Programiranje uz crtanje

U poslednje vreme tržište je prosto zatrpano "revolucionarnim" alatkama koje iz korena menjaju način dizajniranja i održavanja Windows aplikacija, dramatično povećavaju produktivnost, omogućavaju neograničenu proširivost i sve ostalo što se može poželeti. Malo je, međutim, ljudi koji poveruju u sve ove bajke. Zato, hajde da izbliza pogledamo alat koji se još u svojoj prvoj verziji zaista primakao reklamnim fantazijama.

Obrad Bijelić

Microsoft-ov *Visual Basic* se izdvaja se pre svega svojom koncepcijom. Ljudi koji su ga dizajnirali očigledno su znali sve o ciljnom tržištu - sve je odmereno u skladu sa njegovim parametrima. U središtu njihove ideje nalazi se činjenica da razvoj specijalizovanih *Windows* aplikacija može da bude veoma skup ako se koriste standardne Microsoft-ove alatke - *Windows* SDK (*Software Development Kit*) i odgovarajući C prevodilac. Već na prvi pogled je jasno da ova "terminator kombinacija" ne može biti istovremeno podesna za razvoj programa iz *Microsoft Office*-a i malog programa za pristup bazi podataka koja će biti isporučena korisniku u jednom primerku, provodeći svoj radni vek na svega par računara.

DOS deo tržišta o kome govorim, još pre desetak godina, zauzeo je *Clipper*. Lako kreiranje korisničkog interfejsa i jednostavan rad sa bazama podataka, bilo je dovoljno da *Clipper* "kupi" sve programere koji nisu želeli da potroše pola života programirajući prilično dosadne stvari, koje su drugi već uradili mnogo bolje. I zaista, lako kreiranje korisničkog interfejsa je najveći napredak *Visual Basic*-a u odnosu na C i C++. Srećom, projektanti *Visual Basic*-a nisu se zaustavili na korisničkom interfejsu: definisan je kompletan API (*Application Programming Interface*) za podršku proširenja jezika - tako je došlo do kreiranja novog tržišta dodataka za *Visual Basic*, poznatih VBX datoteka. Ovakva koncepcija dovela je do situacije u kojoj glavni problem nije napraviti nešto u *Visual Basic*-u, nego naći pravi VBX. U nekom od narednih brojeva našeg časopisa detaljnije ćemo se pozabaviti proširivanjem *Visual Basic*-a. Do tada, obratićemo više pažnje početnicima.

3, 2, 1...

Koji god programski jezik ili okruženje da koristite, jednom ste morali biti početnik. Ukoliko vam je *Visual Basic* prvi *Windows* jezik sa kojim se upoznajete, počeci zaista mogu da budu gotovo traumatični. Gomila novih ideja, koncepata, pojmova... ukratko, sve vam govori da "treba da zaboravite ono čemu vas je DOS naučio"!

