P R O G R A M I R A N J E
K R O Z
A P L I K A C I J E

Doc. dr Đukanović Slobodan

OSNOVE VBA

Visual Basic je programski jezik visokog nivoa razvijen iz jezika BASIC (Beginner's All-purpose Symbolic Instruction Code). Sam BASIC je nastao 1964.-te godine. Planirano je da to bude jednostavan jezik za učenje programiranja, tj. da približi programiranje korisnicima koji nisu imali veliko matematičko predznanje i nisu bili zainteresovani da ga steknu. Dizajneri su želeli da to čak bude i početni korak u savladavanju moćnijih jezika kao što su FORTRAN ili ALGO.

Od svog postanka, BASIC se razvijao i poboljšavao. U svojim ranim implementacijama, BASIC je bio interpreter. Kod interpretera se svaka linija programa zasebno tumači i ovakvo izvršavanje je obično sporo. Druga mana interpretera je da oni nemaju izvršni kod, odnosno za tumačenje programa neophodno je posedovati odgovarajući program za interpretiranje. Programski paket Matlab radi takođe kao interpreter. Većina modernih dijalekata BASIC-a dozvoljava kompajliranje koda što rezultuje u bržem mnogo izvršavanju koda.

BASIC je dobio na popularnosti 1991.-e kada je Microsoft pustio Visual Basic for Windows koji je popularizovao razvoj samostalna (standalone) aplikacija za Windows. Visual Basic for Applications (VBA) predstavlja implementaciju Visual Basic-a.

Prvi softverski proizvod koji je koristio VBA bio je Excel 5. VBA je vrlo brzo bio prihvaćen i stari makro jezik koji je koristio Excel 4 je bio potisnut. U druge aplikacije Office-a, tj. Word, Access, Power Point, Outlook, VBA je ušao u paketu Office 97. Sa pojavom Front Page-a, VBA postaje i njegov deo. Štaviše, VBA je izašao iz Microsoft Office okvira i biva uključen u Adobe, AutoCAD, Visio, SolidWorks itd. Postoje male razlike u načinu implementacije VBA u različitim aplikacijama, ali se većinom koristi isti jezik, kao što je VB6, i iste biblioteke. Ono što se menja je skup objekata određene aplikacije. Ukratko, VBA može biti ugrađen u aplikacije koje koriste COM (Component Object Model) objekte. COM je Microsoft-ov standard za definisanje interfejsa između objekata. Ključ u korišćenju VBA sa drugim aplikacijama leži u razumevanju objektnog modela aplikacije. VBA radi sa objektima aplikacije i svaka aplikacija (Word, Excel, Visio) ima svoj jedinstveni objektni model.

VBA programski kod je smešten u samom dokumentu (Word dokumentu ili Excel tabeli) i to u kodnim modulima, modulima klasa i korisničkim formama. Unutar kodnog modula se nalaze procedure, dok modul klasa sadrži definicije korisnički definisanih klasa. Korisnička forma predstavlja prozor na kojem se obične nalaze razne korisničke kontrole (dugmad, meniji, liste itd.). Mi ćemo uglavnom raditi sa kodnim modulima i korisničkim formama.

VBA programski kod se unosi u formi procedure. Procedure delimo na subprocedure (komandni makroi) i funkcije. Subprocedure predstavljaju skup VBA naredbi kojima se izvršava određeni zadatak i one ne vraćaju nikakav rezultat. Sa druge strane, funkcija takođe izvršava određeni zadatak i pritom vraća rezultat. Pored ove dve vrste, postoje još i Property procedure koje služe da definišu ili vrate osobinu (property value) za određeni objekat. Osobina može biti visina, širina, boja itd.

Ukratko o VBA

Kao što rekosmo, VBA radi sa objektima same aplikacije. Na primer, objekti kod Excel-a mogu biti radna sveska (workbook), radni list (worksheet), opseg (range), mapa (chart) itd. Klase objekata su hijerarhijski uređene. To znači da objekti mogu služiti kao kontejneri za druge objekte. Na primer, Excel je objekat Application i on sadrži druge objekte, kao što su Workbook

ili CommandBar. Objekat Workbook sadrži objekte Worksheet i Chart. Objekat Worksheet sadrži objekte Range, PivotTable itd.

Slični objekti formiraju kolekciju. Na primer, kolekcija Worksheet se sastoji od svih radnih listova u datoj svesci. Kolekcije za sebe predstavljaju objekte.

Referenciranje određenog objekta u hijerarhiji se vrši korišćenjem operatora tačka (.) koji služi kao separator između kontejnera i člana. Na primer, referenciranje sveske Book1.xls se vrši na sledeći način:

```
Application.Workbooks("Book1.xls")
```

Referenciranje lista Sheet1 sveske Book1 se vrši kao

```
Application.Workbooks("Book1.xls").Worksheets("Sheet1")
```

Slično, referenciranje ćelije A1 lista Sheet1 se vrši kao

```
Application.Workbooks("Book1.xls").Worksheets("Sheet1").Range("A1")
```

Na sreću, ne mora se koristiti ovoliki zapis svaki put za datu ćeliju. Ukoliko znamo da je Sheet1 aktivan list aktivne sveske Book1, gornji zapis se može redukovati na

```
Range("A1")
```

Objekti imaju osobine. Na primer, osobine opsega kao objekta mogu biti `Value` i `Type`. Pomoću VBA se mogu menjati osobine objekta. Referenciranje osobina se vrši kombinovanjem objekta i određene osobine uz pomoć operatora tačka. Na primer, osobina `Value` ćelije A1 sveske Sheet1 se može referencirati sa

```
Worksheets("Sheet1").Range("A1").Value
```

Ovako dobijena vrednost se kasnije može dodeliti nekoj VBA promenljivoj.

Objekti imaju metode. Metod je akcija koja se izvršava nad objektom. Na primer, jedna od metoda objekta `Range` je `ClearContents`, pomoću koje se briše sadržaj opsega. Pozivanje metode se takođe vrši pomoću operatora tačka, koji razdvaja objekat i metod. U prethodnom slučaju bi imali

```
Range("A1").ClearContents
```

Kao i svi moderni programski jezici, VBA poseduje složene tipove podataka, instrukcije za kontrolu toka programa, tj. naredbe uslovnog izvršavanja i cikluse, itd. O VBA sintaksi će biti najviše reči u delu sa Excelom. Sintaksa se, naravno, ne menja u zavisnosti od aplikacije.

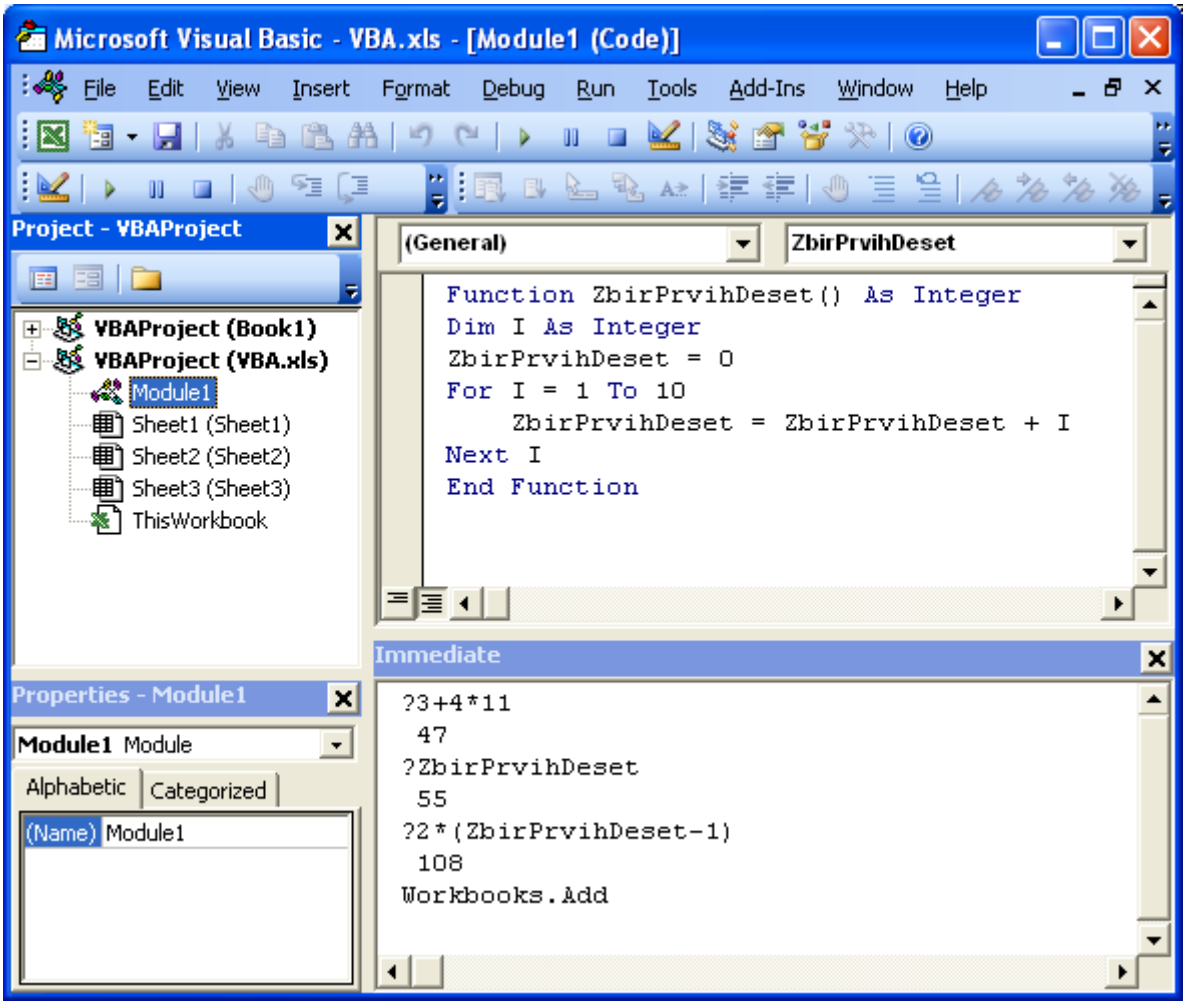
PROGRAMIRANJE EXCEL-A

Za ilustraciju rada sa VBA ćemo koristiti Excel 2003.

Visual Basic Editor

Kod prvih verzija Excel-a koje su imale VBA na raspolaganju, tj. Excel 5 i Excel 95, VBA modul se pojavljivao u formi zasebnog radnog lista. Počev sa Excel-om 97, VBA modulima se pristupa korišćenjem Visual Basic Editora (VBE). Moduli su, naravno, sačuvani zajedno sa radnim sveskama, samo nisu vidljivi dok se ne startuje VBE.

VBE se ne može startovati nezavisno od Excel-a, tj. Excel mora biti otvoren da bi VBE mogao da se startuje. Najjednostavniji način da se startuje VBE je koristeći prečicu **Alt+F11**. Drugi način bi bio pomoću opcije **Tools⇒Macro⇒Visual Basic Editor**. Startovanjem VBE dobija ćete dobiti nešto slično prozoru prikazanom na slici 1.



Slika 1. Prozor Visual Basic Editor-a.

Project Explorer prozor prikazuje strukturu svih trenutno otvorenih radnih sveski u Excel-u. Svaka radna sveska predstavlja jedan *projekat*. Ukoliko ovaj prozor nije vidljiv, može se otvoriti koristeći stavku View iz linije menija ili pomoću prečice **Ctrl+R**.

Code prozor (kodni prozor) sadrži VBA kod. Svaka stavka iz Project Explorer prozora ima pridruženi kodni prozor, koji se aktivira dvoklikom na određenu stavku ili pomoću opcije **View Code** iz padajućeg menija koji se dobija desnim klikom na određenu stavku. Na slici 1. je prikazan kodni prozor modula Module1, gde je definisana funkcija koja sabira prvih deset pozitivnih celih brojeva.

Za direktno izvršavanje VBA naredbi najbolje je koristiti *Immediate* prozor. On može biti vrlo koristan u ispravljanju vašeg koda jer pomoću njega možete testirati napisane procedure korak po korak. Ukoliko ovaj prozor nije vidljiv, može se otvoriti koristeći stavku View iz linije menija ili pomoću prečice **Ctrl+G**. Primer korišćenja ovog prozora je dat na slici 1. Ako želimo da dobijemo numeričku vrednost nekog izraza, koji može sadržati i poziv korisničke funkcije, linija koja se izvršava mora započeti znakom pitanja (prve tri naredbe sa slike 1.). Inače, dovoljno je napisati samo naredbu bez znaka pitanja (naredba `Workbooks.Add` sa slike 1.).

Properties prozor prikazuje osobine trenutno izabranog objekta u Project Explorer prozoru. Ukoliko ovaj prozor nije aktivan, aktivira se opcijom **View⇒Properties Window**.

Rad sa modulima

Ukoliko želimo da naši makroi budu dostupni u svim radnim listovima radne sveske, potrebno ih je snimiti u okviru modula, kao što je to prikazano na slici 1. Dodavanje novog modula se vrši pomoću opcije **Insert⇒Module**.

Brisanje modula se vrši odabirom opcije **Remove** iz padajućeg menija koji se dobija desnim klikom na dati modul.

Eksportovanje modula radi korišćenja u drugim projektima se vrši dabirom opcije **Export File** iz padajućeg menija koji se dobija desnim klikom na dati modul. Fajl eksportovanog modula ima ekstenziju **.BAS** (Basic). Ovaj format koriste Visual Basic moduli, što znači da se eksportovani VBA kod može koristiti u Visual Basic projektima. U suštini, .BAS fajlovi su čisto tekstualni fajlovi i mogu se editovati u Notepad-u ili sličnim tekst-editorima.

Importovanje modula (.BAS fajla) se vrši pomoću opcije **File⇒Import File**.

Promena imena modula se vrši pomoću *Properties* prozora, kao što je prikazano na slici 1.

Subprocedure

Subprocedure, ili komandni makroi, su najčešći tip procedura i one obično sadrže komande koje predstavljaju ekvivalent opcijama menija i drugim programskim komandama. Važna osobina subprocedura je da, kao i obične programske komande, imaju uticaj na svoje okruženje. Na primer, subprocedure mogu menjati format ćelije.

Subprocedure ne vraćaju nikakvu vrednost!

Format subprocedura je sledeći:

```
Sub ImeProcedure(argument1, argument2, ...)
VBA naredbe
End Sub
```

Subprocedure počinju rečju `Sub` i završavaju se sa `End Sub`. Ulazni argumenti subprocedure su razdvojeni zarezima.

Prilikom davanja imena subproceduri moramo ispoštovati par jednostavnih konvencija. Naime, ime procedure ne može biti duže od 255 karaktera, prvi karakter mora biti slovo, a ostali karakteri mogu biti slova, cifre ili karakter podvlaka (_). U imenu procedure nisu dozvoljeni spejsovi i tačke! Na primer, dozvoljena imena subprocedura su

```
PromeniBoju, Para5, Debeli_lad
```

dok su nedozvoljena imena

```
5Para, Mirko&Slavko, Promeni boju
```

Što se tiče imena procedura, VBA nije case-sensitive, tj. ne pravi razliku između velikih i malih slova. Tako se imena PromeniBoju, Promeniboju i PROMENIBOJU odnose na istu proceduru.

Ispred ključne reči `Sub` u zaglavlju subprocedure se mogu naći i dodatne ključne reči koje određuju opseg subprocedure, tj. njenu vidljivost u odnosu na druge module u projektu. Tako se ispred `Sub` mogu naći reči `Private` ili `Public`. `Private` označava da je procedura dostupna samo procedurama iz istog modula, dok `Public` označava da je procedura vidljiva svim procedurama iz svih ostalih modula projekta.

Nezavisno od `Private` i `Public`, ispred `Sub` se može naći i ključna reč `Static` koja označava da se promenljive u proceduri čuvaju i nakon završetka njenog izvršenja.

Izvršavanje subprocedure se može forsirano prekinuti korišćenjem instrukcije `Exit Sub` koja se može naći bilo gde u subproceduri.

U nastavku je data jedna jednostavna procedura koja ispisuje pozdravnu poruku.

```
Sub PocetnaPoruka()
'
' Ovo je prvi makro napisan u okviru ovog kursa
'
Dim Datum As String, Vreme As String
Datum = Date
Vreme = Time
MsgBox "Pozdrav! Ovo je prvi cas VBA." + vbCrLf + _
       "Danas je " + Datum + ", i sad je " + Vreme
End Sub
```

VBA naredbe korišćene u ovoj proceduri će biti objašnjene kasnije. Napomenućemo samo da linije koda koje počinju apostrofom (') predstavljaju komentar u VBA. Komentari mogu poslužiti za dokumentovanje procedure i VBA ih potpuno ignoriše prilikom izvršavanja koda.

Subprocedure se mogu izvršiti na nekoliko načina. Izvršenje iz radne sveske, tj. bez ulaženja u VBE, vrši se odabirom opcije **Tools⇒Macro⇒Macros**. U dobijenom prozoru odaberemo željeni makro (`PocetnaPoruka` u našem slučaju) i kliknemo na dugme **Run**. Pomoću dugmeta **Edit**, koje se nalazi na istom prozoru, možemo startovati VBE da bi editovali naš makro. Za ovako izvršenje subprocedure, ista ne sme imati ulaznih argumenata.

Iz prozora VBE, makro se može izvršiti koristeći Immediate prozor. Ukoliko subprocedura nema argumenata, potrebno je napisati samo njeno ime (bez znaka pitanja). Ukoliko subprocedura ima argumenata, argumente navesti u zagradi, kao što je dato u samoj definiciji subprocedure. Drugi način izvršenja iz prozora VBE je pomoću **Debug** palete alatki. Pozicioniramo se kursorom u kod subprocedure i pritisnemo dugme **Run Sub/User Form**. Slično kao u prethodnom slučaju, za ovako izvršenje, subprocedura ne sme imati ulaznih argumenata.

Funkcije

Drugi često korišćeni tip procedura su korisnički definisane funkcije, koje rade slično kao i ugrađene Excel-ove funkcije (na primer, SUM ili PRODUCT). Osnovna karakteristika im je da primaju argumente, odrade određenu operaciju sa njima i vrate rezultat. Iako mogu uticati na izgled okruženja, dobro definisana funkcija ne bi trebalo da utiče na svoje okruženje.

Sintaksa korisničkih funkcija je sledeća:

```
Function ImeFunkcije (argument1, argument2, ...) as Tip
VBA naredbe
ImeFunkcije = VracenaVrednost
End Function
```

Funkcije se definišu u okviru modula.

Sintaksa funkcija je vrlo slična subprocedurama. Pravila za imenovanje funkcija su ista kao i za subprocedure.

Osobenost ovih funkcija je da ime funkcije predstavlja izlaznu promenljivu. Ovo je ilustrovano linijom `ImeFunkcije = VracenaVrednost`. Tip izlazne promenljive se opciono može navesti u zaglavlju funkcije, nakon zagrade sa listom argumenata. O tipovima VBA promenljivih će biti reči kasnije.

Ispred ključne reči `Function` u zaglavlju funkcije se mogu naći reči `Private` ili `Public`. `Private` označava da je funkcija dostupna samo procedurama iz istog modula, dok `Public` označava da je funkcija dostupna svim procedurama iz svih modula svih aktivnih Excel-ovih projekata.

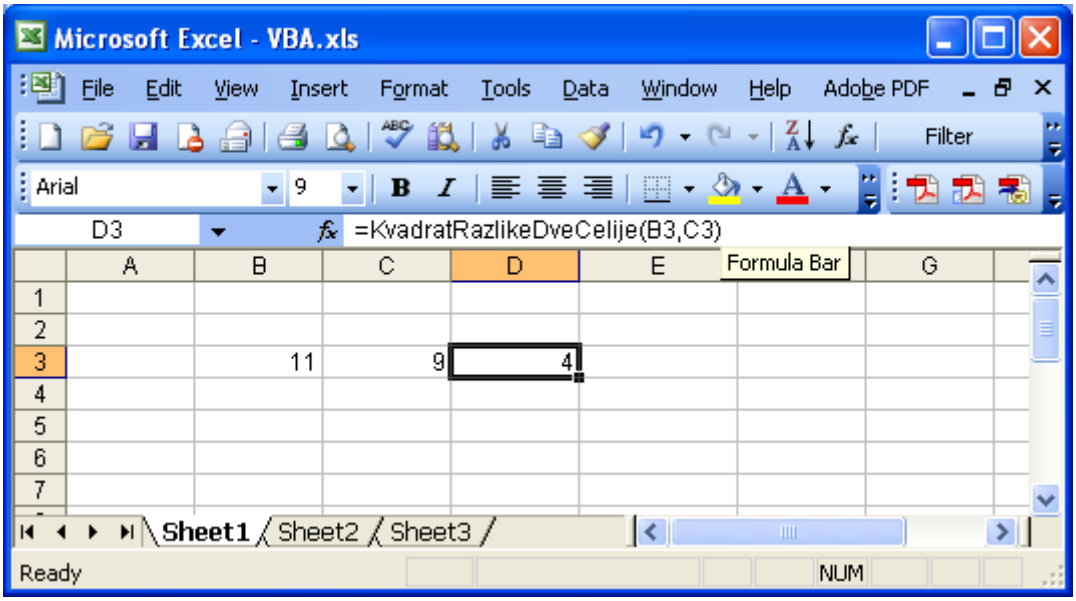
Ispred `Function` se može naći i ključna reč `Static` koja označava da se promenljive deklarisane u funkciji čuvaju između poziva funkcija.

Slično kao kod subprocedura, izvršavanje funkcije se može prekinuti korišćenjem instrukcije `Exit Function`. U tom slučaju je potrebno voditi računa da je imenu funkcije, kao izlaznoj promenljivoj, dodeljena vrednost pre instrukcije `Exit Function`.

Primer jedne korisničke funkcije, koja računa kvadrat razlike dva uneta broja, je dat ispod.

```
Function KvadratRazlikeDveCelije(X As Double, Y As Double) as Double
Dim Z As Double
Z = (X - Y) ^ 2
KvadratRazlikeDveCelije = Z
End Function
```

Ova funkcija se u radnom listu poziva kao i svaka druga Excel-ova funkcija. Argumenti funkcije `X` i `Y` mogu biti proizvoljni brojevi, a mogu biti i ćelije radnog lista, kao što je to prikazano na slici 2.



Slika 2. Izvršenje korisničke funkcije u okviru radnog lista.

Promenljive u VBA

VBA sadrži veliki broj tipova promenljivih. Dajemo u nastavku spisak tipova promenljivih uz kratak opis svake od njih.

`Boolean`

> `Boolean` je promenljiva koja može imati dve vrednosti, *True* (logički tačno) ili *False* (logički netačno). Za postavljanje vrednosti `Boolean` promenljive se mogu koristiti ključne reči `True` i `False`, kao u primeru ispod.
>
> ```
> Dim X As Boolean
> X = True
> ```
>
> Prilikom konvertovanja `Boolean` promenljive u neki drugi tip podataka (na primer, u numeričku vrednost), `True` vraća -1, a `False` vraća 0. Sa druge strane, kada se numerička vrednost konvertuje u `Boolean` vrednost, jedino 0 vraća `False`, a svi ostali brojevi vraćaju `True`. `Boolean` promenljiva zauzimaju dva bajta.

`Byte`

> Ovaj tip podatka se koristi za smeštanje manjih pozitivnih brojeva, od 0 do 255. `Byte` promenljiva zauzima 1 bajt memorije, najmanje memorije u odnosu na bilo koji drugi tip.

`Currency`

> Currency tip služi za rad sa novcem, odnosno tamo gde je vrlo važna tačnost kalkulacija. Podržava pozitivne i negativne brojeve veličine do 15 cifara sa leve strane decimalne tačke i 4 cifre sa desne strane. Opseg ovog tipa je od –922 337 203 685 477.5808 do 922 337 203 685 477.5807. Za razliku od `Single` i `Double` tipova podataka, `Currency` tip je tačan, tj. nije zaokružen. Ovaj tip zauzima 8 bajtova.

`Date`

Ovaj tip podataka se koristi za smeštanje datuma i vremena. Koristi osam bajtova. VBA može da radi sa datumima iz opsega 1.1.100.–31.12.9999. i vremenom od 0:00:00 do 23:59:59. Dodeljivanje vrednosti `Date` promenljivoj se vrši pomoću karaktera taraba (#) sa kojim počinje i završava se vrednost. Na primer, datum 14.9.2009. bi se u `Date` promenljivu `Danas` uneo na bilo koji od sledećih načina:

```
Danas = #September 14, 2009#
Danas = #14/9/2009#
```

Vreme se unosi na sličan način, tj. pomoću karaktera #. Na primer, `Date` promenljivoj `Sad` se dodela vrši na sledeći način:

```
Sad = #17:15 PM#
```

U Code prozoru, VBA konvertuje unešeni podatak u format datuma koji je sistemski podešen. Na primer, ako unesete September 14, 2009, VBA će, verovatno, prikazati 9/14/2009. Slično, vreme 17:15 PM bi se verovatno konvertovalo u 5:15:00 PM.

`Decimal`

`Decimal` promenljive se smeštaju kao 12-bajtni označeni celi brojevi skalirani promenljivim stepenom broja 10. Stepen broja 10 određuje broj decimalnih mesta broja i varira od 0 do 28. Bez decimalnih mesta (stepen broja 10 je 0), granice opsega vrednosti su +/-79 228 162 514 264 337 593 543 950 335. Sa 28 decimalnih mesta, granice opsega vrednosti su +/-7.9228162514264337593543950335, a najmanja nenulta vrednost je $+/-10^{-28}$. `Decimal` tip podataka je podtip tipa `Variant` i nije tip podataka za sebe. To znači da ne možete deklarisati promenljivu tipa `Decimal`.

`Single`

`Single` tip podataka je 4-bajtni tip koji radi sa brojevima u pokretnom zarezu (IEEE format). `Single` može da radi sa negativnim brojevima u opsegu od -3.402823×10^{38} do -1.401298×10^{-45} i pozitivnih u opsegu od 1.401298×10^{-45} do 3.402823×10^{38}.

`Double`

`Double` je 8-bajtni tip podataka, i kao i `Single`, namenjen za brojeve u pokretnom zarezu (IEEE format). U njega se mogu smestiti negativni brojevi od $1.79769313486231\times10^{308}$ do $-4.94065645841247\times10^{-324}$, i pozitivni brojevi od $4.94065645841247\times10^{-324}$ do $1.79769313486232\times10^{308}$.

`Integer`

`Integer` tip podataka je 2-bajtni tip podataka koji može da radi sa cijelim brojevima iz opsega od -32 768 do 32 767. Ovo je možda najčešće korišćeni tip VBA promenljivih jer je dati opseg dovoljan za najveći broj zadataka.

`Long`

`Long` tip je 4-bajtni tip podataka namenjen za rad sa celobrojnim vrednostima iz opsega od -2 147 483 648 do 2 147 483 647.

`Object`

U promenljivima tipa `Object` se smeštaju adrese objekata. Dodeljivanje vrednosti promenljivoj tipa `Object` se vrši naredbom `Set`. `Object` promenljive zauzimaju 4 bajta.

`String`

`String` tip podataka služi za smeštanje teksta. Stringovi mogu da sadrže slova, brojeve, blanko, interpunkcijske znake i specijalne karaktere. Postoje stringovi promenljive i fiksne dužine.

Stringovi promenljive dužine mogu sadržati i do dve milijarde (tačnije, 2^{31}) karaktera; zauzimaju 10 bajtova plus memoriju koja je potrebna za smeštanje stringa.

Stringovi fiksne dužine mogu sadržati od jednog do 64 000 karaktera i oni zauzimaju samo memorijski prostor potreban za smeštanje stringa. Ako je podatak koji je dodeljen stringu kraći od fiksne dužine, VBA dodaje stringu blanko znake da bi ga dopunio do fiksne dužine. Ako je podatak koji je dodeljen stringu duži od fiksne dužine, VBA odseca deo koji je suvišan. VBA broji karaktere sa leve strane stringa. Na primer, ako dodelite string "Programiranje" string promenljivoj fiksne dužine od sedam karaktera, VBA će smestiti samo podstring "Program".

`Variant`

Ovaj tip podataka VBA dodeljuje svim promenljivim čiji tip nije eksplicitno deklarisan, tako da će deklaracija tipa

```
Dim Indeks
```

kreirati `Variant` promenljivu `Indeks`. `Variant` promenljiva se može deklarisati i eksplicitno, na primer

```
Dim Indeks as Variant
```

Promenljive tipa Variant mogu da smeštaju većinu ostalih tipova podataka, pri čemu treba voditi računa o sledećim pojedinostima:

- Promenljive tipa `Variant` ne mogu sadržati stringove fiksne dužine.
- Promenljive tipa `Variant` sadrže i četiri specijalne vrednosti: `Empty` (znači da promenljiva nije inicijalizovana), `Error` (specijalna vrednost koja se koristi za pronalaženje grešaka u proceduri), `Nothing` (specijalna vrednost koja služi za odvajanje promenljive od objekta za koji je bila vezana) i `Null` (koristi se da označi da promenljiva namerno ne sadrži podatke).
- Zbog svojih dodatnih mogućnosti, `Variant` promenljive zauzimaju više memorije od ostalih tipova promenljivih. `Variant` promenljive koje sadrže brojeve zauzimaju do 16 bajtova, a `Variant` promenljive koje sadrže karaktere zauzimaju i do 22 bajta, plus memorijski prostor potreban za smeštanje karaktera.

Deklaracija VBA promenljivih

Deklaracija predstavlja navođenje imena i tipa promenljive koju ćemo koristiti u programu. Deklaracija se vrši korišćenjem reči `Dim`, što je skraćeno od Dimension, i može se vršiti unutar procedure, kad dobijamo promenljivu proceduralnog opsega, ili se može vršiti na vrhu modula, kada dobijamo promenljivu opsega modula. Tako se, na primer, celobrojna promenljiva `X` deklariše na sledeći način:

```
Dim X as Integer
```

Ukoliko se ne navede tip promenljive, podrazumeva se tip `Variant`. Promenljive se, u suštini, mogu deklarisati bilo gde unutar procedure. Jedino ograničenje je da deklaracija promenljive mora prethoditi prvoj upotrebi te promenljive u proceduri. Ipak, dobra je praksa deklaraciju vršiti

odmah na početku procedure, tj. pre ostalih naredbi, jer na taj način vrlo brzo možemo proveriti pravilnost deklaracije svih promenljivih bez lutanja po kodu.

Deklarisanje više promenljivih u jednoj liniji se vrši tako što se deklaracije pojedinih promenljivih odvajaju zarezima, na primer

```
Dim X As Integer, Y As Double, S as String
```

Ukoliko imate više promenljivih istog tipa, tip se mora navesti za svaku promenljivu posebno. Tako će linija

```
Dim X As Integer, Y As Integer
```

deklarisati `X` i `Y` kao celobrojne promenljive, dok će skraćeni oblik

```
Dim X, Y As Integer
```

deklarisati `Y` kao celobrojnu promenljivu i `X` kao `Variant` promenljivu.

Ovakva deklaracija promenljivih, tj. korišćenjem reči `Dim`, se još naziva i *eksplicitnom deklaracijom*. Drugi način je tzv. *implicitna deklaracija*, koja se vrši kada promenljivu koristimo u izrazu bez njenog prethodnog deklarisanja. Ako, na primer, promenljiva `S` nije deklarisana, onda će izraz

```
S = "Ulcinjska solana"
```

deklarisati promenljivu `S` tipa `Variant`.

Prednost implicitne deklaracije je da ne morate da vodite računa o deklaracijama, već da promenljive koristite kako vam zatrebaju (npr. kao u Matlab-u). Međutim, ovakva deklaracija ima i svojih mana. Prva mana je da se lako može napraviti greška kada se radi sa dužim imenima promenljivih. Pretpostavimo, na primer, da u proceduri implicitno deklarišemo promenljivu `ProsekPlata`, i da se kasnije greškom pozovemo na tu promenljivu kao `ProsekPiata`. VBA ovo neće tumačiti kao grešku, već će implicitno deklarisati novu promenljivu koja nema veze sa promenljivom `ProsekPlata`. Pri radu sa velikim programima, ovo može biti veliki problem, jer se, zbog sličnosti imena promenljivih, greške teško pronalaze u šumi izraza. Ovo se ne može desiti ako sve promenljive koje koristite u programu eksplicitno deklarišete. Druga mana je ta da `Variant` promenljive zauzimaju više memorijskog prostora od ostalih tipova. Ova mana je zanemarljiva kod malih programa, ali kod većih može doći do izražaja, pogotovo kod slabijih računara, jer rad sa `Variant` promenljivim zahteva više vremena od rada sa ostalim tipovima. Razlog ovome je stalna provera vrste podataka smeštene u `Variant` promenljivoj.

Pri implicitnoj deklaraciji možemo koristiti tzv. *karakter za definiciju tipa* (*type-declaration character*) koji se nadovezuje na ime promenljive da definiše njen tip. Karakteri za definiciju tipa su dati u tabeli 1.

