na nivou procedure i oslobađa memorijski prostor koji su zauzimale. Ovo predefinisano ponašanje možete prevazići upotrebom ključne reči `Static`, umesto `Dim`, prilikom deklaracije promenljive, ili korišćenjem ključne reči `Static` prilikom deklaracije procedure, tako da sve lokalne promenljive postanu statičke. Vrednost statičke promenljive se zadržava sve dok ne restartujete ili resetujete modul. Životni vek argumenta ne možete da produžite deklarisanjem u listi argumenata procedure.
- Promenljive na nivou modula možete da deklarirate u `Declarations` odeljku modula.
- Vrednosti i objekte možete da prosledujete kao argumente procedure na dva načina: po referenci, tako da procedura prima samu promenljivu, ili po vrednosti, tako da procedura prima samo kopiju promenljive.
- Procedure događaja imaju unapred definisane nazive i liste argumenata.
- Možete da kreirate sopstvene konstante, kako bi Vaš kod bio čitljiviji.
- Za rad sa nekoliko promenljivih istog tipa podataka možete da koristite nizove. Ako su elementi niza tipa `Variant`, elementi mogu biti različitog tipa podataka.
- Možete da kreirate korisničke tipove podataka za promenljive u kojima se pamti nekoliko elemenata različitih tipova podataka.

Sledećim poglavljem kompletiramo uvod u Access VBA objašnjavajući naredbe koje se koriste za kontrolu izvršenja instrukcija i kontrolu broja izvršenja instrukcija.