Excel Expert i VBA

Podsetnik Excel funkcionalnosti

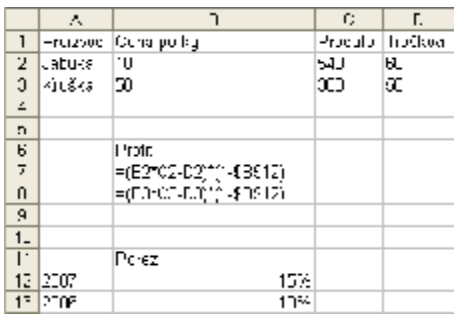
Relativne i apsolutne reference

	A	B	C	D
1	Proizvod	Cena po kg	Prodato	Troškovi
2	Jabuka	10	540	60
3	Kruška	50	300	50
4				
5				
6		Razlika		
7		=B2*C2-D2		
8		=B3*C3-D3		

U formulama možemo referisati druge ćelije relativno ili apsolutno. Relativno referisanje znači da samo navodimo ime ćelije u formuli (npr. B2*C2 – D2). Kada kopiramo formulu sa relativnim referencama na drugo mesto, adrese ćelija se pomeraju za onoliko za koliko se odredište kopiranja pomera od originala. Npr. u primeru na slici – kada kopiramo formulu iz ćelije B7 (koja sadrži reference na drugi red) u ćeliju B8 – sve reference u kopiranoj formuli (B8) će se promeniti tako da pokazuju na treći red.

	A	B	C	D
1	Proizvod	Cena po kg	Prodato	Troškovi
2	Jabuka	10	540	60
3	Kruška	50	300	50
4				
5				
6		Razlika		
7		=$B2*$C$2-$D2		
8		=$B3*$C$2-$D3		

Apsolutne reference se definišu isto kao i relativne s tim što imaju predznak $ ispred oznake kolone odnosno reda. Npr. $A$1 znači apsolutna referenca na kolonu A i red 1. Ako želimo možemo samo fiksirati kolonu: $A1, ili samo red: A$1. Kada kopiramo formule koje sadrže apsolutne reference, reference se ne menjaju bez obzira gde je odredište. U primeru na slici, formula je iskopirana iz ćelije B7 u ćeliju B8 (primetite za koje reference se menja broj reda a za koje ne). (Savet: da bi relativne reference prebacili u apsolutne, selektujte ih i pritisnite F4).



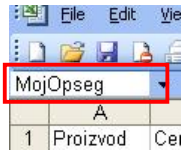
Postoji slučaj kada apsolutne reference ne funkcionišu kako bi možda očekivali. Npr. u tabeli prikazanoj na slici, imamo da se profit računa na osnovu najnovije vrednosti poreza. Kada bi sada dodali novu vrednost poreza za 2008 godinu (desnim dugmetom na 12-ti red, pa *Insert*), bez obzira što smo u formuli u ćeliji B7 fiksirali vrednost $B$12, on će se promeniti da bude $B$13. Ovakva situacija se rešava upotrebom funkcije INDIRECT koja vraća referencu na ćeliju čiji naziv smo naveli pod navodnicima. Npr, u primeru na slici umesto $B$12 bi trebalo da stoji: INDIRECT(„B12“). Sada kada se ubacuje novi red, Excel „neće videti“ da smo imali referencu na B12 zato što je ona u navodnicima (tj. nije direktna referenca) i neće je ni menjati. Ali naša formula funkcioniše zato što se prilikom izračunavanja vrednosti INDIRECT funkcije „B12“ menja sa referencom na B12.

Korišćenje imenovanih opsega (*Range names*)

Opsezi se koriste za lakše snalaženje u radnim listovima: npr, umesto da koristite opseg C2:C45, možete samo koristiti „Prodato“ (tj. Range name koji smo prethodno definisali za opseg C2:C45).

Ime opsega mora počinjati isključivo slovom i mora se sastojati od spojenih karaktera (ako želite da razmaknete dve reči koristite znak „_“). Ne možete imenovati ćeliju tako da se podudara sa postojećim imenima (npr. A1, B58, IS2002 i sl.) U jednom radnom listu može postojati neograničen broj imenovanih opsega, ali ne treba preterivati jer ćemo teže naći opseg koji nam treba i time poništiti njegovu upotrebljivost.

Kreiranje opsega: selektujte jednu ili više ćelija i promenite njihov naziv u posebnoj ćeliji kao na slici. Drugi način: ctrl+F3 (ili *Insert->Name->Define*) Define Name prozor – u ovom prozoru takođe možemo i brisati postojeće opsege.



Korišćenje opsega: u formuli na mestu gde treba ukucati opseg samo ukucamo ime našeg opsega (ili F3 da bi prikazali listu svih definisanih opsega, zatim izaberemo željeni opseg).

Nekad je korisno defnisati opseg tako da zapravo bude konstanta: npr. PDV koji će biti 0,18. Ovakav „opseg“ možemo kasnije koristiti u formulama umesto korišćenja konstante ili referisanja neke ćelije. Da bi definisali konstantu otvorimo *Define Name* prozor (ctrl+F3) i u polje *Refers to* ukucamo željenu konstantu. Primer korišćenja u formuli: =B18 * (1 – PDV).

Da bi videli sve opsege, njihove vrednosti ili gde one referišu, na praznom listu uradite: F3 -> Paste list.

Ono što se verovatno najčešće koristi je automatsko imenovanje opsega. Selektujte opseg u tabeli (selektujete jednu ćeliju i zatim ctrl+a). Pritisnite ctrl+shift+F3 (ili *Insert->Name->Create*) i izaberite odakle Excel treba da izvuče imena opsega (red, kolona, prvi ili poslednji red/kolona...).

Validacija unosa

Validaciju koristimo kada hoćemo da obezbedimo da budući unos na radnom listu zadovoljava neke unapred zadate kriterijume.

Npr, ako želimo da ograničimo unos u ćelijama B2:B30 da budu pozitivni celi brojevi do 100. Selektujemo ćelije B2:B30, u meniju biramo Data->Validation->Whole number, between, min/max vrednost, pri čemu min i max mogu da budu reference na neke ćelije ili imenovane opsege.

Drugi koristan primer bi bio da recimo ograničimo unos duplih vrednosti (npr, za unos liste zaposlenih). Za datu selekciju izabraćemo Custom validaciju, i u prostor za kucanje formule ukucamo:

=COUNTIF($B$2:$B$30; B2) = 1

AutoFilter

Da bi aktivirali AutoFilter moramo imati selektovanu ćeliju koja pripada nekom opsegu. Kada smo selektovali, kliknemo Data->Filter->Auto Filter. Dobićemo za svaku kolonu jedan „dropdown“ meni iz kog možemo birati više opcija koje su generalno gledano samoobjašnjive.

Ograničenje autofilter liste je što može da prikaže samo 1000 redova, i što uopšte gledano za velike liste postaje nepregledno izabrati samo jednu vrednost u listi. Da bi ovo prevazišli poslužićemo se trikom grupisanja vrednosti. Npr, ako imamo liste imena i prezimena (ili uopšte nekog teksta) dodaćemo novu kolonu koja će da izdvaja samo prvo slovo od vrednosti iz prve kolone: =LEFT(A2; 1).

Još jedna zanimljiva opcija je da dodamo jednu kolonu koja će nam služiti za pretraživanje liste. Npr, ako nam se vrednosti od interesa nalaze u koloni A, dodaćemo kolonu B bez naziva kolone. Vrednosti u koloni B ćemo izračunavati po sledećoj formuli (ovde je data formula za ćeliju B2):

=IF(ISNUMBER(SEARCH($B$1;A2)); "Nadjen"; "Nije nadjen")

(ako želite pretragu osetljivu na mala i velika slova, koristite funkciju FIND umesto SEARCH).

Sada, kada ukucamo neku vrednost u ćeliju B1, kolona B će se popuniti vrednostima „Nadjen“ i „Nije nadjen“ u zavisnosti od toga da li je ta vrednost pronađena u odgovarajućem redu A kolone. Ako imamo uključen Auto format sada možemo birati da li želimo da prikažemo one redove koji sadrže ili ne sadrže ovu vrednost, čime smo efektivno napravili pretraživanje po koloni na našem radnom listu.

Sumiranje vrednosti u auto filter listi se razlikuje u zavisnosti od toga da li koristimo SUM ili SUBTOTAL funkciju. SUM će ignorisati prikaz filtera i uračunaće sve vrednosti koje su date u opsegu. SUBTOTAL će računati samo redove koji su prošli filter.

Uslovno formatiranje

Sa uslovnim formatiranjem (conditional formatting) možemo istaći neke određene vrednosti – npr. da li su neke vrednosti ispod dozvoljene i sl.