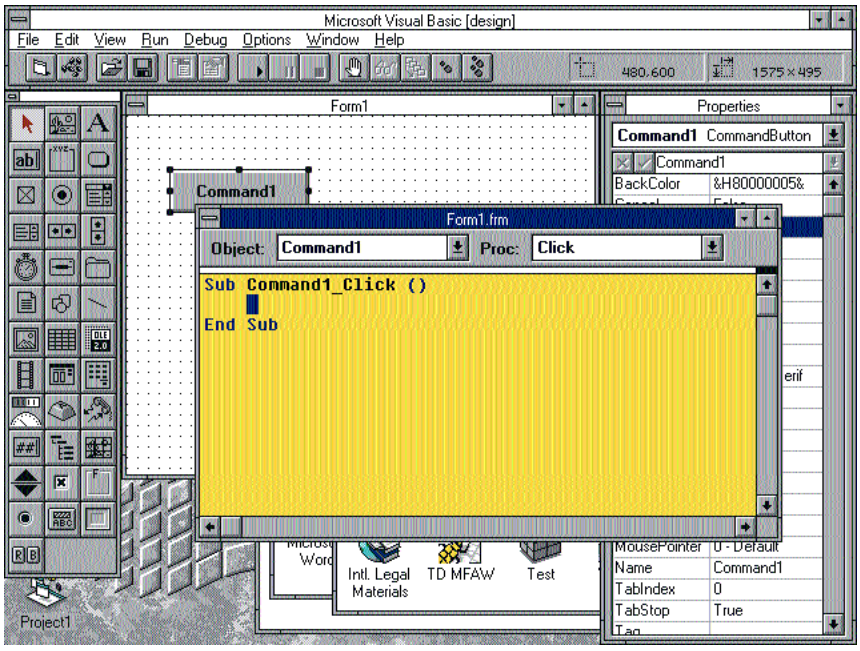
Srećom, stvari ipak ne stoje baš toliko loše. Sve nove (ili "nove") ideje koje donosi *Visual Basic* uvedene su iz jednostavnog razloga: kasnije će vam biti lakše, bez obzira što ćete se na početku pomučiti dok sve proučite. Zbog toga se početnici u svetu *Visual Basic*-a jednostavno moraju osloboditi nekih starih navika (ako ih uopšte imaju), iako će to možda jako boleti - sa nekim navikama pod *Windows*-om, ili bilo kojim grafičkim okruženjem, jednostavno nije moguće živeti. Bolje da ne pominjem legendarnu nemogućnost "poukovanja po memoriji", jer su koncepti kojima ćemo sada posvetiti pažnju ipak mnogo važniji od postojanja ili nepostojanja nekih naredbi.

Dva koncepta koja dominiraju *Visual Basic*-om su programiranje zasnovano na događajima (*event driven programming*) i, delom, objektno orijentisano programiranje (*object oriented programming*). Zašto "delom" - videćemo kasnije.

Po istinitom događaju

Većina "običnih" proceduralnih programa ima nešto što se naziva glavnom petljom. Najčešće je to deo programa koji čeka na akcije korisnika i, u zavisnosti od pritisnutih tastera, aktivira neku od opcija programa, odnosno poziva procedure koje obavljaju odgovarajući posao. Zatim ta procedura preuzima stvar u svoje ruke sve dok korisnik ne pritisne Esc i tako vrati kontrolu glavnoj petlji. U najgorem slučaju, nisu obezbeđene ni "maske za unos", već korisnik mora da se pati sa malo poboljšanim verzijama naredbe INPUT i unosi ili pregleda podatke redosledom koji mu možda uopšte ne odgovara.

Već na prvi pogled jasno je da pisanje programa pod *Windows*-om "nije kompatibilno" sa uobičajenim pristupom. Korisnik veoma često želi da na ekranu ima više prozora, tj. aktivnih opcija, i tako upoređuje podatke i prenosi ih iz jednog prozora u drugi - to su sasvim prirodne operacije koje je veoma teško isprogramirati pristupom "glavne petlje".



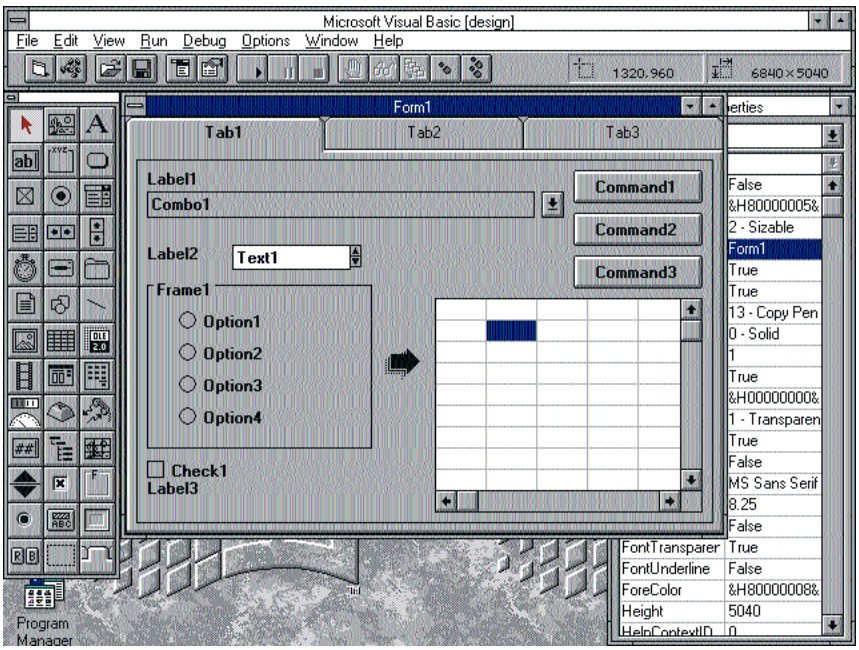
Osim svega ovoga, programi vrlo brzo mogu postati ogromne kolekcije IF / CASE komandi, kojima se ispituju sve moguće reakcije korisnika. Nije potrebno posebno napominjati koliko je teško držati pod kontrolom program sa takvom strukturom, zato ostaje samo jedno pitanje: da li se silna grananja mogu eliminisati? Rešenje je u tome da *Visual Basic* tačno "zna" koju proceduru treba da pozove kada nastupi određeni događaj (*event*) - na primer, pritisak na taster miša. Zbog toga se ovaj pristup zove *programiranje zasnovano na događajima*. Ključna je, dakle, mogućnost *Visual Basic*-a da poveže elemente korisničkog interfejsa sa vašim procedurama koje obavljaju neki koristan posao.