Tip promenljive	Karakter za definiciju tipa
Integer	**%**
Long	**&**
Currency	**@**
Single	**!**
Double	**#**
String	**$**

Tabela 1. Karakteri za definiciju tipa.

Na primer, `Double` promenljivu `Broj` možemo implicitno deklarisati na sledeći način:

```
Broj# = 23.7
```

Eksplicitno deklarisanje se može forsirati čekiranjem opcijom *Require Variable Declaration* koja se nalazi na tabu *Editor* prozora **Tools⇒Options** u VBE. VBE dodaje iskaz `Option Explicit` u sve nove module, koji zahteva eksplicitnu deklaraciju promenljivih u tim modulima.

Opseg VBA promenljivih

Svaka promenljiva ima svoj *opseg*, tj. module i procedure gde se promenljiva može koristiti, i *trajanje* (ili životni vek), koje definiše vreme zadržavanja te promenljive u memoriji. Trajanje promenljive je usko vezano sa njenim opsegom.

U VBA, promenljiva može imati tri tipa opsega:

- *proceduralni* ili *lokalni,*
- *privatni* ili *opseg modula*, i
- *javni*.

Proceduralni opseg

Promenljiva sa proceduralnim opsegom je dostupna sarno u proceduri gde je definisana. Kao rezultat toga, trajanje lokalne promenljive je ograničeno na trajanje same procedure.

Implicitno deklarisane promenljive imaju proceduralni opseg. Eksplicitno deklarisanje lokalne promenljive se vrši pomoću ključnih reči `Dim` ili `Static` unutar procedure.

Privatni opseg

Promenljiva sa privatnim opsegom je dostupna svim procedurama modula gde se ona nalazi, ali ne i procedurama iz drugih modula. Pomoću ovih promenljivih, procedure iz modula mogu međusobno komunicirati. Privatne promenljive zadržavaju svoju vrednost sve dok je predmetni projekat otvoren.

Privatne promenljive se deklarišu pomoću ključnih reči `Dim` ili `Private` na početku modula, ispred prve procedure u modulu.

Reč `Private` se ne može se koristiti unutar procedure i jasnije je koristiti nju, a ne `Dim`, za deklarisanje privatnih promenljivih.

Javni opseg

Promenljiva sa javnim dosegom je dostupna svim procedurama u svim modulima u projektu koji je sadrži.

Javne promenljive se deklarišu korišćenjem ključne reči `Public` ispred prve procedure u modulu.

VBA operatori

Operator dodele vrednosti

Operator dodele vrednosti (=) je verovatno operator koji se najviše koristi. Izraz koji se nalazi sa desne strane ovog operatora se izvrši i rezultat se dodeljuje promenljivoj ili objektu sa leve strane operatora. Na primer:

```
Dim X As Double, Y As Double
X = 2.3
Y = (X-1)^2 - 3
```

Uočimo da izraz tipa

```
2.3 = X
```

nije dozvoljen jer se u konstantu 2.3 ne može upisati vrednost.

Aritmetički operatori

VBA podržava rad sa standardnim aritmetičkim operatorima. Tabela 2 sumira ove operatore.

Operator	Operacija	Primer	Rezultat
+	Sabiranje	10+4	14
-	Negacija	-10	-10
-	Oduzimanje	10-5	5
*****	Množenje	10*7	70
/	Deljenje	10/5	2
****	Celobrojno deljenje	11\5	2
^	Stepenovanje	10^3	1000
Mod	Modulo	10 Mod 5	0

Tabela 2. VBA aritmetički operatori.

Operatori poređenja

Ovi se operatori koriste za poređenje dva broja, stringa, promenljive ili rezultata koje vraćaju funkcije. Ako je izraz tačan, njegova vrednost je logičko True (ekvivalentno bilo kojoj numeričkoj nenultoj vrednosti); u suprotnom je False (ekvivalentno nuli). Operatori poređenja kod VBA su dati u tabeli 3.

Operator	Ime	Primer	Rezultat
=	Jednako	7=3	False
<>	Različito	7<>3	True
>	Veće od	7>3	True
>=	Veće od ili jednako	"a">="b"	False
<	Manje od	"a"<"b"	True
<=	Manje od ili jednako	"a"<="b"	True
Like	Like	"Borba" Like "Bo?ba"	True

Tabela 3. VBA operatori poređenja.

O operatoru `Like` će biti više reči u nastavki.

Logički operatori

Logički operatori se koriste za kombinovanje ili modifikovanje True/False izraza. Najčešće korišćeni VBA logički operatori su dati u tabeli 4.

Operator	Ime	Korišćenje	Rezultat
And	I	`Izr1 And Izr2`	True ako su `Izr1` i `Izr2` True; u suprotnom False.
Or	ILI	`Izr1 Or Izr2`	True ako je `Izr1` ili `Izr2` True; u suprotnom False.
Xor	Ekskluzivno ILI	`Izr1 Xor Izr2`	False ako su `Izr1` i `Izr2` True ili `Izr1` i `Izr2` False; u suprotnom True.
Not	Negacija	`Not Izr`	False ako je `Izr` True i obrnuto.

Tabela 4. VBA logički operatori.

Prvenstvo operatora

Tabela 5 daje prvenstvo operatora u VBA, tj. prvenstvo izvršavanja odgovarajućih operacija, pri čemu su operatori sortirani od onih sa najvećim prvenstvom (vrh tabele) do onih sa najmanjim prvenstvom (dno tabele).

Operator	Operacija
^	Stepenovanje
-	Negacija
***** i **/**	Množenje i deljenje
****	Celobrojno deljenje
Mod	Modulo
+ i **-**	Sabiranje i oduzimanje
&	Nadovezivanje
= < > <= >= <> Like Is	Operacije poređenja
And Or Xor Not	Logičke operacije

Tabela 5. Prvenstvo operatora u VBA.

Matematičke funkcije u VBA

U tabeli 6 je dat spisak najčešće korišćenih matematičkih funkcija u VBA.

Funkcija	Vraćena vrednost
Abs(X)	Apsolutna vrednost broja X.
Sin(X)	Sinus broja X.
Cos(X)	Kosinus broja X.
Tan(X)	Tangens broja X.
Atn(X)	Arkus tangens broja X.

Exp(X)	Eksponent broja X, tj. e^X.
Log(X)	Prirodni logaritam broja X.
Sqr(X)	Kvadratni koren broja X.
Sgn(X)	Znak broja. Vraćena vrednost je **1** ako je X>0, **–1** ako je X<0 i **0** ako je X=0.
Fix(X)	Ceo deo broja X. Ako je X<0, `Fix` vraća prvi negativni broj veći od ili jednak X.
Int(X)	Ceo deo broja X. Ako je X<0, `Int` vraća prvi negativni broj manji od ili jednak X.
Rnd()	Slučajan broj između 0 i 1.

Tabela 6. Matematičke funkcije u VBA.

Kontrola toka programa

Jednostavnije VBA procedure se izvršavaju sekvencijalno, tj. naredbe se izvršavaju redom, jedna za drugom. Jedan primer sekvencijalnih procedura su makroi koji se snimaju opcijom **Tools⇒Macro⇒Record New Macro**. Često se, međutim, zahteva da se u proceduri izvrši jedna radnja ako je ispunjen neki uslov i druga u suprotnom. Ili da se određena radnja ponovi određeni broj puta koji zavisi od unosa korisnika. Tada nam čisto sekvencijalno izvršavanje ne može završiti posao.

VBA poseduje veliki broj naredbi za kontrolu toka programa, koje su uglavnom dosta slične onima iz drugih programskih jezika. Tu ubrajamo:

- `If` naredbu
- `Select Case` naredbu
- `GoTo` naredbu
- `For-Next` petlju
- `Do While` petlje
- `Do Until` petlje

Opisaćemo svaku od njih.

If naredba

Ova se naredba koristi kada je potrebno izvršiti jednu ili više naredbi samo ako je zadovoljen određeni uslov. Najjednostavniji oblik ove naredbe je jednolinijski i izgleda

`If` *`Uslov`* `Then` *`Instrukcije`* `Else` *`Instrukcije2`*

`Uslov` može biti bilo koja kombinacija logičkih uslova, operacija poređenja ili numeričkih izraza. Ukoliko je taj uslov tačan, tj. ima numeričku vrednost različitu od nule, izvršiće se `Instrukcije1`, a u suprotnom će se izvršiti `Instrukcije2`. Ovde je `Else` blok opcion, tj. ne mora postojati ukoliko nema potrebe za njim. Daćemo primer korišćenja obe verzije ovog oblika `If` naredbe.

```
If (X Mod 2 = 0) Then MsgBox "Broj X je paran"
If (X Mod 2 = 0) Then MsgBox "Broj X je paran" Else _
```

```
    MsgBox "Broj X je neparan"
```

U prvoj naredbi će biti ispisana poruka `"Broj X je paran"` ako je `X` paran broj, što se proverava korišćenjem operatora `Mod`, i prećiće se na izvršavanje prve sledeće naredbe. U slučaju da `X` nije paran broj ništa se neće desiti, već će se preći na izvršavanje prve sledeće naredbe. U drugoj naredbi će biti ispisana poruka `"Broj X je paran"` ako je `X` paran broj, ili poruka `"Broj X je neparan"` u suprotnom.

Postoji i drugi oblik ove naredbe koji se piše u više linija. Njegova sintaksa je sledeća:

```
If Uslov1 Then
     Instrukcije1
ElseIf Uslov2 Then
     Instrukcije2
...
ElseIf UslovN Then
     InstrukcijeN
Else
     PodrazumevaneInstrukcije
End If
```

I ovde su svi `ElseIf` i `Else` blokovi opcioni, tako da je najjednostavniji oblik ove naredbe

```
If Uslov Then
     Instrukcije
End If
```

Uočimo postojanje reči `End If` koje označavaju kraj naredbe. U obliku `If` naredbe sa više linija, `End If` je obavezno.

U narednom primeru se ispituje vrednost promenljive koja sadrži broj poena koje je student dobio na ispitu i ispisuje ocenu koju je student dobio znajući da važi sledeća veza:

$$\text{Ocena} = \begin{cases} \text{A}, & \text{BrojPeona} \geq 90 \\ \text{B}, & \text{BrojPeona} \geq 80 \\ \text{C}, & \text{BrojPeona} \geq 70 \\ \text{D}, & \text{BrojPeona} \geq 60 \\ \text{E}, & \text{BrojPeona} \geq 50 \\ \text{F}, & \text{BrojPeona} < 50. \end{cases}$$

```
Poruka = "Student je dobio ocenu "
If BrojPoena >= 90 Then
     Poruka = Poruka + "A"
ElseIf BrojPoena >= 80 Then
     Poruka = Poruka + "B"
ElseIf BrojPoena >= 70 Then
     Poruka = Poruka + "C"
ElseIf BrojPoena >= 60 Then
     Poruka = Poruka + "D"
ElseIf BrojPoena >= 50 Then
     Poruka = Poruka + "E"
Else
     Poruka = Poruka + "F"
End If
MsgBox Poruka
```

Uočimo iz ovog primera da se operator sabiranja može koristiti i za nadovezivanje stringova.

Select Case naredba

`Select Case` naredba je korisna za odabir između 3 ili više opcija, mada radi i sa manje i dobra je alternativa `If` naredbi. `Select Case` ima sledeću sintaksu:

```
Select Case TestniIzraz
     Case Izraz1
          Instrukcije1
     ...
     Case IzrazN
          InstrukcijeN
     Case Else
          PodrazumevaneInstrukcije
End Select
```

Ovde `TestniIzraz` predstavlja izraz (najčešće jedna promenljiva) čija se vrednost proverava i u zavisnosti od toga koji od izraza `Izraz1`, ...,`IzrazN` odgovara toj vrednosti izvršavaju se odgovarajuće instrukcije. Prostije, `TestniIzraz` određuje koje će se instrukcije izvršiti. Rešimo prethodni primer sa ispisom ocene studenta pomoću `Select Case` naredbe.

```
Poruka = "Student je dobio ocenu "
Select Case BrojPoena
     Case Is >= 90
          Poruka = Poruka + "A"
     Case Is >= 80
          Poruka = Poruka + "B"
     Case Is >= 70
          Poruka = Poruka + "C"
     Case Is >= 60
          Poruka = Poruka + "D"
     Case Is >= 50
          Poruka = Poruka + "E"
     Case Else
          Poruka = Poruka + "F"
End Select
MsgBox Poruka
```

Primetimo upotrebu ključne reči `Is` da se specificira opseg vrednosti. Ako pretpostavimo da broj poena može biti samo celi broj, prethodna naredba bi se mogla napisati i na sledeći način:

```
Select Case BrojPoena
     Case Is >= 90
          Poruka = Poruka + "A"
     Case 80 To 89
          Poruka = Poruka + "B"
     Case 70 To 79
          Poruka = Poruka + "C"
     Case 60 To 69
          Poruka = Poruka + "D"
     Case 50 To 59
          Poruka = Poruka + "E"
     Case Else
          Poruka = Poruka + "F"
End Select
```

U ovom obliku smo koristili ključnu reč `To` za specificiranje opsega. Ako je broj poena celi broj, onda se

```
Case 80 To 89
```

može zapisati i kao

```
Case 80, 81, 82, 83, 84, 85, 86, 87, 88, 89
```

gde navodimo sve moguće vrednosti koje može uzeti promenljiva `BrojPoena` i razdvajamo iz zarezima. Ovaj primer baš i ne opravdava upotrebu ovakvog načina zadavanja vrednosti, ali je pogodan za ilustraciju.

Ključna reč `Is` može i da kombinuje izraze, tako da se obuhvataju različiti opsezi koji nisu susedni. Izrazi se odvajaju zarezima. Na primer, slučajem

```
Case Is > 90, 33, 50 To 60
```

bi se obuhvatili brojevi veći od 90, broj 33 i brojevi između 50 i 60.

GoTo naredba

Najjednostavniji način da se promeni sekvencijalni redosled izvršavanja koda je pomoću `GoTo` naredbe. Ova naredba bezuslovno prebacuje izvršenje programa (još se naziva i *grananje*) na određenu instrukciju u kodu procedure koja mora započeti labelom. Dakle, `GoTo` naredba ne može da grana van procedure. Labela je oznaka instrukcije i služi da jednoznačno odredi instrukciju na koju se ide naredbom `GoTo`. Labela može biti tekstualni string praćen dvotačkom (:) ili broj bez dvotačke. U VBA proceduri može postojati proizvoljan broj labela.

Naredbu `GoTo` bi trebalo koristiti što štedljivije ili čak ne koristiti je uopšte. Razlog je u teškom prepravljanju većih procedura koje sadrže ovu naredbu. Na primer, `GoTo` se može koristiti za iskakanje iz petlji. Ova naredba narušava strukturnu organizaciju programa. Jedina upotreba koja može opravdati potrebu za ovom naredbom je *upravljanje greškama* (error handling), o čemu će više reči biti kasnije.

U narednoj funkciji dajemo primer korišćenja `GoTo` naredbe.

```
Function RadSaPrirodnim(X As Integer) As Integer
Dim I As Integer
RadSaPrirodnim = -1
If X < 1 Then GoTo Greska
RadSaPrirodnim = 0
For I = 1 To X
     RadSaPrirodnim = RadSaPrirodnim + I
Next I
Greska: Exit Function
End Function
```

Funkcija `RadSaPrirodnim` ima za ulazni argument celi broj `X` i treba da izvrši određenu operaciju samo ako je `X` prirodan broj. Ukoliko `X` nije prirodan broj, funkcija vraća `-1`. Ovo smo uradili pomoću naredbe `GoTo` koja grana na labelu `Greska`, a ta labela odgovara instrukciji forsiranog izlaska iz funkcije. Naravno, sve ovo se moglo uraditi i bez naredbe `GoTo`.

For-Next petlja

Kad-tad ćemo doći u situaciju da određenu operaciju izvršimo određeni broj puta, koji može biti unapred poznat ili zavisiti od vrednosti neke promenljive u kodu. Tada moramo koristiti cikluse ili petlje. Najjednostavnija petlja je `For-Next` čija je sintaksa

```
For Brojac = Start To End Step Korak
```

```
    Instrukcije
Next Brojac
```

`Brojac` je VBA promenljiva po kojoj se petlja izvršava uzimajući na početku vrednost `Start`, a završavajući sa vrednošću `End`. U ovoj petlji je opcion podatak `Korak`, koji predstavlja korak promene vrednosti promenljive `Brojac`. Opcion je i zapis `Next Brojac` koji se može skratiti sa `Next`. Dajemo primer dve `For-Next` petlje.

```
S = 0
For I = 1 To 99
    S = S + I
Next I

S = 0
For I = 1 To 99 Step 2
    S = S + I
Next
```

Prvom petljom vršimo sabiranje svih prirodnih brojeva manjih od 100, a drugom svih prirodnih neparnih brojeva manjih od 100. U oba slučaja je početna vrednost promenljive `I` jednaka 1, a krajnja 99, pri čemu u prvoj petlji `I` uzima vrednosti 1,2,3,...,99, a u drugoj 1,3,5,...,99.

Forsirani izlazak iz `For-Next` petlje se vrši pomoću naredbe `Exit For` koja se može smestiti bilo gde unutar petlje. Kada se prilikom izvršavanja naiđe na ovu naredbu, iz petlje se automatski izlazi i prelazi na izvršenje prve naredbe nakon petlje.

Do While petlje

`Do While` predstavlja drugi tip petlji u VBA. Za razliku od brojačke `For-Next` petlje, ova se petlja izvršava sve dok je ispunjen određeni uslov. `Do While` petlja može uzeti jedan od sledeća dva oblika:

```
Do While Uslov
    Instrukcije
Loop
```

ili

```
Do
    Instrukcije
Loop While Uslov
```

`While` uslov se može staviti na početak ili na kraj petlje. Razlika između ova dva oblika `Do While` petlje je u trenutku kad se proverava tačnost uslova. U prvom obliku se uslov proverava odmah na početku petlje i može se desiti da se u petlju uopšte ne uđe. U drugom obliku se uslov proverava na kraju petlje i petlja se izvršava minimum jedanput.

Forsirani izlazak iz `Do While` petlje se vrši naredbom `Exit Do`.

Uradimo primer sa sabiranjem brojeva manjih od 100 koristeći oba oblika `Do While` petlje.

```
S = 0: I = 1
Do While I < 100
    S = S + I
    I = I + 1
Loop

S = 0: I = 0
```

```
Do
     I = I + 1
     S = S + I
Loop While I < 99
```

Moramo voditi računa da se promenljiva od koje zavisi uslov izvršenja petlje menja tako da jednog trenutka taj uslov ne bude zadovoljen. Ukoliko to nije ispunjeno, dobićemo beskonačnu petlju. U našem slučaju, promenljiva `I` raste dok ne postane 100 u slučaju prve petlje i 99 u slučaju druge petlje. Dakle, pri izlasku iz petlje, promenljiva od koje zavisi uslov petlje ima prvu vrednost za koju taj uslov nije zadovoljen.

Uočimo korišćenje dvotačke radi grupisanja više VBA instrukcija u jednu liniju.

Napomenimo i to da je deo `While` *`Uslov`* u sintaksi petlje opcion, tj. ne mora se navoditi uslov. Tada se u petlji mora naći `Exit Do` radi izlaska iz petlje. Prethodni primer urađen koristeći ovaj oblik `Do While` petlje je dat ispod.

```
S = 0: I = 1
Do
     S = S + I
     I = I + 1
     If I = 100 Then Exit Do
Loop
```

Do Until petlje

Ovaj tip petlje je vrlo sličan `Do While` petlji, sa tom razlikom da se `Do Until` petlja izvršava sve dok se ne ispuni određeni uslov. Sintaksa ove petlje takođe dozvoljava proveru uslova petlje na njenom početku i na kraju.

```
Do Until Uslov
     Instrukcije
Loop
```

ili

```
Do
     Instrukcije
Loop Until Uslov
```

Petlja se u oba oblika izvršava sve dok uslov nije ispunjen. Kada se uslov ispuni, izlazi se iz petlje i nastavlja sa prvom sledećom instrukcijom. Kod oblika sa proverom uslova na kraju, petlja se izvršava bar jedanput.

Uradimo primer sa sabiranjem brojeva manjih od 100 koristeći oba oblika `Do Until` petlje.

```
S = 0: I = 1
Do Until I = 100
     S = S + I
     I = I + 1
Loop

S = 0: I = 0
Do
     I = I + 1
     S = S + I
Loop Until I = 99
```

Forsirani izlazak iz `Do Until` petlje se takođe vrši naredbom `Exit Do`.

P R O G R A M I R A N J E
K R O Z
A P L I K A C I J E

Doc. dr Đukanović Slobodan

D R U G I T E R M I N

Nizovi

Niz predstavlja grupu elemenata koji imaju isti tip i ime, pri čemu se određenom elementu niza pristupa koristeći ime niza i indeks elementa. Indeks elementa predstavlja redni broj elementa u nizu i navodi se u malim zagradama. Kao kod C-a, indeks prvog elementa niza je **0**. Tako bi elementima niza `X`, dužine 10, pristupali na sledeći način:

```
X(0), X(1), ..., X(9)
```

Deklaracija nizova

Nizovi se deklarišu koristeći ključne reči `Dim` ili `Public`, kao i ostale VBA promenljive. Pri deklaraciji niza se može promeniti indeks prvog elementa i to na sledeći način:

```
Dim X(1 To 10) As Integer
```

Prethodnom deklaracijom smo deklarisali niz od 10 celih brojeva, pri čemu je indeks prvog elementa 1, a poslednjeg 10. U zagradi se, dakle, navode indeksi prvog i poslednjeg elementa, povezanih rečju `To`. Ako bi u deklaraciji naveli samo gornji indeks 10, tj.

```
Dim X(10) As Integer
```

deklarisali bismo niz od 11 elemenata, jer je podrazumevano indeks prvog elementa 0.

Navođenjem

```
Option Base 1
```

na početku modula, indeks prvog elementa niza će biti 1 u svim procedurama tog modula.

Indeks prvog i poslednjeg elementa niza možemo saznati pomoću funkcija `LBound(`*`ImeNiza`*`)`, koja vraća indeks prvog elementa, i `UBound(`*`ImeNiza`*`)`, koja vraća indeks poslednjeg elementa niza.

Višedimenzioni nizovi

Pored prethodno opisanih jednodimenzionih nizova, u VBA možemo raditi i sa višedimenzionim nizovima. VBA podržava do 60 dimenzija nizova, ali se retko kad koristi više od trodimenzionih nizova. Najčešći primer višedimenzionih nizova su dvodimenzioni nizovi ili matrice.

Deklaracija višedimenzionih nizova je ista kao i deklaracija jednodimenzionih nizova, pri čemu se, za svaku dimenziju, indeks prvog i poslednjeg elementa navode u zagradama. Na primer, sa

```
Dim M(1 To 10, 1 To 10) as Integer
```

deklarišemo dvodimenzioni niz `M` od 100 elemenata, raspoređenih u 10 vrsta i 10 kolona. Prilikom pristupanja elementima niza M, moramo navesti redni broj vrste i kolone u kojoj se nalazi element. Na primer, izrazom

```
M(3,4) = 106
```

se vrši promena vrednosti elementa niza `M` u preseku treće vrste i četvrte kolone.

Dinamički nizovi

Mana prethodno opisanih nizova je ta da se svaki put mora izvršiti predimenzionisanje nizova prilikom deklarisanja, kako bi se smestili podaci za slučaj najzahtevnije moguće obrade. Često se deklariše nekoliko puta više memorijskog prostora od stvarno potrebnog. Neiskorišćena zauzeta memorija je nedostupna za smeštaj drugih promenljivih. Ovaj se problem može rešiti korišćenjem dinamičkih nizova.

Dinamički nizovi nemaju unapred definisan broj elemenata. Dakle, pri deklaraciji dinamičkog niza, ne navodi se indeks poslednjeg elementa u zagradi, tj.

```
Dim M() As Integer
```

Međutim, pre prve upotrebe niza u programu, mora se definisati broj elemenata niza, što se radi naredbom `ReDim` na sledeći način:

```
ReDim M(10)
```

Pomoću naredbe `ReDim` se može vršiti i promena broja elemenata niza (redimenzionisanje). Redimenzionisanje niza se u proceduri može vršiti proizvoljan broj puta. Treba voditi račun da se prilikom redimenzionisanja niza gubi vrednost elemenata. Za očuvanje vrednosti elemenata niza potrebno je koristiti naredbu `Preserve`, kao na primer

```
ReDim Preserve M(10)
```

Rad sa stringovima

Pored numeričkih podataka, Excel vrlo često radi sa stringovima i poseduje veliki broj funkcija za njihovu obradu. VBA takođe poseduje veliki broj korisnih funkcija za obradu stringova. Već smo videli da VBA može da radi stringovima fiksne i promenljive dužine. Ukoliko se u deklaraciji stringa navede i njegova dužina, dobijamo string fiksne dužine. U suprotnom, ako se ne navede dužina, dobijamo string promenljive dužine. Navodimo primer deklaracije stringa fiksne dužine `S1`, koji ima 40 karaktera, i stringa `S2` promenljive dužine.

```
Dim S1 as String * 40
Dim S2 as String
```

U tabeli 7 je dat spisak korisnih funkcija za manipulaciju stringovima u VBA.

Funkcija	Opis
`Asc(`***`string`***`)`	Vraća ANSI kod prvog karaktera u stringu ***`string`***. *Primer*: `Asc("A12")` vraća broj `65`.
`AscW(`***`string`***`)`	Vraća Unicode kod prvog karaktera u stringu ***`string`***. *Primer*: `AscW("Žarko")` vraća broj `381`.
`Chr(`***`kodkar`***`)`	Vraća karakter koji odgovara ANSI kodu ***`kodkar`***. *Primer*: `Chr(65)` vraća karakter `"A"`.
`ChrW(`***`kodkar`***`)`	Vraća karakter koji odgovara Unicode kodu ***`kodkar`***. *Primer*: `ChrW(382)` vraća karakter `"ž"`.
`Len(`***`string`***`)`	Vraća broj karaktera u stringu ***`string`***. *Primer*: `Len("žabac&komarac")` vraća broj `13`.
`Str(`***`broj`***`)`	Konvertuje broj ***`broj`*** u string.

	Primer: `Str(412.41)` vraća string `"412.41"`.

`Val(`***`string`***`)`	Vraća broj sadržan u stringu ***`string`***. Čitanje broja započinje s leva i završava se kad se naiđe na prvi nenumerički karakter. Funkcija `Val` prepoznaje tačku kao decimalni separator, ali ne i zarez, ignoriše tabove i blanko znake.
	Primer: `Val("301*44")` vraća broj `301.`
	`Val("301.1*44")` vraća broj `301.1`
	`Val("301..1*44")` vraća broj `301`

`InStr(`***`string1,string2`***`)`	Vraća poziciju prve pojave stringa ***`string1`*** u stringu ***`string2`***. *Primer*: `InStr("Blagdan","dan")` vraća broj `5.`
`LCase(`***`string`***`)`	Vraća string kod koga su sva velika slova konvertovana u mala, a ostali karakteri su neizmenjeni. *Primer*: `LCase("#A1G2")` vraća `"#a1g2"`
`UCase(`***`string`***`)`	Vraća string kod koga su sva mala slova konvertovana u velika, a ostali karakteri su neizmenjeni. *Primer*: `UCase("*b1c2")` vraća `"*B1C2"`
`Left(`***`string,duzina`***`)`	Vraća string koji se sastoji od ***`duzina`*** karaktera stringa ***`string`*** gledano sa leva. *Primer*: `Left("Dobar dan",4)` vraća `"Doba"`
`Right(`***`string,duzina`***`)`	Vraća string koji se sastoji od ***`duzina`*** karaktera stringa ***`string`*** gledano sa desna. *Primer*: `Right("Dobar dan",4)` vraća `" dan"`
`Mid(`***`string,start,duzina`***`)`	Vraća string koji se sastoji od ***`duzina`*** karaktera stringa ***`string`*** počev od karaktera čija je pozicija ***`start`***. ***`duzina`*** je opcion argument i ako se ne navede ide se do kraja stringa. *Primer*: `Mid("Program",2,4)` vraća `"rogr"` `Mid("Program",2)` vraća `"rogram"`
`LTrim(`***`string`***`)`	Eliminiše spejsove kojima počinje string ***`string`***. *Primer*: `LTrim("  Makro")` vraća `"Makro"`
`RTrim(`***`string`***`)`	Eliminiše spejsove kojima se završava string ***`string`***. *Primer*: `LTrim("Makro  ")` vraća `"Makro"`
`Trim(`***`string`***`)`	Eliminiše spejsove kojima počinje i završava se string ***`string`***. *Primer*: `Trim("  Makro  ")` vraća `"Makro"`
`StrComp(`***`string1,string2,nacin`***`)`	Poredi stringove ***`string1`*** i ***`string2`*** i vraća **0** ako su jednaki, **-1** ako je ***`string1`*** manji od ***`string2`*** i **1** ako je ***`string2`*** manji od ***`string1`*** . ***`nacin`*** predstavlja način poređenja stringova i može biti binaran (`nacin=0` ⇒ pravi se razlika između velikih i malih slova) i tekstualan (`nacin=1` ⇒ ne pravi se razlika između velikih i malih slova). Ako se ***`nacin`***

	ne navede, podrazumevano je binarno poređenje. *Primer*: `StrComp("VBA","vba",0)` vraća `-1` `StrComp("VBA","vba",1)` vraća `0`

Tabela 7. Funkcije za rad sa stringovima u VBA.

Za sve gore pomenute funkcije koje vraćaju string, vraćeni string je tipa `Variant`. Ukoliko želimo da string bude tipa `String`, potrebno je na ime funkcije nadovezati karakter $. Tako će, na primer, ekvivalent funkcije `Lcase` koji vraća `String` tip biti `Lcase$`.

Za očekivati je da se, pošto string u suštini predstavlja niz karaktera, pojedinim karakterima stringa može pristupiti navođenjem pozicije karaktera u malim zagradama, kao što se radi kod numeričkih nizova. Međutim, kod VBA to nije slučaj. Pristupanje karakterima stringa, kao i njihova izmena, se vrši funkcijom `Mid`, kao što je prikazano u prethodnoj tabeli. Na primer, instrukcijom

```
Mid(S,1,3) = "123"
```

bi se zamenila prva tri karaktera stringa S sa stringom `"123"`.

Osim sa funkcijom `StrComp`, stringovi se mogu porediti i operatorom ispitivanja jednakosti =. Ovaj operator vrši binarno poređenje. To je podrazumevani način poređenja stringova sa ovim operatorom, kao što je kod funkcije `StrComp`. Podrazumevani način poređenja stringova se može promeniti ako se na vrhu modula navede

```
Option Compare Text
```

čime podrazumevani način postaje tekstualni. Brisanjem ovog iskaza ili pomoću

```
Option Compare Binary
```

podrazumevani način opet postaje binarni.

VBA dozvoljava vrlo jednostavno nadovezivanje (konkatenaciju) stringova korišćenjem operatora + ili `&`. Tako će izraz

```
S = "Black" + "White"
```

rezultovati u stringu `"BlackWhite"`. Treba biti oprezan kada se radi sa `Variant` promenljivim jer `Variant` podaci menjaju tip u zavisnosti od toga šta radimo sa njima. Uzmimo, na primer, string `S` u koji je upisan string `"105"` i posmatrajmo sledeće dve VBA instrukcije:

```
S = 2 * S
S = S + S
```

Nakon prve instrukcije bi `S` imalo numeričku vrednost 210, a nakon druge instrukcije `S` bi bio string `"105105"`. Pri radu sa stringovima, operator + uvek vrši nadovezivanje. Međutim, ako radite sa `Variant` promenljivom u koju je prvobitno upisan string, a nakon toga se promeni tip podatka, operator + više neće vršiti nadovezivanje.

Rad sa slovima š, đ, č, ć i ž

VBE ne dozvoljava unos naših slova š, đ, č, ć i ž, malih i velikih. Postavlja se pitanje na koji način raditi sa tim slovima. Odgovor je - pomoću funkcije `ChrW`. Jedino što treba da znamo jesu Unicode kodovi naših slova i oni su dati u tabeli ispod.

Slovo	Š	Đ	Č	Ć	Ž	š	đ	č	ć	ž
Kod	**352**	**272**	**268**	**262**	**381**	**353**	**273**	**269**	**263**	**382**

Tabela 8. Unicode kodovi naših slova.

Na primer, string `"Šećer"` bi se definisao na sledeći način:

```
S = ChrW(352) + "e" + ChrW(263) + "er"
```

Ipak, na ovaj način ne možemo prikazati ova slova u korisničkim formama. Da bi to uradili, sistemski moramo podesiti opciju **Language for non-Unicode programs** na Serbian (Latin). Ova opcija se nalazi na tabu **Advanced** prozora **Regional and Language Options** kod Windows XP i na tabu **Administrative** prozora **Regional and Language Options** kod Windows Vista. Sa ovom opcijom mi čak omogućavamo unos naših slova u prozor VBE-a, što znači da prva rečenica ovog poglavlja nije ispravna ☺.