Selektujemo oblast za koju želimo da primenimo uslovno formatiranje, zatim iz *Format* menija izaberemo *Conditional Formatting*. Zadamo kriterijum za formatiranje, i izaberemo format teksta koji treba primeniti ako je uslov zadovoljen. Npr. „Cell Value Is - less than - 3000“ će primeniti format koji izaberemo na sve ćelije koje imaju brojnu vrednost manju od 3000. Slično možemo dodavati i više uslovnih formata.

Ako imamo nazive kolona ili redova koji su zapravo brojevi (npr. godina – 2008), da bi sprečili da uslovno formatiranje utiče i na njih trebalo bi da ih promenimo tako što ćemo ih uneti sa jednim apostrofom pre broja (‘2008). Time smo označili da je ćelija tipa teksta i njena vrednost će se ignorisati pri poređenju sa brojevima u uslovnom formatiranju.

Pored osnovnih uslova za formatiranje moguće je uneti i uslov u obliku formule. Iskoristićemo ovo na primeru senčenja parnih redova (npr. zbog lakšeg čitanja u štampanoj verziji). Kao kriterijum biramo „Formula Is“, a ukucavamo sledeću formulu: =MOD(ROW(); 2) = 0. U ovoj formuli MOD daje ostatak pri deljenju sa 2, a ROW() daje broj reda za koji Excel trenutno ispituje da li zadovoljava uslov formatiranja. Ukoliko je ovaj red neparni (npr, 3, 5, 7 itd…) pri deljenju sa 2 ostatak će biti 1 i neće proći uslov. Ako je paran – uslov je zadovoljen (zato što je jednako nula) i primenjuje se uslovno formatiranje.

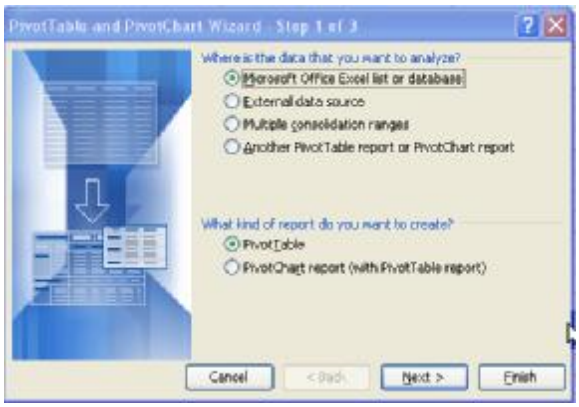
Dinamički opseg

Da bi napravili dinamički opseg za rad, umesto definisanja fiksnog opsega („ćelije od – do“ princip), možemo kreirati region koji će se automatski proširivati kako budu dodavani redovi odnosno kolone. Definicija regiona je data tekstom formule:

=OFFSET(Trgovina!$A$1,0,0,COUNTA(Trgovina!$A:$A), COUNTA(Trgovina!$1:$1))

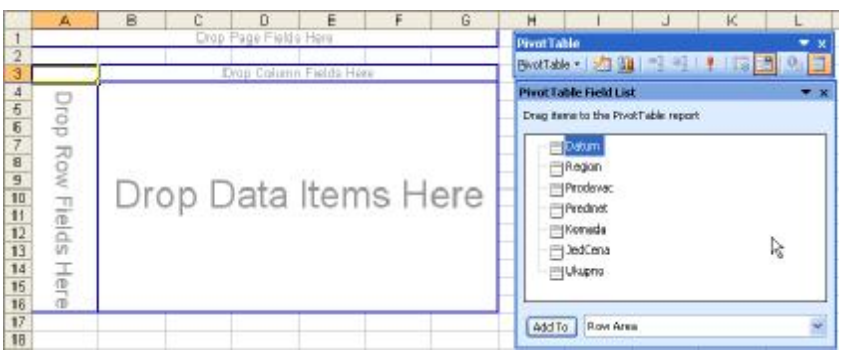
Prva vrednost OFFSET funkcije je referenca na prvu-levu ćeliju opsega, druga i treća treba da budu 0, a pretposlednja i poslednja definišu koliko opseg ima redova i kolona (respektivno). Da bi se ovaj broj stalno uvećavao kako se redovi i kolone menjaju (dodaju ili brišu) korišćena je funkcija COUNTA (*count all*), koja će brojati koliko ima popunjenih redova u koloni A, odnosno kolona u redu 1. Zbog ovoga ne smemo dodavati neke podatke na ovaj list ako ne pripadaju našem opsegu (ili bar ih treba čuvati tako da se ne nalaze u prvoj koloni/redu).

Pivot tabela, Pivot grafik



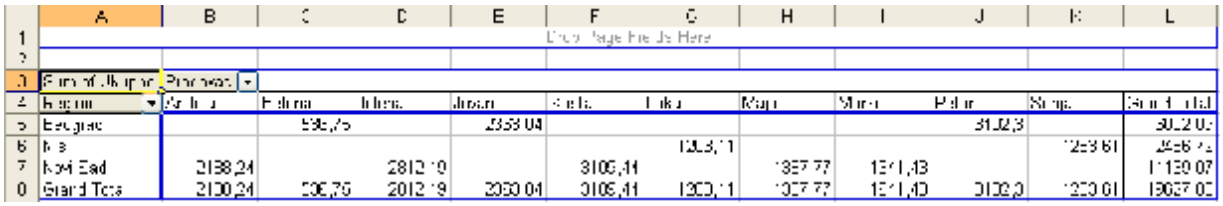
Pivot tabela je veoma koristan alat za brzo sumiranje i pregled podataka. Pivot tabelu kreiramo iz menija *Data->PivotTable and PivotChart Report*. Počinje Wizard koji nas vodi kroz definisanje podataka neophodnih za Pivot tabelu.

U prvom koraku biramo odakle želimo da dohvatimo podatke – ako su podaci u Excel-u (najčešći slučaj) biramo prvu opciju. U drugom koraku izaberemo opseg u kome se podaci nalaze, i nakon toga konačno biramo i gde želimo da se tabela kreira (na istom ili na novom listu).



Kreiranje sadržaja PivotTabele se vrši prostim prevlačenjem naziva kolona na željeno mesto. Npr. u primeru koji je podeljen na vežbama probajte da prevučete „Region“ u *Row*

Fields deo, i „Ukupno“ u *Data Items*. Ovako smo dobili listu po regionima i ukupne ostvarene promete po svakom gradu. Ako želimo da vidimo koji prodavac je prodavao gde, prevući ćemo kolonu „Prodavac“ u *Column Fields*. Na kraju, lista bi trebalo da izgleda ovako:



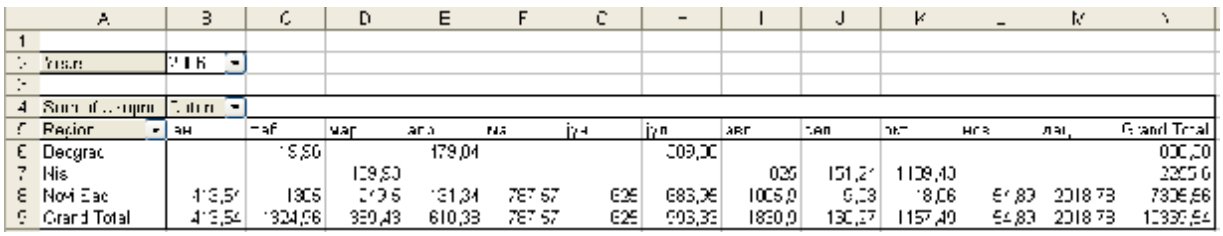
Da bi dobili prikaz tabele po gradovima i listu proizvoda koji su tamo prodati, prevući ćemo „Region“ u *Page Fields*, a u *Row Fields* ćemo sad staviti „Predmet“. Rezultat bi trebalo da bude sličan kao na slici.

	A	B	C	D	E
1	Region	Beograd			
2					
3	Sum of Ukupno	Prodavac			
4	Predmet	Helena	Jovan	Petar	Grand Total
5	Gumica		565,22	1183,26	1748,48
6	Heftalica	57,71	858,76	1619,19	2535,66
7	Hemijska	479,04	575,36	299,85	1354,25
8	Olovka		363,7		363,7
9	Grand Total	536,75	2363,04	3102,3	6002,09

Promenom regiona biramo prikaz po gradovima (ovde su prikazani podaci za Beograd).

Duplim klikom na „Sum of Ukupno“ možemo promeniti prikaz u procentualni. Prvo proširimo prikaz više opcija (klik na dugme *Options…*), i izaberemo *„% of row“*.

Grupisanje podataka: Dodajte kolonu „Datum“ deo sa kolonama a uklonite „Prodavac“. Da bi prikazali prodaju po mesecima i godini, kliknućemo desnim dugmetom na kolonu „Datum“*->Group and Show Data->Group…* U ponuđenoj listi selektujemo i Months i Years. Dobili smo nov izgled tabele grupisan po vrednostima za svaki mesec svake godine. Radi bolje preglednosti, prevući ćemo *Years* da bude u *Page Field* delu. Izgled tabele bi trebalo da izgleda slično kao na slici (izabrana je 2006 godina u *Page Field* delu).