Navedeno povezivanje elemenata korisničkog interfejsa sa vašim procedurama potpuno je automatsko. Pogledajmo primer: ako se naš program sastoji od jednog prozora (u *Visual Basic* terminologiji - "formulara") sa jednim komandnim tasterom čije je programsko ime *Command1*, pritisak mišem na taj taster automatski će izazvati poziv procedure *Command1_Click*. Vama ostaje samo da napišete telo ove procedure koje obavlja željeni posao. Vidimo da *Visual Basic* gradi ime procedure koja reaguje na događaj na osnovu imena elementa korisničkog interfejsa (u ovom slučaju *Command1*) i događaja (u ovom slučaju *Click* - pritisak na **levi** taster miša).

Svakako, *Click* nije jedini događaj koji korisnik može da "izazove" - u *Help* stranama *Visual Basic* naći ćete opise desetina drugih događaja, čijim preuzimanjem (tzv. "nasleđivanjem") možete do detalja da kontrolišete sve akcije korisnika. Komandni taster, kao što je *Command1*, samo je jedna od mnogih tzv. "kontrola" koje možete kreirati (npr. *liste, edit-polja, radio tasteri* itd.). Ostaje samo još jedna nedoumica: šta se dešava ako korisnik pritisne taster *Command1*, a procedura *Command1_Click* ne postoji? Srećom, *Visual Basic* neće prijaviti grešku, već će jednostavno ignorisati taj događaj - pritisak na miša neće proizvoditi nikakav efekat. Slično, *Visual Basic* neće praviti probleme ako u programu postoji procedura *Command1_Click* a kontrola *Command1* je u međuvremenu obrisana.

Objekti i metodi

Već smo pomenuli par novih izraza: *formular, taster, lista*... Sve ovo, a i mnogo toga drugog, u *Visual Basic*-u, jeste **objekat**. Naravno, na svaki pomen objekata, konzervativni bejzikoljupci dobiju ospice, ali uskoro ćemo zaključiti da se, bez objekata, u *Visual Basic*-u jednostavno - ne može.



Da vidimo i zašto. Svaki od elemenata korisničkog interfejsa - formular, lista, meni - osim mogućnosti da "emituje" događaje, mora da ima i određene osobine (*properties*). Na primer, tekstualna kontrola (natpis, labela) u okviru forme, ima definisan font kojim se tekst u toj kontroli ispisuje, njegovu veličinu, boju teksta i još dvadesetak drugih svojstava. Svaka od ovih informacija zapravo je neka vrsta promenljive koja pripada odgovarajućem objektu, a čija se vrednost čita na potpuno isti način kao da je u pitanju neka promenljiva, dok joj se konkretna vrednost dodeljuje najobičnijim operatorom dodeljivanja, znakom =. Ime objekta i ime promenljive razdvaja se (kao i u mnogim drugim objektnim jezicima) tačkom: da biste, na primer, promenili font neke tekstualne kontrole "Label1", kucate: *Label1.FontName = "Times New Roman"*. Slično tome, da biste na formularu ispisali naziv aktuelnog fonta, koristite *Print Form1.FontName* ili jednostavno *Print FontName*. Naime, ako ne navedete ime objekta na koji se neka osobina odnosi (u ovom slučaju *FontName*), podrazumeva se ime tekućeg formulara.