Like operator

Ovaj operator može biti vrlo koristan jer pruža vrlo moćno i elegantno poređenje stringova. Primenjuje se u sledećem obliku:

```
string Like obrazac
```

`obrazac` predstavlja string koji može sadržati tri džoker znaka (wildcards):

- **?** - menja bilo koji karakter,
- ***** - menja proizvoljan broj karaktera, i
- # - menja bilo koju cifru.

Par jednostavnih primera korišćenja operatora `Like` je dato ispod.

```
"Koren" Like "Kor?n"        vraća True
"Koren" Like "Ko?n"         vraća False
"Koren" Like "K*"           vraća True
"2009" Like "####"          vraća True
"Koren2009" Like "K*##09"   vraća True
```

Radi povećanja fleksibilnosti, pored ovih džokera, operator `Like` dozvoljava i pretragu samo određenih karaktera, koji se navode u srednjim zagradama `[]`. Tako će, na primer,

```
"Koren" Like "Kor[ae]n"
```

našem slučaju vratiti True, jer će, na četvrtoj poziciji, VBA samo tražiti karaktere `a` i `e`. U ovim se zagradama može navesti i opseg karaktera, koristeći karakter `-`. Tako će, na primer, `[a-z]` zamenjivati bilo koje malo slovo, `[A-Z]` zamenjivati bilo koje veliko slovo, `[A-D]` zamenjivati velika slova A, B, C i D itd. Važno je znati da se prilikom korišćenja opsega karaktera, opseg mora navesti u rastućem redosledu. Na primer, zapis `[Z-A]` bi doveo do greške u izvršavanju.

Ako bi se ispred opsega karaktera stavio znak uzvika `!`, onda bi se proveravalo da li karakter ne pripada datom opsegu. Na primer, sa `[!a-z]` bi ispitivali da li dati karakter nije malo slovo. Dajemo nekoliko primera.

```
"H" Like "[A-Z]"            vraća True
"H" Like "[!A-Z]"           vraća False
```

`"H5N1" Like "H#[!C-E]#"` vraća True

Ukoliko je aktivna opcija `Option Compare Text`, poređenje sa `"[A-Z]"` i `"[a-z]"` će dati iste rezultate.

Sada ćemo, kao primer, dati funkcijsku proceduru koja radi potpuno istu stvar kao i funkcija `Lcase$`.

```
Function UMala(S As String) As String
Dim I As Integer
UMala = S
For I = 1 To Len(S)
    If Mid(UMala, I, 1) Like "[A-Z]" Then
        Mid(UMala, I, 1) = Chr(Asc(Mid(UMala, I, 1)) + _
        Asc("a") - Asc("A"))
    End If
Next I
End Function
```

Konverzija velikih slova u mala se vrši instrukcijom

```
Mid(UMala, I, 1) = Chr(Asc(Mid(UMala, I, 1)) + Asc("a") - Asc("A"))
```

Na kodnu poziciju tekućeg karaktera (`Asc(Mid(UMala,I,1)`) se dodaje razlika kodnih pozicija malih i odgovarajućih velikih slova (`Asc("a")-Asc("A")`). Funkcija `Chr` daje karakter koji odgovara dobijenoj kodnoj poziciji.

Funkcija Format

`Format` funkcija predstavlja vrlo pogodan alat pri formatiranju numeričkih vrednosti, vremena i datuma. Mi ćemo koristiti njen skraćeni oblik:

`Format(`*`izraz`*`,`*`format`*`)`

gde *`izraz`* predstavlja izraz koji treba formatirati u skladu sa formatom *`format`*. Funkcija vraća string koji predstavlja formatirani izraz. Ovde ćemo raditi samo sa numeričkim izrazima, vremenima i datumima.

Pri formatiranju izraza, možemo koristiti *predefinisane* i *korisničke* formate.

Predefinisani numerički formati

VBA podrazumevano prikazuje brojeve na najjednostavniji način, tj. bez separatora hiljada i dodatnih simbola. Na primer, broj 12345 će biti prikazan kao `12345`. Ovaj se format može promeniti korišćenjem nekog od predefinisanih numeričkih formata iz tabele 9.

Format	Opis
`General Number`	Podrazumevani format *Primer*: `Format(12345,"General Number")` vraća `"12345"`.
`Currency`	Prikazuje se separator hiljada, oznaka valute i dva decimalna mesta. Oznaka valute zavisi od sistemskog podešavanja. *Primer*: `Format(12345,"Currency")` vraća `"£12,345.00"`.
`Fixed`	Prikazuje se najmanje jedna cifra levo od decimalnog zareza i najmanje dve cifre desno od decimalnog zareza. *Primer*: `Format(1222.345,"Fixed")` vraća `"1222.35"`.

`Standard`	Prikazuje se separator hiljada, najmanje jednom cifrom levo od decimalnog zareza i najmanje dve cifre desno od decimalnog zareza. *Primer*: `Format(1222.345,"Standard")` vraća `"1,222.35"`.
`Percent`	Prikazuje se broj pomnožen sa 100,dve cifre desno od decimalnog zareza, i znak za procenat (`%`) desno od decimalnog zareza. *Primer*: `Format(0.12345,"Percent")` vraća `"12.35%"`.
`Scientific`	Standardni scientific zapis. Cifra najveće težine je prikazana levo od decimalnog zareza, i prikazuje se 2 do 30 decimalnih mesta praćeno sa `"E"` i eksponentom. *Primer*: `Format(12300,"Scientific")` vraća `"1.23E+04"`.
`Yes/No`	Vraća `No` ako je broj jednak `0` i `Yes` za sve ostale vrednosti. *Primer*: `Format(123,"Yes/No")` vraća `"Yes"`.
`True/False`	Vraća `False` ako je broj jednak `0` i `True` za sve ostale vrednosti. *Primer*: `Format(123,"True/False")` vraća `"True"`.
`On/Off`	Vraća `Off` ako je broj jednak `0` i `On` za sve ostale vrednosti. *Primer*: `Format(0,"On/Off")` vraća `"Off"`.

Tabela 9. Predefinisani numerički formati u VBA.

Kao što možemo videti kroz primere, format se navodi kao string.

Predefinisani formati za datum i vreme

Podrazumevani datum i vreme zavise od sistemskih podešavanja (Control Panel/Regional Options). Tabela 10. prikazuje kompletnu listu predefinisanih VBA formata za datum i vreme.

Format	Opis
`General Date`	Podrazumevani format.
`Long Date`	Prikazuje se datum u skladu sa sistemski definisanim long date formatom (na primer, `19 September 2009`).
`Medium Date`	Prikazuje se datum u skladu sa sistemski definisanim medium date formatom (na primer, `19-Sep-09`).
`Short Date`	Prikazuje se datum u skladu sa sistemski definisanim short date formatom (na primer, `19/09/2009`).
`Long Time`	Prikazuje se vreme u skladu sa sistemski definisanim long time formatom (na primer, `12:32:45`).
`Medium Time`	Prikazuje se vreme u skladu sa sistemski definisanim medium time formatom (na primer, `12:32 PM`).
`Short Time`	Prikazuje se vreme u skladu sa sistemski definisanim short time formatom (na primer, `12:32`).

Tabela 10. Predefinisani formati za datum i vreme u VBA.

Korisnički numerički formati

Predefinisani formati u VBA pružaju određenu slobodu u formatiranju izraza. Ipak, postoje i određena ograničenja. Na primer, kod `Currency` formata, oznaka valute (£, €, Din) se ne može promeniti pomoću predefinisanih formata, već se sistemski mora menjati. Ovakva ograničenja se mogu prevazići kreiranjem sopstvenih numeričkih formata.

VBA numerički format se sastoji od četiri dela, razdvojenih tačka-zarezom:

```
pozitivni format; negativni format; format nule; null format
```

Prvi deo definiše prikaz pozitivnih brojeva, drugi negativnih, treći definiše prikaz nule i četvrti definiše prikaz null vrednosti. Postoji i format sa manje od četiri dela i tad se brojevi kontrolišu na sledeći način:

Tri dela: `pozitivni format; negativni format; format nule`
Dva dela: `pozitivni i format nule; negativni format`
Jedan deo: `pozitivni, negativni i format nule`

Simbol	Opis
#	Jedna cifra. Ništa se ne prikazuje ako nema unosa.
0	Jedna cifra. Prikazuje nulu ako nema unosa.
.	Pozicija decimalne tačke.
,	Pozicija separatora hiljada. Označava poziciju samo prve hiljade.
%	Množi broj sa 100 (samo za prikaz) i dodaje karakter %.
E+ e+ E- e-	Prikaz brojeva u scientific formatu. E- and e- smeštaju znak - u eksponent; E+ and e+ smeštaju znak + u eksponent.
$ ( ) - + <space>	Prikazuje date karaktere.
\	Umetanje karaktera koji dolazi posle \.
"text"	Umetanje teksta pod znacima navoda.

Tabela 11. Specijalni simboli koji se koriste pri definisanju numeričkih formata.

U tabeli 11. su dati specijalni simboli koji se koriste pri definisanju ovih delova formata.

Navedimo nekoliko primera.

`Format(6353.4, "##,##0.00")` vraća `"6,353.40"`
`Format(334.9, "###0.00 kg")` vraća `"334.90 kg"`
`Format(4, "0.00%")` vraća `"400.00%"`
`Format(6353.4, "00.00000E+")` vraća `"63.53400E+2"`
`Format(6353.4, "#.#E+")` vraća `"6.4E+3"`
`Format(1234567, "00-00-000")` vraća `"12-34-567"`

Korisnički formati za datum i vreme

U tabeli 12. su dati simboli za formatiranje datuma i vremena.

Simbol	Opis
Formati za datum	
c	Prikazuje datum kao ddddd i vreme kao ttttt.
d	Broj dana bez prateće nule (1 do 31).
dd	Broj dana sa pratećom nulom (01 do 31).
ddd	Skraćenica dana od 3 slova (na primer, Fri).
dddd	Puno ime dana (na primer, Friday).
ddddd	Prikazuje kompletan datum koristeći sistemski short date format.

dddddd	Prikazuje kompletan datum koristeći sistemski long date format.
m	Broj meseca bez prateće nule (1 do 12).
mm	Broj meseca sa pratećom nulom (01 do 12).
mmm	Skraćenica meseca od 3 slova (na primer, Jul).
mmmm	Puno ime meseca (na primer, Jul).
q	Kvartal godine (1 do 4).
w	Dan u nedelji, kao broj (od 1, za Sunday, do 7, za Saturday).
ww	Nedelja u godini (1 do 54).
y	Dan u godini (1 do 366).
yy	Godina kao dvocifreni broj (00 do 99).
yyyy	Pun naziv godine (1900 do 2078).
Formati za vreme	
ttttt	Prikazuje kompletno vreme koristeći podrazumevani sistemski format.
h	Sat bez prateće nule (0 do 24).
hh	Sat sa pratećom nulom (00 do 24).
m	Minut bez prateće nule (0 do 59).
mm	Minut sa pratećom nulom (00 do 59).
n	Minut bez prateće nule (0 do 59).
nn	Minut sa pratećom nulom (00 do 59).
s	Sekunda bez prateće nule (0 do 59).
ss	Sekunda sa pratećom nulom (00 do 59).
AM/PM, am/pm	Prikaz vremena koristeći 12-časovni sat.
/ : . -	Separatori datuma ili vremena.

Tabela 12. Simboli za formatiranje datuma i vremena.

Dajemo par primera da ilustrujemo korišćenje simbola datih u prethodnoj tabeli.

`Format(#17:02:04#, "h:m:s")` vraća `"17:2:4"`
`Format(#17:02:04#, "hh:mm:ss")` vraća `"17:02:04"`
`Format(#17:02:04#, "hh:mm:ss AM/PM")` vraća `"05:02:04 PM"`
`Format(#21/9/2009#, "dddd, mmm d yyyy")` vraća `"Monday, Sep 21 2009"`

P R O G R A M I R A N J E
K R O Z
A P L I K A C I J E

Doc. dr Đukanović Slobodan

TREĆI TERMIN

Objektni model Excel-a

Dosadašnji deo materijala je bio vezan za čisto programiranje u VBA i kao takav bi se mogao posmatrati nezavisno od same aplikacije. Drugim rečima, možemo reći da je dosadašnji deo univerzalan i dele ga sve aplikacije koje koriste VBA. Došlo je vreme da odemo korak dalje i da zagazimo u vode objektnog modela MS Excel-a.

Ključ u programiranju date aplikacije korišćenjem VBA leži u razumevanju objektnog modela te aplikacije. Sam objektni model Excel-a je hijerarhijski uređen. Na vrhu modela se nalazi objekt `Application`, koji predstavlja sam program, tj. Excel. Objekt `Application` sadrži druge objekte. U tabeli 13. je dato nekoliko objekata koje sadrži objekt `Application`, uz kratko objašnjenje svakog od njih.

Objekt	Kolekcija	Opis
`Workbook`	`Workbooks`	Otvorena radna sveska. Objekt `Workbooks` predstavlja kolekciju svih otvorenih radnih sveski.
`Window`	`Windows`	Otvoren prozor. Objekt `Windows` predstavlja kolekciju svih otvorenih prozora.
`Name`	`Names`	Definisano ime opsega. Objekt `Names` predstavlja kolekciju svih definisanih imena u svim otvorenim radnim sveskama.
`Dialog`	`Dialogs`	Ugrađen Excel-ov prozor za dijalog. Objekt `Dialogs` predstavlja kolekciju svih ugrađenih prozora za dijalog.

Tabela 13. Neki objekti koje sadrži objekt Application.

Objekti mogu imati svoje objekte. Neki od objekata koje sadrži objekt `Workbook` su dati u tabeli 14.

Objekt	Kolekcija	Opis
`Worksheet`	`Worksheets`	Otvoren radni list. Objekt `Worksheets` predstavlja kolekciju radnih listova date radne sveske.
`Chart`	`Charts`	Otvorena mapa. Objekt `Charts` predstavlja kolekciju svih mapa date radne sveske.
`Name`	`Names`	Definisano ime opsega. Objekt `Names` predstavlja kolekciju svih definisanih imena u datoj radnoj svesci.

Tabela 14. Neki objekti koje sadrži objekt Workbook.

Možemo ići i korak niže u hijerarhiji. Objekat `Worksheet` sadrži dosta drugih objekata, od kojih su neki dati u tabeli 15.

Objekt	Kolekcija	Opis
`ChartObject`	`ChartObjects`	Postojeće mape na tekućem radnom listu. Objekt `ChartObjects` predstavlja kolekciju svih mapa na tekućem radnom listu.
`Range`	–	Predstavlja ćeliju, vrstu, kolonu ili proizvoljnu dozvoljenu selekciju ćelija (jedan ili više blokova neprekinutih ćelija).
`Comment`	`Comments`	Postojeći komentari na tekućem radnom listu. Objekt `Comments` predstavlja kolekciju svih komentara na tekućem radnom listu.

`Shape`	`Shapes`	Postojeći oblici (npr. AutoShape, korisnički crtež, OLE objekt ili slika) na tekućem radnom listu. Objekt `Shapes` predstavlja kolekciju svih objekata na tekućem radnom listu.

Tabela 15. Neki objekti koje sadrži objekt Worksheet.

Drugi ključni koncept u VBA programiranju su *kolekcije*. Kolekcija predstavlja grupu objekata iste klase. Kolekcija za sebe predstavlja objekat. U prethodne tri tabele se vidi veza kolekcija sa odgovarajućim objektima. Možemo raditi sa čitavom kolekcijom ili sa pojedinim objektima te kolekcije.

Referenciranje objekta u kolekciji se vrši navođenjem imena objekta ili njegovog rednog broja u malim zagradama nakon imena kolekcije. Tako se, na primer, prvi radni list, koji se zove Sheet1, može referencirati na bilo koji od sledeća dva načina:

```
Worksheets("Sheet1")
Worksheets(1)
```

Drugi radni list se referencira sa `Worksheets(2)`.

Pored kolekcije `Worksheets` postoji i kolekcija `Sheets`, koja se sastoji od svih listova u radnoj svesci, uključujući radne listove i mapne listove (chart sheets).

Zgodno je mesto da se naglasi da se mnogim objektima u VBA ne može jednostavno pristupiti koristeći samo njihovo ime. Takvim objektima se mora pristupiti kao elementima odgovarajuće kolekcije. Na primer, radnoj svesci VBA.xls se ne može pristupiti samo navođenjem imena radne sveske, tj. VBA.xls. Umesto toga, svesci se može pristupiti na sledeći način:

```
Workbooks("VBA.xls")
```

Referenciranje objekata

Referenciranje određenog objekta u hijerarhiji se vrši korišćenjem operatora tačka (.) koji služi kao separator između kontejnera i člana. Puna adresa ćelije A1 prvog radnog lista radne sveske VBA.xls bi bila

```
Application.Workbooks("VBA.xls").Worksheets(1).Range("A1")
```

Rad sa ovakvim punim adresama doprinosi obimu i nepreglednosti koda. Tako se objekat `Application` može praktično izostaviti pri referenciranju ostalih objekata u hijerarhiji. Ako radimo samo sa jednom radnom sveskom onda nema potrebe navoditi o kojoj se radnoj svesci radi. Tako se referenciranje ćelije A1 svodi na

```
Worksheets(1).Range("A1")
```

Ukoliko je prvi radni list aktivan, onda se prethodni oblik može svesti na

```
Range("A1")
```

Za očekivati je da u VBA postoji objekat `Cell`, koji bi predstavljao ćeliju radnog lista. Međutim, objekat `Cell` ne postoji i njegovu ulogu vrši objekt `Range`. Objekt `Range`, osim što može predstavljati jednu ćeliju, npr. `Range("C4")`, može predstavljati i selekciju ćelija, na primer `Range("C4:D8")` ili `Range("A1,B2:B10,C4:D8").` Štaviše, ako ste u radnom listu imenovali određeni opseg, možete mu pristupiti zapisom `Range("Ime opsega")`.

Referenciranje objekata samo po sebi ne vrši nikakvu konkretnu radnju, već omogućava pristup datom objektu. Konkretna radnja bi podrazumevala čitanje ili promenu osobine objekta ili pozivanje određenog metoda koji bi nešto uradio sa objektom.

Dodela objekta promenljivoj

Kao što smo već rekli, u VBA postoji i `Object` tip promenljivih. U promenljivu ovog tipa se može smestiti bilo koji Excel-ov objekt. Dodela vrednosti `Object` promenljivoj se vrši naredbom `Set`, u zapisu:

```
Set ImePromenljive = Objekt
```

Na primer, ukoliko želimo da dodelimo radni list Sheet1 aktivne radne sveske promenljivoj `List`, potrebno je uraditi sledeće:

```
Dim List As Object
Set List = ActiveWorkbook.Worksheets("Sheet1")
```

U nastavku koda se ovaj radni list jednostavno može referencirati sa `List`.

Radi bržeg izvršavanja, potrebno je koristiti objekte odgovarajućeg tipa u procedurama. Na primer, u prethodnom primeru, umesto generičkog tipa `Object`, za deklararaciju promenljive `List` se mogao koristiti tip `Worksheet`, tj.

```
Dim List As Worksheet
```

U cilju daljeg poboljšanja performansi koda, objektne promenljive se mogu dealocirati, tj. memorija zauzeta njima se može osloboditi (kada više nisu potrebne) upisivanjem vrednosti `Nothing` u te promenljive. U prethodnom primeru bi to bilo:

```
Set List = Nothing
```

Osobine objekata

Svaki VBA objekt ima određeni skup karakteristika koje se nazivaju *osobine* objekta. Osobine definišu izgled i poziciju objekta. Na primer, svaki objekt `Window` ima osobinu `WindowState` kojom se dati prozor može prikazati kao minimizovan, maksimizovan ili normalan. Objekt `Range` ima osobinu `Value`, pomoću koje se vrednost ćelije može očitati ili promeniti.

Osobina objekta se takođe referencira koristeći operator tačka, tj. u obliku

```
Objekt.Osobina
```

Na primer, vrednost ćelije A1 prvog radnog lista se može dobiti na sledeći način:

```
Worksheets(1).Range("A1").Value
```

i ta se vrednost može prikazati ili dodeliti nekoj promenljivoj. Naravno, u prethodnom zapisu, `Worksheets(1).Range("A1")` predstavlja objekt, a `Value` osobinu.

Promena vrednosti osobine objekta se vrši na sledeći način:

```
Objekt.Osobina = Vrednost
```

gde `Vrednost` predstavlja izraz čija se vrednost dodeljuje datoj osobini objekta. Vrednost može biti bilo kojeg tipa VBA promenljivih, pri čemu ćemo najčešće raditi sa numeričkim, stringovnim i logičkim tipom. Naredne instrukcije ilustruju rad sa svakim od ovih tipova.

```
Range("A1").Value = 23.11
Range("A1").Font.Size = 12
Range("A1").Font.Name = "Times New Roman"
Range("A1").Font.Bold = True
```

Većina objekata ima podrazumevanu osobinu, koja se pri referenciranju može izostaviti. Na primer, podrazumevana osobina objekta `Range` je `Value`. Na primer, instrukcijom

```
Range("A1") = 23.11
```

bi postigli isti efekat kao i sa `Range("A1").Value = 23.11`. Ipak, smatra se dobrom programerskom praksom stalno uključivati ime podrazumevane osobine.

Važno je znati i to da osobine mogu vratiti referencu na objekat. Na primer, u instrukciji

```
Range("A1").Font.Bold = True
```

osobina `Font` vraća objekt `Font` sadržan u objektu `Range`. Može delovati pomalo zbunjujuće da `Font` predstavlja i osobinu i objekat. I to nije jedini slučaj.

Metode objekata

Pored osobina, koje definišu izgled i poziciju objekata, objekti imaju i *metode*, koje predstavljaju akcije koje možemo vršiti nad objektima. Metod se takođe poziva koristeći operator tačka, tj. u obliku

```
Objekt.Metod
```

Na primer, objekt `Range` ima metodu `Clear` koja briše sadržaj i formatiranje predmetnog opsega. Na primer, instrukcija

```
Worksheets("Sheet1").Range("A1,B2,C3").Clear
```

briše sadržaj i formatiranje ćelija A1, B2 i C3. Brisanje sadržaja opsega, uz očuvanje formata, se vrši metodom `ClearContents`.

Neki metodi zahtevaju argumente kojima se specificira radnja. Na primer, ukoliko želimo da iskopiramo sadržaj jednog opsega u drugi, potrebno je koristiti metod `Copy`. Ovaj metod ima jedan argument koji je opcion i koji predstavlja destinaciju kopiranja. Na primer, ukoliko želimo da iskopiramo sadržaj opsega A1:C3 prvog radnog lista u opseg A4:C7 drugog radnog lista potrebno je pozvati metod `Copy` na sledeći način:

```
Worksheets(1).Range("A1:C3").Copy Worksheets(2).Range("A4:C7")
```

Ovde `Worksheets(2).Range("A4:C7")` predstavlja destinaciju kopiranja. Destinacija se kao argument može izostaviti i u tom slučaju se sadržaj opsega kopira na Clipboard.

Argumenti kod metoda i osobina

Argumenti kod metoda služe da pobliže odrede radnju koju vrši dati metod, dok kod osobina argumenti služe da specificiraju vrednost osobine. Argumenti mogu biti obavezni i opcioni.

Ako metod koristi argumente, oni se smeštaju nakon imena metoda i razdvajaju se zarezima. Opcioni argumenti se mogu izostaviti u pozivu metoda, kada njihova pozicija ostaje upražnjena.

Posmatrajmo metod `Protect` objekta `Workbook`, koji štiti radnu svesku, tj. ne dozvoljava njenu modifikaciju. Ovaj metod ima tri opciona argumenta:

- `Password` - string koji predstavlja case-sensitive lozinku za radni list.
- `Structure` - True ili False vrednost kojom se štiti struktura radne sveske (relativna pozicija radnih listova). Podrazumevana vrednost je False.
- `Windows` - True ili False vrednost kojom se štite prozori radne sveske.

Na primer, ako želimo da zaštitimo radnu svesku VBA.xls, možemo koristiti sledeću naredbu:

```
Workbooks("VBA.xls").Protect "MojaLozinka", True, False
```

čime se radna sveska štiti lozinkom `"MojaLozinka"`. Drugim argumentom, `True`, se štiti struktura radne sveske, dok se trećim argumentom, `False`, onemogućava zaštita prozora radne sveske. U ovom pozivu se argumenti moraju navesti u pravilnom redosledu, tj. prvo ide `Password`, pa `Structure`, pa `Windows`.

Ako ne dodelimo lozinku, tj. ne navedemo prvi argument, onemogućićemo zaštitu radne sveske. Ovo se radi na sledeći način:

```
Workbooks("VBA.xls").Protect , True, False
```

Prvi argument je izostavljen, ali smo ostavili praznu poziciju za njega.

Drugi način poziva metoda je pomoću imenovanih argumenata. Na primer, prethodni poziv odrađen preko imenovanih argumenata bi bio:

```
Workbooks("VBA.xls").Protect Structure:=True, Windows:=False
```

Na ovaj način možemo specifikovati samo one opcione argumente kojima želimo da damo vrednost, i to u proizvoljnom redosledu. Znači, potpuno istu stvar bi dobili pozivom

```
Workbooks("VBA.xls").Protect Windows:=False, Structure:=True
```

Uočimo način definisanja imenovanih argumenata pomoću operatora `:=`. Prednost korišćenja imenovanih argumenata dolazi do izražaja pri radu sa metodama koje imaju veliki broj opcionih argumenata, kada je dovoljno navesti samo one argumente koje želimo da definišemo. Ne mora se ostavljati prazan prostor za ostale argumente.

Ukoliko metod vraća rezultat koji želimo da upišemo u neku promenljivu, listu argumenata metode je potrebno navesti u malim zagradama.

Ako osobina koristi argumente, argumente je potrebno navesti u malim zagradama. Na primer, osobina `Resize` objekta `Range` vrši promenu veličine opsega i ima dva opciona argumenta, `RowSize` i `ColumnSize`, koji predstavljaju novi broj vrsta i kolona opsega. Specificiranje prvog argumenta na način

```
Worksheets(1).Range("A1").Resize 4
```

je neispravno jer nismo naveli argument u zagradama. Pravilno bi bilo

```
Worksheets(1).Range("A1").Resize(4)
```

ili

```
Worksheets(1).Range("A1").Resize(RowSize:=4)
```

Aktivni objekti

Pri radu se Excel-om, samo jedna radna sveska može biti trenutno aktivna. Slično, kod aktivne radne sveske, samo jedan radni list može biti trenutno aktivan i samo jedna ćelija je aktivna ćelija, čak i kada se radi sa opsegom koji obuhvata više ćelija. Uzimajući ovo u obzir, VBA nam dopušta da aktivnim objektima pristupamo na jednostavniji način.

Aktivnim objektima pristupamo preko osobina objekta `Application`. Na primer, objekt `Application` ima osobinu `ActiveCell` koja vraća referencu na aktivnu ćeliju. Tako, na primer, ako želimo da u aktivnu ćeliju upišemo broj 32.1, dovoljno je napisati

```
ActiveCell.Value = 32.1
```

Namerno je izostavljen objekt `Application` jer se podrazumeva. Važno je znati da se prethodna instrukcija neće izvršiti ukoliko aktivni list nije radni list, već mapa.

Ukoliko je selektovan opseg unutar radnog lista, aktivna ćelija će biti jedna ćelija tog opsega.

Objekt `Application` ima osobinu `Selection` koja vraća referencu na selektovani objekt, bila to jedna ćelija, opseg ćelija, mapni objekt, tekst boks ili oblik.

Pored osobina `ActiveCell` i `Selection`, korisne osobine objekta `Application` su `ActiveChart` (aktivna mapa ili mapni objekt na radnom listu), `ActiveWindow` (aktivni prozor), `ActiveWorkbook` (aktivna radna sveska), `RangeSelection` (selektovane ćelije na radnom listu), `ThisWorkbook` (radna sveska koja sadrži proceduru koja se trenutno izvršava).

Pogodnost rada sa ovim osobinama je da nema potrebe znati aktivnu ćeliju, radni list ili radnu svesku. Ovo dalje omogućava pisanje VBA koda koji nije vezan ni za jednu posebnu radnu svesku, radni list ili opseg.

`ActiveCell`, kao i `Font`, može predstavljati i osobinu i objekt. Tačnije, objekt `Window` ima osobinu `ActiveCell`, a već smo videli da `ActiveCell` predstavlja `Range` objekt.

Range objekti

`Range` objekt se nalazi unutar `Worksheet` objekta, i sastoji se od jedne ćelije ili opsega ćelija na jednom radnom listu. `Range` objektima se u VBA može pristupati koristeći:

- osobinu `Range` objekata `Worksheet` i `Range`,
- osobinu `Cells` objekata `Worksheet` i `Range`,
- osobinu `Offset` objekta `Range`.

Osobina Range

Osobina `Range` vraća objekt `Range` i referencira se na sledeći način:

```
Objekt.Range(Opseg)
```

`Opseg` može biti jedna ćelija, opseg ćelija, unija opsega, presek opsega ili definisano ime opsega. Navodimo nekoliko primera korišćenja ove osobine:

```
Worksheets(1).Range("C2").Value = 1
Worksheets(1).Range("C2:D7").Value = 2
ActiveSheet.Range("C2:D7,F4,H8").Value = 3
```

```
ActiveSheet.Range("C2:D7 B3:E5").Value = 4
ActiveSheet.Range("MojOpseg").Value = 5
```

Prva dva primera su, nadamo se, jasna. U trećem primeru imamo opseg koji predstavlja uniju opsega C2:D7 i F4:H8. Dakle, unija opsega se formira razdvajajući opsege zarezima. U četvrtom primeru imamo presek opsega C2:D7 i B3:E5. Kod preseka opsega, opsege razdvajamo spejsovima. Konačno, u petom primeru pristupamo opsegu čije je ime `"MojOpseg"`, koji je definisan u okviru aktivnog radnog lista.

Ovo su bili primeri korišćenja osobine `Range` na `Worksheet` objektima. Kao što je rečeno, ova se osobina može koristiti i sa `Range` objektima. Na primer, instrukcijom

```
ActiveCell.Range("C3") = 25.5
```

se pristupa ćeliji koja se nalazi u odnosu na aktivnu ćeliju isto kao što se ćelija C3 nalazi u odnosu na ćeliju A1. Dakle, C3 predstavlja relativnu adresu ćeliju u odnosu na aktivnu ćeliju. U prethodnoj instrukciji smo izostavili podrazumevanu osobinu `Value`.

Pored ovog načina relativnog pristupa ćelijama, postoji još jedan, jednostavniji, način relativnog pristupa, korišćenjem osobine `Offset`, koji ćemo objasniti nešto kasnije.

Osobina Cells

Kao i osobina `Range`, osobinu `Cells` imaju `Worksheet` i `Range` objekti. Postoje tri sintakse korišćenja ove osobine:

```
Objekt.Cells(rowIndex,columnIndex)
Objekt.Cells(rowIndex)
Objekt.Cells
```

Najjednostavniji i najlogičniji je prvi način korišćenja. Na primer, instrukcijom

```
Worksheets("Sheet1").Cells(3,5) = 51
```

se pristupa ćeliji u preseku treće vrste i pete kolone, tj. ćeliji E3, i u nju se upisuje broj 51. S' obzirom da radni list u Excel-u ima 65536 vrsta i 256 kolona, indeks vrste u osobini `Cells` je broj između 1 i 65536, dok je broj kolone između 1 i 256. Specijalno, `Cells(1,1)` odgovara ćeliji A1.

U drugom načinu korišćenja ove osobine se navodi samo broj vrste, i u tom slučaju se ćelijama opsega pristupa kao da je opseg razvijen u niz, vrstu po vrstu. Na primer, instrukcijama:

```
Worksheets("Sheet1").Cells(8) = 11
Worksheets("Sheet1").Cells(258) = 12
```

se pristupa ćelijama H1 i B2, respektivno. Poslednjoj ćeliji radnog lista se pristupa sa

```
Worksheets("Sheet1").Cells(16777216)
```

Treća sintaksa, bez navođenja broja vrsta i kolona, vraća sve ćelije referencirane radne sveske. Prethodne dve sintakse su vraćale jednu ćeliju. Na primer, instrukcija

```
ActiveSheet.Cells = 45
```

u čitav radni list upisuje broj 45, dok instrukcija

```
ActiveSheet.Cells.ClearContents
```

briše sadržaj čitavog radnog lista.

Kada se osobina `Cells` koristi sa `Range` objektima, imamo sličnu situaciju kao kod osobine `Range`, tj. argumenti u zagradi će definisati relativnu adresu ćelije u odnosu na aktivnu ćeliju, koja se nalazi u gornjem levom uglu referenciranog opsega. Na primer, ako je aktivna ćelije B5, instrukcijama

```
ActiveCell.Cells(1,1) = 5
ActiveCell.Cells(2,2) = 6
```

ćemo menjati sadržaj ćelijama B5 (aktivna) i C6. Kada se izostavi broj kolona, onda se ćelijama opsega pristupa kao da je opseg razvijen u niz, vrstu po vrstu. Na primer, instrukcija

```
Range("A1:D5").Cells(5)
```

vraća ćeliju A2. Ovim zapisom nismo ograničeni samo na pristup ćelijama iz opsega A1:D5, već možemo pristupati i ćelijama tog opsega. Na primer, instrukcijom

```
Range("A1:D5").Cells(22)
```

bismo referencirali ćeliju B6.