Na sličan način možemo grupisati i druge tipove podatka (npr. po broju godina, po koraku od 10 godina bi dobili grupe 10-20 god, 20-30, itd...).

Da bi od Pivot tabele dobili Pivot grafik, dovoljno je prvo kliknuti na neku ćeliju tabele, zatim na Chart Wizard ikonicu kao na slici.

Rad sa nizovima (*Array*)

Nizovi (*Array*) predstavljaju niz od dve ili više vrednosti. Npr. kada koristimo funkciju za sumiranje: `SUM(A1:A50)` opseg od `A1:A50` predstavlja jedan niz vrednosti.

Postoje neke formule koje ne mogu da se izračunaju na standardan način, odnosno red po red. Npr:
`=SUM(A1:A50*B1:B50)`

Funkcija SUM bi mogla da izračuna ovaj izraz kada bi to bio samo jedan opseg. Međutim, ovde ima dva opsega i nije najjasnije koje vrednosti iz prvog opsega se množe sa kojim vrednostima iz drugog opsega. Da bi ovo obradili, formulu je potrebno uneti umesto sa standardnim pritiskom tastera **enter** na kraju, sa **ctrl+shift+enter**. Na ovaj način, specificirali smo da je ovo *Array* formula, i ona se od ostalih razlikuje po tome što Excel ispisuje vitičaste zagrade („{“ i „}“) oko tela formule. Funkcionisanje array formule (na prethodnom primeru): svaki opseg se obrađuje red po red, odnosno, prvo će se raditi sa prvim redom oba opsega (pomnožiće se vrednosti A1 i B1), zatim sa drugim itd... Dobijeni rezultati se interno u Excel-u smeštaju u nov Array na kome se konačno može primeniti formula SUM. Očekivano, opsezi moraju imati jednak broj redova – u suprotnom dobićemo grešku (*#N/A*).

Česta greška je da se posle izmene Array formule ista ne potvrdi (ne unese) sa *ctrl+shift+enter*, već samo sa *enter*, što će uslediti greškom (*#VALUE*). Da ispravite ovu grešku, kliknite na ćeliju, pritisnite F2 da bi menjali njen sadržaj i potvrdite je sa *ctrl+shift+enter*. Array formulu ne možete uneti tako što ćete ručno upisivati zagrade oko vaše formule - `{=SUM(A1:A50*B1:B50)}` - Array formule se mogu unositi samo na prethodno opisani način.

„Greška“ u sabiranju

Ukoliko radite sa decimalnim vrednostima, a u tabeli ih prikazujete ih kao celobrojne, možete doći u situaciju da se brojevi nekorektno sabiraju. Posmatraćemo na primeru sledeće tabele:

	A	B	C
1	Decimalne vrednosti	Formatirane kao celobrojne	
2	1,49	1	
3	1,49	1	
4	2,98	3	=B2+B3

Prikazano je kao da je 1+1=3. Problem je nastao zbog toga što će Excel zaokružiti decimalne vrednosti prilikom njihovog prikazivanja i to na najbližu vrednost i tu vrednost će prikazati. Međutim, prilikom računanja koristi se prava vrednost (a ne prikazana), odnosno – u ćeliji B4 na prikazanom primeru, izračunata vrednost je 2.98 bez obzira što se sabiraju vrednosti čiji je prikaz u formatu celih brojeva. Naravno, kada se ta vrednost od 2.98 zaokruži na najbliži celi broj, dobićemo broj 3. Ovo za velike liste smanjuje grešku pri sabiranju, ali postoje slučajevi (pogotovo za male tabele) kada nam je potrebno da sabiramo samo vrednosti koje su zaokružene. Za rešenje ovog problema postoje dva načina.

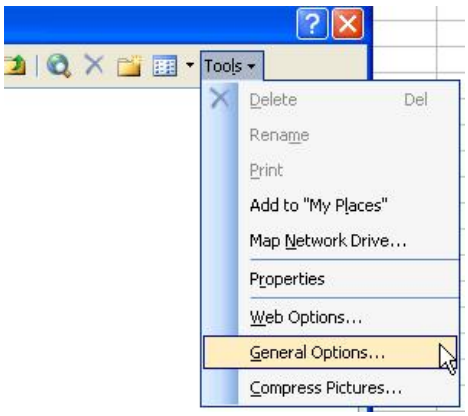
1. Izaberite Tools->Options->Calculations i štiklirajte opciju „Precision As Displayed“. Ovime se sve vrednosti u ćelijama odsecaju na onoliko decimala koliko je definisano formatom ćelije. Ovu akciju nije moguće kasnije poništiti.
2. Koristite Array formule sa zaokruživanjem – za gornji primer: {=SUM(ROUND(B2:B3;0))}.

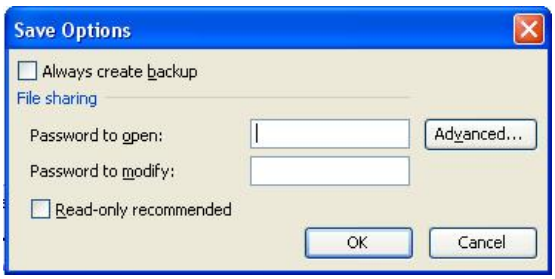
Zaštita radnog lista

Sve ćelije su po *default*-u zaključane (*locked*). Međutim, ovo postaje aktivno tek kada se aktivira zaštita radnog lista (*worksheet*). Pre zaštite, na probnoj tabeli, selektujte nekoliko ćelija i u *format cells* meniju, u poslednjoj stavci (*Protection*) odštiklirajte Locked, a na drugom opsegu štiklirajte *Hidden* da bi demonstrirali upotrebu ove opcije. Kada ste završili aktivirajte zaštitu lista: *Tools->Protection->Protect Sheet*. Izaberite opcije koje želite da dozvolite/zabranite, opciono unesite šifru za zaštitu i kliknite OK. Sada je onemogućena izmena ćelija koje su zaključane, a ne vidi se sadržaj formula u ćelijama koje imaju štiklirano „Hidden".

Moguće je posle zaštite dozvoliti izmenu nekih opsega po korisnicima i eventualno uz šifru. Korisnici moraju biti *user*-i Windows operativnog sistema na kome se pokreće Excel (ili članovi Domena ukoliko se radi o mrežnom okruženju). Ovo je dostupno preko opcije: *Tools->Protection->Allow Users To Edit Ranges*. Za svakog korisnika možemo definisati posebnu šifru, i možemo za isti opseg ćelija dodati više od jednog korisnika u listu.

Zaštita celog dokumenta se vrši prilikom njegovog čuvanja. File->Save As. Izaberite Tools->General Options





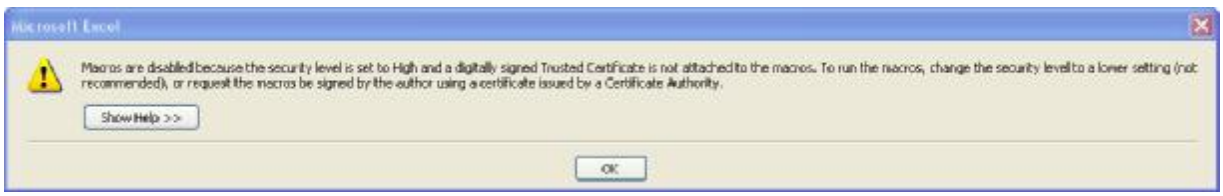
Unesite polja „Password to open" i „Password to modify" u zavisnosti od toga da li želite da zaštite dokument od otvaranja ili od izmena.

Napomena: Prilikom korišćenja zaštite radnog lista, ne smete računati na pouzdanost zaštite – svaka zaštita šifrom se može probiti, samo je pitanje vremena. Zaštita ima smisla ako želite da zaštitite podatke od slučajne izmene ili nasumičnog pokušaja zloupotrebe od običnih korisnika, ali ako neko zaista želi da otvori vaš dokument ili da vrši izmene, uvek postoji način (ovo vam zapravo može biti korisno u slučajevima kad zaboravite šifru).

VBA / Makroi

Makro je sekvenca instrukcija kojima se automatizuje neki proces. Na primer, jedan makro bi mogao da bude: postavi pozadinsku boju ćelije na zeleno, postavi boju teksta na belo, podesi font da bude veličine 12. Ako primetimo da neki proces stalno ponavljamo najverovatnije je da bi nam se isplatilo da napravimo makro koji jednom snimimo i kasnije možemo da ga koristimo svaki put kad nam to zatreba. U Excelu makroi se pišu koristeći varijantu jezika Visual Basic (VBA = Visual Basic for Applications). VBA je standardni jezik za više Microsoft-ovih proizvoda, i principe koje naučite u Excelu možete primenjivati npr. u Wordu, itd… Takođe, makroi se često koriste i u mnogim drugim aplikacijama (Photoshop npr – Window->Actions, itd…) mada ne koriste VBA već imaju svoju sintaksu.