Zaključili smo da objekti mogu da emituju događaje (*Events*) i da poseduju određene osobine (*Properties*). Ako u *Help*-u za *Visual Basic* pogledate opis neke vrste objekta (tzv. klasu), videćete da se osim događaja i osobina stalno pominju i **metodi**. Metodi su procedure i funkcije koje obavljaju neku operaciju nad odabranim objektom. Na primer, ako *Visual Basic* dozvoljava kreiranje formulara (*Forms*), moglo bi se uvesti nekoliko naredbi koje operišu sa formularima: kucali biste, recimo, *Show Form1, Move Form1, Hide Form1*, itd. Možda bi ovakve naredbe lepo radile da imamo samo objekte tipa *formular*, ali se stvari komplikuju jer bi *Show* trebalo da radi istovetan posao i sa komandnim tasterima, listama, menijima, bazama podataka... ukratko, sa bilo kakvim objektima. Mogle bi se uvesti i naredbe kao što su *FormShow Form1*, pa zatim *MoveForm Form1* itd, ali bi takva koncepcija ipak bila daleko od svake elegancije. Zbog toga su uvedeni metodi: da bi se prikazao formular *Form1*, koristi se konstrukcija *Form1.Show*. Isto važi i za objekte bilo koje druge klase - naravno, ako oni podržavaju metod *Show*. Tačku možemo shvatiti kao nalog: "uradi **nad** objektom". Osim što je ovakav pristup fleksibilan, on je i prava blagodet za početnike - ima li jednostavnije i kraće naredbe štampaču da pređe na sledeću stranu od *Printer.NewPage*? Ukoliko metod traži neke dodatne parametre, oni se navode iza njegovog imena, npr. *Command1.Move 5, 5, 10, 10*. Metod može da bude i funkcija(*Print Clipboard.GetText(CF_TEXT)*). Kao i kod osobina objekata, ukoliko se ime objekta ispusti, podrazumeva se aktivni formular. Legendarna naredba *Print "Zdravo,";Ime$* postaje, dakle, metod koji se izvršava nad tekućim formularom.

Uopšte, svemu što ima veze sa određenim objektom pristupa se preko operatora "tačka" - metodi se izvršavaju sa *Objekat.Metod*, a osobinama objekta se pristupa sa *Objekat.Osobina*. Šteta je što procedure koje reaguju na događaje, vezane za neki objekat, nemaju identičnu sintaksu sa metodima, već se mora koristiti kostrukcija *Objekat_Događaj*.

Elementi jezika

Kada se radi o *Visual Basic*-u, početnici obično nemaju mnogo problema sa samim jezikom. Štaviše, *Visual Basic* je u dovoljnoj meri kompatibilan sa svojim prethodnicima pa se, bez većih izmena, pokrenću standardni DOS programi napisani u *Quick Basic*-u. *Visual Basic* nudi dobre zamene za neke od mnogih ne-baš-dalekovidih konvencija svojih prethodnika: brojeve linija još uvek možete da koristite, kao i Goto i GoSub naredbe, ali u primerima uz *Visual Basic* nigde nećete pronaći ni tragove ovih i sličnih naredbi. Umesto njih, *Visual Basic* nudi procedure i funkcije sa parametrima, nalik svim strukturnim jezicima.