Ukoliko želimo da obiđemo selektovani opseg, ćeliju po ćeliju, i da izvršimo određenu operaciju (recimo, upis slučajnog broja koji vraća funkcija `Rnd`) sa svakom ćelijom, to možemo efikasno uraditi koristeći dve petlje i osobinu `Cells` na sledeći način:

```
Dim I As Integer, J As Integer
For I = 1 To Selection.Rows.Count
    For J = 1 To Selection.Columns.Count
        Selection.Cells(I, J) = Rnd
    Next
Next
```

Broj vrsta i kolona predmetnog opsega se dobija pomoću osobine `Count` objekta `Range`.

Opseg se može obići i koristeći osobinu `Cells` sa jednim argumentom, što je dato ispod.

```
Dim I As Integer
For I = 1 To Selection.Count
   Selection.Cells(I) = Rnd
Next
```

Osobina Offset

Kao i osobine `Range` i `Cells`, osobina `Offset` vraća `Range` objekt, ali za razliku od ove dve osobine, `Offset` se odnosi samo na `Range` objekte i nijedne više. Sintaksa ove osobine je:

```
Objekt.Offset(rowOffset,columnOffset)
```

Argumenti ove osobine definišu relativni pomeraj (offset) ćelije u odnosu na gornji levi ugao opsega. Ovi argumenti mogu biti pozitivni (krećemo se dole ili desno), negativni (krećemo se gore ili levo), ili nula. Na primer, ukoliko je aktivna ćelija C3, sa instrukcijama

```
ActiveCell.Offset(1,0).Value = 5
ActiveCell.Offset(-1,0).Value = 6
```

menjamo vrednost ćelijama C4 i C2, respektivno. Ukoliko je aktivna ćelija A1, onda argumenti ne mogu biti negativni, tj. desiće se greška pri pokušaju unosa negativnog pomeraja, jer odgovarajuće ćelije ne postoje. `Offset(0,0)` odgovara aktivnoj ćeliji.

Rad sa objektima i kolekcijama

Pri radu sa VBA, dosta vremena ćemo provesti radeći sa objektima i kolekcijama. Većina objekata ima veliki broj osobina i metoda, i vrlo često se nad jednim objektom izvršava veliki broj akcija (promena vrednosti osobina i pozivanje metoda). Ovo može značajno povećati obimnost koda kada se radi sa dugim imenima.

VBA pruža mogućnost elegantnog i efikasnog rada sa objektima i kolekcijama pomoću `With-End With` konstrukcije i `For Each-Next` petlje.

With-End With konstrukcija

Ova konstrukcija omogućava da se izvrši veliki broj operacija na jednom objektu. Rad sa ovom naredbom je najbolje objasniti na sledećem primeru. Pretpostavimo da želimo da definišemo boju ćelija i prelom teksta u opsegu A1:D5, kao i nekoliko osobina fonta u tom opsegu. To se može uraditi na sledeći način:

```
Range("A1:D5").Interior = RGB(5,23,109)
Range("A1:D5").WrapText = True
Range("A1:D5").Font.Name = "Courier New"
Range("A1:D5").Font.Bold = True
Range("A1:D5").Font.Size = 13
Range("A1:D5").Font.Underline = xlUnderlineStyleSingle
Range("A1:D5").Font.ColorIndex = 34
```

S' obzirom da se sve naredbe odnose na isti opseg, one se mogu grupisati naredbom `With-End With` na sledeći način:

```
With Range("A1:D5")
    .Interior.Color = RGB(59, 123, 19)
    .WrapText = True
    .Font.Name = "Courier New"
    .Font.Bold = True
    .Font.Size = 13
    .Font.Underline = xlUnderlineStyleSingle
    .Font.ColorIndex = 34
End With
```

Prvi set naredbi je jasniji, ali se drugi set naredbi izvršava dosta brže, jer se objekt ne referencira eksplicitno u svakoj naredbi. S' obzirom da se u drugom setu naredbi ponavlja objekt `Font`, može se ugnezditi jedna `With-End With` naredba kojom bi se grupisale osobine ovog objekta, što je dato ispod.

```
With Range("A1:D5")
    .Interior.Color = RGB(59, 123, 19)
    .WrapText = True
    With .Font
        .Name = "Courier New"
        .Bold = True
        .Size = 13
        .Underline = xlUnderlineStyleSingle
        .ColorIndex = 34
    End With
End With
```

Iz prethodnih naredbi vidimo da se boja fonta može odrediti pomoću funkcije `RGB`, koja se poziva na sledeći način:

```
RGB(crvena,zelena,plava)
```

Ova funkcija ima tri obavezna argumenta, koji predstavljaju udeo crvene, zelene i plave boje. U pitanju su celi brojevi od 0 do 255. Iako se na ovaj način može definisati 256^3 boja, mi možemo koristiti samo 56 boja, što je posledica činjenice da svaka radna sveska ima dodeljenu paletu od 56 boja. Ova se paleta može menjati na tabu **Color** prozora **Tools⇒Options** kod MS Excel-a 2003. Umesto boje koju smo definisali, biće prikazana boja iz palete koja joj najbolje odgovara.

Drugi način da se definiše boja u VBA je pomoću osobine `ColorIndex`, koja može uzeti vrednosti od 0 (bez boje) do 56. Dobro pogađate zašto baš 56.

For Each-Next petlja

Ova petlja predstavlja varijaciju `For-Next` petlje i namenjena je radu sa kolekcijama, tj. kada je potrebno izvršiti određenu operaciju nad svim objektima u kolekciji. Sintaksa `For Each-Next` petlje je:

```
For Each Element in Group
    Instrukcije
Next Element
```

`Group` predstavlja ime kolekcije i to može biti i niz. Iz `For Each-Next` petlje se takođe izlazi koristeći `Exit For`.

Navedimo prvo jedan jednostavan primer korišćenja ove naredbe sa nizom od 10 celih brojeva.

```
Dim Niz(1 To 10) As Integer
For I = 1 To 10
    Niz(I) = I ^ 2
Next I
For Each n In Niz
    Debug.Print n
Next n
```

Prvom `For-Next` petljom se formira niz, dok se `For Each-Next` petljom štampaju elementi niza. Uočimo da pri korišćenju `For Each-Next` petlje ne moramo znati koliko ima elemenata u nizu.

Posmatrajmo sledeći primer.

```
Dim List As Worksheet
For Each List In ActiveWorkbook.Worksheets
    MsgBox List.Name
Next List
```

Ovde smo prošli kroz čitavu kolekciju `Worksheets` aktivne radne sveske i pomoću MsgBox-a prikazali ime svakog radnog lista. U svakoj iteraciji naredbe, objektna promenljiva `List` postaje tekući radni list. Na ovaj način ostvarujemo vrlo elegantan rad sa radnim listovima. Opet, nema potrebe da znamo koliko ima radnih listova u svesci, ovaj će zapis proći kroz svaki element kolekcije.

U narednom primeru se svakoj ćeliji iz selekcije dodeljuje slučajan cijeli broj između 0 i 100.

```
Dim Cell As Range
For Each Cell In Selection
    Cell.Value = Fix(Rnd * 101)
Next Cell
```

Završićemo poglavlje sa primerom kojim se zatvaraju sve radne sveske osim aktivne.

```
Dim Sveska as Workbook
For Each Sveska In Workbooks
    If Sveska.Name <> ActiveWorkbook.Name Then Sveska.Close
Next Sveska
```

Osobine i metode objekta Application

U tabelama 16. i 17. su respektivno nabrojane neke od korišćenih osobina i metoda objekta `Application`, koji, kako je već rečeno, predstavlja sam Excel. Za spisak ostalih osobina i metoda najlakše je konsultovati Help.

Osobina	Opis
`StandardFont`	Vraća ili menja standardni Excel-ov font. Pri promeni fonta koristiti sledeću sintaksu: `Application.StandardFont = "Courier New"`
`StandardFontSize`	Vraća ili menja veličinu standardnog Excel-ovog fonta. Na primer: `Application.StandardFontSize = 14`
`SheetsInNewWorkbook`	Vraća ili menja broj radnih listova u novoj radnoj svesci.
`DefaultFilePath`	Vraća ili menja početni folder koji se pojavljuje kada se prikaže Open ili Save As dijalog boks. Na primer: `Application.DefaultFilePath = "C:\Temp"`
`UserName`	Vraća ili menja korisničko ime za aplikaciju.
`OperatingSystem`	Vraća ime i verziju operativnog sistema.
`CutCopyMode`	Vraća ili menja Excel-ov status za operacije Cut i Copy. Ukoliko u kodu kopiramo objekt `Range`, Excel ostaje u Copy modu i nakon operacije Paste, odnosno biće prikazan pokretni okvir oko kopiranog opsega. Da bismo izašli iz ovog moda, potrebno je izvršiti: `Application.CutCopyMode = False`
`DisplayAlerts`	Vraća i menja mogućnost prikaza Excel-ovih alarma i poruka. Ukoliko ne želimo da nas Excel prekida sa svojim porukama, u ovu osobinu treba da se upiše `False`.
`MemoryFree`	Vraća veličinu slobodne memorije koja je dostupna Excel-u
`MemoryTotal`	Vraća veličinu ukupne sistemske memorije dostupne Excel-u
`MemoryUsed`	Vraća veličinu sistemske memorije koju je Excel trenutno zauzeo.

Tabela 16. Neke od osobina objekta Application.

Metod	Opis
`Calculate`	Izvršavanje svih otvorenih radnih sveski.
`FindFile`	Otvara se Open dijalog boks.

`Wait`	Pauzira izvršavanje makroa dok se ne dostigne specificirano vreme. Sintaksa je:
	`Application.Wait(Vreme)`
	gde `Vreme` predstavlja vreme kad makro nastavlja sa izvršavanjem. Na primer, ako želimo da pauziramo izvršenje makroa 35 sekundi, dovoljno je izvršiti:
	`Application.Wait Now + TimeValue("00:00:05")`
	U prethodnoj naredbi, `Now` je funkcija koja vraća tekući datum i vreme, dok `TimeValue` predstavlja funkciju koja vraća `Date` podatak koji sadrži

	specificirano vreme.
`Quit`	Zatvaranje Excel-a. Ukoliko postoji više otvorenih radnih svesaka sa nesnimljenim promenama, Excel će pitati da li želimo da snimimo ove promene. will ask if you want to save the changes. Ako ne želimo da nas Excel pita za snimanje, možemo snimiti sve radne sveske pre zatvaranja, ili da podesimo osobinu `DisplayAlerts` na False (tada se radne sveske *ne snimaju*!)

Tabela 17. Neke od metoda objekta Application.

Osobine i metode objekta Workbook i kolekcije Workbooks

U tabelama 18. i 19. su nabrojane neke od korišćenih osobina i metoda objekta `Workbook` (koji u hijerarhiji dolazi odmah ispod objekta `Application`) i kolekcije `Workbooks`, respektivno. Za spisak ostalih osobina i metoda najlakše je konsultovati Help.

Osobina	Opis
Objekt Workbook	
`FullName`	Vraća ime radne sveske, uključujući i put do nje.
`Name`	Vraća ime radne sveske.
`Path`	Vraća put do radne sveske. Kod novih, nesnimljenih radnih sveski, osobina `Path` je prazan string (`""`).
`Saved`	Vraća podatak o tome da li je bilo promena u radnoj svesci od trenutka poslednjeg snimanja. Ukoliko promena nije bilo, `Saved` vraća `False`.
Kolekcija Workbooks	
`Count`	Vraća broj trenutno otvorenih radnih sveski.

Tabela 18. Neke od osobina objekta Workbook i kolekcije Workbooks.

Metod	Opis
Objekt Workbook	
`Activate`	Aktiviranje specificirane radne sveske. Ta sveska postaje `ActiveWorkbook`.
`Close`	Zatvaranje radne sveske. Ovaj metod ima (pojednostavljenu) sintaksu: `Close(SaveChanges, FileName)` gde `SaveChanges` specificira da li želimo da snimimo promene (True) ili ne (False), dok `FileName` predstavlja ime pod kojim želimo da snimimo izmenjeni fajl. Fajl će biti snimljen samo ako je bilo promena u fajlu i `SaveChanges` ima vrednost True.
`PrintOut`	Štampanje specificirane radne sveske. Pogledati Help radi više informacija.
`PrintPreview`	PrintPreview prikaz radne sveske.
`Protect`	Zaštita radne sveske. Za više informacija pogledati sekciju *Argumenti kod metoda i procedura*.
`Unprotect`	Uklanjanje zaštite sa radne sveske. Sintaksa je: `Unprotect(Password)` gde `Password` predstavlja lozinku koju smo postavili metodom `Protect`.
`Save`	Snimanje radne sveske. Ukoliko je u pitanju nova radna sveska, koristiti

	metod SaveAs.
SaveAs	Snimanje specificirane radne sveske pod drugim imenom. Ovaj metod ima (pojednostavljenu) sintaksu: SaveAs(FileName) gde FileName predstavlja puno ime fajla, uključujući i put do njega.
Kolekcija Workbooks	
Add	Kreiranje nove radne sveske. Sintaksa je: Add(Template) gde je Template opcioni argument koji predstavlja šablon na osnovu kojeg se formira nova radna sveska. To može biti string koji specificira neki Excel-ov fajl koji može poslužiti kao šablon, a može biti i konstanta koja definiše šablon. Ukoliko se Template izostavi, kreira se radna sveska sa brojem radnih listova definisanih osobinom SheetsInNewWorkbook objekta Application.
Open	Otvaranje postojeće radne sveske. Pojednostavljena sintaksa ovog metoda je: Open(FileName) gde je FileName string koji predstavlja puno ime fajla, uključujući i put. Ukoliko se fajl nalazi u tekućem folderu (određenog osobinom DefaultFilePath objekta Application), dovoljno ja navesti samo ime fajla, na primer: Workbooks.Open "VBA.xls"

Tabela 19. Neke od metoda objekta Workbook i kolekcije Workbooks.

Osobine i metode objekta Worksheet i kolekcije Worksheets

U tabelama 20. i 21. su respektivno nabrojane neke od korišćenih osobina i metoda objekta Worksheet i kolekcije Worksheets. Za spisak ostalih osobina i metoda konsultovati Help.

Osobina	Opis
Objekt Worksheet	
Name	Vraća ili menja ime radnog lista.
StandardHeight	Vraća standardnu visinu svih ćelija u datoj radnoj svesci.
StandardWidth	Vraća standardnu dužinu svih kolona u datoj radnoj svesci.
Visible	Određuje da li se dati radni list može videti (True) ili ne (False).
Rows	Vraća objekt Range koji predstavlja specificiranu vrstu datog radnog lista. Na primer, instrukcijom Worksheets(1).Rows(5).Delete brišemo petu vrstu u prvoj radnom listu tekuće radne sveske.
Columns	Vraća objekt Range koji predstavlja specificiranu kolonu datog radnog lista. Na primer, instrukcijom Worksheets(1).Columns(1).Font.Italic = True se postavlja Italic stil slova u prvoj koloni prve radne sveske.
Cells	Vraća Range objekat specificiran argumentima osobine Cells.
Kolekcija Worksheets	
Count	Vraća broj radnih listova u aktivnoj radnih sveski.

Tabela 20. Neke od osobina objekta Worksheet i kolekcije Worksheets.

Metod	Opis
Objekt Worksheet	
Activate	Aktiviranje specificiranog radnog lista. Taj list postaje ActiveSheet.
Select	Selektovanje specificiranog radnog lista.
Calculate	Kalkulacija specificiranog radnog lista.

Copy	Kopiranje specificiranog radnog lista. Sintaksa metoda je:
	`Copy(Before, After)`
	Before predstavlja radni list ispred kojeg smeštamo kopirani list, dok After predstavlja radni list iza kojeg smeštamo kopirani list. Ne mogu se istovremeno specificirati oba argumenta. Ako se oba argumenta izostave, Excel otvara novu radnu svesku za kopirani list. Na primer, instrukcijom
	`Worksheets("Sheet1").Copy Before:=Worksheets("Sheet3")`
	kopiramo Sheet1 ispred lista Sheet3.

Move	Pomeranje specificiranog radnog lista. Sintaksa metoda je:
	`Move(Before, After)`
	Before predstavlja radni list ispred kojeg smeštamo pomereni list, dok After predstavlja radni list iza kojeg smeštamo pomereni list. Ne mogu se istovremeno specificirati oba argumenta. Ako se oba argumenta izostave, Excel otvara novu radnu svesku za pomereni list. Na primer, instrukcijom
	`Worksheets("Sheet1").Move After:=Worksheets("Sheet3")`
	pomeramo Sheet1 iza lista Sheet3.

Delete	Brisanje specifiranog radnog lista.
Protect	Zaštita specificiranog radnog lista. Pojednostavljena sintaksa ovog metoda je: `Protect(Password)` gde Password predstavlja lozinku kojom se štiti radni list.
Unprotect	Uklanjanje zaštite sa radnog lista. Sintaksa je: `Unprotect(Password)` gde Password predstavlja lozinku koju smo postavili metodom Protect.
Kolekcija Worksheets	

Add	Dodavanje novih radnih lista u datu radnu svesku. Sintaksa metoda je:
	`Add(Before, After, Count, Type)`
	Before predstavlja radni list ispred kojeg smeštamo kopirani list, dok After predstavlja radni list iza kojeg smeštamo kopirani list. Ne mogu se istovremeno specificirati oba argumenta. Ako se oba argumenta izostave, novi radni list se dodaje ispred aktivne radne sveske. Count je broj listova koji se dodaje i ukoliko se izostavi, podrazumevano se dodaje jedan list. Type predstavlja tip radnog lista i podrazumevano je xlWorksheet. Na primer, instrukcijom
	`Worksheets.Add After:=Worksheets("Sheet1"), Count:=2`
	se dodaju dva radna ispred lista Sheet1.

Tabela 21. Neke od osobina objekta Worksheet i kolekcije Worksheets.

Osobine i metode objekta Range

Sa objektom Range smo se već nekoliko puta sretali i videli smo na koje se sve načine može pristupiti jednom Range objektu. Poznavanje rada sa Range objetima, tj. ćelijama, opsezima i

imenovanim opsezima, je od ključne važnosti, jer, kako god se okrenemo, uvek ćemo nešto raditi sa sadržajem ćelija.

Da se ne bi previše ponavljali, u tabelama 22. i 23. su date osobine i metode objekta `Range` koje do sad nismo pominjali, ili ih bar nismo detaljno objasnili.

Osobina	Opis
`Address`	Vraća adresa, u vidu teksta, specifiranog opsega.
`Row`	Vraća broj prve vrste specifiranog opsega. Na primer, `Range("C4:G7").Row` vraća broj 4.
`Column`	Vraća broj prve kolone specifiranog opsega. Na primer, `Range("C4:G7").Column` vraća broj 3.
`Count`	Vraća broj ćelija specifiranog opsega. Na primer, `Range("C4:G7").Count` vraća broj 20.
`CurrentRegion`	Vraća `Range` objekt koji predstavlja region u kom se nalazi specificiran opseg. Region je pravougaoni blok ćelija okružen sa najmanje jednom praznom vrstom iznad i ispod, i najmanje jednom praznom kolonom sa leve i desne strane. Ova osobina je vrlo korisna kad ne znamo veličinu opsega sa kojim radimo, ili kad se taj opseg menja dodajući ili oduzimajući vrste i kolone.
`Formula`	Vraća ili upisuje formulu u specificirani opseg.
`Comment`	Vraća `Comment` objekt. Kolekcija `Comments` sadrži sve komentare tekućeg radnog lista. Na primer, `Range("C4").Comment.Text "Novi komentar"` će promeniti komentar u ćeliji C4. Ukoliko komentar ne postoji, VBA će javiti grešku.

Tabela 22. Neke od osobina objekta Range.

Metod	Opis

`Rows`	Vraća vrstu opsega. Ima sintaksu
	`Rows(Index)`
	gde `Index` predstavlja redni broj vrste opsega. Ukoliko se `Index` izostavi, metod vraća kolekciju svih vrsta opsega. Na primer,
	`Range("MojOpseg").Rows.Count`
	vraća broj vrsta opsega `MojOpseg`.

`Columns`	Vraća vrstu opsega. Ima sintaksu `Columns(Index)` gde `Index` predstavlja redni broj kolone opsega. Ukoliko se `Index` izostavi, metod vraća kolekciju svih kolona opsega.

`Cut`	Premeštanje opsega na Clipboard ili na novu destinaciju. Sintaksa je:
	`Cut(Destination)`
	gde `Destination` predstavlja ćeliju ili opseg gde smeštamo opseg. Ukoliko se `Destination` izostavi, opseg se smešta na Clipboard. Na primer, sa
	`Range("A1:C4").Cut Destination:=Range("D5")`

	premeštamo opseg A1:C4 u opseg gde se ćelija D5 nalazi u gornjem levom uglu, tj. u opseg D5:F8.
`Copy`	Kopiranje opsega na Clipboard ili na novu destinaciju. Sintaksa je: `Copy(Destination)` gde `Destination` predstavlja ćeliju ili opseg gde kopiramo opseg.
`Clear`	Brisanje kompletnog sadržaja opsega, uključujući i format i komentare.
`ClearComments`	Brisanje komentara iz opsega.
`ClearContents`	Brisanje sadržaja iz opsega.
`ClearFormats`	Brisanje formata opsega.
`Delete`	Brisanje specificiranog opsega.
`Resize`	Promena veličine specifiranog opsega. Sintaksa je `Resize(RowSize,ColSize)` `RowSize` i `ColSize` su novi broj vrsta i kolona opsega, respektivno.
`Select`	Selektovanje specifiranog opsega.
`AddComment`	Dodavanje komentara u specifiranu ćeliju. Moguće je dodati komentar u samo jednu ćeliju. Ukoliko ćelija već sadrži komentar, VBA će javiti grešku.

Tabela 23. Neke od metoda objekta Range.

P R O G R A M I R A N J E
K R O Z
A P L I K A C I J E

Doc. dr Đukanović Slobodan

ČETVRTI TERMIN

Programiranje događaja u Excel-u

Excel može da prati veliki broj događaja koji se dešavaju tokom izvršenja. Ove događaje možemo klasifikovati na:

- Događaje vezane za radne sveske. Primeri ovih događaja su `Open` (radna sveska je otvorena ili kreirana), `BeforeSave` (radna sveska treba da se snimi), `BeforeClose` (radna sveska treba da se zatvori) i `NewSheet` (dodavanje novog radnog lista).
- Događaje vezane za radne listove. Primeri ovih događaja su `Change` (ćelija radnog lista je promenjena), `SelectionChange` (promena selekcije na radnom listu) i `Calculate` (radna sveska je nanovo proračunata).
- Događaje vezane za mape. Ovi događaji uključuju `Select` (objekt na mapi je selektovan) i `SeriesChange` (promena vrednosti podatka u sekvenci koja je grafički prikazana.
- Događaji vezani za aplikaciju. Ovde imamo događaje `NewWorkbook` (nova radna sveska je kreirana), `WorkbookBeforeClose` (bilo koja radna sveska treba da se zatvori), `SheetChange` (ćelija u bilo kojoj radnoj svesci je promenjena). Za praćenje događaja na nivou aplikacije, potrebno je koristiti klasni modul.
- Događaji vezani za korisničke forme. Ovde su uključeni `Initialize` (dešava se pre prikaza korisniče forme), `Click` (dešava se kad se klikne CommandButton).
- Događaji koji nisu vezani za objekte. Ovde imamo dva korisna događaja na nivou aplikacije, `OnTime` i `OnKey`, koji rade drugačije od ostalih događaja.

Procedure za upravljanje događajima (*event-handler procedures*) moraju biti smeštene u okviru odgavarajućih modula. Na primer, procedure koje upravljaju događajima vezanim za radne sveske se smeštaju u okviru `ThisWorkbook` modula, koje upravljaju događajima vezanim za radne listove se smeštaju u okviru modula radne sveske (npr., Sheet1, Sheet2).

Svaka procedura za upravljanje događajima ima predodređeno ime, koje direktno ukazuje na događaj o kome se radi. Tako, na primer, nije teško zaključiti o kojim se događajima radi iz imena `Workbook_Open`, `Worksheet_Calculate`, `Chart_Activate`. VBE omogućava elegantan unos zaglavlja procedure iz padajućeg menija koji se aktivira sa vrha kodnog prozora odgovarajućeg modula.

Na slici 3. je dat primer odabira procedure `Open` vezane za tekuću radnu svesku, čime se u kodni prozor unosi:

```
Private Sub Workbook_Open()

End Sub
```

Ostatak procedure je, naravno, naš.

Neke procedure za upravljanje događajima imaju argumente. Na primer, ono što bi nam uneo VBE pri odabiru procedure za upravljanje događajem `NewSheet` radne sveske je:

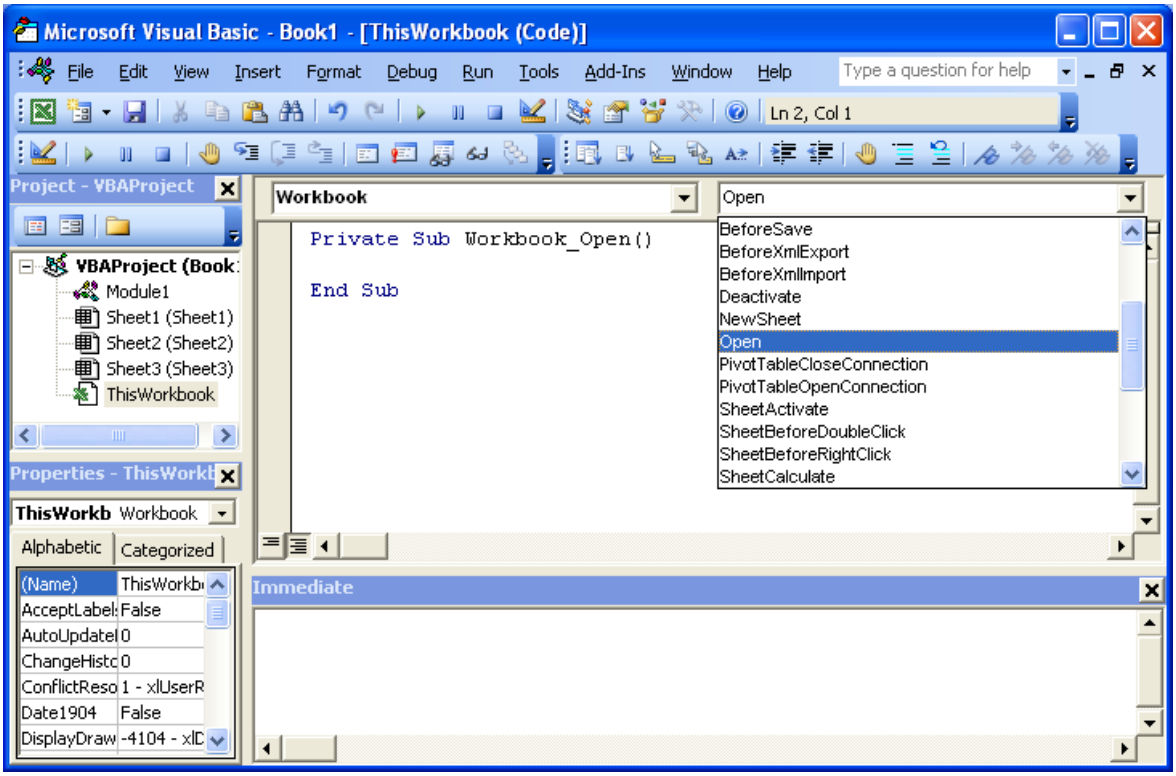
```
Private Sub Workbook_NewSheet(ByVal Sh As Object)

End Sub
```

gde se novi radni list prosleđuje kao argument (`Sh`). U ovom slučaju se `Sh` prosleđuje kao tip `Object` jer, pored radnog lista, u pitanju može biti i mapni list.

Nekoliko procedura za upravljanje događajima imaju `Boolean` argument nazvan `Cancel`, kojim se dati događaj može poništiti. Na primer, procedura za upravljanje događajem `BeforePrint` radne sveske ima deklaraciju:

```
Private Sub Workbook_BeforePrint(Cancel As Boolean)
```

gde je vrednost argumenta `Cancel` jednaka `False`. Ukoliko bi `Cancel` podesili na `True` u okviru procedure, otkazali bi štampanje.



Slika 1. Odabir odgovarajuće procedure za upravljanje događajima.

Događaji na nivou radne sveske

Događaji na nivou radne sveske su događaji koji se dešavaju u okviru određene radne sveske. Procedure za upravljanje ovim događajima se smeštaju u kodni modul `ThisWorkbook` objekta. U tabeli 24. je dat spisak jednog broja ovih događaja, uz akcije koja okidaju ove događaje. Nakon toga dajemo nekoliko urađenih primera za pojedine događaje.

Događaj	Akcija koja okida događaj
`Activate`	Radna sveska je aktivirana.
`BeforeClose`	Radna sveska treba da se zatvori.
`BeforePrint`	Radna sveska treba da se štampa ili pregleda sa Print Preview.
`BeforeSave`	Radna sveska treba da se snimi.
`Deactivate`	Radna sveska je deaktivirana.
`NewSheet`	Novi list je dodat u radnu svesku.
`Open`	Radna sveska je otvorena.
`SheetActivate`	Bilo koji radni list je aktiviran.
`SheetCalculate`	Bilo koja radna sveska je proračunata ili nanovo proračunata.
`SheetChange`	Bilo koja radna sveska je promenjena.

SheetDeactivate	Bilo koja radna sveska je deaktivirana.
SheetSelectionChange	Selekcija na bilo kojoj radnoj svesci je promenjena.
WindowActivate	Bilo koji prozor radne sveske je aktiviran.
WindowDeactivate	Bilo koji prozor radne sveske je deaktiviran.
WindowResize	Veličina bilo kojeg prozora radne sveske je promenjena.

Tabela 1. Neki događaji na nivou radne sveske.

Napominjemo još jednom da su ovo događaji vezani za konkretnu radnu svesku. Ukoliko želimo da pratimo događaje vezano za sve radne sveske, onda radimo sa događajima na nivou aplikacije.

Događaj Open

Ovo je jedan od događaja koji se najčešće prati. Događaj se okida otvaranjem radne sveske, a odgovarajuća procedura je `Workbook_Open procedure`. Ova se procedura najčešće koristi za prikaz poruka, otvaranje drugih radnih svesaka, postavljanje korisničkih menija i paleta alatki, aktiviranje određenog radnog lista ili ćelije itd.

Ovde ćemo uraditi jedan primer programiranja ovog događaja. Želimo, recimo, da vodimo evidenciju o otvaranju postojeće radne sveske. U tu svrhu ćemo koristiti jedan radni list, nazvan Evidencija, u koji ćemo upisivati vreme i datum svakog otvaranja radne sveske. Upisivanje vršimo u ćelijama A1, A2 itd. Ukoliko radni list Evidencija ne postoji, mi ćemo ga kreirati. Naravno, kreiranje se vrši samo pri prvom otvaranju radne sveske.

```
Private Sub Workbook_Open()
Dim List As Worksheet
Dim Postoji As Boolean, RadList As Integer, I As Integer
Postoji = False
For Each List In ThisWorkbook.Worksheets
    If List.Name = "Evidencija" Then Postoji = True
Next
RadList = ThisWorkbook.Worksheets.Count
If Postoji = False Then
    ThisWorkbook.Worksheets.Add after:=ThisWorkbook.Worksheets(RadList)
    ThisWorkbook.Worksheets(RadList + 1).Name = "Evidencija"
End If
ThisWorkbook.Worksheets("Evidencija").Select
I = 1
Do While ActiveSheet.Cells(I, 1).Value <> ""
    I = I + 1
Loop
ActiveSheet.Cells(I,1).Value = "Otvorena " & Date & " u " & Time
ThisWorkbook.Worksheets(1).Select
ThisWorkbook.Save
End Sub
```

Uočimo korišćenje funkcija `Date` i `Time`, koje respektivno vraćaju tekući datum i vreme.

Događaj NewSheet

Ovaj događaj se okida dodavanjem novog lista u predmetnu radnu svesku. List može biti radni list ili mapni list. Mi ćemo raditi uglavnom sa radnim listovima.

Ovde ćemo programirati proceduru `Workbook_NewSheet` koja će pre unošenja novog lista u radnu svesku korisniku prikazati InputBox u koji korisnik treba da unese ime radnog lista. Ukoliko radni list sa tim imenom već postoji, potrebno je prikazati novi InputBox, sa porukom o greški prilikom unosa, u koji će korisnik uneti novo ime. Unešeni radni list se smešta nakon svih postojećih radnih listova.

```
Private Sub Workbook_NewSheet(ByVal Sh As Object)
Dim Ime As String, ind As Boolean, L As Worksheet
ind = False
Ime = InputBox("Uneti ime: ", "Ime radnog lista")
For Each L In ThisWorkbook.Worksheets
    If StrComp(UCase(L.Name), UCase(Ime)) Then ind = True
Next
Do While ind = True
    Ime = InputBox("Pogresno ime! Uneti opet: ", "Ime radnog lista")
    ind = False
    For Each L In ThisWorkbook.Worksheets
        If StrComp(UCase(L.Name), UCase(Ime)) = 0 Then ind = True
    Next
Loop
Sh.Name = Ime
Sh.Move After:=ThisWorkbook.Worksheets(ThisWorkbook.Worksheets.Count)
End Sub
```

Događaj BeforePrint

Ovaj događaj se dešava neposredno pred štampu ili pregled pred štampu (Print Preview). Ne postoji način da se odredi o kom od ova dva zahteva se zapravo radi. Događaj koristi argument `Cancel`, kojim se može poništiti štampa radne sveske. To se postiže postavljanjem vrednosti argumenta `Cancel` na `True`.