Nakon default instalacije MS Office-a, makroi koje otvaramo sa dokumentima ne funkcionišu. Ovo je učinjeno sa razlogom - ranije su se često upisivali virusi u obliku makroa i ubacivali u dokumente. Samo otvaranje takvog dokumenta bi zatim izazvalo aktiviranje virusa i zbog toga su proizvođači odlučili da onemoguće pokretanje makroa osim ako korisnik to eksplicitno ne želi. Ako su vam podešavanja takva da je pokretanje makroa onemogućeno ispisaće vam se sledeća poruka prilikom pokretanja makroa:



Međutim, ovo važi samo za makroe koje dobijamo iz drugih dokumenata – makroe koje sami pišemo (koje smo sami kreirali) su podrazumevano sigurni i Excel će vam dozvoliti da ih pišete i pokrećete.

Ako želite da smanjite nivo sigurnosti da bi omogućili pokretanje tuđih (nepotpisanih) makroa, koristite opciju: *Tools->Options->Security->Macro Security*. Po *default*-u izabran je nivo bezbednosti *High* – spustite na *Medium* ako želite da vas pita pre svakog pokretanja makroa (što je i preporuka ovde), ili na *Low* ako želite da svi makroi mogu da se izvršavaju bez vaše prethodne saglasnosti. Da bi promene koje ste ovde napravili postale aktuelne morate zatvoriti i ponovo otvoriti Excel.

Snimanje makroa

Najjednostavnija upotreba makroa je snimanje akcija korisnika. Ne morate da poznajete VBA da bi mogli da ga koristite. Da ne bi smo previše koristili Tools->Macro meni dodaćemo novi *Toolbar*: kliknite desnim dugmetom na *Toolbar* deo i izaberite *Visual Basic* (*Toolbar* su sve ikonice/dugmići koji se najčešće nalaze odmah ispod padajućeg menija – *Font, New, Open, Save*, itd…). Za sad ćemo koristiti samo prva dva: pokretanje i snimanje makroa. Treće dugme pokreće podešavanja za bezbednost (isto kao Tools->Options->Security), a ostale dugmiće ćemo koristiti nešto kasnije.

Izaberite jednu ćeliju. Kliknite na dugme za snimanje i pojaviće se prozor u kome definišete podatke o makrou:

- ***Macro Name***: Naziv makroa. U našem primeru nazvaćemo ga "Zeleno"
- ***Shortcut key***: Prečica za pokretanje makroa. Kliknite na ovo polje, držite Shift i pritisnite "L" (ili neko drugo slovo). Sledeći put kad budemo hteli da pokrenemo makro, dovoljno je da pritisnemo ctrl+shift+L (ne moramo da biramo naš makro iz menija).
- ***Store macro in***. Ovde definišemo gde želimo da snimimo naš makro. Postoje tri opcije:
    - *This workbook* – makro će se snimiti u workbook-u u kome radimo, i biće dostupan samo dok je on otvoren
    - *New Workbook* – napraviće se novi workbook, i makro će se snimiti u njega
    - *Personal Macro Workbook* – makro će se sačuvati u posebnom fajlu personal.xls. Ovaj fajl je poseban po tome što se nalazi u /XLStart folderu (najčešće to je putanja: *C:\Documents and Settings\myUsername\Application Data\Microsoft\Excel\XLSTART*), što znači da se otvara automatski pri pokretanju Excel-a. Excel ovaj dokument otvara i automatski primenjuje *Hide* za workbook (podsetnik: da sakrijete/prikažete otvorene *workbook*-ove koristite meni: *Window->Hide/Unhide…*). Osim toga, ovo je sasvim obična Excel tabela, baš kao i bilo koja druga. U Personal.xls čuvamo sve makroe koji nisu specifične prirode, tj. koje možemo da primenjujemo na više dokumenata.

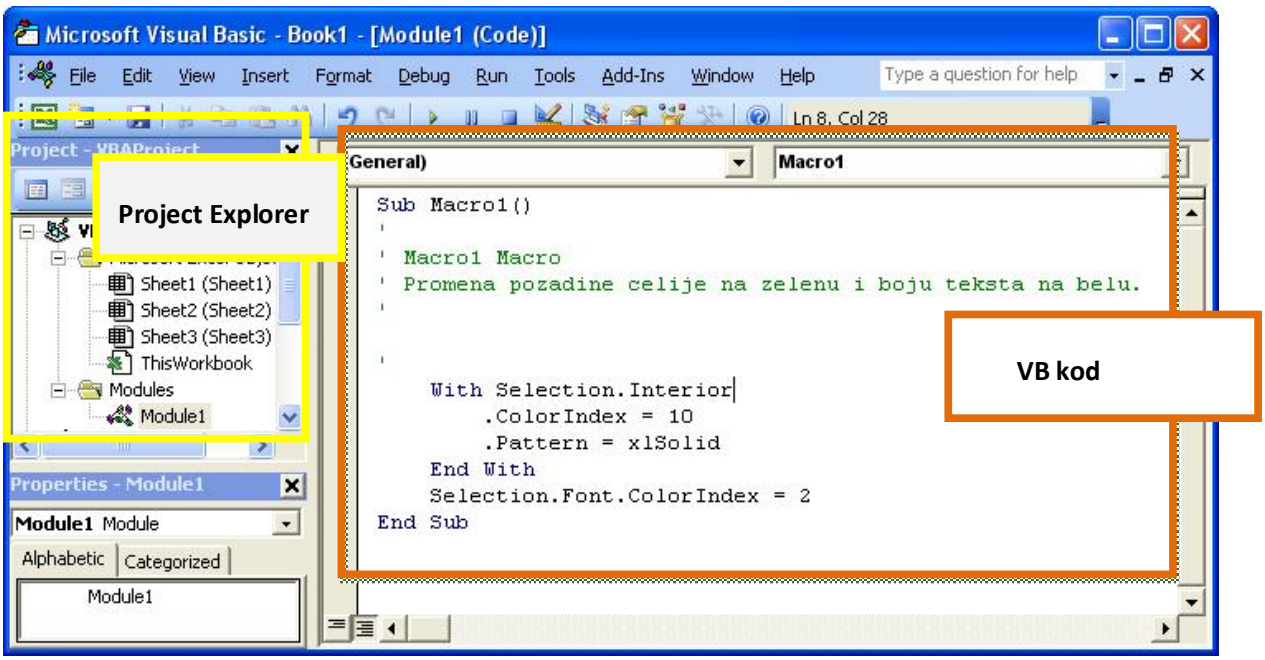
Kad ste završili sa definisanjem makroa, počinje snimanje makroa: promenite pozadinu ćelije u zelenu, promenite boju teksta u belu, i pritisnite stop dugme. Upravo smo napravili makro koji će izabranu ćeliju promeniti tako da ima zelenu pozadinu i beli tekst. Da bi pokrenuli makro, prvo selektujte neke ćelije koje želite da promenite, kliknite na dugme koje podseća na "Play", izaberite makro "Zeleni" i zatim *Run*. Naša selekcija se promenila tako da odgovara onome što smo definisali kad smo snimali makro.

Da bi videli koje akcije smo snimili u makro, u meniju za pokretanje makroa (dugme Play, ili Alt+F8, ili *Tools->Macro*) izaberite željeni makro i zatim "Edit" (ako ste snimali u *Personal Macro Workbook* moraćete da ga prvo učinite vidljivim – *Window->Unhide*). Otvoriće se Visual Basic Editor (VBE) u kome su prikazani makroi svih dokumenata koji su trenutno aktivni. U ovom prozoru možemo menjati kod koji smo prethodno snimili, ali i pisati nov kod.

Treba primetiti da smo prvo selektovali ćeliju, a zatim počeli snimanje makroa. Da smo slučajno uradili drugačije selekcija ćelije bi se upisala u makro, i svaki put kad pokrenemo makro prvo bi se selektovala ta ćelija, a zatim bi se radile ostale akcije.

Visual Basic Editor

Ovo je program koji je zaseban i ne zavisi od Excel-a ali "tesno sarađuju". Da bi pokrenuli ovaj editor možemo izabrati u Excel-u: *Tools->Macro->Visual Basic Editor*, ili Alt+F11 (ili, kao što je opisano ranije – kada želimo da izmenimo neki makro i kliknemo na "Edit"). Prikazaće se sledeći prozor:



U ovom prozoru je prikazan kod svih trenutno otvorenih dokumenata. U *Project Explorer*-u se nalazi prikaz svih otvorenih dokumenata, grupisanih u *VBAProject* (jedan workbook = jedan VBAProject). Svaki projekat sadrži *Microsoft Excel Objects* u kome su svi spreadsheet-ovi, kao i ceo Workbook. Drugu grupu čine moduli (*Modules*). Moduli su "kutije" (eng. *containers*) u kojima se čuva kod. Videćete da se naš makro upisao u jedan modul (Module1). Kod možemo pisati ili u modulima ili vezano za određeni Sheet.