Procedure se deklarišu sa *Sub Ime (Parametri)*, pri čemu se parametri prenose po imenu - to znači da se iz procedure može uticati na originalnu vrednost promenljive koja je prenesena kao parametar. S obzirom da ova osobina može da zada dosta glavobolja, uvedena je i klauzula *ByVal*, kojom se navodi da se određeni parametar prenosi striktno po vrednosti - promena vrednosti ovakvog parametra neće uticati na stanje promenljive pre poziva procedure. Napuštanje procedure odnosno funkcije, postiže se naredbom *Exit Sub* (*Exit Function*). Deklaracija funkcije u *Visual Basic*-u slična je deklaraciji procedure (koristi se ključna reč *Function*, naravno), osim što je u telu funkcije moguće određivanje povratne vrednosti, na način koji se koristi u paskalu: pre naredbe *Exit Function*, fiktivnoj promenljivoj istog imena kao i sama funkcija, dodeljuje se povratna vrednost. Zaglavlje funkcije mora odrediti i tip povratne vrednosti - na primer: *Function Rez(Par As Integer) As Integer*.

Što se podržanih tipova promenljiva tiče, *Visual Basic*, slično svojim starijim rođacima, radi sa celobrojnim promenljivama (kratke - *Integer*, duge - *Long*), racionalnim (kratke - *Single*, duge - *Double*), stringovima (tip *String*, maksimalna dužina 64 kilobajta), a obezbeđen je i tip *Currency*, namenjen poslovnoj aritmetici, kao i tip *Variant* koji može da sadrži vrednost proizvoljnog tipa (slično *Clipper*-ovim promenljivama). Najzad, postoji i mogućnost kreiranja struktura (naredba *Type*) koje sadrže više promenljivih različitih tipova, uz uobičajeni pristup preko imena i tačke (npr. *Radnik.Ime = "Petar"*).

Tipovi promenljivih mogu se navoditi implicitno ili eksplicitno. Predlažem da u *declarations* sekciju svakog programa ugradite *Option Explicit* - tako ćete morati da deklarišete svaku promenljivu (npr. *Dim Ime As String * 50*), ali će vam se taj posao kasnije višestruko isplatiti. Pre svega, program će biti robusniji i brži nego kada promenljive ne deklarišete, a sve reference na promenljive pogrešnog tipa biće prijavljene još pri prevođenju (odn. pre samog starta programa). Najzad, korišćenje jedne promenljive za različite namene je veoma loša navika koja će vaš kod učiniti veoma nečitljivim. *Visual Basic*, istina, dopušta i deklaracije u "preistorijskom" maniru: pozdrazumeva se, recimo, da je promenljiva *Ime$* string, jer je svakom od postojećih tipova pridružen jedan specijalan znak koji mora biti poslednji u imenu promenljive. Stvar je ukusa volite li da program vrvi od specijalnih znakova, ali mislim da je bolje da promenljive eksplicitno deklarišete naredbom *Dim*. Tako bi promenljiva *Ime$* postala samo *Ime*, što je, priznaćete, ipak lepše.

Visual Basic, naravno, podržava i nizove, i to čak u nešto raskošnijoj varijanti od svojih starijih rođaka. Za deklarisanje se koristi naredba *Dim*, pri čemu se specificira veličina niza i tip promenljivih koje ga čine: naredba *Dim A(100) As Integer* deklariše niz od sto celih brojeva (indeksi počinju od nule). Kao i obično, elementima niza pristupa se preko malih zagrada.

Osim što je moguće deklarisati "obične", jednodimenzionalne, VB podržava i višedimenzionalne nizove kojima se mogu zadati donje i gornje granice. Na primer, savršeno je legalno deklarisati matricu stringova čiji indeksi idu od -100 do 100 i od 1 do 31: *Dim Temp(-100 To 100, 1 To 31) As String*. Osim toga, *Visual Basic* podržava i dinamičke nizove koji su u starim interpreterima uvek predstavljani kao "high-tech". Naime, pri deklaraciji niza jednostavno se propusti navođenje granica, a zatim se po potrebi niz redimenzioniše naredbom *ReDim*.

Bez obzira na solidne mogućnosti koje pruža *Visual Basic* na polju nizova, treba voditi računa o jednoj, za korisnika bitnoj stvari: brzini. *Visual Basic* je interpretator: čak i ako ste generisali samostalnu EXE datoteku (opcijom *Make EXE File*), ona sadrži *p-kod* koji se interpretira. Bez obzira što je tehnologija interpretiranja takvog p-koda prilično dobra, treba biti pažljiv pri radu sa velikim nizovima - neracionalno napisana petlja ili predimenzionisani nizovi i stringovi mogu da dovedu vaš program do potpune neupotrebljivosti.