Ovaj događaj postoji samo na nivou radne sveske, tj. ne postoji na nivou radnog lista, pa se ne može odrediti šta će biti štampano..

Dajemo jednostavan primer korišćenja ovog događaja. Pretpostavimo da je korisnik ostao bez tonera u štampaču i da želi da mu pri svakom zahtevu za štampu date radne sveske procedura podseća da je toner potrošen.

```
Private Sub Workbook_BeforePrint(Cancel As Boolean)
Dim Poruka As String, Odgovor As Integer
Poruka = "Ponestalo je tonera!" & vbCrLf & _
"Zelite li da nastavite sa stampom?"
Odgovor = MsgBox(Poruka, vbYesNo, "Podsecanje na toner")
If Odgovor = vbNo Then Cancel = True
End Sub
```

Događaji na nivou radnog lista

Događaji na nivou radnog lista su događaji koji se dešavaju u okviru određene radnog lista. Procedure za upravljanje ovim događajima se smeštaju u kodni modul odgovarajućeg radnog lista. U tabeli 25. je dat spisak nekih događaja na nivou radnog lista, uz akcije koja okidaju ove događaje. Nakon toga dajemo nekoliko urađenih primera za pojedine događaje.

Događaj	Akcija koja okida događaj

`Activate`	Radni list je aktiviran.
`BeforeDoubleClick`	Dvoklik na radnom listu.
`BeforeRightClick`	Desni klik na radnom listu.
`Calculate`	Radni list je proračunat ili nanovo proračunat.
`Change`	Ćelija radnog lista je promenjena.
`Deactivate`	Radni list je deaktiviran.
`SelectionChange`	Selekcija na radnom listu je promenjena.

Tabela 2. Neki događaji na nivou radnog lista.

Događaj BeforeRightClick

Ovaj događaj se dešava kada uradimo operaciju desni klik, neposredno pre podrazumevane radnje. Kao što znamo, podrazumevana radnja posle desnog klika je pojava padajućeg menija. Odgovarajuća procedura ima dva argumenta, `Target` i `Cancel`. `Target` označava tekuću selekciju na radnom listu ili, ako ništa nije selektovano, ćeliju na koju je korisnik kliknuo, a `Cancel` određuje da li će se izvršiti podrazumevana radnja.

Uradimo primer u kome programiramo ovaj događaj tako da se prazne ćelije selekcije ispunjavaju brojem kojeg unosimo pomoć InputBox-a. Onemogućiti pojavu padajućeg menija. Ukoliko nijedna ćelija nije prazna, sve negativne brojeve u opsegu zameniti nulom.

```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range,_
Cancel As Boolean)
Dim I As Integer, J As Integer, broj As Double, ImaPraznih As Boolean
ImaPraznih = False
Cancel = True
For I = 1 To Target.Rows.Count
    For J = 1 To Target.Columns.Count
        If Target.Cells(I, J) = "" Then ImaPraznih = True
    Next
Next
If ImaPraznih = True Then
    broj = Val(InputBox("Uneti broj: ", "Unos broja"))
    For I = 1 To Target.Rows.Count
        For J = 1 To Target.Columns.Count
            If Target.Cells(I, J) = "" Then Target.Cells(I, J) = broj
        Next
    Next
Else
    For I = 1 To Target.Rows.Count
        For J = 1 To Target.Columns.Count
            If Target.Cells(I, J) < 0 Then Target.Cells(I, J) = 0
        Next
    Next
End If
End Sub
```

Događaji na nivou aplikacije

Ukoliko želimo da pratimo događaje vezane za sve otvorene radne sveske ili sve radne listove, koristićemo događaje na nivou aplikacije. Odgovarajuće procedure se smeštaju u klasni modul i zahtevaju neka podešavanja. Tabeli 26. daje spisak nekih događaja na nivou aplikacije, uz akcije koja okidaju ove događaje.

Događaj	Akcija koja okida događaj
`NewWorkbook`	Nova radna sveska je kreirana.
`SheetActivate`	Bilo koji radni list je aktiviran.
`SheetCalculate`	Bilo koji radni list je proračunat ili nanovo proračunat.
`SheetChange`	Ćelija bilo kojeg radnog lista je promenjena.
`SheetDeactivate`	Bilo koji radni list je deaktiviran.
`SheetSelectionChange`	Selekcija na bilo kojoj radnoj svesci je promenjena.
`WindowActivate`	Bilo koji prozor radne sveske je aktiviran.
`WindowDeactivate`	Bilo koji prozor radne sveske je deaktiviran.
`WindowResize`	Veličina bilo kojeg prozora radne sveske je promenjena.
`WorkbookActivate`	Bilo koja radna sveska je aktivirana.
`WorkbookBeforeClose`	Bilo koja radna sveska treba da se zatvori.
`WorkbookBeforePrint`	Bilo koja radna sveska treba da se štampa.
`WorkbookBeforeSave`	Bilo koja radna sveska treba da se snimi.
`WorkbookDeactivate`	Bilo koja radna sveska je deaktivirana.
`WorkbookNewSheet`	Novi radni list je kreiran u bilo kojoj radnoj svesci.
`WorkbookOpen`	Radna sveska je otvorena.

Tabela 3. Neki događaji na nivou aplikacije.

Da bi se omogućili događaji na nivou aplikacije, potrebno je uraditi sledeće:

1. Kreirati novi klasni modul. Podrazumevano će ime klasnog modula biti `Class1`, `Class2`. Ime se može promeniti koristeći Properties prozor;
2. U klasnom modulu, deklarisati javni `Application` objekt koristeći ključnu reč `WithEvents`. Na primer:

   ```
   Public WithEvents AP As Application
   ```

3. Kreirati promenljivu koja će nam služiti kao referenca na deklarisani `Application` objekt u klasnom modulu. Ovo treba da je objektna promenljiva na nivou modula, deklarisana u običnom VBA modulu, a ne u klasnom modulu. Na primer, ukoliko je ime modula `Klasa`, deklaracija može biti:

   ```
   Dim X As New Klasa
   ```

4. Povezati deklarisani i `Application` objekt, što se obično radi u proceduri `Workbook_Open`. Povezivanje se vrši na sledeći način:

   ```
   Set X.AP = Application
   ```

5. Napisati proceduru za upravljanje događajem u klasnom modulu.

Događaji koji nisu pridruženi objektu

Događaji opisani u prethodnim poglavljima su bili vezani za objekte. Pored takvih događaja, postoje i događaji koji nisu pridruženi objektima, kao što su `OnTime` i `OnKey`. Ovim se

događajima pristupa koristeći metode objekta `Application`. Događaji `OnTime` i `OnKey` se programiraju u opštem VBA modulu.

P R O G R A M I R A N J E
K R O Z
A P L I K A C I J E

Doc. dr Đukanović Slobodan

Korisničke forme

Korisničke forme, ili korisnički dijalog box-ovi, predstavljaju najvažniji element korisničkog interfejsa u Windows okruženju. Zastupljeni su, bez sumnje, u svakom Windows programu. Rad sa njima je dosta intuitivan i lak.

Excel, takođe, praktično svu komunikaciju sa korisnikom ostvaruje koristeći dijalog box-ove. Štaviše, pomoću VBA možemo kreirati sopstvene dijalog box-ove kojima bismo prilagodili interfejs Excel-a našim potrebama. Pre nego što pređemo na kreiranje korisničkih formi, dajmo kratak prikaz onih Excel-ovih formi koje smo do sad koristili. U pitanju su Message box i Input box, a pored njih postoji još nekoliko vrlo korisnih predefinisanih formi.

Message box

VBA funkcija `MsgBox` omogućava jednostavan način prikaza poruke korisniku, pri čemu od korisnika možemo dobiti jednostavnu povratnu informaciju. Sintaksa ove funkcije je:

```
Odgovor = MsgBox(prompt, buttons, title, helpfile, context)
```

gde su objašnjenja pojedinih argumenata data ispod.

- `prompt` – tekst koji se prikazuje korisniku u box-u;
- `buttons` – numerički izraz koji određuje koja dugmad i ikonice će biti prikazane korisniku;
- `title` – naslov prozora message box-a;
- `helpFile`, `context` – određuju help fajl i stavku help-a. Zadaju se u paru.

Od argumenata message box-a, jedino je `prompt` obavezan argument.

Argument `buttons` pruža veliki broj opcija vezanih za dugmad koja se mogu pojaviti na message box-u. Pomoću njega se može specifiicirati koja će se dugmad i ikonice pojaviti, kao i koje dugme je podrazumevano. Konstante koje određuju dugmad na message box-u su date u tabeli 27.

Konstanta	Vrednost	Opis
`vbOKOnly`	0	Samo OK dugme
`vbOKCancel`	1	Dugmad OK i Cancel
`vbAbortRetryIgnore`	2	Dugmad Abort, Retry i Ignore
`vbYesNoCancel`	3	Dugmad Yes, No i Cancel
`vbYesNo`	4	Dugmad Yes i No
`vbRetryCancel`	5	Dugmad Retry i Cancel
`vbCritical`	16	Ikonica Critical Message
`vbQuestion`	32	Ikonica Warning Query
`vbExclamation`	48	Ikonica Warning Message
`vbInformation`	64	Ikonica Information Message
`vbDefaultButton1`	0	Prvo dugme je podrazumevano
`vbDefaultButton2`	256	Drugo dugme je podrazumevano
`vbDefaultButton3`	512	Treće dugme je podrazumevano
`vbDefaultButton4`	768	Četvrto dugme je podrazumevano
`vbSystemModal`	4096	Sve aplikacije su suspendovane dok korisnik ne klikne

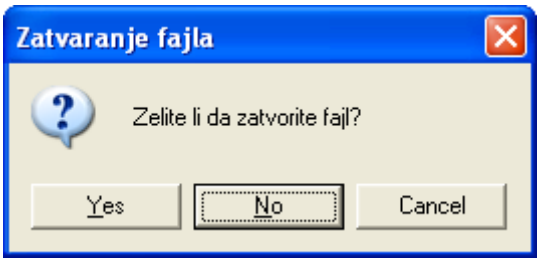
		neko dugme na message box-u.

Tabela 27. Konstante koje definišu dugmad na message box-u.

Kombinovanjem konstanti za dugmad, ikonice i podrazumevanu dugmad ostvarujemo željeni izgled prozora. Na primer, pozivom funkcije

```
MsgBox("Zelite li da zatvorite fajl?", _
       vbYesNoCancel + vbQuestion + vbDefaultButton2, "Zatvaranje fajla")
```

će se pojaviti prozor prikazan na slici 4.



Slika 4. Jedan primer message box-a.

Nadamo se da je ovaj poziv jasan.

Odgovor korisnika na message box se dobija preko vrednosti koju vraća funkcija `MsgBox`. U zavisnosti od dugmeta koje korisnik pritisne, funkcija vraća jednu od celobrojnih vrednosti datih u tabeli 28. Vraćena vrednost može dalje odrediti odgovarajuću akciju.

Konstanta	Vrednost	Pritisnuto dugme
`vbOK`	1	OK
`vbCancel`	2	Cancel
`vbAbort`	3	Abort
`vbRetry`	4	Retry
`vbIgnore`	5	Ignore
`vbYes`	6	Yes
`vbNo`	7	No

Tabela 28. Konstante koje vraća funkcija `MsgBox`.

Konstante date u prvoj koloni tabele 27. i 28. su deo specifikacije VBA i mogu se koristiti bilo gde u kodu umesto odgovarajućih numeričkih vrednosti. Lakše je pamtiti ove konstante, jer im je ime usko vezano za odgovarajuće dugme.

Ukoliko želimo da nam funkcija ne vraća rezultat, onda se argumenti ne navode u zagradi. Na primer, poziv

```
MsgBox "Danas je lep dan"
```

bi ispisao dati string i ne bi se vratio nikakav rezultat.

Input box

Input box predstavlja jednostavnu korisničku formu koja korisniku omogućava da unese jedan podatak. Vraćeni podatak je string, pa ga je potrebno dodatno obraditi ukoliko želimo da ga tretiramo drugačije (npr. ako želimo da ga tretiramo kao broj).

Input box se aktivira pomoću VBA funkcije `InputBox`, čija je sintaksa

```
Podatak = InputBox(prompt, title, default, xpos, ypos, helpfile, context)
```

gde je

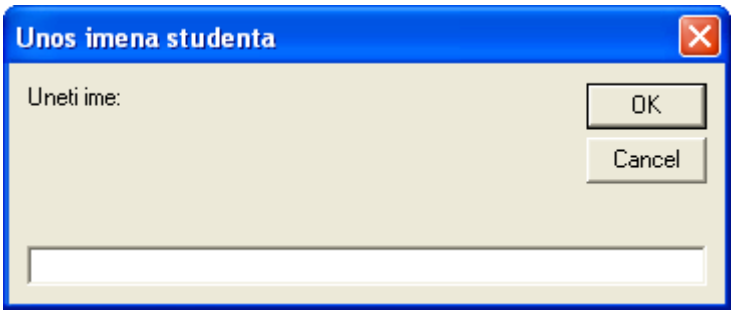
- `prompt` – tekst koji se prikazuje korisniku u input box-u;
- `title` – naslov prozora input box-a;
- `default` – podrazumevana vrednost prikazana u input box-u;
- `xpos`, `ypos` – ekranske koordinate gornjeg levog ugla prozora input box-a;
- `helpFile`, `context` – određuju help fajl i stavku help-a. Zadaju se u paru.

Od svih argumenata input box-a, jedino je `prompt` obavezan argument. Maksimalna dužina prompt-a je približno 1024 karaktera, zavisno od širine korišćenih karaktera.

Na primer, poziv

```
Ime = InputBox("Uneti ime: ", "Unos imena studenta")
```

će dati input box prikazan na slici 5.



Slika 5. Jedan primer input box-a.

Ime koje smo uneli u tekst polje će biti dodeljeno promenljivoj `Ime`.

Korisnički definisani dijalog box-ovi

Korisnički dijalog box-ovi se kreiraju na korisničkoj formi, UserForm, koja se u dati projekt unosi opcijom **Insert⇒UserForm**. Nakon toga se na formu obično dodaju kontrole, podese se njihove osobine i napišu procedure za upravljanje događajima vezanim za te kontrole (npr. programiramo šta ćemo uraditi kad se pritisne određeno dugme na formi). Ove procedure se nalaze u kodnom prozoru UserForm-a.

Prikaz dijalog box-a

Korisnička forma se prikazuje pomoću metode `Show` objekta `UserForm`. Instrukcija kojom se prikazuje forma se mora naći u standardnom VBA modulu, a ne u kodnom prozoru forme. Obično se u ovu svrhu kreira posebna procedura. Na primer, procedura

```
Sub PrikaziFormu()
    UserForm1.Show
```

```
End Sub
```

će prikazati dijalog box forme `UserForm1`.

VBA poseduje i naredbu `Load` pomoću koje se vrši učitavanje forme u memoriju, na sledeći način:

```
Load UserForm1
```

Ipak, ovako učitana forma nije vidljiva dok se pozove metod `Show`. Naredba `Load` se obično koristi kod složenih formi koje se učitavaju u memoriju pre nego što zatrebaju kako bi se skratilo vreme prikaza forme metodom `Show`.

Počev sa Excel-om 2000, korisničke forme mogu biti "modeless", što znači da korisnik može pristupiti radnoj svesci, ili aplikaciji, bez zatvaranja forme. Ovaj način rada se postiže na sledeći način:

```
UserForm1.Show vbModeless
```

Podrazumevani način prikaza korisničke forme je modalan, tj. bez mogućnosti pristupa aplikaciji bez prethodnog zatvaranja forme.

Za zatvaranje forme se može koristiti naredba `Unload`. Na primer, forma `UserForm1` se može zatvoriti na bilo koji od sledeća dva načina:

```
Unload UserForm1
Unload Me
```

`Me` je VBA ključna reč pomoću koje forma referencira samu sebe. Koristi se radi skraćenja referenciranje forme i odnosi se na onu formu u kojoj se nalazi procedura koja sadrži reč `Me`.

Nakon zatvaranja forme naredbom `Unload` kontrole na formi su resetovane na originalne vrednosti. Isti efekat se postiže pritiskom dugmeta za zatvaranje forme (× u naslovnoj liniji forme). Pritisak na ovo dugme aktivira UserForm događaje `QueryClose` i `Terminate`.

Forma se može zatvoriti koristeći metod `Hide`. Pozivom ovog metoda dijalog box nestaje se ekrana, ali ostaje učitan u memoriji, tako da naš kod i dalje može pristupiti osobinama kontrola. Ova metoda se poziva na sledeći način:

```
UserForm1.Hide
Me.Hide
```

Dodavanje kontrola formi

Kontrola se na formu dodaju koristeći Toolbox, koji se, ako nije aktiviran, aktivira opcijom **View⇒Toolbox**. VBE nema mogućnost dodavanja kontrola iz menija. Toolbox sa kontrolama je prikazan na slici 6.

Slika 6. Toolbox sa kontrolama za korisničke forme

Dodavanje kontrola se vrši prevlačenjem kontrole sa Toolbox-a na formu. Osobine dodate kontrole se mogu menjati pomoću Properties prozora.

Podrazumevano ime dodate kontrole se sastoji od tipa kontrole i rednog broja kontrole tog tipa koja se dodaje. Na primer, prvi tekst box koji se dodaje se zove TextBox1, drugi TextBox2 itd. Poželjno je ova imena uskladiti sa našim potrebama, tj. svakoj kontroli dati ime koje asocira na ulogu kontrole.

U nastavku je dat spisak kontrola koje možemo dodati preko Toolbox-a, uz objašnjenje šta svaka od njih omogućava. Kontrole navodimo sa vrha naniže, sa leva udesno. Nešto kasnije ćemo dati jedan detaljno urađen primer pomoću kojeg ćemo ilustrovati rad sa najčešće korišćenim kontrolama.

Labela (Label)

Labela služi da prikaže tekst na dijalog box-u. Jednu ovakvu kontrolu smo imali kod input box-a. Tekst “Uneti ime” sa slike 5. odgovara ovoj kontroli.

Tekst box (TextBox)

Ova kontrola dozvoljava korisniku da unese tekst. Jednu ovakvu kontrolu smo imali kod input box-a sa slike 5.

Kombo box (ComboBox)

Ova kontrola nam daje padajući meni sa kojeg možemo izabrati jednu opciju ili nam dopušta da sami unesemo podatak. Na primer, pri snimanju fajla opcijom Save As, polja za unos imena i tipa fajla su kombo box-ovi.

List box (ListBox)

List box kontrola nam daje listu stavki sa koje korisnik može izabrati jednu ili više stavki. Ova kontrola je vrlo slična kombo box-u, s tim što kombo daje padajući meni za odabir jedne stavke. Druga razlika je da kod kombo box-a korisnik može uneti stavku koja se ne nalazi na listi.

Ček box (CheckBox)

Ček box kontrola se upotrebljava kada želimo da definišemo neki binaran izbor, da ili ne, istinito ili neistinito, uključeno ili isključeno. Ova kontrola ima dve moguće vrednosti, True ili False, koje respektivno znače da je kontrola čekirana (štiklirana) ili nije čekirana.

Opciono dugme (OptionButton)

Opciona dugmad se koriste kada korisnik treba da odabere jednu od nekoliko opcija. Opciona dugmad se nalaze u grupama od po minimum dva dugmeta. Kada se jedno dugme u grupi

aktivira ostala se deaktiviraju. Ukoliko na dijalog box-u treba da se nađe više od jedne grupe opcionih dugmadi, svaka grupa dugmadi mora različitu `GroupName` osobinu, inače će se sva dugmad tretirati kao da pripadaju istoj grupi. Drugi način je da se grupe opcionih dugmadi odvoje pomoću okvira koji automatski grupiše dugmad.

Toggle dugme (**ToggleButton**)

Slično kao ček box i opciona dugmad, ovo dugme ima dva moguća stanja, uključeno i isključeno. Klik na dugme menja trenutno stanje u ono drugo i menja izgled dugmeta. Vrednost dugmeta je True (pritisnuto) ili False (nije pritisnuto).

Okvir (**Frame**)

Okvir služi da obuhvati druge kontrole, bilo iz estetskih razloga, bilo zbog grupisanja kontrola na logičan način. Ova kontrola može biti vrlo korisna u grupisanju opcionih dugmadi kada naš dijalog box ima više od jedne grupe ovih dugmadi.

Komandno dugme (**CommandButton**)

Ovo dugme služi da izvrši određenu radnju koju progamiramo u okviru događaja `Click` ovog dugmeta. Ova kontrola se najviše koristi i svaki dijalog box kojeg kreiramo će imati bar jedno komandno dugme. Na primer, message box sa slike 4. ima tri komandna dugmeta.

Tab strip (**TabStrip**)

Ova kontrola služi za kreiranje dijalog box-ova sa više tabova (stranica). Jedan takav dijalog box se dobija opcijom **Tools⇒Options**. Ova kontrola, podrazumevano, ima 2 stranice. Dodavanje novih stranica i brisanje postojećih se vrši sa padajućeg menija koji se dobija desnim klikom na kontrolu.

Multi page (**MultiPage**)

Slično kao tab strip, multi page kontrola kreira dijalog box-ove sa više stranica. Ova kontrola pruža više mogućnosti i jednostavnija je za rad sa dijalog box-ovima od tab strip kontrole. Za razliku od multi page kontrole, tab strip ne služi kao kontejner za druge objekte.

Klizač (**ScrollBar**)

Klizač je kontrola pomoću koje, na jednostavan način, možemo odabrati jednu od vrednosti iz određenog opsega. Odabir se vrši prevlačenjem klizača.

Spin dugme (**Spin Button**)

Spin dugme omogućava odabir vrednosti kontrole klikanjem jedne od dve strelice, pri čemu jedna povećava, a druga smanjuje vrednost. Ova kontrola je slična klizaču, pri čemu kod klizača prevlačenjem možemo postići veće korake promene. Spin dugme se obično koristi zajedno sa tekst box-om ili labelom koji prikazuju tekuću vrednost kontrole.

Image kontrola (**Image control**)

Ova kontrola se koristi za prikaz slike koja može biti zasebni fajl ili se uzeti sa Clipboard-a. Nakon smeštanja u kontrolu, slika postaje deo odgovarajuće radne sveske.

RefEdit kontrola (**RefEdit**)

RefEdit kontrola se koristi za odabir opsega u radnom listu.

Pozivanje korisničke forme sa palete alatki

Vrlo efektan i korisniku blizak načina pozivanja kreirane forme je preko odgovarajućeg dugmeta sa palete alatki. U tu svrhu, možemo kreirati sopstvene palete alatki i sa njih pozivati našu dijalog box, ili ga možemo pozivati sa neke od postojećih paleta alatki. Ovde ćemo ilustrovati oba načina.

Za oba načina je zajedničko da se dodavanje dugmeta koje aktivira formu najčešće vrši pri otvaranju radne sveske, tj. u okviru procedure `Workbook_Open`.

Dodavanje dugmeta postojećoj paleti alatki (Standard)

Dugme koje se dodaje paleti alatki je VBA objekt `CommandBarButton`. Deklaracija dugmeta i dodavanje Standard paleti alatki se vrši na sledeći način:

```
Dim Dugme As CommandBarButton
Set Dugme = Application.CommandBars("Standard").Controls.Add( _
         Type:=msoControlButton, Temporary:=True)
```

Dodavanje kontrole se vrši metodom `Add` kolekcije `Controls`. Argument `Type` ove metode određuje tip kontrole koju dodajemo paleti alatki (u našem slučaju `msoControlButton`), dok argument `Temporary` određuje da li se kontrola automatski briše nakon zatvaranja aplikacije (True) ili ne (False). Podrazumevana vrednost je False.

Nakon dodavanja dugmeta kontroli, treba da definišemo osobine dugmeta. Na primer, konstrukcija

```
With Dugme
     .Style = msoButtonCaption
     .Caption = "Dugme"
     .OnAction = "Procedura"
End With
```

bi definisala dugme samo sa tekstom (osobina `Style`), tekst koji će biti ispisan na dugmetu (osobina `Caption`), kao i koja se procedura poziva pritiskom na dugme (osobina `OnAction`). Ime procedure se navodi u obliku stringa.

Ako želimo da, umesto teksta, definišemo ikonicu koja će biti prikazana na dugmetu, imali bi konstrukciju sličnu sledećoj:

```
With Dugme
     .Style = msoButtonIcon
     .Picture = LoadPicture("C:\Temp:\Slika.bmp")
     .OnAction = "Procedura"
End With
```

Dozvoljena je i kombinacija teksta i ikonice na dugmetu, kada je stil kontrole `msoButtonIconAndCaption`.

Procedura koju poziva kreirano dugme treba da sadrži naredbu za prikaz korisničke forme (prisetite se metode `Show`) i treba da se nalazi u standardnom modulu.

Kreiranje nove palete alatki

Paleta koja se dodaje je VBA objekt `CommandBar`. Dodavanje palete se vrši metodom `Add` kolekcije `CommandBars`.

```
Dim Paleta As CommandBar
Set Paleta = Application.CommandBars.Add(Name:="NasaPaleta", _
```

```
        Position:=msoBarTop, Temporary:=True)
Paleta.Visible = True
```

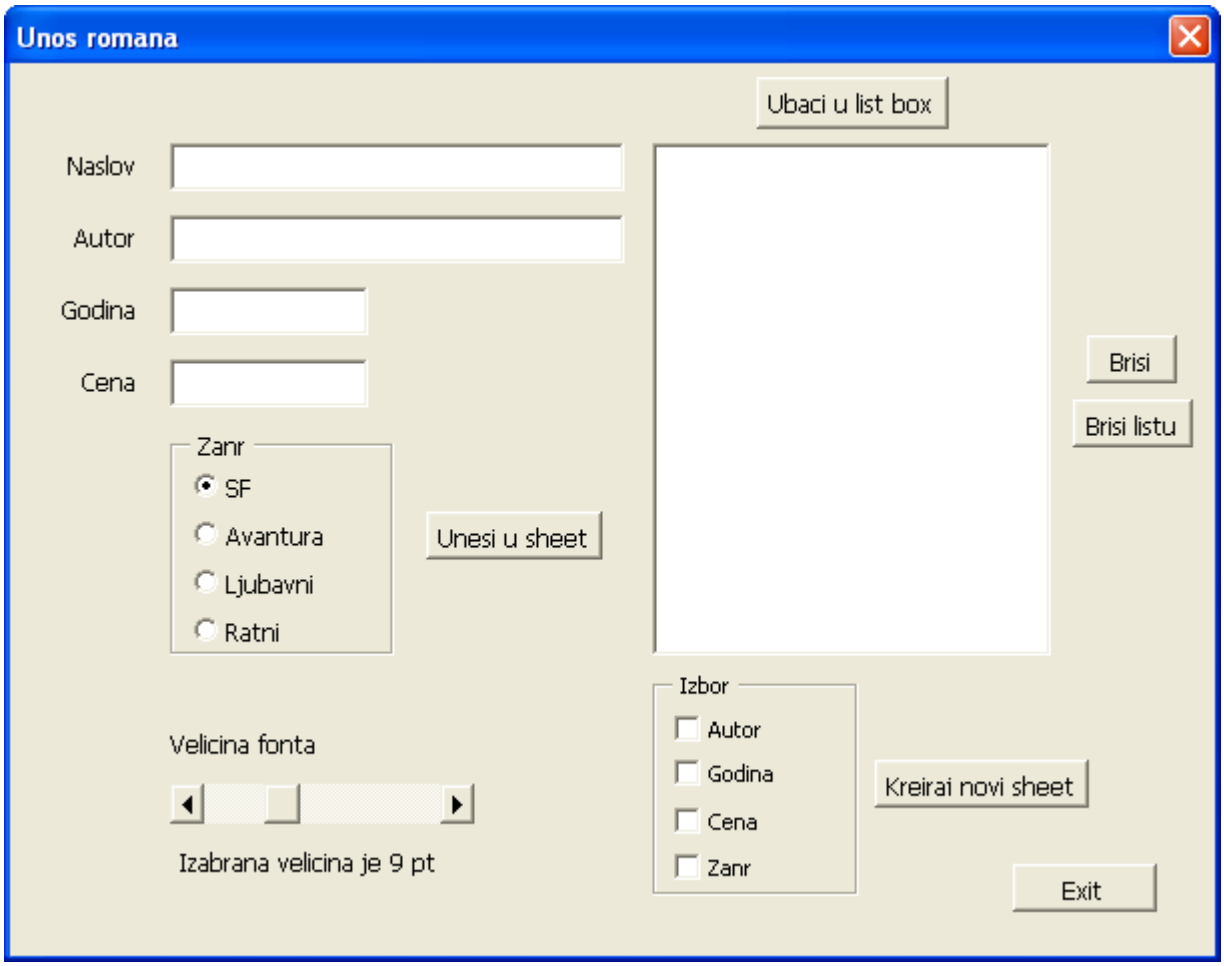
Argument `Name` definiše ime palete. Argument `Position` definiše poziciju palete u odnosu na prozor aplikacije. Konstantom `msoBarTop` se paleta smešta na vrh prozora, sa drugim paletama alatki. Za ostale konstante pogledajte dokumentaciju VBA. Argument `Temporary` određuje da li se paleta automatski briše nakon zatvaranja aplikacije (True) ili ne (False). Podrazumevana vrednost je False.

Kreirana paleta podrazumevano nije vidljiva. Da bi je učinili vidljivom, potrebno je podesiti njenu osobinu `Visible` na True.

Dodavanje dugmeta na paletu `"NasaPaleta"` se vrši na način pokazan u prethodnom poglavlju, s tim što se kreirana paleta referencira sa `CommandBars("NasaPaleta")`. Ostalo je isto.

PRIMER:

Napravimo korisničku formu prikazanu na slici 7.



Slika 7. Korisnička forma za unos podataka o romanima.

Pomoću ove forme ćemo unositi podatke o romanima radnil list Romani. Od podataka koje unosimo imamo naslov romana, autora, godinu izdanja, cenu i žanr. Prva četiri podatka se unose preko odgovarajućih tekst box-ova, dok se žanr unosi preko opcionih dugmadi. Dugme *Unesi u sheet* služi da unese podatke u list. Naslove svih postojećih romana možemo prebaciti u list box, a kasnije, na osnovu podataka iz list box-a, može kreirati novi radni list sa imenima romana i

podacima po želji (ček box-ovi ispod list box-a). Forma se zatvara pritiskom dugmeta *Exit*. Dodatna funkcionalnost forme je mogućnost promene veličine fonta u tabeli sa romanima od 9 do 16 pt.