Treba razlikovati tri vrste koda koje ćemo pisati:

1. Procedure
2. Funkcije
3. *Event code* – odnosno, kod koji se izvršava kada nastupi određeni događaj (dupli klik, otvaranje dokumenta, promena selekcije, itd…)

Svaka vrsta ima svoje specifičnosti zbog kojih se drastično razlikuje od drugih.

Procedure

Procedure možete shvatiti kao komande koje se izvršavaju bilo na zahtev korisnika ili iz neke druge procedure. Primer procedure bi bio: "osenči ćeliju da bude siva, upiši neki tekst", i sl. U opštem slučaju

procedure se koriste kada treba nešto napraviti, promeniti, kreirati… Za razliku od funkcija, procedure nemaju povratnu vrednost.

Primer:

```
Sub Macro1()
' Promena pozadine celije na zelenu i boju teksta na belu.
    With Selection.Interior
        .ColorIndex = 10
        .Pattern = xlSolid
    End With
    Selection.Font.ColorIndex = 2
End Sub
```

Funkcije

Funkcije se pišu veoma slično kao procedure, međutim, funkcije uvek vraćaju jednu vrednost (baš kao i funkcije u Excel-u – npr. SUM, COUNT, itd…). Možemo ih koristiti unutar drugih funkcija/procedura, ili u radnim listovima. **Funkcije koje koristimo u worksheet-u ne mogu da ga menjaju!** Ako pokušamo da izmenimo worksheet bilo gde unutar funkcije ona će vratiti grešku (iako funkcije koje pozivamo iz procedura mogu da menjaju izgled worksheeta, ta praksa se ne preporučuje).

Primer:

```
Function Saberi(a1, a2)
      'Sabiranje brojeva a1 i a2
      Saberi = a1 + a2 'vraćanje vrednosti funkcije vršimo tako što imenu funkcije
                       'dodelimo vrednost
End Function
```

Event Code

To je kod koji će se izvršavati svaki put kada nastupi neki događaj (*event*=događaj). Po suštini, event code su procedure. Npr. možemo napisati kod koji se izvršava svaki put kad korisnik promeni selekciju ćelije (tj. selektuje neku drugu ćeliju), u trenutku pre štampe, kad se aktivira list, itd... *Event* kod se ne može pisati u modulima, nego samo u kodu koji je predviđen za worksheet/workbook (ali naravno – možemo odatle pozivati druge procedure/funkcije koje možemo smestiti bilo gde, pa i u modul).

Primer:

```
Private Sub Workbook_Open()
    MsgBox "Dobrodosli u workbook sa makroima"
End Sub
```

Zaključak:

Kada počinjete da pišete kod, i kad odlučujete šta vam treba, koristite funkcije ako vam treba nešto da se **izračuna**; ako treba nešto **uraditi/promeniti** – koristite procedure; ako treba **reagovati** na neke događaje onda pišite event code. Ali čak i ako pogrešite i počnete npr. da pišete funkciju a treba vam procedura – izmene je lako napraviti zato što je “telo”, tj. sadržaj koda isti i dovoljno je promeniti zaglavlje gde se definiše procedura odnosno funkcija.

Objekti

Objekti

Objekti čine najbitniju stavku u VBA i od suštinskog je značaja pravilno ih shvatiti. Svaki objekat definiše svoja ponašanja (tj. metode) i osobine (tj. podešavanja). Objekti su npr. Application, Worksheet, Workbook, selektovani opseg ćelija, jedna ćelija, itd… Osobine npr. ćelije bi bili okviri (borders), kom redu pripada, kojoj koloni, sadržaj ćelije, koja je pozadinska boja, boja fonta, veličina itd…; ponašanje: aktiviraj ćeliju (tj. da ćelija dobije fokus), dodaj komentar, uradi merge, obriši sadržaj, itd…

Kolekcije

Objekti iste vrste se nekad smeštaju u kolekcije: npr. više *Worksheet*-ova smeštamo zajedno u *Worksheet**s*** kolekciju, više ćelija u *Cells*… Da bi dobili određeni objekat iz kolekcije objekata navodimo njegovo ime u zagradama, npr za određeni *worksheet*: *Worksheets("MojSheet")*. Sada na dalje možemo raditi sa ovim izrazom potpuno iste akcije koje bi inače radili na jednom *sheet*-u.

Hijerarhija

Svaki objekat pored svojih osobina i ponašanja može sadržati druge objekte – npr. Workbook sadrži Worksheet objekte, Worksheet dalje ima Range objekte (opseg ćelija), koji opet ima ćelije itd… Da bi pristupili nekom objektu u VBA kodu, specificiramo njeno mesto u hijerarhiji koristeći znak "." (tačka). Na primer:

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

Ako izostavimo početak hijerarhije podrazumevaće se trenutno aktivni objekat. U prethodnom primeru, ako izostavimo prva dva objekta i napišemo:

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

to znači da referišemo Workbook koji je trenutno aktivan. Ako izostavimo i Worksheets("Sheet1"), znači da radimo na trenutno aktivnom sheet-u i možemo samo pisati Range("A1"). Ovo je korisno zato što isti makro možemo koristiti na više worksheet-ova, u zavisnosti od toga koji je trenutno aktivan.

Programiranje

Osnovni principi

Linija koda može biti proizvoljno dugačka. Ako želimo možemo je nastaviti u sledećem redu ako red prekinemo sa znakom „_" . VB će onda taj znak ignorisati i preći će da obrađuje sledeći red kao da je u istom redu sa prethodnim. Na primer:

```
varA = Cells(1,2).Value * Cells(3,3).Value – Cells(3,4).Value * _
       Cells(5,5).Value / _
       Cells(2,2).Value - 3
```

Obratiti pažnju na uvlačenje u svim sledećim primerima – nije obavezno ali veoma pomaže da lakše sagledamo koji red pripada gde (da li je „ugnježden“ ili ne, da li se nastavlja od prethodnog reda itd…).

Komentari koje dodajemo se ne izvršavaju. Svaki komentar počinje znakom apostrofa (’) i važi do kraja linije. Drugim rečima, VB će ignorisati sve što piše posle apostrofa. Zbog toga, možemo pisati nešto ovako:

```
Sub Macro1()
' Komentar koji stoji samostalno u redu
    With Selection.Interior ' komentar iza linije koda
        .ColorIndex = 6  ' Boja 6 je žuta boja – opisujemo kod koji nije najjasniji
        .Pattern = xlSolid
    End With
    Selection.Font.ColorIndex = 3 'Boja teksta – crvena (tj. 3)
End Sub
```

Dodela vrednosti se vrši prostim operatorom “=”. U odredište koje je sa leve strane jednakosti se upisuje izračunata vrednost sa desne strane. Uvek se prvo izračuna desna strana jednakosti, a zatim se to upisuje u odredište. Na taj način, možemo zapisati:

```
varA = 0 'u varA upisujemo vrednost nula (0)
varA = varA + 1 'u varA upisujemo vrednost: varA+1, tj. 0+1, tj. 1
```

U ovom primeru, matematički gledano poslednji red je netačan / neispravan, ali kao što smo rekli ranije – prvo se izračuna vrednost sa desne strane jednakosti, i zatim se to upiše u varA. Leva strana jednakosti nas ne interesuje do samog kraja kada treba u nju da upišemo vrednost.

Kada želimo da dodelimo promenljivoj vrednost nekog objekta, moramo prvo napisati ključnu reč “Set”:

```
Dim varA As Range
Set varA = ActiveCell
```

Sad na dalje možemo koristiti `varA` isto kao što bi koristili `ActiveCell` zato što pokazuju na isti objekat.

Još jedna “vrsta” dodele je kada određenim paramentrima neke procedure/funkcije dodeljujemo vrednosti. U tom slučaju, parametrima dodeljujemo vrednosti sa “:=”. Na primer:

```
Workbook.Open fileName:="Book1.xls", password:="pswd"
```

Promena osobine – osobine možete shvatiti i kao „podešavanja objekta“. Npr. jedno od podešavanja za ćeliju je njena vrednost, tj. ono što u njoj piše. Sledeći kod će promeniti vrednost ćelije A3 tako da u njoj piše reč „Test“:

```
Range(“A3”).Value = “Test”
'sledeći red ima potpuno isti efekat, ali ćeliji pristupa na drugi način:
Cells(3,1).Value = “Test” ' treći red, prva kolona – u vrednost upiši Test
```

U prethodnom primeru na dva različita načina smo došli do istog objekta – ćelije A3. Koji ćete način da koristite je potpuno nebitno, ali da vam treba referenca na neki opseg (npr. A2:A40, ili imenovani opseg – “podaci”, “vrednosti” itd…) morali bi da koristite Range.