Visual Basic je od predaka u velikoj meri nasledio šarenilo kontrolnih struktura. Ima smisla koristiti svega nekoliko: *Do / Loop* i *For / Next* petlje kao i *If / EndIf* i *Select Case / End Select* strukture.

U dijalogu *Environment* (*Options* meni) može da se uključi opcija *Syntax Checking*. *Visual Basic* će testirati sintaksnu ispravnost svake linije nakon što ste je uneli. Osim provere, vaš kod se uvek "sređuje": imena rezervisanih reči, naredbi i funkcija biće kapitalizovana u skladu sa *Microsoft*-ovim idejama: prvi znak reči piše se velikim slovom, ostalo malim. Ova mogućnost je prilično korisna, osim ako vam u sred pisanja neke linije padne na pamet da promenite nešto u drugom delu programa: *Visual Basic* vam neće dati da idete dalje dok ne ispravite sintaksnu grešku ili barem ne isključite proveru sintakse. Imajte u vidu da se ispravljaju isključivo čiste sintaksne greške; VB se, na sreću (?), neće buniti ako ste upravo uneli liniju u kojoj se pominje nedeklarisana promenljiva - ta greška će biti prijavljena tek pri prevođenju programa.

Korisnički interfejs

Kako smo upoznali koncepciju i osnovne pojmove u *Visual Basic*-u, red bi bio da uradimo nešto konkretnije. Interesantno je da čovek može da napravi i pokrene *Visual Basic* program pritiskom na tri tastera, računajući tu i uključivanje računara, pod pretpostavkom da je WIN poslednja linija u AUTOEXEC-u. Dakle, dovoljno je uključiti računar, startovati *Visual Basic* (Ctrl Alt V, na mojoj instalaciji *Windows*-a), pritisnuti taster F5 ili kliknuti mišem na ikonu sličnu PLAY dugmetu na kasetofonu... vaš prvi *Visual Basic* program je već startovan! On, naravno, ne radi baš ništa korisno, ali to ipak znači da već imamo kostur aplikacije - samo još da napravimo telo.

Komandni interfejs prosečne *Windows* aplikacije oslanja se pre svega na menije i komandne tastere. Ove prve kreirate pritiskom na Ctrl+M kada se otvara prozor namenjen dizajniranju menija. Svakoj stavci u meniju pridružuje se i identifikator - kada korisnik klikne na neku opciju, *Visual Basic* poziva funkciju *IdentifikatorOpcije_Click*. Aktiviranje opcije u meniju je sa programske strane identično pritisku na neki komandni taster.

U skladu sa prvom rečju svog imena, *Visual Basic* dozvoljava da se elementi interfejsa kreiraju tako što se iz palete (sa leve strane ekrana) odabere sličica odgovarajuće kontrole, a zatim i mišem odredi mesto te kontrole na formularu. Nakon pozicioniranja, sa novokreiranim objektom, možete uraditi tri stvari: izbrisati ga (pritiskom na taster Del), podesiti njegove osobine ili dva puta kliknuti mišem na njega kako biste počeli da pišete konkretne procedure koje reaguju na akcije preduzete nad tim objektom od strane korisnika.

Podešavanje osobina objekta može se obaviti iz prozora koji dobijate pritiskom na F4; imajte u vidu da se podešavanje osobina objekta preko ovog prozora (*Properties*) odnosi isključivo na inicijalno stanje. Ipak, neka od svojstava objekta nije moguće promeniti programski, a neka druga ne mogu ni da se postave pri dizajniranju programa - zavisno od prirode objekta.

Ukoliko želite da odredite neku od procedura koje reaguju na događaje, dovoljno je da dva puta kliknete na kontrolu i otvoriće vam se prozor sa editorom, sličan onome na slici. Pri vrhu editorskog prozora imate dve liste: u jednoj birate objekat koji pripada trenutno aktivnom formularu, a u drugom neki od odgovarajućih događaja koje program obrađuje.