Dajemo ispod kompletan kod, a nakon toga i objašenjenje pojedinih elemenata.

```
Private Sub Workbook_Open()
Dim Dugme As CommandBarButton
Set Dugme = Application.CommandBars("Standard").Controls.Add( _
Type:=msoControlButton, Temporary:=True)

With Dugme
    .Style = msoButtonCaption
    .Caption = "Romani"
    .OnAction = "Romani"
End With
End Sub
```

```
Sub Romani()
Dim I As Integer
FormRomani.Show vbModeless
FormRomani.TextNaslov.SetFocus
I = 1
Do While ActiveSheet.Cells(I, 1) <> ""
    I = I + 1
Loop
ActiveSheet.Cells(I, 1).Activate
End Sub
```

```
Private Sub CommandBrisiElement_Click()
If Me.ListBoxRomani.ListIndex <> -1 Then
    Me.ListBoxRomani.RemoveItem Me.ListBoxRomani.ListIndex
End If
End Sub
```

```
Private Sub CommandBrisiListu_Click()
Me.ListBoxRomani.Clear
End Sub
```

```
Private Sub CommandExit_Click()
Unload Me
End Sub
```

```
Private Sub CommandFormirajList_Click()
Dim I As Integer
Call CommandBrisiListu_Click
ActiveSheet.Range("A2").Activate
I = 2
Do While Cells(I, 1) <> ""
    Me.ListBoxRomani.AddItem Cells(I, 1).Value
    I = I + 1
Loop
```

```
End Sub
```

```
Private Sub CommandKreirajNoviSheet_Click()
Dim I As Integer, J As Integer, K As Integer
Dim NoviS As Worksheet, RomOpseg As Range
Set NoviS = Worksheets.Add(after:=Worksheets(Worksheets.Count))
NoviS.Name = "Romani" & Worksheets.Count
NoviS.Cells(1, 1).Value = "Naslov romana"
K = 1
If Me.CheckBoxAutor Then
    K = K + 1
    NoviS.Cells(1, K).Value = "Autor"
End If
If Me.CheckBoxGodina Then
    K = K + 1
    NoviS.Cells(1, K).Value = "Godina izdanja"
End If
If Me.CheckBoxCena Then
    K = K + 1
    NoviS.Cells(1, K).Value = "Cena"
End If
If Me.CheckBoxZanr Then
    K = K + 1
    NoviS.Cells(1, K).Value = ChrW(381) + "anr"
End If
Set RomOpseg = Worksheets("Romani").Range("A1").CurrentRegion
For I = 0 To Me.ListBoxRomani.ListCount - 1
    K = 1
    For J = 2 To RomOpseg.Rows.Count
        If Me.ListBoxRomani.List(I) = RomOpseg.Cells(J, K) Then
            K = 1
            NoviS.Cells(I + 2, K).Value = RomOpseg.Cells(J, K).Value
            If Me.CheckBoxAutor Then
                K = K + 1
                NoviS.Cells(I + 2, K).Value = RomOpseg.Cells(J, 2).Value
            End If
            If Me.CheckBoxGodina Then
                K = K + 1
                NoviS.Cells(I + 2, K).Value = RomOpseg.Cells(J, 3).Value
            End If
            If Me.CheckBoxCena Then
                K = K + 1
                NoviS.Cells(I + 2, K).Value = RomOpseg.Cells(J, 4).Value
            End If
            If Me.CheckBoxZanr Then
                K = K + 1
                NoviS.Cells(I + 2, K).Value = RomOpseg.Cells(J, 5).Value
            End If
        End If
    Next
Next
End Sub
```

```
Private Sub CommandUnesi_Click()
Dim Odgovor As Integer
```

```
If Me.TextNaslov = "" Then
    MsgBox "Mora se uneti naslov romana!", vbOKOnly, "Pogresan unos"
    Exit Sub
End If
Odgovor = vbYes
If Me.TextAutor = "" Or Me.TextGodina = "" Or Me.TextCena = "" Then
    Odgovor = MsgBox("Niste uneli sva polja." & vbCrLf & _
    "Zelite li da nastavite?", vbYesNo, "Nepotpun unos")
End If
If Odgovor = vbYes Then
    ActiveCell.Cells(1, 1).Value = Me.TextNaslov.Text
    ActiveCell.Cells(1, 2).Value = Me.TextAutor.Text
    ActiveCell.Cells(1, 3).Value = Me.TextGodina.Text
    ActiveCell.Cells(1, 4).Value = Me.TextCena.Text
    If Me.OptionAvantura Then ActiveCell.Cells(1, 5).Value = "Avantura"
    If Me.OptionLjubavni Then ActiveCell.Cells(1, 5).Value = "Ljubavni"
    If Me.OptionRatni Then ActiveCell.Cells(1, 5).Value = "Ratni"
    If Me.OptionSF Then ActiveCell.Cells(1, 5).Value = "SF"
    ActiveCell.Cells(2, 1).Activate
    Me.TextNaslov.Text = ""
    Me.TextAutor.Text = ""
    Me.TextGodina.Text = ""
    Me.TextCena.Text = ""
    Me.TextNaslov.SetFocus
Else
    If Me.TextNaslov = "" Then
        Me.TextNaslov.SetFocus
    ElseIf Me.TextAutor = "" Then
        Me.TextAutor.SetFocus
    Else
        Me.TextCena.SetFocus
    End If
End If
End Sub
```

```
Private Sub ScrollFont_Change()
Me.LabelFontSize2.Caption = "Izabrana velicina je " & _
     ScrollFont.Value & " pt"
Range("A1").CurrentRegion.Font.Size = ScrollFont.Value
End Sub
```

U funkciji `Workbook_Open` smo dodali dugme Romani standardnoj paleti alatki i podesili da se klikom na to dugme aktivara procedura `Romani`, koja je definisana u okviru standardnog VBA modula. U okviru procedure `Romani` se prikazuje korisnička forma i inicijalizuju opciona dugmad i klizač `ScrollFont`. Početna vrednost klizača kojim se definiše veličina fonta u tabeli sa romanima je 9, i klikom na strelice se menja vrednost klizača za 1. Sa klizačem je povezana i labela `LabelFontSize2`, koja prikazuje trenutnu vrednost veličine fonta. Sa `Do While` petljom se pozicioniramo u prvu slobodnu ćeliju za upis. Uočimo korišćenje metode `SetFocus` na tekst box-u `TextNaslov`. Pomoću ove metode se postiže da se, nakon startovanja korisničke forme, kursor nalazi u ovom tekst box-u.

Unos podataka se vrši preko tekst box-ova `TextNaslov`, `TextAutor`, `TextGodina` i `TextCena`. Pomoću dugmeta *Unesi u sheet* (`CommandUnesi`) se upisuju podaci za unetu knjigu. Ukoliko ime knjige nije definisano javlja se poruka o greški prilikom unosa, dok ako nije definisan neki drugi podatak, korisnik se pomoću message box-a pita da li želi da unese

nepotpune podatke. Pomoću grupe opcionih dugmadi `OptionAvantura`, `OptionLjubavni`, `OptionRatni` i `OptionSF` se definiše žanr romana.

Naslovi romana iz radnog lista se pomoću dugmeta *Ubaci u list box* (`CommandFormirajList`) mogu ubaciti u list box. Dodavanje elemenata u list box se vrši pomoću metode `Additem`. Pre ubacivanja elemenata u listu vrši se brisanje čitave liste pozivanjem procedure `CommandBrisiListu_Click`, koja pomoću metode `Clear` briše čitav list box.

Brisanje pojedinačnih elemenata iz list box-a se vrši dugmetom *Brisi* (`CommandBrisiElement`) u kome se poziva metoda `RemoveItem` za brisanje elementa. Argument ove procedure je redni broj elementa, pri čemu redni brojevi počinju od nule. Briše se selektovani element, a ukoliko nijedan element nije selektovana, osobina `ListIndex` vraća -1.

Pomoću dugmeta *Kreiraj novi sheet* (`CommandKreirajNoviSheet`) se kreira novi radni list u koji se upisuju samo oni romani koji se trenutno nalaze u list box-u. Šta će osim naslova knjiga biti upisano zavisi od toga koji smo ček box štiklirali. Protumačite sami ovu proceduru.

Konačno, zatvaranje forme vrši dugme *Exit* (`CommandExit`) koje koristi naredbu `Unload`.

P R O G R A M I R A N J E
K R O Z
A P L I K A C I J E

Doc. dr Đukanović Slobodan

ŠESTI TERMIN

GetOpenFilename metod

Ovaj metod se koristi kad od korisnika tražimo ime nekog fajla, uključujući i put do njega. `GetOpenFilename` je metod objekta `Application` i prikazuje klasičan Open dijalog prozor, kao onaj što se dobija opcijom **File⇒Open**, ali ne otvara odabrani fajl, već samo vraća ime fajla. Sintaksa ovog metoda je

```
Ime = Application.GetOpenFilename(FileFilter, FilterIndex, Title,
ButtonText, MultiSelect)
```

gde je

- `FileFilter` – string koji određuje tipove fajlova koji će biti prikazani u padajućoj listi *Files of type* prozora Open. Zove se još i filter fajlova;
- `FilterIndex` – opcija iz padajućeg menija koja će biti podrazumevano prikazana;
- `Title` – naslov dijalog prozora. Ukoliko se izostavi, naslov je „Open“;
- `ButtonText` – samo za Macintosh računare;
- `MultiSelect` – ukoliko je True, može se selektovati viša fajlova u Open prozoru, a ako je False, može se selektovati samo jedan fajl. Podrazumevano je False.

Svi argumenti metoda `GetOpenFilename` su opcioni.

Od ovih argumenata, reći ćemo par reči više o argumentu `FileFilter`. Kao što je rečeno, ovaj argument je string koji će biti prikazan u padajućoj listi *Files of type* prozora Open. Ako, na primer, želimo samo da prikažemo tekstualne fajlove, filter fajlova bi se definisao na sledeći način:

```
NasFilter = "Tekstualni fajlovi (*.txt),*.txt"
```

Generalno, ovaj string ima dva dela, odvojena zarezom. Prvi deo stringa `NasFilter` (`Tekstualni fajlovi (*.txt)`) je tekst koji se prikazuje u padajućoj listi *Files of type*, dok drugi deo (`*.txt`) određuje tip fajlova koji će biti prikazani u listi.

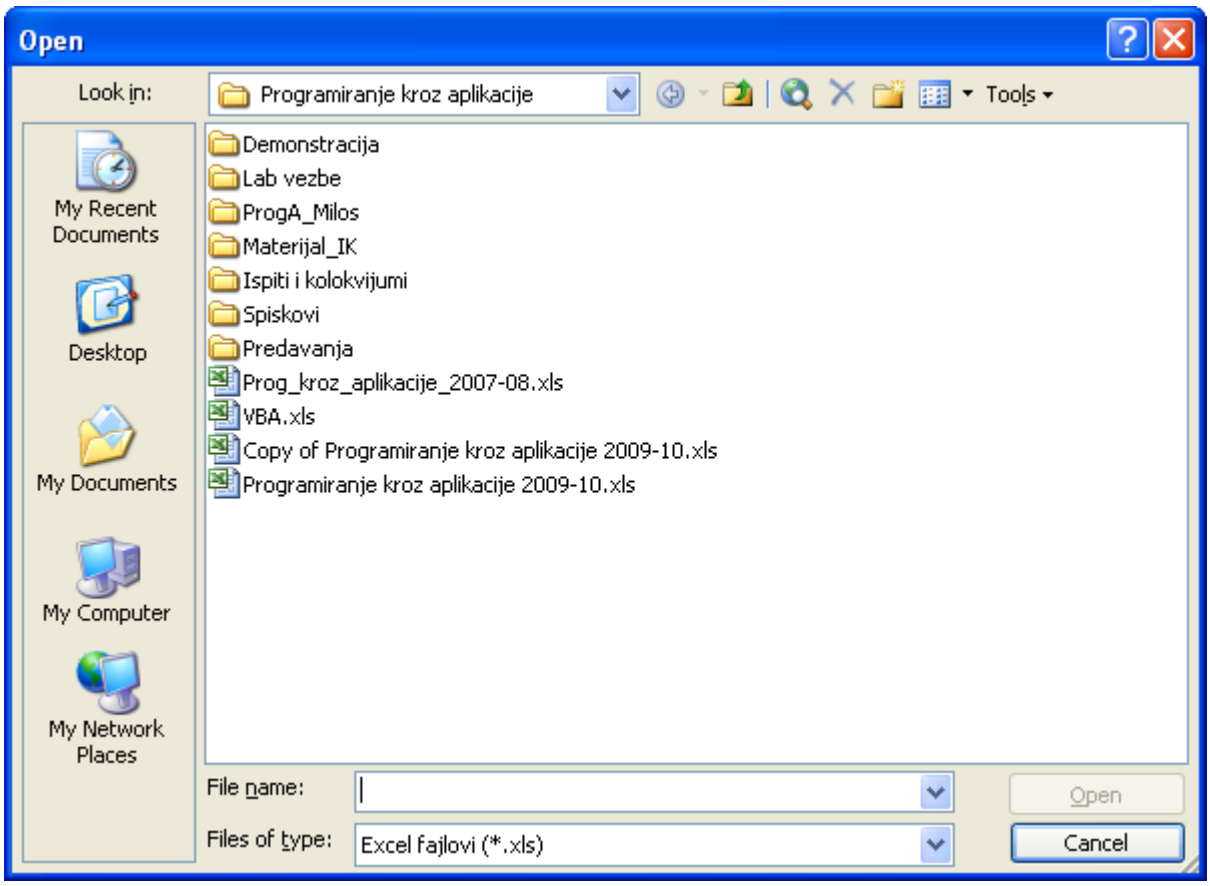
Ukoliko želimo da prikažemo više tipova fajlova u padajućoj listi, onda bi se filter definisao na sledeći način:

```
NasFilter = "Tekstualni fajlovi (*.txt),*.txt," & _
            "Word fajlovi (*.doc),*.doc," & _
            "Excel fajlovi (*.xls),*.xls," & _
            "Svi fajlovi (*.*),*.*"
```

Ispis stringa `NasFilter` u više redova je čisto radi preglednosti. Vidimo da se pojedinačni tipovi fajlova razdvajaju zarezima. Na primer, poziv metoda `GetOpenFilename` sa ovako definisanim filterom fajlova, uz `FilterIndex=3`, tj.

```
Ime = Application.GetOpenFilename(NasFilter,3)
```

bi dao dijalog box prikazan na slici 8.



Slika 8. Primer Open dijalog box-a sa korisnički definisanim filterom fajlova.

Metod `GetOpenFilename` vraća False ako korisnik pritisne dugme Cancel na Open dijalog box-u.

Metod `GetSaveAsFilename` obavlja sličnu operaciju kao i `GetOpenFilename`. On prikazuje Save As dijalog box i čeka da korisnik selektuje fajl. Vraća ime fajla, uključujući i put, i ne vrši nikakvu akciju nad selektovanim fajlom.

Odabir foldera pomoću FileDialog objekta

Metode `GetSaveAsFilename` i `GetOpenFilename` služe da korisnik odabere fajl. Odabir foldera se najjednostavnije vrši pomoću `FileDialog` objekta. Ovaj objekt postoji počev od Excel-a 2002, tako da u ranijim verzijama Excel-a ovakav način odabira foldera neće raditi. Osim foldera, pomoću ovog objekta se mogu selektovati i fajlovi.

Objekt `FileDialog` se dobija pomoću `FileDialog` osobine objekta `Application`. `FileDialog` osobina ima jedan argument, `fileDialogType`, koji definiše tip fajl dijaloga i ima četiri moguće vrednosti date ispod:

`msoFileDialogFilePicker` – dijalog za odabir fajla;

`msoFileDialogFolderPicker` – dijalog za odabir foldera;

`msoFileDialogOpen` – dijalog za otvaranje fajla;

`msoFileDialogSaveAs` – dijalog za snimanje fajla.

U nastavku je dat primer u kome se kreira `FileDialog` objekt i pomoću njega od korisnika traži odabir foldera. Ukoliko je korisnik odabrao neki folder, pomoću message box-a će biti prikazano

ime tog foldera. Ukoliko je korisnik pritisnuo dugme Cancel, biće prikazana poruka "Selekcija otkazana!"

```
Dim FD As FileDialog
Set FD = Application.FileDialog(msoFileDialogFolderPicker)
With FD
    .InitialFileName = "C:\Temp\"
    .Title = "Odaberite folder"
    .Show
End With
If FD.SelectedItems.Count = 0 Then
    MsgBox "Selekcija otkazana!"
Else
    MsgBox FD.SelectedItems(1)
End If
```

Pomoću osobine `InitialFileName` definišemo početni folder, tj. onaj u kome će se otvoriti dijalog boks. Pomoću metode `Show` prikazujemo dijalog box, dok pomoću osobine `SelectedItems` dobijamo ime selektovanog foldera.

Objekt `FileDialog` ima i osobinu `AllowMultiSelect` koja omogućava selekcija više fajlova u dijalog box-u. Ova osobina nema efekat na dijalog box-ove za odabir foldera i snimanje fajlova jer se ove operacije ne mogu izvršiti nad više objekata. Zato jedini argument osobine `AllowMultiSelect` koji ima smisao u prethodnom primeru je 1.

Rad sa fajlovima u VBA

Spisak korisnih VBA komandi za rad sa fajlovima je dat u tabeli 29.

Komanda	Opis
`ChDir`	Promena tekućeg foldera.
`CurDir`	Dobijanje tekućeg foldera.
`ChDrive`	Promena tekućeg drajva.
`Dir`	Dobijanje imena fajla ili foldera koje odgovara određenom obrascu, atributu fajla ili labeli drajva.
`FileCopy`	Kopiranje fajla.
`FileDateTime`	Dobijanje datuma i vremena kad je fajl poslednji put modifikovan.
`FileLen`	Dobijanje veličine fajla, izražene u bajtima.
`GetAttr`	Dobijanje atributa fajla.
`Kill`	Brisanje fajla.
`MkDir`	Kreiranje novog foldera.
`Name`	Promena imena fajla ili foldera.
`RmDir`	Brisanje praznog foldera.
`SetAttr`	Promena atributa fajla.

Tabela 29. VBA komandi za rad sa fajlovima.

U nastavku ćemo reći par reči o funkciji `Dir`. Za ostale funkcije iz prethodne tabele konsultujte Help.

Dir funkcija

Pomoću funkcije `Dir` dobijamo string koji predstavlja ime fajla ili foldera koje odgovara određenom obrascu, atributu fajla ili labeli drajva. Ova funkcija ima dva opciona argumenta i njena sintaksa je:

```
ImeFajla = Dir(pathname, attributes)
```

gde je `pathname` string koji određuje ime fajla (može uključiti i put do njega), dok je `attributes` VBA konstanta ili broj koji određuje atribute fajla. Ukoliko se ovaj argument izostavi, `Dir` vraća fajl koji nema atributa. Atributi fajla su dati u tabeli 30.

Konstanta	Vrednost	Opis
`vbNormal`	0	Određuje fajl bez atributa (podrazumevano).
`vbReadOnly`	1	Određuje read-only fajlove u dodatku fajlova bez atributa.
`vbHidden`	2	Određuje sakrivene fajlove u dodatku fajlova bez atributa.
`VbSystem`	4	Određuje sistemske fajlove u dodatku fajlova bez atributa.
`vbVolume`	8	Određuje oznaku volume-a; ako je bilo koji drugi atribut specificiran, `vbVolume` se ignoriše.
`vbDirectory`	16	Određuje foldere u dodatku fajlova bez atributa.
`vbAlias`	64	Specificirano ime fajla je alias. Samo za Macintosh.

Tabela 30. Atributi fajla za funkciju Dir.

Ukoliko `Dir` ne pronađe traženi fajl, vraća prazan string. Ova se osobina može iskoristiti za proveru da li postoji traženi fajl.

Za specifikovanje više fajlova, u funkciji `Dir` se mogu koristiti džoker karakteri ***** (menja proizvoljan broj karaktera) i **?** (menja jedan karakter). Ovo radi samo u Microsoft Windows okruženju.

Da bi obišli sve fajlove u folderu, ili sve fajlove određenog tipa, potrebno je koristiti oblik funkcije `Dir` bez argumenata, pri čemu se pri prvom pozivu funkcije `Dir` mora specifirati put, inače će doći do greške. Kada prođe kroz sve fajlove u folderu, `Dir` vraća prazan string. U narednom pozivu funkcije `Dir` se mora navesti put da ne bi došlo do greške. Takođe, put se može promeniti i kad Dir ne dođe do poslednjeg fajla u datom folderu, tj. ne mora se čekati da se obiđu svi fajlovi.

U nastavku dajemo dva primera rada sa funkcijom `Dir`.

Primer 1: U ovo primeru ćemo izlistati sve fajlove iz određenog foldera u aktivni radni list.

```
Sub ListanjeFajlova()
Dim I As Integer, Fajl as String
Folder = "C:\Temp\"
I = 1
With ActiveSheet
    .Cells(I, 1) = "Ime fajla"
    .Cells(I, 2) = "Velicina (bytes)"
    .Cells(I, 3) = "Datum / Vreme"
End With
Range("A1:C1").Font.Bold = True
Fajl = Dir(Folder)     ' Uzimanje prvog fajla iz foldera Folder
Do While Fajl <> ""
```

```
    I = I + 1
    With ActiveSheet
        .Cells(I, 1) = Fajl
        .Cells(I, 2) = FileLen(Folder & Fajl)
        .Cells(I, 3) = FileDateTime(Folder & Fajl)
    End With
    Fajl = Dir         ' Uzimanje sledeceg fajla
Loop
End Sub
```

U prvom pozivu funkcije `Dir` navodimo put do foldera iz kojeg listamo fajlove, dok u svakom narednom pozivu samo koristimo zapis `f = Dir` kojim se vraća ime sledećeg fajla. Pomoću funkcija `FileLen` i `FileDateTime` dobijamo veličinu fajla i vreme njegove poslednje modifikacije, respektivno.

Ovde smo izlistali sve fajlove koji se nalaze u datom folderu. Ukoliko, na primer, želimo da listamo samo Word dokumente, potrebno je u prvom pozivu funkcije `Dir` to naglasiti na sledeći način:

```
f = Dir(Folder & "*.doc")
```

Primer 2: U ovom primeru ćemo obrisati folder koji je odabran pomoću FileDialog objekta. Za brisanje foldera se može koristiti naredba `RmDir`. Međutim, ova naredba briše samo prazne foldere, pa je pre brisanja foldera potrebno obrisati sve fajlove iz foldera, što se radi naredbom `Kill`. Procedura je data ispod.

```
Sub BrisanjeFajlova()
Dim Fajl As String, FD As FileDialog
Set FD = Application.FileDialog(msoFileDialogFolderPicker)
FD.Title = "Odaberite folder"
FD.Show
Fajl = Dir(FD.SelectedItems(1) & "\*.*")
Do While Fajl <> ""
    Kill FD.SelectedItems(1) & "\" & Fajl
    Fajl = Dir
Loop
ChDir ".."
RmDir FD.SelectedItems(1)
End Sub
```

Nadamo se da je ova procedura jasna. Napomenimo da se pomoću `RmDir` ne može obrisati folder u kom se trenutno nalazimo. Stoga smo pre naredbe `RmDir` promenili tekući folder pomoću naredbe `ChDir`. Argument `".."` ove naredbe znači da se pomeramo jedan folder gore u hijerarhiji, tj. tekući folder postaje onaj koji sadrži folder koji se briše.

FileSearch objekt

Osobina `FileSearch` objekta `Application` vraća `FileSearch` objekt. Ovaj objekt omogućuje pretragu fajlova, slično Windows-ovoj aplikacije za pretragu fajlova. Fajl se može tražiti prema datoj specifikaciji, ili čak po reči ili frazi koju sadrži. `FileSearch` objekt se koristi od Excel-a 97 nadalje.

Neke od ključnih metoda i osobina objekta `FileSearch` su date u tabeli 31.

Osobina ili metod	Opis
`FileName`	Osobina koja daje ime traženog fajla, pri čemu se mogu koristiti džoker

	karakteri.
`FoundFiles`	Osobina koja vraća `FoundFiles` objekt koji sadrži imena fajlova pronađenih tokom pretrage.
`LookIn`	Osobina kojom se specificira folder u kojem se vrši pretraga.
`SearchSubfolders`	Osobina kojom se specificira da li želimo da pretražujemo podfoldere foldera u kojem se vrši pretraga. `True` uključuje i podfoldere.
`Execute`	Metod koji vrši pretragu.
`NewSearch`	Metod koji resetuje `FileSearch` objekt.
`TextOrProperty`	Osobina koja vraća ili menja reč ili frazu koja se traži. Pretraga može uključiti i džoker karaktere.

Tabela 31. Neke od metoda i osobina objekta FileSearch.

U nastavku dajemo proceduru `PretragaFajlova` koja traži .doc fajlove koji sadrže reč "VBA" u folder C:\Temp. Pretraživaćemo samo fajlove koji su modifikovani tokom ovog meseca (osobina `LastModified`). Ukoliko nije pronađen nijedan fajl, potrebno je ispisati poruku `"Nije pronadjen nijedan fajl"`.

```
Sub PretragaFajlova()
Dim FS As FileSearch
Set FS = Application.FileSearch
ActiveSheet.Cells(1, 1) = "Ime fajla"
Range("A1").Font.Bold = True
With FS
    .NewSearch
    .LookIn = "C:\Temp"
    .SearchSubFolders = True
    .TextOrProperty = "VBA"
    .MatchTextExactly = False
    .Filename = "*.doc"
    .LastModified = msoLastModifiedThisMonth
    .Execute
End With
If FS.FoundFiles.Count = 0 Then
    ActiveSheet.Cells(2, 1) = "Nije pronadjen nijedan fajl"
Else
    For I = 1 To FS.FoundFiles.Count
        ActiveSheet.Cells(I + 1, 1) = FS.FoundFiles(I)
    Next I
End If
End Sub
```

Metod Execute treba pozivati nakon definisanja svih kriterijuma pretrage. Osobina `Count` objekta `FoundFiles` daje broj pronađenih fajlova. Ukoliko nije pronađen nijedan fajl, ova osobina ima vrednost 0.

Rad sa tekstualnim fajlovima

Kao i drugi programski jezici, VBA podržava rad sa tekstualnim fajlovima. Naredbe postoje u VBA omogućuju elegantan i moćan rad sa tekstualnim fajlovima.

Najčešći pristup tekstualnom fajlu je tzv. *sekvencijalni pristup*, koji omogućava čitanje i upis pojedinačnih karaktera i čitavih linija teksta. Drugim rečima, sekvencijalni pristup znači da čitanje fajla počinje na početku fajla i linije se čitaju redom.

Druga dva načina pristupa tekstualnim fajlovima su *slučajni* i *binarni*, i njih nećemo obrađivati.

Otvaranje fajla

Pre čitanja iz fajla, ili upisa u njega, fajl se mora otvoriti. Za otvaranje fajla se koristi VBA naredba `Open` (isto kao i metod `Open` objekta `Application`, stoga voditi računa).

Naredba `Open` ima malo složeniju sintaksu, a mi ćemo ovde navesti nama koristan pojednostavljeni oblik:

```
Open pathname For mode As #filenumber
```

gde je

- `pathname` – obavezan argument koji definiše ime fajla, zajedno sa putom;
- `mode` – obavezan argument koji definiše način otvaranja fajla i može biti (navodimo samo nama korisne opcije):
  - `Append` – sekvencijalan pristup koji omogućava čitanje fajla ili dopisivanje teksta na kraj fajla;
  - `Input` – sekvencijalan pristup koji omogućava čitanje fajla, bez mogućnosti upisa;
  - `Output` – sekvencijalan pristup koji omogućava čitanje i upis u fajl. Na ovaj način se kreiraju novi fajlovi, dok se brišu postojeći fajlovi sa istim imenom.
- `filenumber` - obavezan argument koji predstavlja broj fajla (broj između 1 i 511); još se naziva i *handle fajla*. Radi dobijanja broja fajla koji je dostupan (tj. koji nije u upotrebi), može se koristiti funkcija `FreeFile`.

Upotreba karaktera # ispred broja fajla je opciona, ali ga je poželjno koristiti radi poboljšanja preglednosti naredbe.

Na primer, naredbom

```
Open "C:\Temp\Fajl.txt" For Input As #1
```

se otvara Fajl.txt radi čitanja. Ukoliko ne znamo broj fajla koji je dostupan, možemo koristiti sledeći kod:

```
Dim Dostupan As Integer
Dostupan = FreeFile
Open "C:\Temp\Fajl.txt" For Input As #Dostupan
```

Zatvaranje fajla

Fajl koji je otvoren naredbom `Open` se zatvara naredbom `Close`, u sledećem zapisu:

```
Close #filenumber
```

Ovom se naredbom može zatvoriti više fajlova, na sledeći način:

```
Close #filenumber1, #filenumber2, ..., #filenumberN
```

Ukoliko se navede samo naredba `Close`, bez navođenja broja fajlova, zatvaraju se svi trenutno otvoreni fajlovi. Svi fajlovi otvoreni naredbom `Open` se alternativno mogu zatvoriti naredbom `Reset`.

Čitanje iz tekstualnog fajla

Kod VBA postoje 3 naredbe za čitanje podataka iz tekstualnog fajla:

- `Input` – funkcija za čitanje određenog broja karaktera iz fajla;
- `Input #` – čitanje podataka iz fajla i dodeljivanje tih podataka promenljivim koje su odvojene zarezima;
- `Line Input #` – čitanje čitave linije podataka.

Funkcija `Input` čita određeni broj karaktera iz specificiranog fajla i vraća string pročitanih karaktera. Poziva se na sledeći način:

```
Procitano = Input(number, #filenumber)
```

gde je `number` broj karaktera koji se čita, a `filenumber` je handle za dati fajl. Na primer, sekvenca naredbi

```
Open "C:\temp\Fajl.txt" For Input As #1
Procitano = Input(12, #1)
```

će pročitati prvih 12 karaktera iz fajla Fajl.txt. U nastavku je data procedura `BrojCifaraUFajlu` kojom se broji koliko ima cifara u fajlu Fajl.txt. Dobijeni broj se prikazuje pomoću MsgBox-a.

```
Sub BrojCifaraUFajlu()
Dim BrCif As Integer, Procitano As String
Open "C:\temp\Fajl.txt" For Input As #1
BrCif = 0
Do While Not EOF(1)
    Procitano = Input(1, #1)
    If Procitano Like "[0-9]" Then
        BrCif = BrCif + 1
    End If
Loop
MsgBox "Ima " & BrCif & " cifara"
Close #1
End Sub
```

U prethodnoj proceduri je korišćena VBA funkcija `EOF` koja vraća True kada se dođe do kraja fajla i False u suprotnom. Argument ove funkcije je handle na otvoreni fajl, naveden bez karaktera #. Ukoliko ne bismo koristili ovu funkciju došlo bi do greške prilikom pokušaja čitanja nakon poslednjeg karaktera u fajlu. Ovo važi kod sve tri naredbe za čitanje iz fajla.

Čitanje čitavog fajla odjednom se vrši na sledeći način:

```
Procitano = Input(LOF(filenumber), #filenumber)
```

gde je `LOF` funkcija koja vraća broj bajta u fajlu, što je u slučaju tekstualnog fajla jednako broju karaktera.

Naredba `Input #` je pogodna ukoliko svaki red fajla sadrži fiksan broj podataka i kad je potrebno raditi sa tim podacima pojedinačno. Ovu naredbu ćemo ilustrovati na sledećem, nama bliskom primeru. Pretpostavimo da svaki red fajla Ispit.txt sadrži ime i prezime studenta i broj poena koje je taj student dobio na ispitu. Ime i prezime su zadati u formi jednog stringa pod znacima navoda, dok je broj poena realan broj. Student je položio ispit ukoliko je dobio više od 50 poena. Procedura `Polozili` određuje koliko je studenata položilo ispit i taj broj prikazuje pomoću MsgBox-a.

```
Sub Polozili()
Dim ImePrez As String, Poeni As Single, BrPol As Integer
Open "C:\Temp\Ispit.txt" For Input As #1
BrPol = 0
Do While Not EOF(1)
    Input #1, ImePrez, Poeni
    If Poeni > 50 Then BrPol = BrPol + 1
Loop
MsgBox "Polozilo je " & BrPol & " studenata"
Close #1
End Sub
```

Ime i prezime se ne moraju navesti pod znacima navoda, ali se onda obavezno moraju odvojiti zarezima.

Naredba `Line Input #` čita jednu liniju teksta (do znaka za novi red) i pročitanu liniju smešta u String promenljivu. Sintaksa ove naredbe je

```
Line Input #filenumber, varname
```

gde je `varname` ime promenljive u koju smeštamo pročitanu liniju.

Upis u tekstualni fajl

Za sekvencijalni upis u tekstualni fajl koristimo naredbe `Write #` i `Print #`.

Naredba `Write #` služi za upis niza vrednosti. Upisane vrednosti su razdvojene zarezima, dok se stringovi nalaze pod znacima navoda. Podaci upisani u fajl pomoću naredbe `Write #` se obično čitaju naredbom `Input #`. Sintaksa ove naredbe je:

```
Write #filenumber, outputlist
```

gde je `outputlist` lista promenljivih razdvojenih zarezima koje se upisuju u fajl. Naravno, umesto promenljive se može navesti i konstantna vrednost, bilo numerička, stringovna ili Boolean. Štaviše, određeni broj spejsova i tabova se može dodati u `outputlist` funkcijama `Spc` i `Tab`, respektivno. Argument ove dve funkcije je broj spejsova i tabova koje želimo da upišemo u fajl. Ukoliko se `outputlist` izostavi, upisuje se prazna linija u fajl.

Ukoliko želimo da nakon ispisa ostanemo u istom redu, naredbu `Write #` treba završiti tačka-zarezom.

Naredba `Print #` takođe služi za upis niza vrednosti i njena sintaksa je:

```
Print #filenumber, outputlist
```

gde je `outputlist` lista vrednosti koje želimo upisati u fajl. Ukoliko `outputlist` ima više vrednosti, te vrednosti treba razdvajati tačka-zarezom. Ako se razdvajaju zarezima, u fajlu će odgovarajuće vrednosti biti razdvojene tabovima. Tekst je upisan onako kako smo mi naveli, bez dodavanja znaka navoda oko stringova i zareza koji razdvajaju pojedinačne unose. Ukoliko se `outputlist` izostavi, upisuje se prazna linija u fajl. Podaci upisani u fajl naredbom `Print #` se obično čitaju naredbama `Input` i `Line Input #`.

Isto kao kod naredbe `Write #`, ako želimo da nakon ispisa ostanemo u istom redu, naredbu `Print #` treba završiti tačka-zarezom.