Čitanje osobine je veoma slično upisivanju vrednosti, samo što sada umesto upisivanja vrednosti u osobinu, čitamo osobinu (tj. smeštamo je na desnu stranu) a na levu stranu ćemo staviti neku promenljivu:

```
varA = Cells(1,3).Value ' upisujemo u varA vrednost ćelije C1
Cells(1,4).Value = varA ' u D1 upisujemo vrednost koju smo malopre pročitali iz C1
Cells(2,2).Value = Cells(5,1) 'u B2 upisujemo vrednost iz A5
```

Matematičke operacije su veoma slične drugim programskim jezicima. Na primer:

```
varA = varB + varC * varD / varB – varD^2
' imamo redom: sabiranje, množenje, deljenje, oduzimanje, stepenovanje stepenom 2
```

Logičke operacije su opet, veoma slične onome što ste verovatno sretali. Primer:

```
varA = (varB > varC) ' u varA će stajati TRUE ili FALSE
varA = varC=varD 'prva jednakost je za dodelu (uvek), a druga deo logičke operacije
varA = varD<=varE 'da li je varD manje ili jednako od varE
varA = varE<>varA 'da li je varE različito od varA
```

Promenljive u VB mogu da se definišu eksplicitno: `Dim varA as Integer`, ali imate slobodu da jednostavno koristite promenljive i koje niste definisali, i VB će automatski da ih definiše za vas. Nazivi promenljivih moraju početi slovom, mogu sadržati brojeve, ali ne i znak razmaka (ako želite da razmak bude deo naziva promenljive koristite znak „_“). Primer:

```
Dim varA as Integer 'eksplicitno deklarišemo da je varA celobrojnog tipa
varA = 12 'dodeljujemo joj vrednost 12
varB = varA * 2 'iako nismo deklarisali varB, upisujemo vrednost u nju
varA = varB / 4 'varB kasnije možemo koristiti potpuno ravnopravno
```

Tip podatka predstavlja vrstu podatka sa kojim radimo. Postoje više standardnih tipova: Integer (celobrojni), Double (decimalni), String (tekstualni), Boolean (logički), itd... Takođe, svaki objekat u Excel-u je poseban tip podatka – Workbook, Worksheet, itd... Tip podatka određuje šta možemo sa tim podatkom da radimo. VisualBasic ne kontroliše u trenutku pisanja koda da li ono što radimo odgovara tipu podatka: npr, možemo napisati: `3 * Workbooks(„Sheet1“)`, ali će zato prijaviti grešku kada bude pokušao da izvrši kod. Neki smatraju da je ovo glavni nedostatak VB jezika zato što dozvoljava veliki broj grešaka u vremenu izvršavanja (nije *strong typed*).

Velika i mala slova ne prave razliku za VB – ako i stavimo negde malo slovo gde treba veliko, VB će prepoznati na koju promenljivu/objekat se odnosi (*case insensitive*).

Kontrolisanje toka izvršavanja

Kod se standardno izvršava jedan red za drugim. Međutim, ima slučajeva kada nam je potrebno da neki kod izvršimo pod nekim uslovima, da neki deo koda ponavljamo određen broj puta itd… Zbog ovoga postoje različite dodatne komande kojima možemo da kontrolišemo kako će da teče izvršavanje koda. Svaka od dole navedenih konstrukcija može da se ugnježdava u samu sebe ili u druge komande, tj. može se posmatrati kao jedna zaokružena jedinica koda.

If-Then-Else konstrukcija omogućava da izvršavamo neki kod samo pod određenim uslovom. Sintaksa:

```
If uslov Then
       Naredba1
       Naredba2
       …
Else
       Naredba3
       …
End If
```

Ako je *uslov* tačan (tj. izračunava se na `TRUE`) onda će se izvršiti *Naredba1*, *Naredba2* i ostale ako postoje. Ako je *uslov* netačan, izvršava se deo koda u `Else` grani – tj. *Naredba3* (i ostale ako postoje). Ako pišemo `If-Then-Else` u više redova kao u gornjem primeru, obavezno je da naredbu završimo sa `End If`. Ako nam `Else` deo nije potreban možemo izostaviti celu tu granu..

Konkretan primer:

```
If Cells(1, 1).Value = True Then
    Cells(2, 2).Value = "Tacno"
Else
    Cells(2, 2).Value = "Netacno"
End If
```

Ili:

```
If Cells(1, 1).Value = True Then
    Cells(2, 2).Value = "Tacno"
End If
```

Ili:

```
If Cells(1, 1).Value = True Then Cells(2, 2).Value = "Tacno"
```

Select-Case je naredba koja je po funkciji slična If-Then-Else naredbi sa tom razlikom što se upoređuje više vrednosti, bez potrebe za ugnježdavanjem If delova. Sintaksa:

```
Select Case var
       Case uslov1
              Naredba1
              …
       Case uslov2
              Naredba2
              …
       …
End Select
```

Promenljiva čiju vrednost ispitujemo je označena sa *var.* U svakom *Case* delu ta vrednost se ispituje da li je jednaka uslovu *uslov1, uslov2* itd… Ako je tačan neki uslov, tada se izvršava odgovarajući set naredbi. Npr, ako je tačan *uslov1*, onda će se izvršavati *Naredba1* (i ostale ako postoje, sve do sledećeg *Case* dela), ako je tačan *uslov2,* izvršava se *Naredba2*, itd… Ako u uslovima treba da iskoristimo vrednost *var* (npr. *var*>2), onda pišemo i ključnu reč *Is*: *Case Is > 2*. Za operator jednakosti to nije potrebno (Case 4 će ispitivati da li je var = 4).

Primer:

```
Select Case ActiveCell.Value
       Case Is < 0
              ActiveCell.Font.ColorIndex = 3
       Case 0
              ActiveCell.Font.ColorIndex = 5
       Case Is >0
              ActiveCell.Font.ColorIndex = 1
End select
```

For-Next petlja omogućava da neki deo koda ponavljamo određeni broj puta. Sintaksa:

```
For var = x To y
       Naredbe…
Next var
```

Promenljiva *var* koristi kao brojač prolaza kroz petlju – pri svakom prolazu uvećava se za 1 (ime promenljive je proizvoljno). U prvom prolazu ova promenljiva dobija vrednost *x*, a petlja će ići dok god ne dostigne vrednost *y*. U svakom prolazu izvršava se set komandi koji je unutar `For` petlje (obeležen sa *Naredbe…*). `For` petlju obavezno završavamo sa `Next` *var* (ime promenljive nije neophodno, ali je poželjno jer olakšava razumevanje koda).

Konkretan primer:

```
Suma = 0
For iBr = 0 To 100
  Suma = Suma + iBr
Next iBr
```

Ovaj primer će sabrati brojeve od 0 do 100. Rezultat sabiranja imamo u promenljivoj Suma. Sledeći primer upisuje redne brojeve u prvih 100 ćelija kolone A:

```
For iBr = 1 To 100
   Cells(iBr, 1).Value = iBr
Next iBr
```

Postoji još jedna konstrukcija koja se koristi za For naredbu kada je potrebno da "prođemo" kroz sve objekte neke kolekcije objekata. Tada će For naredba početi sa: `For Each` *obj* `In` *Coll* (ostalo je sve isto), odnosno u prevodu: "ponavljaj za svaki objekat obj u kolekciji Coll". U tom slučaju, objekat se ne uvećava za jedan kao u prethodom slučaju već će za svaku sledeću iteraciju dobiti sledeću vrednost u kolekciji (zbog toga je bitno da tu kolekciju ne menjamo unutar petlje).

Na primer, ako želimo da svaku ćeliju selekcije promenimo tako da sadrži svoj redni broj:

```
varI = 0
For Each Item In Selection
       varI = varI + 1
       Item.Value = varI
Next Item
```

Ovo je vrlo korisno kada radimo sa nekom kolekcijom u kojoj ne znamo tačan broj objekata a želimo da ih prođemo sve.

With je ključna reč koju ćemo koristiti u prilikama kada često pristupamo istom objektu. Nema neki funkcionalni značaj za program osim što nam skraćuje vreme pisanja. Sintaksa:

```
With objekat
      Naredbe
      ...
End With
```

Objekat je neki objekat čije osobine ili čijim objektima želimo da pristupamo u delu *Naredbe*. Na primer, umesto da pišemo dugački kod:

```
Selection.HorizontalAlignment = xlCenter
Selection.VerticalAlignment = xlCenter
Selection.WrapText = False
```

možemo da pišemo skraćeno:

```
With Selection
      .HorizontalAlignment = xlCenter
      .VerticalAlignment = xlCenter
      .WrapText = False
End With
```

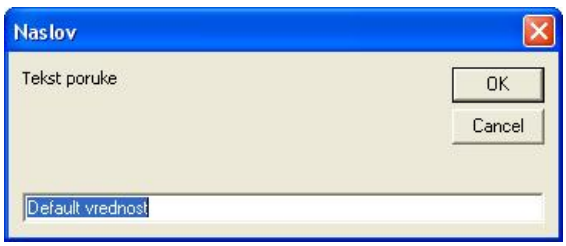
Interakcija sa korisnikom

Za sada koristićemo samo dve osnovne funkcije za interakciju – InputBox i MsgBox, kojima ćemo uzimati podatke od korisnika odnosno prikazivati poruke korisniku (respektivno). Detaljniju i mnogo opširniju grafičku komunikaciju radićemo u delu sa UserForms.