U listi objekata je posebno značajna stavka *general* (engl. opšte): sve procedure koje se nalaze u *general* sekciji mogu se koristiti u bilo kojoj proceduri koja spada u "nadležnost" aktivnog formulara. Međutim, procedure iz jednog formulara nisu vidljive u drugim formularima. Da bismo razumeli zašto stvari tako stoje, moramo se upoznati sa pojmom modula.

Moduli

Svaki program u *Visual Basic*-u jeste jedan projekat (datoteka sa nastavkom MAK). Projekat se može sastojati iz više modula različitih tipova.

Moduli formulara (tekstualne datoteke sa nastavkom FRM) su najčešći: u njima je definisan izgled formulara onakav kakvog ste ga "nacrtali", spisak svih kontrola koje formular sadrži zajedno sa njihovim osobinama (*properties*), kao i tekst svih procedura i funkcija koje spadaju u nadležnost tog formulara: sve procedure koje obrađuju događaje sadržane u formularu (razni *Form_Load*, *Command1_Click* i slični), kao i procedure lokalne za taj formular. Recimo da se funkcija *BrojAktivnih()* koristi samo u formularu 'Korisnici', pa ima smisla da se ona nađe upravo u modulu koji se bavi korisnicima (npr. KORISNIC.FRM). Ono što je naročito važno je da su sve procedure, funkcije i promenljive, koje se nalaze u FRM datoteci, lokalne za taj formular - ne može im se pristupiti iz drugih formulara. Promenljivoj je, ipak, moguće dodeliti atribut *Global*, čime ona automatski postaje vidljiva za sve module.

Često su potrebne procedure i funkcije koje se pozivaju iz bilo kog dela projekta. U tom slučaju, dodajemo novi modul (*New Module* iz **File** menija) koji dobija ime Module1.Bas (pri snimanju se ime lako menja). Modul sa ekstenzijom BAS može da sadrži funkcije, procedure i promenljive interesantne za čitav projekat (osim ako nisu eksplicitno proglašene privatnim za taj modul - upotrebom ključne reči *Private*). Tako razvijete modul za, na primer, proračun kamata, a zatim taj modul ubacivati u bilo koji projekat u kome je obračun kamata potreban. Svakako, ako se mehanizam tog proračuna promeni, nećete biti primorani da jurite po disku tražeći ga u sto raznih projekata, već ćete izmenu izvršiti nad samo jednom BAS datotekom.

Moduli sa nastavkom VBX sadrže proširenja *Visual Basic*-a u vidu dodatnih kontrola. Na primer, datoteka TABELA.VBX bi mogla da sadrži mogućnost da na formular postavite tabelu i da programski upravljate njom na o kome smo već nešto rekli (kontrolni tasteri). Iako sam sadržaj VBX datoteka korisniku nije preterano bitan, pomenuću da se zapravo radi o dinamičkoj biblioteci (DLL, *dynamic link library*) koja zadovoljava određene zahteve koje definiše *Visual Basic* API. Za razvoj VBX-ova, neophodan je *Visual Basic Control Development Kit* (VB CDK) koji se dobija uz profesionalno izdanje *Visual Basic*-a. Ako program treba da instalirate na nekom drugom računaru, osim same EXE datoteke morate isporučiti i sve VBX datoteke koje program koristi - njihovo prikupljanje automatizuje *Setup Wizard*. Analizirajući projektnu datoteku, *Setup Wizard* pravi spisak svih potrebnih fajlova, i kreira instalacione diskete.

I pored načelne orijentacije ka objektima i klasama, *Visual Basic* je daleko od pravog objektno-orijentisanog jezika. Pre svega, ne mogu se kreirati nove klase, što je ogroman nedostatak - ostaje samo da se nadamo da će u budućnosti biti ispravljen. Nasleđivanje postojećih klasa je moguće na rudimentarnom nivou, tek toliko da zadovolji u primenama zbog kojih je napravljeno - realizacija *event driven* pristupa pri radu sa korisničkim interfejsom. Bilo kako bilo, *Visual Basic* je pravi jezik za početnike u svetu *Windows* programiranja, jer najbrže dovodi do rezultata! U svetu u kakvom živimo, to je očigledno najvažnije.