Primeri rada sa tekstualnim fajlovima

Primer 1. Iz tekstualnog fajla C:\Temp\Fajl.txt prepisati sve linije koje sadrže reč "VBA" u radni list. Pored pronađene linije, u radni list upisati i redni broj linije u fajlu i njenu dužinu.

```
Sub Podstring()
Dim I As Integer, BrLin As Integer, Linija As String
Open "C:\Temp\Fajl.txt" For Input As #1
Cells(1, 1) = "Linija"
Cells(1, 2) = "Broj linije"
Cells(1, 3) = "Duzina linije"
Range("A1:C1").Font.Bold = True
I = 1
BrLin = 0
Do While Not EOF(1)
    Line Input #1, Linija
    BrLin = BrLin + 1
    If InStr(Linija, "VBA") <> 0 Then
       I = I + 1
       Cells(I, 1) = Linija
       Cells(I, 2) = BrLin
       Cells(I, 3) = Len(Linija)
    End If
Loop
Close #1
End Sub
```

Primer 2. Formirati tekstualni fajl „Vrednosti.txt" u koji su upisane numeričke vrednosti ćelija selektovanog opsega (svaka u zasebnom redu tekstualnog fajla) u formatu:

```
vrsta: I, kolona: J, vrednost: X
```

gde je X vrednost ćelije u I-toj vrsti i J-toj koloni selektovanog opsega. Ne upisivati prazne ćelije i one koje ne sadrže broj. Procedura se startuje na desni klik miša u tekućem radnom listu.

```
Sub Vrednosti()
Dim I As Integer, J As Integer, Cel As Variant
Open "C:\Temp\Vrednosti.txt" For Output As #1
For I = 1 To Selection.Rows.Count
    For J = 1 To Selection.Columns.Count
        Cel = Selection.Cells(I, J)
        If Cel <> "" And IsNumeric(Cel) Then
            Print #1, "vrsta: " & I & ", kolona: " & J & _
                    ", vrednost: " & Cel
        End If
    Next
Next
Close #1
End Sub
```

Primer 3. Napisati proceduru koja formira tekstualni fajl u koji upisuje sledeće podatke vezane za selektovani opseg:

- Ime radne sveske selektovanog opsega
- Ime radnog lista selektovanog opsega
- Adresu selektovanog opsega (bez karaktera $ i :)
- Ukupan broj karaktera u selektovanom opsegu

- Broj slova u selektovanom opsegu
- Broj cifara u selektovanom opsegu.

Tekstualnom fajlu dati ime sledećeg formata:

RadnaSveska_RadniList_AdresaSelektovanogOpsega.txt

Na primer, ako se radna sveska zove **VBA.xls** i selektujemo opseg **C1:D12** u radnom listu **Sheet1**, ime odgovarajućeg tekstualnog fajla treba da bude **VBA.xls_Sheet1_C1D12.txt**.

```
Sub Statistika()
Dim Cel As Range, ImeFajla As String
Dim UK As Integer, Slova As Integer, Cifre As Integer
UK = 0: Slova = 0: Cifre = 0
For Each Cel In Selection
    UK = UK + Len(Cel.Text)
    Slova = Slova + BrojKaraktera(Cel.Text, "Slova")
    Cifre = Cifre + BrojKaraktera(Cel.Text, "Cifre")
Next
ImeFajla = CurDir & "\" & ThisWorkbook.Name & "_" & ActiveSheet.Name
ImeFajla = ImeFajla & "_" & AdrSel(Selection.Address) & ".txt"
Open ImeFajla For Output As #1
Print #1, "Radna sveska: " & ThisWorkbook.Name
Print #1, "Radni list: " & ActiveSheet.Name
Print #1, "Adresa opsega: " & AdrSel(Selection.Address)
Print #1, "Ukupno karaktera: " & UK
Print #1, "Ukupno slova: " & Slova
Print #1, "Ukupno cifara: " & Cifre
Close #1
End Sub
```

```
Function BrojKaraktera(S As String, T As String) As String
Dim I As Integer
BrojKaraktera = 0
If StrComp(T, "Slova", 1) = 0 Then
    For I = 1 To Len(S)
        If Mid(S, I, 1) Like "[a-z]" Or Mid(S, I, 1) Like "[A-Z]" Then
            BrojKaraktera = BrojKaraktera + 1
        End If
    Next
ElseIf StrComp(T, "Cifre", 1) = 0 Then
    For I = 1 To Len(S)
        If Mid(S, I, 1) Like "[0-9]" Then
            BrojKaraktera = BrojKaraktera + 1
        End If
    Next
End If
End Function
```

```
Function AdrSel(S As String) As String
Dim I As Integer
AdrSel = ""
For I = 1 To Len(S)
    If Mid(S, I, 1) <> "$" And Mid(S, I, 1) <> ":" Then
        AdrSel = AdrSel & Mid(S, I, 1)
    End If
```

```
Next
End Function
```

U ovom primeru, funkcija `BrojKaraktera` služi da izbroji slova i cifre u stringu koji predstavlja prvi ulazni argument. Drugi ulazni argument, string `T`, određuje da li brojimo slova ili cifre.

Funkcija `AdrSel` za ulazni argument ima string koji predstavlja adresu opsega, a vraća taj string bez karaktera **$** i **:**. Na primer, ako je ulazni string "$A$3:$C$7", izlazni string će biti "A3C7". Razlog uklanjanja ovih karaktera iz adrese opsega je da se oni ne mogu naći u imenu tekstualnog fajla.

P R O G R A M I R A N J E
K R O Z
A P L I K A C I J E

Doc. dr Đukanović Slobodan

SEDMI TERMIN

Objektni model Word-a je, takođe, hijerarhijski uređen. Na vrhu modela se nalazi objekt `Application`, koji predstavlja sam program, tj. Word. Objekt `Application` sadrži druge objekte, od kojih je najčešće korišćeni objekt `Document`, koji predstavlja otvoreni Word-ov dokument. Krenimo od objekta `Application` i njegovih osobina i metoda.

Objekt Application

U tabelama 32. i 33. su respektivno nabrojane neke od korišćenih osobina i metoda objekta `Application`. Za spisak ostalih osobina i metoda možete konsultovati Help.

Osobina	Opis
`ActiveDocument`	Vraća `Document` objekt koji predstavlja aktivni dokument. Ako nijedan dokument nije otvoren, dešava se greška.
`ActivePrinter`	Vraća ili menja aktivni štampač. Na primer: `Application.ActivePrinter = "HP LaserJet 6 local on LPT1:"`
`CapsLock`	Vraća stanje CapsLock dugmeta; True ako je aktivno, False u suprotnom.
`NumLock`	Vraća stanje NumLock dugmeta; True ako je aktivno, False u suprotnom.
`Caption`	Vraća ili menja tekst u naslovnoj liniji prozora Word-ovih dokumenata. Kod otvorenog dokumenta, ovo je tekst koji dolazi nakon imena dokumenta.
`Documents`	Vraća `Documents` kolekciju koja sadrži sve otvorene dokumente.
`DisplayAlerts`	Vraća i menja mogućnost prikaza Word-ovih alarma i poruka. Ukoliko ne želimo da nas Word prekida sa svojim porukama, u ovu osobinu treba da se upiše False.
`WindowState`	Vraća ili menja stanje prozora određenog dokumenta. Stanje može biti konstanta `wdWindowStateMaximize`, `wdWindowStateNormal` ili `wdWindowStateMinimize`.
`PrintPreview`	Prikazivanje izgleda aktivnog dokumenta pred štampu. Moguće vrednosti su True i False.

Tabela 32. Neke od osobina objekta `Application`.

Metod	Opis
`ChangeFileOpenDirectory`	Određuje folder koji će biti podrazumevano prikazan u Open dijalog box-u (**File⇒Open**). Sintaksa je: `ChangeFileOpenDirectory(Path)` gde je `Path` put do foldera. Na primer, `ChangeFileOpenDirectory "C:\Temp\"`
`Move`	Pomera prozor aktivnog dokumenta. Sintaksa je: `Application.Move(Left, Top)` gde `Left` i `Right` predstavljaju koordinate gornjeg levog ugla prozora, date u pikselima. Greška se javlja ukoliko je prozor dokumenta maksimizovan ili minimizovan. Na primer,

	`Application.Move 100, 100`
`Resize`	Menja veličinu prozora aktivnog dokumenta. Sintaksa je: `Application.Resize(Width, Height)` gde `Width` i `Height` predstavljaju novu širinu i visinu prozora. Greška se javlja ukoliko je prozor dokumenta maksimizovan ili minimizovan. Na primer, `Application.Resize 500, 350`
`CentimetersToPoints`	Pretvaranje centimetara u tačke.
`MillimetersToPoints`	Pretvaranje milimetara u tačke.
`InchesToPoints`	Pretvaranje inči u tačke.
`Quit`	Zatvaranje Word-a. Ukoliko postoji više otvorenih radnih svesaka sa nesnimljenim promenama, Word će pitati da li želimo da snimimo ove promene. Ako ne želimo da nas Word pita za snimanje, možemo snimiti sva dokumenta pre zatvaranja, ili da podesimo osobinu `DisplayAlerts` na False (tada se dokumenti *ne snimaju*!)

Tabela 33. Neke od metoda objekta `Application`.

Document objekti i rad sa njima

U objektnoj hijerarhiji Microsoft Word-a, objekt `Document` dolazi direktno ispod objekta `Application`. VBA nam pruža veliku slobodu u radu sa Word dokumentima, uključujući najčešće operacije kao što su otvaranje postojećih dokumenata, kreiranje novih dokumenata, brisanje dokumenata, snimanje dokumenata pod drugim imenom.

Referenciranje Document objekta

Način pristupanja `Document` objektu "po definiciji" je preko kolekcije `Documents`. Videli smo da osobina `Documents` objekta `Application` vraća kolekciju `Documents`, koja sadrži sve otvorene dokumente. Isto kao kod kolekcije `Worksheets` u Excel-u, argument kolekcije `Documents` u Word-u je naziv dokumenta ili redni broj u kolekciji. Tako, na primer, ako je naš dokument VBA.doc drugi u kolekciji `Documents` (drugi otvoren), njemu se može pristupiti na bilo koji od sledeća dva načina:

```
Documents("VBA.doc")
Documents(2)
```

Aktivni dokument se može referencirati preko `ActiveDocument` objekta, kojeg vraća osobina `ActiveDocument` objekta `Application`.

Konačno, `ThisDocument` objekt predstavlja dokument gde se nalazi VBA kod koji se trenutno izvršava. Ako se naš kod odnosi samo na objekte iz tekućeg dokumenta, onda je zgodnije koristiti objekat `ActiveDocument`. Međutim, ako naš kod radi i sa drugim dokumentima, onda je najjedonostavniji način referenciranja dokumenta koji sadrži taj kod pomoću `ThisDocument` objekta.

Otvaranje dokumenta

Za otvaranje `Document` objekta se koristi metod `Open`. `Open` je metod kolekcije `Documents` i ima 10 argumenata, od kojih je obavezan samo argument koji predstavlja put do dokumenta. Mi ćemo koristiti pojednostavljenu sintaksu ovog metoda gde koristimo samo obavezni argument, tj.:

```
Documents.Open(FileName)
```

gde je `FileName` ime fajla, koje uključuje kompletnu putanju do fajla. Na primer, Word-ov fajl VBA.doc, koji se nalazi u folderu C:\Temp, se otvara na sledeći način:

```
Documents.Open "C:\Temp\VBA.doc"
```

Kreiranje novog dokumenta

Za kreiranje novog Word-ovog dokumenta se koristi `Add`, metod kolekcije `Documents`, na sledeći način:

```
Documents.Add(Template, NewTemplate)
```

gde je

- `Template` – opcioni argument koji određuje šablon (template) fajl na osnovu kojeg se kreira novi dokument. `Template` se navodi kao string koji predstavlja ime .dot fajla, zajedno sa putem do tog fajla. Ukoliko se ovaj argument izostavi, za kreiranje se koristi fajl Normal.dot.
- `NewTemplate` – opcioni argument koji određuje da li kreiramo standardni Word-ov dokument (.doc) ili šablon (.dot). Vrednost True je za .doc fajlove, a False za .dot fajlove.

Osobine i metode objekta Document i kolekcije Documents

U tabelama 34. i 35. su nabrojane neke od korišćenih osobina i metoda objekta `Document` i kolekcije `Documents`, respektivno. Za spisak ostalih osobina i metoda konsultovati Help.

Osobina	Opis
Objekt Document	
`Characters`	Vraća kolekciju `Characters`, koja predstavlja sve karaktera datog dokumenta.
`Sentences`	Vraća kolekciju `Sentences`, koja predstavlja sve rečenice datog dokumenta.
`Tables`	Vraća kolekciju `Tables`, koja predstavlja sve tabele datog dokumenta.
`Comments`	Vraća kolekciju `Comments`, koja predstavlja sve komentare datog dokumenta.
`GrammarChecked`	Vraća True ako je proverena gramatika u datom dokumentu i False u suprotnom.
`SpellingChecked`	Vraća True ako je proveren pravopis u datom dokumentu i False u suprotnom.
`Name`	Vraća ime dokumenta.
`FullName`	Vraća ime dokumenta, uključujući i put do njega.
`Path`	Vraća put do dokumenta. Kod novih, nesnimljenih dokumenata, osobina `Path` je prazan string

	("").
Saved	Vraća podatak o tome da li je bilo promena u dokumentu od trenutka poslednjeg snimanja. Ukoliko promena nije bilo, Saved vraća False.
Content	Vraća Range objekt koji predstavlja sadržaj čitavog dokumenta.
Kolekcija Documents	
Count	Vraća broj trenutno otvorenih dokumenata.

Tabela 34. Neke od osobina objekta Document i kolekcije Documents.

Metod	Opis
Objekt Document	
Activate	Aktiviranje specificiranog dokumenta. Taj dokument postaje ActiveDocument.
CheckGrammar	Provera ispravnosti gramatike u datom dokumentu.
CheckSpelling	Provera ispravnosti pravopisa u datom dokumentu.
Close	Zatvaranje dokumenta. Ovaj metod ima (pojednostavljenu) sintaksu: Close(SaveChanges) gde SaveChanges specificira da li želimo da snimimo promene. Ovaj argument ima tri moguće vrednosti: wdSaveChanges (promene se snimaju pre zatvaranja), wdDoNotSaveChanges (promene se ne snimaju pre zatvaranja) i wdPromptToSaveChanges (korisnik se pita za snimanje promena).

GoTo	Vraća Range objekt koji predstavlja početnu poziciju specificiranog objekta, kao što je stranica, tabela ili bookmark. Sintaksa je:
	GoTo(What, Which, Count, Name)
	What predstavlja tip objekta na koji idemo. Postoji 17 konstanti koje se mogu koristiti u ovu svrhu, a mi ćemo navesti samo najčešće korišćene: wdGoToBookmark, wdGoToComment, wdGoToField, wdGoToFootnote, wdGoToGraphic, wdGoToLine, wdGoToObject, wdGoToPage, wdGoToSection, wdGoToTable.
	Which je konstanta koja određuje kako idemo prema specificiranom objektu. Ovde imamo sledeće konstante: wdGoToAbsolute, wdGoToFirst, wdGoToLast, wdGoToNext, wdGoToPrevious, wdGoToRelative.
	Count određuje redni broj specifiranog objekta.
	Name je ime objekta ukoliko je What jedna od sledećih konstanti wdGoToBookmark, wdGoToComment, wdGoToField ili wdGoToObject.
	Na primer, ukoliko želimo da pređemo na prvu tabelu u aktivnom dokumentu, dovoljno je da izvršimo sledeći deo koda:
	`Dim r As Range`
	`Set r = ActiveDocument.GoTo(What:=wdGoToTable, _`
	`  Which:=wdGoToFirst)`
	`r.Select`
	ili
	`Dim r As Range`
	`Set r = ActiveDocument.GoTo(What:=wdGoToTable, _`
	`  Which:=wdGoToAbsolute, Count:=1)`
	`r.Select`

`PrintOut`	Štampanje specificiranog dokumenta. Metod ima 18 argumenata, a mi ćemo koristiti sledeću pojednostavljenu sintaksu: `PrintOut(Range, From, To, Copies, Pages, ActivePrinter, PrintToFile)` `Range` određuje opseg teksta za štampanje i može biti `wdPrintAllDocument` (čitav dokument), `wdPrintCurrentPage` (tekuća stranica), `wdPrintFromTo` (opseg stranica koji se zadaje preko početne i krajnje stranice), `wdPrintRangeOfPages` (opseg stranica koji može obuhvatiti i nesusedne stranice), `wdPrintPrintSelection` (selekcija). `From` je početna stranica za štampanje ukoliko je Range jednak `wdPrintFromTo`. `To` je krajnja stranica za štampanje ukoliko je Range jednak `wdPrintFromTo`. `Copies` predstavlja broj štampanih kopija. Podrazumevano je 1. `Pages` određuje opseg stranica koji se štampa ukoliko je `Range` jednak `wdPrintRangeOfPages` (na primer, `"1,4-8,10"`). `ActivePrinter` određuje štampač. `PrintToFile` određuje da li Word štampa dokument u fajl (vrednost True) i pita korisnika za ime fajla.
`PrintPreview`	PrintPreview prikaz dokumenta.
`Save`	Snimanje dokumenta. Ukoliko je u pitanju novi dokument, koristiti metod `SaveAs`.
`SaveAs`	Snimanje specificiranog dokumenta pod drugim imenom. Ovaj metod ima (pojednostavljenu) sintaksu: `SaveAs(FileName)` gde `FileName` predstavlja puno ime fajla, uključujući i put do njega.
`Undo`	Vraćanje određenog broja akcija. Sintaksa je `Undo(Times)` gde je `Times` broj akcija koje vraćamo. Metod vraća True ukoliko je operacija uspešno izvršena.
`Redo`	Nanovo izvršavanje operacija koje su vraćene operacijom `Undo`. Sintaksa je `Redo(Times)` gde je `Times` broj akcija koje nanovo izvršavamo. Metod vraća True ukoliko je operacija uspešno izvršena.
`Range`	Vraćanje objekta `Range` koji predstavlja tekst dokumenta određen početnom i krajnjom pozicijom, koje su argumenti metoda. Sintaksa je `Range(Start, End)` gde su `Start` i `End` pozicija početnog i krajnjeg karaktera. Na primer, `Range(1,50)` vraća prvih 50 karaktera dokumenta.
Kolekcija Documents	
`Add`	Kreiranje novog dokumenta, što je već obrađeno.
`Open`	Otvaranje postojećeg dokumenta, što je već obrađeno.

Tabela 35. Neke od metoda objekta `Document` i kolekcije `Documents`.

Događaji na nivou dokumenta

Za razliku od Excel-a, gde imamo mnoštvo događaja vezane za radne sveske i radne listove, kod Word-a je situacija nešto jednostavnija. Tu imamo mali broj događaja vezanih za dokumente. Od ovih događaja, pomenućemo tri najvažnija: `Close`, `Open` i `New`.

Akcija koja okida događaj `Close` je zatvaranje dokumenta, bilo opcijom **File⇒Close** ili pozivom metoda `Close`. Procedura koja se vezana za ovaj događaj je `Document_Close` i ona se izvršava pre nego se dokument zatvori.

Događaj `New` se odnosi samo na šablone i akcija koja okida ovaj događaj je kreiranje novog dokumenta zasnovanog na šablonu ili kad se pozove metod `Add` i specifira se šablon.

Akcija koja okida događaj `Open` je otvaranje dokumenta, bilo opcijom **File⇒Open** ili pozivom metoda `Open`.

Objekti koji predstavljaju tekst u Word-u

Word obiluje objektima koji pružaju mnoštvo načina za rad sa tekstom. Preciznije, tu imamo `Range` objekte, koji predstavljaju proizvoljnu povezanu sekciju teksta, `Characters` objekte pomoću kojih možemo pristupiti pojedinačnim karakterima teksta, `Words` objekte koji omogućavaju rad sa rečima, `Sentences` objekte koji omogućavaju rad sa rečenicama, `Paragraph` objekte koji omogućavaju rad sa pasusima. Znači, tekstu možemo pristupati praktično na svim nivoima. U nastavku ćemo opisati svaki od ovih objekata.

Objekt Range

Već smo videli da u Excel-u ne postoji objekt koji predstavlja jednu ćeliju radnog lista, već da se ćelija predstavlja pomoću objekta `Range`. Slično, u Word-u ne postoji objekt koji predstavlja jedan karakter ili reč, već se u tu svrhu koristi objekt `Range`.

Objekt `Range` predstavlja neprekinutu sekciju teksta u dokumentu, tako da on može predstavljati sve između jednog karaktera i čitavog dokumenta.

Objekt `Range` se može dobiti pomoću metode `Range`, koja se odnosi na objekt `Document`, što je već objašnjeno u tabeli 35. Drugi način dobijanja ovog objekta je pomoću osobine `Range`. U pitanju je osobina velikog broja Word-ovih objekata, uključujući objekte `Paragraph` i `Selection`. Ova osobina je važna jer kod ovih objekata ponekad predstavlja jedini jednostavan način rada sa tekstom. Na primer, objekt `Paragraph` nema osobinu `Font`, pa se podešavanja fonta vrši preko osobine `Range`, kao što je dato sledećom naredbom:

```
Documents(1).Paragraphs(1).Range.Font.Bold = True
```

Osobine i metode objekta Range

U tabelama 36. i 37. su date neke od korišćenih osobina i metoda objekta `Range`. Osim njih, Word pruža pravo obilje osobina i metoda ovog objekta, što ostavljamo vama da istražite.

Osobina	Opis
`Bold`	Vraća True ako je dati opseg boldovan, False ako nijedan karakter opsega nije boldovan i `wdUndefined` ako je deo opsega boldovan. Ova se osobina može setovati na True (sve se bolduje), False (uklanja se boldovanje) ili `wdToggle` (True se menja u False i obrnuto).
`Italic`	Vraća True ako je dati opseg prikazan u italiku, False ako nijedan karakter

	opsega nije u italiku i `wdUndefined` ako je deo opsega u italiku. Ova se osobina može setovati na True (sve u italik), False (uklanja se italik) ili `wdToggle` (True se menja u False i obrnuto).
`Case`	Vraća ili menja veličinu slova datog opsega. Ova osobina koristi razne Word-ove konstante, uključujući `wdLowerCase`, `wdTitleSentence`, `wdTitleWord`, `wdToggleCase` i `wdUpperCase`. Na primer `Activedocument.Paragraphs(1).Range.Case = wdUpperCase` će sva slova u prvom pasusu aktivnog dokumentu prebaciti u odgovarajuća velika.
`Characters`	Vraća kolekciju `Characters` koja predstavlja sve karaktere opsega.
`Words`	Vraća kolekciju `Words` koja predstavlja sve reči opsega.
`Sentences`	Vraća kolekciju `Sentences` koja predstavlja sve rečenice (tj. `Sentence` objekte) opsega.
`Paragraphs`	Vraća kolekciju `Paragraphs` koja predstavlja sve pasuse (tj. `Paragraph` objekte) opsega.
`Start`	Pozicija prvog karaktera opsega.
`End`	Pozicija poslednjeg karaktera opsega.
`Font`	Vraća ili menja `Font` objekt koji određuje formatiranje karaktera u opsegu.
`Text`	Vraća ili menja tekst opsega. Na primer `Activedocument.Paragraphs(1).Range.Text = "Prvi pasus"` će zameniti prvi pasus u aktivnom dokumentu sa tekstom datim pod znacima navoda.

Tabela 36. Neke od osobina objekta `Range`.

Metod	Opis
`CheckGrammar`	Provera ispravnosti gramatike u datom opsegu.
`CheckSpelling`	Provera ispravnosti pravopisa u datom opsegu.
`Collapse`	Ukoliko je opseg trenutno selektovan, ova metoda uklanja selekciju (tzv. kolapsiranje) i pomera kursor u skladu sa sintaksom: `Collapse(Direction)` gde `Selection` određuje poziciju kursora nakon uklanjanja selekcije. Moguće vrednosti su `wdCollapseStart` (kursor na početku opsega, i ovo je podrazumevana opcija) i `wdCollapseEnd` (kursor na kraju opsega). Ekvivalentna radnja je levi klik miša na početak i kraj selekcije.
`Copy`	Kopiranje sadržaja opsega na Clipboard.
`Cut`	Premeštanje sadržaja opsega na Clipboard.
`Delete`	Koristi se za brisanje dela opsega ili čitavog opsega. Ukoliko se navede bez argumenata, briše se ceo opseg. Ako se navede sa argumentima, na sledeći način: `Delete(Unit,Count)` briše se `Count` jedinica `Unit`, gde `Unit` može biti `wdCharacter` (podrazumevano) ili `wdWord`. Ukoliko je `Count` pozitivan broj, brišu se jedinice unapred, a ako je negativan brisanje se vrši unazad.

`InsertAfter`	Umetanje teksta posle specificiranog opsega. Sintaksa je `InsertAfter(Text)` gde `Text` predstavlja tekst koji se umeće.
`InsertBefore`	Umetanje teksta ispred specificiranog opsega. Sintaksa je `InsertBefore(Text)` gde `Text` predstavlja tekst koji se umeće.
`InsertParagraphAfter`	Umetanje karaktera za novi pasus posle specificiranog opsega. Nakon primene metoda, opseg se proširuje tako da uključi novi pasus.
`InsertParagraphBefore`	Umetanje karaktera za novi pasus ispred specificiranog opsega. Nakon primene metoda, opseg se proširuje tako da uključi novi pasus.
`Paste`	Smešta sadržaj Clipboard-a na tekuću poziciju opsega. Da bi izbegli prepisivanje preko selektovanog opsega, iskoristite metod `Collapse` pre metoda `Paste`.
`Select`	Selektovanje specificiranog opsega.

Tabela 37. Neke od metoda objekta `Range`.

Objekt Selection

Objekt `Selection` predstavlja tekuću selekciju u Word-ovom dokumentu, tj. sve što je u datom trenutku selektovano (blok teksta, tekst box, tabela, slika, …). Ako ništa nije selektovano, `Selection` označava tekuću tačku za umetanje. U svakom trenutku u dokumentu može postojati samo jedna selekcija, a ukoliko postoji više otvorenih dokumenata, samo jedna selekcija može biti aktivna.

Objekti `Range` i `Selection` su dosta slični. Razlikuju se po tome da `Selection` odgovara jednoj aktivnoj selekciji, a `Range` postoji nezavisno od kursora i selekcije, uvek se sastoji od teksta i može se pristupiti bilo kojem broju Range objekata.

Objekti `Range` i `Selection` dele veliki broj osobina i metoda. Mi ćemo ovde dati samo jedan broj osobina i metoda objekta `Selection` koji nisu navedeni kod objekta `Range`. Ovi metodi su dati u tabeli 38.

Metod	Opis
`Type`	Osobina koja vraća tip selekcije.
`TypeBackspace`	Metod koji briše karakter koji prethodi selekciji. Ukoliko selekcija nije kolapsirana, briše se selekcija. Ovaj metod je funkcionalno potpuno isti kao dugme Backspace na tastaturi.
`TypeParagraph`	Metod koji umeće novi, prazni, pasus. Ukoliko selekcija nije kolapsirana, zamenjuje se umetnutim pasusom. Ako ne želimo da brišemo sadržaj selekcije, potrebno je koristiti metode `InsertParagraphAfter` i `InsertParagraphBefore`, koje su opisane u prethodnoj tabeli. Ovaj metod je funkcionalno potpuno isti kao dugme Enter na tastaturi.
`TypeText`	Metod za umetanje teksta u dokument. Sintaksa je: `TypeText(Text)` gde `Text` predstavlja tekst koji se umeće. Na primer, ukoliko želimo da ispred tekuće selekcije umetnemo tekst `"Dobar dan"`, potrebno je da

	uradimo sledeće:
	`Selection.Collapse wdCollapseStart`
	`Selection.TypeText "Dobar dan"`

`Move`	Pomeranje selekcije. Sintaksa je:
	`Move(Unit,Count)`
	gde `Unit` može biti `wdCharacter` (podrazumevano), `wdWord`, `wdSentence`, `wdParagraph`, `wdSection`, `wdStory`, `wdCell`, `wdColumn`, `wdRow` ili `wdTable`. Ukoliko je `Count` pozitivan broj, vrši se pomeranje unapred, a ako je negativan unazad. `Count` je podrazumevano 1. Metod vraća broj izvršenih pomeraja. Ukoliko je vraćena vrednost 0, do pomeranja nije ni došlo. Na primer,
	`Selection.Move wdParagraph, 3`
	`Selection.Move wdParagraph, -3`
	pomeraju selekciju 3 pasusa nadole i nagore, respektivno.

`MoveLeft`	Pomeranje selekcije ulevo. Sintaksa je: `MoveLeft(Unit, Count, Extend)` gde `Unit` i `Count` imaju isto značenje kao kod metode `Move`, dok `Extend` može uzeti dve vrednosti, `wdMove` (podrazumevano) ili `wdExtend`. Ako se koristi `wdMove`, selekcija se kolapsira nakon svog poslednjeg karaktera i pomera se ulevo, a ako se koristi `wdExtend`, selekcija se širi ulevo.
`MoveRight`	Pomeranje selekcije udesno. Sintaksa je: `MoveRight(Unit, Count, Extend)` gde `Unit` i `Count` imaju isto značenje kao kod metode `Move`, dok `Extend` može uzeti dve vrednosti, `wdMove` (podrazumevano) ili `wdExtend`. Ako se koristi `wdMove`, selekcija se kolapsira nakon svog poslednjeg karaktera i pomera se ulevo, a ako se koristi `wdExtend`, selekcija se širi ulevo.
`MoveUp`	Pomeranje selekcije naviše. Sintaksa je: `MoveUp(Unit, Count, Extend)` gde `Unit` može biti `wdLine` (podrazumevano), `wdParagraph`, `wdWindow` ili `wdScreen`, a `Count` je broj jedinica za koje se pomeramo (podrazumevano 1). `Extend` može biti `wdMove` (podrazumevano) ili `wdExtend`. `wdMove` kolapsira selekciju nakon poslednjeg karaktera i pomera selekciju naviše, dok `wdExtend` širi selekciju naviše.
`MoveDown`	Pomeranje selekcije naviše. Sintaksa je: `MoveDown(Unit, Count, Extend)` gde svi argumenti imaju isto značenje kao kod metoda `MoveUp`.
`MoveStart`	Pomeranje početka selekcije. Sintaksa je: `MoveStart(Unit,Count)` gde `Unit` može biti `wdCharacter` (podrazumevano), `wdWord`, `wdSentence`, `wdParagraph`, `wdSection`, `wdStory`, `wdCell`, `wdColumn`, `wdRow` ili `wdTable`. Ukoliko je `Count` pozitivan broj, vrši se pomeranje unapred, a ako je negativan unazad. Count je podrazumevano 1.
`MoveEnd`	Pomeranje kraja selekcije. Sintaksa je: `MoveEnd(Unit,Count)` gde `Unit` i `Count` imaju isto značenje kao kod metoda `MoveStart`.
`Start`	Osobina koja vraća ili menja poziciju početka selekcije (redni broj karaktera u dokumentu).
`End`	Osobina koja vraća ili menja poziciju kraja selekcije (redni broj karaktera u dokumentu).

`StartOf`	Pomera ili širi početnu poziciju selekcije do početka najbliže specificirane tekstualne jedinice. Sintaksa je: `StartOf(Unit, Extend)` gde Unit uzima iste vrednosti kao kod metoda `MoveDown`, pri čemu je podrazumevana vrednost `WdWord`. `Extend` može biti `wdMove` (podrazumevano) ili `wdExtend`.
`EndOf`	Pomera ili širi krajnju poziciju selekcije do kraja najbliže specificirane tekstualne jedinice. Sintaksa je: `EndOf(Unit, Extend)` gde `Unit` i `Count` imaju isto značenje kao kod metoda `StartOf`.
`Expand`	Proširenje opsega. Sintaksa je: `Expand(Unit)` gde Unit uzima iste vrednosti kao kod metoda `MoveDown`, pri čemu je podrazumevana vrednost `WdWord`.

`Extend`	Proširenje opsega do datog karaktera, na sledeći način:
	`Extend(Character)`
	gde je karakter do kojeg širimo opseg. Na primer, pozivom
	`Selection.Extend "T"`
	se proširuje selekciju do prvog velikog slova T.

Tabela 38. Neke od osobina i metoda objekta `Selection`.

Objekt Characters

`Characters` je kolekcija koja predstavlja sve karaktere datog objekta, bilo da je to reč, rečenica, pasus, selekcija ili ceo dokument. Na primer,

```
ActiveDocument.Words(3).Characters
```

vraća kolekciju karaktera u objektu `Range` koji predstavlja treću reč aktivnog dokumenta. Dakle, osobina `Characters` vraća kolekciju `Characters`. Osim objekta `Range`, ovu osobinu imaju objekti `Selection` i `Document`.

Osobina `Characters` nam pruža mogućnost rada na nivou karaktera. Pristupanje pojedinačnim karakterima se vrši navođenjem rednog broja karaktera u zagradi. Na primer, naredbom

```
ActiveDocument.Words(1).Characters(1).Font.Color = wdColorPink
```

pristupamo prvom karakteru prve reči dokumenta i menjamo mu boju u ljubičastu. Uočimo način zadavanja boje preko VBA konstante, što nismo mogli raditi u Excel-u.