InputBox je zapravo funkcija koja prikazuje korisniku poruku i pruža mu prostor za unošenje proizvoljnog teksta. Kada korisnik klikne OK vrednost koju je ukucao je sadržana u povratnoj vrednosti funkcije. Primer:

```
inp = InputBox("Tekst poruke", "Naslov", "Default vrednost")
```

U ovom primeru, kada se pokrene makro, korisniku se prikazuje sledeći prozor:



Kada klikne OK, InputBox će vratiti vrednost koju je on uneo kao rezultat funkcije, i u našem primeru to ćemo sačuvati u promenljivu `inp`. Ako korisnik klikne na „Cancel“ InputBox će vratiti vrednost praznog stringa: „“. *Naslov* i *Default vrednost* u pozivu ove funkcije možemo izostaviti, dok je tekst poruke obavezan.

MsgBox je takođe funkcija ali za razliku od InputBox ova funkcija samo ispisuje poruku korisniku. Kao povratnu vrednost vraća informaciju da li je korisnik kliknuo na OK, Cancel, ili sl. Ove povratne vrednosti su konstante definisane objektom VbMsgBoxResult. Npr, ako je kliknuo OK, MsgBox će vratiti VbMsgBoxResult.vbOK, ako je kliknuo Cancel, vratiće vrednost VbMsgBoxResult.vbCancel, itd... vrednost ovih konstanti možete uvek dobiti tako što ćete napisati VbMsgBoxResult i zatim kad pritisnete tačku prikazaće vam se sve konstante koje su definisane. Primer:

```
inp = MsgBox("Tekst poruke", definicijaTastera, "Naslov")
```

Tekst poruke je tekst koji se ispisuje korisniku. Definicija tastera podrazumeva jednu od konstanti u `vbMsgBoxStyle` objektu (npr. `vbOKOnly`, `vbYesNo`, `vbOKCancel`, itd...). Naslov je naslov poruke.

Jedan od primera bi mogao da bude sledeći:

```
inp = MsgBox("Nedovoljno podataka. Nastaviti obradu?", vbYesNo, "Naslov")
```

Ovde pitamo korisnika da li želi da nastavi obradu i prikazujemo mu `Yes` i `No` dugmiće. Njegov izbor pamtimo u promenljivoj `inp`. Nakon toga, možemo da ispitujemo šta je kliknuo tako što ćemo porediti `inp` sa `vbMsgBoxResult.vbNo` i `vbMsgBoxResult.vbYes`.

Primeri – Procedure

Procedura koja ispisuje nazive worksheet-ova aktivnog Workbook-a u kolonu A

```
Sub PopuniSheetove()
    Dim jedanSheet As Worksheet

    Cells(1, 1).Value = "Lista sheet-ova" 'zadajemo naziv kolone
    i = 2
    For Each jedanSheet In Worksheets 'krećemo se kroz listu sheet-ova
        Cells(i, 1).Value = jedanSheet.Name 'upisujemo vrednosti u ćelije
        i = i + 1 ' uvećavamo u kom redu će se sledeći put upisati vrednost
    Next jedanSheet
End Sub
```

Verzija 2: Ispis ide počevši od trenutno aktivne ćelije. Na kraju reformatirati ispis (AutoFit).

```
Sub PopuniSheetove2()
    Dim jedanSheet As Worksheet
    Dim C As Range

    Set C = ActiveCell
    i = 1

    C.Value = "Lista sheet-ova"
    For Each jedanSheet In Worksheets
        C.Offset(i, 0).Value = jedanSheet.Name
        ' u prethodnom redu – offset vraća referencu koja je za i redova
        ' ispod aktivne ćelije, i za 0 kolona u desno (što znači da će biti
        ' u istoj koloni gde je i aktivna ćelija)
        i = i + 1
    Next jedanSheet

    'Kad smo završili, uradimo AutoFit na koloni u kojoj se nalazi naša ćelija
    Columns(C.Column).AutoFit
    'C.Column će vratiti broj kolone u kojoj se nalazi aktivna ćelija, a Columns
    'kolekcija će zatim vratiti objekat kolone (Column) na kome primenjujemo AutoFit
End Sub
```

Zadatak 1: Napisati makro koji će napraviti okvire (borders) za tabelu sa 5 kolona i 10 redova, a počev od trenutno aktivne ćelije. Spoljašnji okvir tabele treba da bude duplih ivica, a unutrašnje ivice između ćelija standardne debljine i stila.

- Selektujte neki opseg ćelija.
- Snimite makro u kome ćete promeniti sve ivice tako da odgovaraju zahtevu. Zaustavite snimanje
- Promenite dobijeni makro tako da odgovara zahtevima u zadatku.

Zadatak 2: Napisati makro u kome se od korisnika traži da unese neku vrednost (Vr). Zatim, pitati ga da li želi da koristi tu vrednost za uvećanje vrednosti. Ako odgovori pozitivno ćelije od B1 do B20 uvećati, u suprotnom umanjiti za vrednost koju je prethodno uneo (Vr).

- Koristiti InputBox za dobijanje vrednosti Vr
- Koristiti MsgBox sa dugmićima Yes i No (vbYesNo)
- Ćelije menjati u For petlji koristeći Cells(red, kolona).Value

Primeri – Funkcije

Funkcija koja sabira brojeve od zadatog broja do krajnjeg broja:

```
Function Saberi(a1, a2)
    s = 0
    For i = a1 To a2
        s = s + i
    Next i
    Saberi = s
End Function
```

	A
1	1
2	5
3	=Saberi(A1;A2)

Funkcija koja vraća kvadrat najvećeg broja u opsegu

```
Function KvadratMax(opseg)
    maxVr = 0
    For Each cel In opseg
        If (cel.Value > maxVr) Then
            maxVr = cel.Value
        End If
    Next cel
    KvadratMax = maxVr ^ 2
End Function
```

	A
1	1
2	2
3	7
4	4
5	5
6	=KvadratMax(A1:A5)

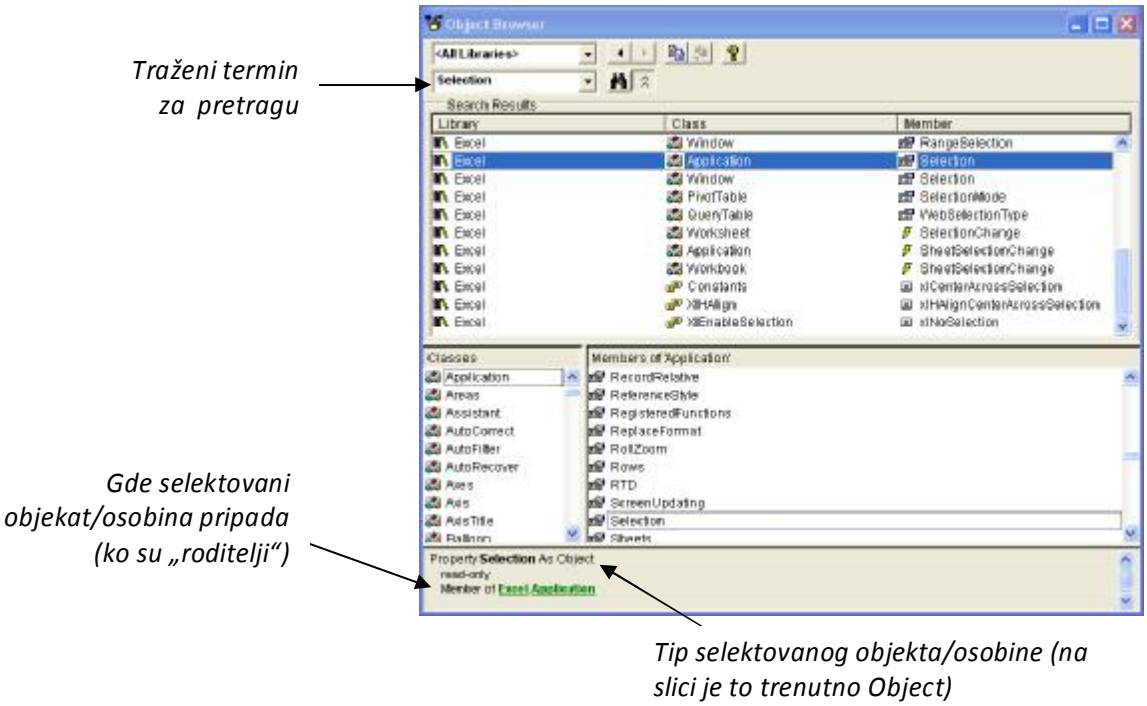
Zadatak 1: Napisati funkciju koja vraća prosečnu vrednost za zadati opseg.