Od korisnih osobina pomenućemo samo osobinu `Count` koja vraća broj karaktera u datoj kolekciji. Ispod je data procedura `DuzeOd10Podvuci`, koja u dokumentu pronalazi sve reči duže od 10 karaktera i podvlači ih.

```
Sub DuzeOd10Podvuci()
Dim Rec As Range
For Each Rec In ThisDocument.Words
    If Rec.Characters.Count > 10 Then
        Rec.Font.Underline = wdUnderlineSingle
    End If
Next
```

```
End Sub
```

Objekt Words

Obrada reči predstavlja sledeći korak u obradi teksta. `Words` je kolekcija koja predstavlja sve reči specificiranog objekta. Na primer,

```
ActiveDocument.Words
Activedocument.Sentences(2).Words
```

vraćaju sve reči aktivnog dokumenta i reči druge rečenice aktivnog dokumenta, respektivno. Dakle, osobina `Words` objekta vraća kolekciju `Words`. Pored objekta `Document`, objekti `Paragraph`, `Range` i `Selection` imaju ovu osobinu, s' tim što se kod objekta `Paragraphs` osobini `Words` pristupa preko osobine `Range`.

Već smo rekli da Word VBA ne sadrži objekt koji predstavlja reč kao takvu. Umesto toga, reči se klasifikuju kao objekti tipa `Range`. Samim tim, važe pomenute osobine i metode. Kao primer, navedimo proceduru koja pomoću MsgBox-a javlja koliko reči u aktivnom dokumentu počinje velikim slovom.

```
Sub PocinjuVelikim()
Dim Rec As Range, Br As Integer
Br = 0
For Each Rec In ActiveDocument.Words
    If Mid(Rec.Text, 1, 1) Like "[A-Z]" Then
        Br = Br + 1
    End If
Next
If Br = 0 Then
    MsgBox "Nema reci koje pocinju velikim slovom."
Else
    MsgBox "Ima " & Br & " reci koje pocinju velikim slovom."
End If
End Sub
```

Objekt Sentences

Pored slova i reči, Word omogućava rad sa tekstom na nivou rečenica. Osobina `Sentences` objekata `Document`, `Range` ili `Selection` vraća objekt `Sentences`, koji predstavlja kolekciju svih rečenica specificiranog objekta. Pojedinim rečenicama se pristupa navođenjem rednog broja rečenice u zagradi. Kao i reči, rečenice su takođe objekti tipa `Range`. Kao mali primer, daćemo naredbu koja bolduje poslednju rečenicu u dokumentu.

```
ActiveDocument.Sentences(ActiveDocument.Sentences.Count).Font.Bold = True
```

Objekt Paragraph

Sledeći logičan korak je obrada pasusa. Objekt `Paragraph` je član kolekcije `Paragraphs` koja predstavlja sve pasuse specificiranog objekta tipa `Document`, `Range` ili `Selection`. Pojedini članovi kolekcije `Paragraphs` se takođe navode pomoću njihovog rednog broja.

Osobine i metode objekta `Paragraph` su date u tabelama 39. i 40., respektivno.

Osobina	Opis
`LeftIndent`	Vraća ili menja uvlačenje pasusa (mereno tačkama) sa leve strane.
`RightIndent`	Vraća ili menja uvlačenje pasusa (mereno tačkama) sa desne strane.
`LineSpacing`	Vraća ili menja veličinu proreda (mereno tačkama) za dati pasus.
`SpaceAfter`	Vraća ili menja veličinu razmaka (mereno tačkama) između specificiranog i narednog pasusa.
`SpaceBefore`	Vraća ili menja veličinu razmaka (mereno tačkama) između specificiranog i prethodnog pasusa.
`Alignment`	Vraća ili menja poravnanje pasusa. Na primer, konstanta koja definiše centralno poravnanje je `wdAlignParagraphCenter`. Spisak konstanti koje definišu poravnanje se može dobiti traženjem `WdParagraphAlignment` u prozoru Object Browser-a (ovaj se prozor dobija opcijom **View⇒Object Browser** u VBE prozoru).
`Style`	Vraća ili menja stil specificiranog pasusa. Word sadrži veliki broj konstanti za predstavljanje predifinisanih stilova. Na primer, konstanta koja definiše stil Heading 1 je `wdStyleHeading1`. Spisak konstanti se može dobiti traženjem `wdBuiltInStyle` u prozoru Object Browser-a.

Tabela 39. Neke od osobina objekta `Paragraph`.

Metod	Opis
`Indent`	Uvlačenje pasusa do pozicije sledećeg taba.
`Outdent`	Vraćanje uvlačenja pasusa do pozicije prethodnog taba.
`Next`	Pomeranje unapred u dokumentu od specificiranog pasusa. Sintaksa je: `Next(Count)` gde `Count` predstavlja broj pasusa preko kojih se pomeramo.
`Previous`	Pomeranje unazad u dokumentu od specificiranog pasusa. Sintaksa je: `Previous(Count)` gde `Count` predstavlja broj pasusa preko kojih se pomeramo.
`Space1`	Postavlja prored pasusa na jednostruk (single).
`Space15`	Postavlja prored pasusa na 1.5.
`Space2`	Postavlja prored pasusa na dvostruk (double)

Tabela 40. Neke od metoda objekta `Paragraph`.

Kod tabele sa osobinama objekta `Paragraph` vidimo da se uvlačenja, proredi i razmaci definišu u tačkama. U ovu svrhu mogu poslužiti metode koje pretvaraju nama bliske jedinice, kao što su milimetri, centimetri, ili inči, u tačke. U pitanju su metode `MillimetersToPoints`, `CentimetersToPoints` i `InchesToPoints`, respektivno. Ovo su metode objekta `Application` (videti tabelu 33).

P R O G R A M I R A N J E
K R O Z
A P L I K A C I J E

Doc. dr Đukanović Slobodan

OSMI TERMIN

Objekt Paragraph

Sledeći logičan korak je obrada pasusa. Objekt `Paragraph` je član kolekcije `Paragraphs` koja predstavlja sve pasuse specificiranog objekta tipa `Document`, `Range` ili `Selection`. Pojedini članovi kolekcije `Paragraphs` se takođe navode pomoću njihovog rednog broja.

Osobine i metode objekta `Paragraph` su date u tabelama 39. i 40., respektivno.

Osobina	Opis
`LeftIndent`	Vraća ili menja uvlačenje pasusa (mereno tačkama) sa leve strane.
`RightIndent`	Vraća ili menja uvlačenje pasusa (mereno tačkama) sa desne strane.
`LineSpacing`	Vraća ili menja veličinu proreda (mereno tačkama) za dati pasus.
`SpaceAfter`	Vraća ili menja veličinu razmaka (mereno tačkama) između specificiranog i narednog pasusa.
`SpaceBefore`	Vraća ili menja veličinu razmaka (mereno tačkama) između specificiranog i prethodnog pasusa.
`Alignment`	Vraća ili menja poravnanje pasusa. Na primer, konstanta koja definiše centralno poravnanje je `wdAlignParagraphCenter`. Spisak konstanti koje definišu poravnanje se može dobiti traženjem `WdParagraphAlignment` u prozoru Object Browser-a (ovaj se prozor dobija opcijom **View⇒Object Browser** u VBE prozoru).
`Style`	Vraća ili menja stil specificiranog pasusa. Word sadrži veliki broj konstanti za predstavljanje predifinisanih stilova. Na primer, konstanta koja definiše stil Heading 1 je `wdStyleHeading1`. Spisak konstanti se može dobiti traženjem `wdBuiltInStyle` u prozoru Object Browser-a.

Tabela 39. Neke od osobina objekta `Paragraph`.

Metod	Opis
`Indent`	Uvlačenje pasusa do pozicije sledećeg taba.
`Outdent`	Vraćanje uvlačenja pasusa do pozicije prethodnog taba.
`Next`	Pomeranje unapred u dokumentu od specificiranog pasusa. Sintaksa je: `Next(Count)` gde `Count` predstavlja broj pasusa preko kojih se pomeramo.
`Previous`	Pomeranje unazad u dokumentu od specificiranog pasusa. Sintaksa je: `Previous(Count)` gde `Count` predstavlja broj pasusa preko kojih se pomeramo.
`Space1`	Postavlja prored pasusa na jednostruk (single).
`Space15`	Postavlja prored pasusa na 1.5.
`Space2`	Postavlja prored pasusa na dvostruk (double)

Tabela 40. Neke od metoda objekta `Paragraph`.

Kod tabele sa osobinama objekta `Paragraph` vidimo da se uvlačenja, proredi i razmaci definišu u tačkama. U ovu svrhu mogu poslužiti metode koje pretvaraju nama bliske jedinice, kao što su milimetri, centimetri, ili inči, u tačke. U pitanju su metode `MillimetersToPoints`, `CentimetersToPoints` i `InchesToPoints`, respektivno. Ovo su metode objekta `Application` (videti tabelu 33).

U nastavku je data procedura `Pasusi` koja sve pasuse sa više od 7 rečenica uvlači po 1cm sa obe strane, postavlja im obostrano poravnanje i dvostruki prored, razmiče ih 1.5cm od susednih pasusa i postavlja okvir koji se sastoji od jedne linije iznad i ispod pasusa. Linije okvira formatirati po želji.

```
Sub Pasusi()
Dim Pas As Paragraph
For Each Pas In ThisDocument.Paragraphs
    If Pas.Range.Sentences.Count > 7 Then
        With Pas
            .Alignment = wdAlignParagraphJustify
            .LeftIndent = CentimetersToPoints(1)
            .RightIndent = CentimetersToPoints(1)
            .SpaceAfter = CentimetersToPoints(1.5)
            .SpaceBefore = CentimetersToPoints(1.5)
            .Space2
            With .Borders(wdBorderTop)
                .LineStyle = wdLineStyleSingle
                .LineWidth = wdLineWidth600pt
                .Color = wdColorDarkRed
            End With
            With .Borders(wdBorderBottom)
                .LineStyle = wdLineStyleDashDot
                .LineWidth = wdLineWidth075pt
                .Color = wdColorAutomatic
            End With
        End With
    End If
Next
End Sub
```

Objekt Table

Pored teksta, malo pažnje ćemo posvetiti i radu sa tabelama u Wordu. Tabele se u Word VBA predstavljaju pomoću objekta `Table`, odnosno pomoću kolekcije `Tables` koja predstavlja sve tabele specificirane selekcije, opsega ili čitavog dokumenta. Osobina `Tables` objekta `Document` vraća kolekciju `Tables`. Pojedinačne tabele se dobijaju navođenjem rednog broja tabele u kolekciji `Tables`. Na primer,

```
Documents("VBA.doc").Tables(1)
```

vraća prvu tabelu u dokumentu VBA.doc.

Osobine i metode objekta `Table` i kolekcije `Tables` su date u tabelama 41. i 42., respektivno.

Osobina	Opis
Objekt Table	
`AllowAutoFit`	Omogućava automatsko podešavanje veličine tabele u odnosu na njen sadržaj (opcija AutoFit to Contents). Moguće vrednosti su True i False.

`Borders`	Vraća kolekciju `Borders` koja predstavlja sve ivice date tabele. Na primer, naredbom
	`Documents(1).Tables(1).Borders(wdBorderHorizontal). _`
	`LineWidth = wdLineWidth150pt`
	debljinu svih horizontalnih linija prve tabele postavljamo na 1.5 tačaka.

`Shading`	Omogućava dodavanje boje i teksture pozadini tabele. Na primer, naredbama
	`Tables(1).Shading.BackgroundPatternColor = wdColorBlue`
	`Tables(1).Shading.Texture = wdTextureVertical`
	se prvoj tabeli aktivnog dokumenta respektivno menja boja pozadine u plavu i tekstura u vidu horizontalnih crta.

`Columns`	Vraća kolekciju `Columns` koja predstavlja sve kolone tabele. Na primer, naredba `Documents(1).Tables(1).Columns(2).Delete` briše drugu kolonu prve tabele datog dokumenta.
`Rows`	Vraća kolekciju `Rows` koja predstavlja sve vrste tabele.
`Uniform`	Vraća True ako sve vrste tabele imaju isti broj kolona.
Kolekcija Tables	
`Count`	Vraća broj tabela u referenciranom dokumenta.

Tabela 41. Neke od osobina objekta `Table` i kolekcije `Tables`.

Metod	Opis
Objekt Table	
`AutoFitBehavior`	Određuje način automatske promene veličine tabele kada je aktivna opcija AutoFit, kada imamo podešavanje veličine tabele u zavisnosti od veličine prozora dokumenta ili u zavisnosti od teksta u tabeli. Sintaksa je: `AutoFitBehavior(Behavior)` gde `Behavior` definiše automatsko podešavanje veličine tabele i može biti `wdAutoFitContent`, `wdAutoFitWindow` i `wdAutoFitFixed`. Poslednja vrednost zapravo isključuje mogućnost automatskog podešavanja veličine tabele.
`AutoFormat`	Dodavanje predefinisanog stila tabeli. Pojednostavljena sintaksa je: `AutoFormat(Format)` gde `Format` predstavlja predifinisani stil. Spisak Word VBA konstante koje predstavljaju predefinisane stilove se može dobiti traženjem `wdTableFormat` u prozoru Object Browser-a. Na primer, konstanta `wdTableFormatGrid1` definiše stil Table Grid1.

`Cell`	Vraća objekt `Cell` koji predstavlja jednu ćeliju tabele. Sintaksa je:
	`Cell(Row, Column)`
	gde `Row` predstavlja redni broj vrste, a `Column` redni broj kolone date ćelije. Oba argumenta su obavezna. Na primer, naredba
	`Tables(1).Cell(1,1).Delete`
	briše prvu ćeliju prve tabele aktivnog dokumenta.

`ConvertToText`	Konvertuje tabelu u tekst i vraća `Range` objekt koji predstavlja dobijeni tekst. Pojednostavljena sintaksa je:
	`ConvertToText(Separator)`
	gde `Separator` predstavlja karakter koji razdvaja konvertovane kolone. Svaka konvertovana vrsta se nalazi u novom redu. `Separator` može biti jedna od sledećih konstanti:
	`wdSeparateByCommas`
	`wdSeparateByDefaultListSeparator`
	`wdSeparateByParagraphs`

	`wdSeparateByTabs` (podrazumevano).
`Delete`	Brisanje tabele.
`Select`	Selektovanje tabele.
`Sort`	Sortiranje tabele. Pogledati Help za više informacija.
`SortAscending`	Sortiranje vrsta tabele u rastući alfa-numerički redosled. Prva vrsta se tretira kao zaglavlje tabele i ne uzima se u obzir pri sortiranju. Za uključivanje prve vrste u sortiranje, koristiti metod `Sort`.
`SortDescending`	Sortiranje vrsta tabele u opadajući alfa-numerički redosled. Prva vrsta se tretira kao zaglavlje tabele i ne uzima se u obzir pri sortiranju.
`Split`	Deli tabelu umetanjem praznog pasusa iznad specificirane vrste i vraća `Table` objekt koji sadrži specificiranu vrstu i vrste nakon nje. Sintaksa je: `Split(BeforeRow)` gde je `BeforeRow` redni broj vrste iznad koje umećemo prazan pasus.
Kolekcija Tables	

`Add`	Kreiranje nove tabele. Pojednostavljena sintaksa metode je:
	`Add(Range, NumRows, NumColumns)`
	gde `Range` predstavlja opseg u dokumentu gde smeštamo tabelu, dok `NumRows` i `NumColumns` predstavljaju broj vrsta i kolona koje ima nova tabela. Ova tri argumenta su obavezna. Na primer, naredba
	`Documents(1).Tables.Add Documents(1).Range(0,0), 2, 3`
	će kreirati praznu tabelu od 2 vrste i 3 kolone na početku prvog dokumenta, dok će naredbe
	`Set Selek = Documents(1).Content`
	`Selek.Collapse wdCollapseEnd`
	`Documents(1).Tables.Add Selek, 2, 3`
	kreirati ovu tabelu na kraju prvog dokumenta. Uočite način prelaska na kraj dokumenta.

Tabela 42. Neke od osobina objekta objekta `Table` i kolekcije `Tables`.

Pored osobina i metoda objekta `Table` i kolekcije `Tables`, korisno je navesti i metodu `Add` kolekcija `Rows` i `Columns`, kojom se u tabelu dodaju nove vrste i kolone. U oba slučaja, ovaj metod ima jedan obavezan argument, koji predstavlja redni broj vrste, odnosno kolone, ispred koje su ubacuje nova vrsta, odnosno kolona. Brisanje specificiranih vrsta i kolona se vrši metodom `Delete`.

U nastavku dajemo primer rada sa tabelama. U folderu C:\Temp\VBA se nalazi određeni broj Word dokumenata, pri čemu u svakom dokumentu prvi pasus sadrži ime i prezime studenta, drugi pasus sadrži jedinstveni kod pridružen tom studentu (proizvoljna kombinacija cifara i velikih slova) i treći pasus sadrži naslov njegovog diplomskog rada. Napisati proceduru `Diplomski` koja uzima podatke iz prva tri pasusa svih dokumenata u datom folderu i smešta ih u tabelu u dokumentu gde se nalazi makro. Tabela sadrži tri kolone, pri čemu prva kolona odgovara imenu i prezimenu studenta, druga jedinstvenom kodu i treća naslov diplomskog rada. Tabela treba da se nađe na kraju dokumenta.

```
Sub Diplomski()
Dim Dok As Document, R As Range, Tab1 As Table
Dim I As Integer, S As String, Tekst As String, Putanja As String
Set R = ThisDocument.Content
```

```
R.Collapse wdCollapseEnd
Set Tab1 = ThisDocument.Tables.Add(R, 1, 3)
With Tab1
    .AutoFitBehavior wdAutoFitContent
    .Borders.InsideLineStyle = wdLineStyleSingle
    .Borders.OutsideLineStyle = wdLineStyleDouble
End With
Tab1.Cell(1, 1).Range.Text = "Ime i prezime"
Tab1.Cell(1, 2).Range.Text = "Jedinstveni kod"
Tab1.Cell(1, 3).Range.Text = "Naslov diplomskog rada"
Putanja = "C:\Temp\VBA"
S = Dir(Putanja & "\*.doc")
I = 1
Do While S <> ""
    Tab1.Rows.Add
    I = I + 1
    Set Dok = Documents.Open(Putanja & "\" & S)
    Tekst = Dok.Paragraphs(1).Range.Text
    Tab1.Rows(I).Cells(1).Range.Text = Mid(Tekst, 1, Len(Tekst) - 1)
    Tekst = Dok.Paragraphs(2).Range.Text
    Tab1.Rows(I).Cells(2).Range.Text = Mid(Tekst, 1, Len(Tekst) - 1)
    Tekst = Dok.Paragraphs(3).Range.Text
    Tab1.Rows(I).Cells(3).Range.Text = Mid(Tekst, 1, Len(Tekst) - 1)
    Dok.Close
    S = Dir
Loop
End Sub
```

Daćemo i primer procedure `Fontovi` koja pomoću MsgBox-a korisniku javlja koji su sve fontovi korišćeni u aktivnom dokumentu.

```
Sub Fontovi()
Dim Font(1 To 100) As String, TekFont As String, Ispis As String
Dim I As Integer, J As Integer, kar As Range, Ind As Boolean
I = 0
For Each kar In ActiveDocument.Characters
    TekFont = kar.Font.Name
    Ind = False
    For J = I To 1 Step -1
        If StrComp(Font(J), TekFont) = 0 Then Ind = True
    Next
    If Ind = False Then
        I = I + 1
        Font(I) = TekFont
    End If
Next
Ispis = ""
For J = 1 To I
    If J < I Then
        Ispis = Ispis & Font(J) & vbCrLf
    Else
        Ispis = Ispis & Font(J)
    End If
Next
Ispis = "U dokumentu " & ActiveDocument.Name & " su korisceni fontovi:" _
```

```
        & vbCrLf & Ispis
MsgBox Ispis
End Sub
```

U prethodnoj proceduri smo koristili niz `Font` u koji smo smeštali sve pronađene fontove u dokumentu. Od svakog karaktera u dokumentu smo uzimali font i pretragom po nizu `Font` smo utvrđivali da li taj font već postoji u dokumentu. Ukoliko postoji, promenljivoj `Ind` smo davali vrednost `True`.

P R O G R A M I R A N J E
K R O Z
A P L I K A C I J E

Doc. dr Đukanović Slobodan

DEVETI TERMIN

Rad sa Word dokumentima u Excel-u i obrnuto

Iz jednog Word dokumenta možemo kreirati, modifikovati, zatvoriti ili brisati Excel tabele i obrnuto. Povezivanje ove dve aplikacije ćemo ilustrovati kroz dva primera.

Primer 1. Word dokument Racun.doc predstavlja račun koji podnosi firma koja se bavi prodajom računarskih komponenti. Ovaj dokument sadrži dve tabele, kao što je prikazano ispod.

Šifra	Količina	Naziv	Jedinična cena	Ukupno
102	2			
101	1			
115	2			
103	3			

Ukupno	
PDV	
Za naplatu:	

Popuni

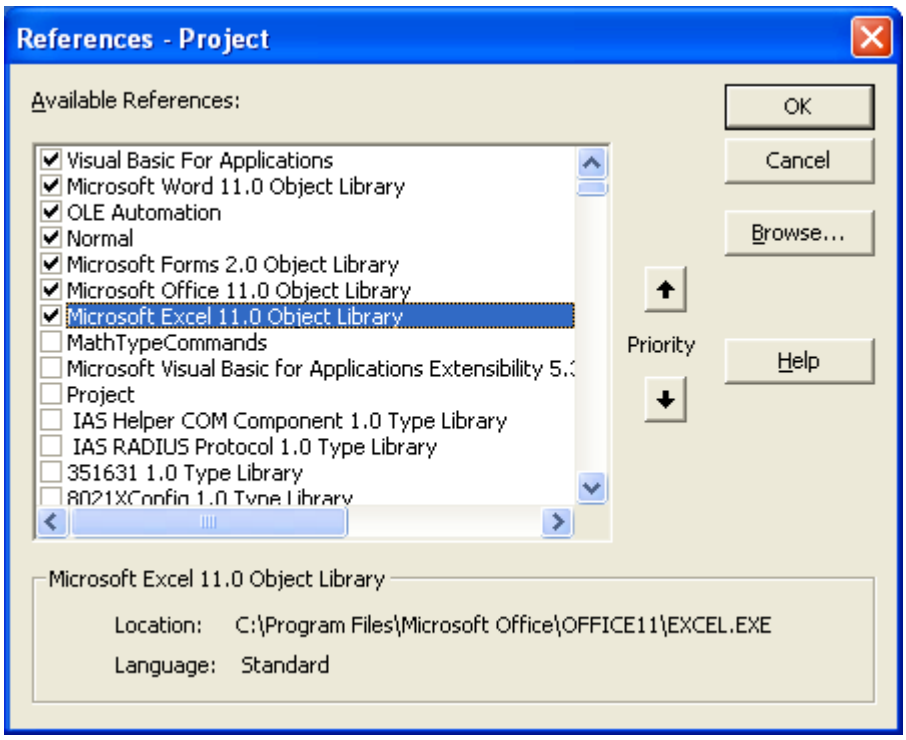
U folderu gde je Racun.doc, nalazi se i radna sveska Zalihe.xls u kojoj su date šifre proizvoda, nazivi proizvoda, jedinične cene proizvoda i trenutna količina, na način prikazan ispod.

	A	B	C	D	E
1	**Šifra**	**Naziv**	**Jed. cena**	**Količina**	
2	101	Monitor	120,00 €	8	
3	102	Tastatura	10,00 €	11	
4	103	PC	440,00 €	5	
5	104	Modem	15,00 €	7	
6	105	Štampač	75,00 €	3	
7	106	Miš	5,00 €	15	
8					
9					

Klikom na dugme Popuni, koje se nalazi u Racun.doc, potrebno je:

- Popuniti kolone *Naziv* i *Jedinična cena* sa podacima uzetim iz sveske Zalihe.xls. Ukoliko tražena šifra ne postoji, u data polja ovih kolona upisati *Pogrešna šifra*, dok ako je tražena količina veća od one koja postoji na zalihama, potrebno je upisati *Manjak zaliha*.
- Popuniti kolonu *Ukupno* kao proizvod jedinične cene i količine;
- Popuniti donju tabelu, pri čemu *Ukupno* predstavlja zbir elemenata kolone Ukupno iz gornje tabele, dok PDV iznosi 17%. Vrednost *Za naplatu:* se dobija kao zbir polja *Ukupno* i *PDV* iz iste tabele.
- Ažurirati kolonu *Količina* u svesci Zalihe.xls, tj. umanjiti brojeve u ovoj koloni za vrednosti date u koloni *Količina* iz Racun.doc.

Prvo, da bi iz Word dokumenta mogli raditi sa Excel radnim sveskama, potrebno je u VBE prozoru podesiti referencu na objektnu biblioteku Excel-a. Ovo se radi pomoću opcije **Tools⇒References**, čijim se odabirom dobija dijalog box sličan onom sa slike 9. Referenca se traži u polju **Available References**.



Slika 9. Dijalog box za odabir referenci u VBE.

Drugo, do sad nismo radili sa kontrolama u dokumentu, već samo na korisničkim formama. U ovom primeru se podrazumeva da se tabela popunjava pritiskom na dugme *Popuni* koje se nalazi u samom dokumentu. **Control Toolbox** paleta alatki omogućava rad sa kontrolama. Potrebno je aktivirati ovu paletu, što se radi na standardan način, pozicionirati se gde želimo da ubacimo kontrolu i kliknuti na odgovarajuću kontrolu. Da bi se kontrola mogla isprogramirati, potrebno je aktivirati *Design Mode*, što se radi istoimenim dugmetom sa ove palete. Ovaj mod se prepoznaje po tome što se prilikom selektovanja kontrole oko nje pojavljuje isti okvir koji se pojavljuje kada selektujemo sliku u Word dokumentu. Dvoklikom na kontrolu odlazimo u kodni prozor gde programiramo datu kontrolu. Kodovi svih kontrola kreiranih na ovaj način se nalaze u okviru objekta `ThisDocument`. Nakon programiranja, potrebno je izaći iz Design Mode-a da bi testirali i koristili našu kontrolu.

U nastavku su date procedure kojima se izvršava dati zadatak.

```
Private Sub CommandButton1_Click()
Call Popuni
End Sub
```

```
Sub Popuni()
Dim ExApp As New Excel.Application, RS As Workbook
Dim Uk As Double, sif As Long, Ime As String, k as Integer
Dim JedCen As Double, CenaKomp As Double, Ind As Integer, Poruka As String
Set RS = ExApp.Workbooks.Open(ActiveDocument.Path & "\Zalihe.xls")
With ActiveDocument.Tables(1)
    Uk = 0
    For k = 2 To .Rows.Count
        sif = Val(.Cell(k, 1).Range)
        vr = 2
        Ind = 0
        Poruka = "Pogresna sifra"
        Do While RS.Worksheets(1).Cells(vr, 1) <> ""
```

```
            If RS.Worksheets(1).Cells(vr, 1) = sif Then
                If RS.Worksheets(1).Cells(vr,4)<Val(.Cell(k,2).Range) Then
                    Ind = 1
                    Poruka = "Manjak zaliha"
                Else
                    Ime = RS.Worksheets(1).Cells(vr, 2)
                    JedCen = RS.Worksheets(1).Cells(vr, 3)
                    CenaKomp = Val(.Cell(k, 2).Range) * Val(JedCen)
                    Uk = Uk + CenaKomp
                    Ind = 2
                    RS.Worksheets(1).Cells(vr, 4) = _
                    RS.Worksheets(1).Cells(vr,4) - Val(.Cell(k, 2).Range)
                    RS.Save
                End If
                Exit Do
            End If
            vr = vr + 1
        Loop
        If Ind = 0 Or Ind = 1 Then
            .Cell(k, 3).Range = Poruka
            .Cell(k, 4).Range = Poruka
            .Cell(k, 5).Range = Format(0, " 0.00 €")
        Else
            .Cell(k, 3).Range = Ime
            .Cell(k, 4).Range = Format(JedCen, " 0.00 €")
            .Cell(k, 5).Range = Format(CenaKomp, " 0.00 €")
        End If
    Next
End With
With ActiveDocument.Tables(2)
    .Cell(1, 2).Range = Format(Uk, "0.00 €")
    .Cell(2, 2).Range = Format(Uk * 0.17, "0.00 €")
    .Cell(3, 2).Range = Format(Uk * 1.17, "0.00 €")
End With
RS.Close
ExApp.Quit
Set RS = Nothing
Set ExApp = Nothing
End Sub
```

Procedura `CommandButton1_Click` treba da se nađe u okviru objekta `ThisDocument`, dok procedura `Popuni` treba da se nađe u standardnom modulu.

Novina u odnosu na dosadašnje znanje jeste deklaracija

```
Dim ExApp As New Excel.Application
```

gde je korišćena ključna reč `New`. Ova reč omogućava implicitno kreiranje objekta (u našem slučaju aplikacija Excel), tj. kreiranje objekta prilikom njegovog prvog referenciranja. Na ovaj način se ne mora koristiti ključna reč `Set`. Podsetimo se da se pomoću `Set` vrši dodela postojećeg objekta deklarisanom objektu.

U proceduri `Popuni`, promenljiva `Ind` nam služi kao indikacija da li smo pronašli traženu komponentu i da li je tražena količina veća od trenutno postojeće. Ima 3 moguće vrednosti, 0 (tražena šifra ne postoji), 1 (tražena količina je veća od trenutno postojeće količine) i 2 (sve je u redu).

Nadamo se da je sam kod manje-više jasan. Potrebno je biti za nijansu oprezniji prilikom referenciranja ćelija tabela u Word-u i Excel-u, koji su dosta slični. Sadržaj ćelije tabele iz Word dokumenta je string, pa ako želimo da ga tumačimo kao broj, potredno je koristiti funkciju `Val`. U proceduri je korišćena i funkcija `Format` pomoću koje postiže željeni format ispisa sa dve decimale i znakom €.

Primer 2. U ovom primeru ćemo ažurirati stanje zaliha (fajl Zalihe.xls) iz prethodnog primera. Spisak novopristiglih komponenti je dat u Word dokumentu Doprema.doc, koji se nalazi u istom folderu kao i Zalihe.xls, i to u obliku tabele date ispod.

Šifra	Naziv	Količina
101	Monitor	10
105	Štampač	5
106	Miš	20
109	Skener	3

Prvoj radnom listu sveske Zalihe.xls dodati dugme kojim se vrši ažuriranje stanja zaliha. Ažuriranje se vrši tako što se komponentama koje već postoje poveća količina za iznos dat u fajlu Doprema.doc, dok za komponente koje trenutno ne postoje u spisku zaliha treba dodati novu vrstu. Jediničnu cenu novounete komponente definiše korisnik preko input box-a.

```
Private Sub CommandButton1_Click()
Call Azuriraj
End Sub
```

```
Sub Azuriraj()
Dim WApp As New Word.Application, Dok As Document, list As Worksheet
Dim Postoji As Boolean, vr As Integer, k As Integer, JedCen As Double
Dim Ime As String
Set list = ActiveWorkbook.Worksheets(1)
Set Dok = WApp.Documents.Open(ActiveWorkbook.Path & "\Doprema.doc")
With Dok.Tables(1)
    For k = 2 To .Rows.Count
        Postoji = False
        vr = 2
        Do While list.Cells(vr, 1) <> ""
            If list.Cells(vr, 1) = Val(.Cell(k, 1).Range) Then
                list.Cells(vr, 4) = _
                list.Cells(vr, 4) + Val(.Cell(k, 3).Range)
                Postoji = True
            End If
            vr = vr + 1
        Loop
        If Postoji = False Then
            list.Cells(vr, 1) = Val(.Cell(k, 1).Range)
            Ime = .Cell(k, 2).Range
            Ime = Mid(Ime, 1, Len(Ime) - 1)
            list.Cells(vr, 2) = Ime
            JedCen = InputBox("Uneti jedinicnu cenu za " & Ime)
            list.Cells(vr, 3) = Val(JedCen)
            list.Cells(vr, 4) = Val(.Cell(k, 3).Range)
        End If
    Next
End With
ActiveWorkbook.Save
```

```
Dok.Close
WApp.Quit
Set Dok = Nothing
Set WApp = Nothing
End Sub
```

Procedura `CommandButton1_Click` treba da se nađe u okviru radnog lista gde se nalazi kreirano dugme, dok procedura `Azuriraj` treba da se nađe u okviru standardnog modula.

Kao u prvom primeru, i ovde je potrebno podesiti referencu na objektnu biblioteku Word-a. Ovo se radi na način prikazan na slici 9.

U proceduri `Azuriraj`, implicitno se kreira Word aplikacija pomoću ključne reči `New`. Dalje, promenljiva Postoji služi kao indikacija da li trenutna komponenta iz dokumenta Doprema.doc već postoji u spisku zaliha, što se određuje poređenjem šifara. Ukoliko data komponenta ne postoji, vršimo dodavanje nove vrste u radni list popunjavajući odgovarajuće ćelije podacima pročitanim iz Word tabele. Pošto u Word tabeli nije definisana jedinična cena komponente, od korisnika tražimo unos ovog podatka.