Zadatak 2: Napisati funkciju koja vraća polovinu minimalne vrednosti.

Rad sa objektima

Object Model

Često nam zatreba da vidimo koji objekat je kog tipa, ili da pronađemo gde u hijerarhiji objekata možemo naći traženi objekat/osobinu, itd... Ovo sve se može pronaći u tzv Object Model-u u Visual Basic Editor-u. Object Model je prozor koji služi isključivo za pružanje informacija i pretragu po objektima. Aktivira se u meniju pod *View->Object Model* ili pritiskom na taster F2.



Najčešći način upotrebe: ukucavamo neki termin koji želimo da nađemo gde pripada (npr, želimo da vidimo odakle sve možemo da pristupimo objektu *Line*), i zatim odlučujemo šta nam treba u zavisnosti od „roditelja“ koji su ponuđeni (tj. kojoj klasi pripada).

Utvrđivanje tipa objekta

Kada posle imena objekta stavimo tačku, očekujemo da nam Excel izlista sve osobine, metode, decu objekte i događaje koje taj objekat podržava. Međutim, za neke ovo nije moguće – npr. ako stavimo tačku posle *Selection* objekta očekujemo da dobijemo sve njegove osobine, ali problem je u tome što se taj objekat stalno menja – nekad korisnik selektuje jednu ćeliju, nekad selektuje neki crtež na worksheet-u, nekad grafik, itd... Zbog toga, *Selection* je objekat opšteg tipa – *Object*, koji nema nijednu mogućnost. Tek se u trenutku izvršavanja koda zna šta je zapravo sadržano u tom *Selection* objektu, i kod koji smo kucali ili može ili ne može da se izvrši (npr, ako koristimo Selection da bi pristupili nizu ćelija sve će biti

OK ako je zaista i selektovan niz ćelija, ali ono što je problem je što ćemo dobiti grešku ako je selektovana linija, kvadrat ili sl. iz *Drawing toolbar*-a).

Da bi predupredili ove greške koristićemo funkciju TypeName koja će vratiti trenutni tip podatka koji je prosleđen kao parametar. Npr. TypeName(Selection) će vratiti *Line* ako je selektovana linija, *Range* ako je selektovana jedna ili više ćelija itd... Sledeći kod će uvećati vrednost u selektovanim ćelijama samo ako je zaista i selektovan opseg ćelija (tj. Range):

```
Sub Uvecaj()
    If TypeName(Selection) = "Range" Then
        For Each C In Selection
            C.Value = C.Value + 1
        Next C
    End If
End Sub
```

Kada smo utvrdili da se radi o objektu željenog tipa, možemo tu informaciju koristiti da radimo sa promenljivima koje su tačno definisanog tipa:

```
Sub Uvecaj()
    If TypeName(Selection) = "Range" Then
        Dim c As Range
        For Each c In Selection
            c.Value = c.Value + 1
        Next c
    End If
End Sub
```

Sada kad god budemo koristili „c“ u kodu imaćemo sve prednosti toga što tačno znamo koje osobine i metode „c“ podržava – kad stavimo tačku iza „c“ dobićemo kompletan listing, zato što je c tačno određenog tipa – *Range*.

UserForms

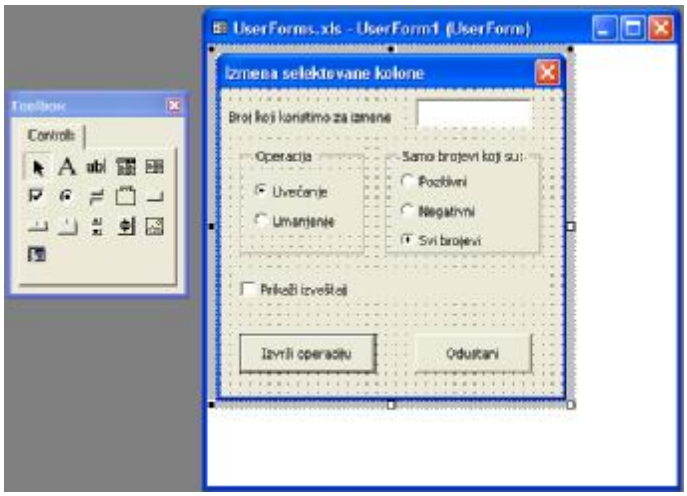
UserForms su zapravo Windows prozori čiji izgled i ponašanje mi definišemo. Koristimo ih da bi dobili informacije od korisnika, i za lakše kontrolisanje korisnikovog unosa, i bilo šta drugo što inače ne bi mogli da dobijemo upotrebom standardnih Excel funkcija, ili upotrebom naših makroa.

Pravljenje UserForms prozora se vrši u dve faze: prvo raspoređujemo sve elemente po prozoru: tekst, dugmiće, checkBox-ove, opcije, itd..., a zatim pišemo makroe kojima definišemo kako će se koji element ponašati (npr, šta treba uraditi ako klikne dugme OK, šta ako klikne „Unesi“ itd...).

Formiranje UserForms ekrana

Otvorite Visual Basic Editor (Tools->Macro->Visual Basic Editor, ili F11, ili ako ste aktivirali Visual Basic toolbar imate VBE ikonicu). Kliknite desnim dugmetom na ime projekta u koji želite da smestite ekran – npr. *VBAProject(Book1)*, zatim *Insert ->UserForm*. Prikazuje se ekran za izmene u dizajn modu. Postoje

dva moda rada: *design* – u kome dodajemo i menjamo elemente; i *runtime* – kada se prozor prikazuje korisniku, tj. izvršava se.



Dobijamo ekran koji predstavlja radnu površinu za naš nov ekran – na slici je predstavljeno spoljašnjim prozorom „UserForms.xls – UserForm1 (UserForm)“. Unutar tog prozora mi definišemo izgled našeg prozora koji ćemo u daljem tekstu zvati radnom površinom (na slici predstavljeno prozorom „Izmena selektovane kolone“). Kad se prvi put kreira UserForms ovaj prozor će biti prazan. Popunjavamo ga koristeći elemente prikazane u Toolbox-u: element koji želimo da koristimo prostim prevlačenjem smestimo na radnu površinu. Ako zaustavimo kursor miša iznad kontrole, pojaviće se naziv kontrole. Mi ćemo u ovom kursu raditi samo sa vrstama kontrola koje su izlistane u tabeli. Svaka kontrola ima svoj set osobina (properties) koje menjamo u *Properties* prozoru (View->Properties, ili F4). Najbitnije osobine su date uz svaku kontrolu u tabeli.

Kontrola	**Opis**	**Bitan property**	**Komentar**
Label	Predstavlja običan tekst koji ispisujemo na ekranu	Caption	Najčešće se koristi da opiše korisniku šta se traži od njega, funkciju prozora, dodatna pojašnjenja i sl.
TextBox	Mesto gde korisnik unosi tekst	MultiLine – mogućnost unosa više redova Text – tekst koji je upisan u kontrolu	Tekst koji korisnik unese može biti bilo koji niz karaktera, o čemu treba voditi računa pri upotrebi ove vrednosti u kodu
CheckBox	„Kutija“ za štikliranje	Value – True/False – tj. da li je štiklirano ili nije	Najčešće koristimo kad treba da dobijemo „da/ne“ odgovor od korisnika.
OptionButton	Biranje jedne od opcija	Value – True/False, tj. da li je opcija izabrana ili ne	Za svaku opciju dodajemo poseban option button, a korisnik na jednom prozoru može da izabere samo jednu. Ukoliko nam treba više grupacija za opcije na istom prozoru, možemo ih grupisati po frejmovima
ToggleButton	Dugme za uključivanje/isključivanje	Value – True/False, tj. da li je dugme utisnuto ili ne	Slična upotreba kao CheckBox, samo sa drugačijim izgledom
Frame	Kontrola čija jedina uloga je da u sebi čuva druge kontrole	Caption	-
CommandButton	Glavni „hvatač“ događaja	Default, Cancel – ako korisnik pritisne <enter> ili <esc> da li se automatski poziva Click ovog dugmeta	Sve glavne događaje ćemo vezivati za Click dugmeta.

Sve kontrole koje dodajemo na UserForm imaju više osobina koje su im zajedničke, od kojih su ovde izlistane neke najbitnije/najkorisnije:

- Caption – koji određuje šta na kontroli piše.
- (Name) – identifikator kontrole koji ćemo često koristiti u našem kodu prilikom obrade unosa. Postoje neki standardi za dodeljivanje imena kontrolama koji nam olakšavaju da prepoznamo tip kontrole iz njenog imena. Npr. TextBox-ovi uvek počinu sa txt a zatim se nastavlja sa logičkim imenom – txtBroj, txtIme, txtLozinka, itd... Labele sa lbl, OptionButton – opt, CheckBox – chk, CommandButton – cmd, ToggleButton – tgl, itd... Nije bitno koristiti baš tri slova, niti da to budu baš ova slova, ali koju god sintaksu da uvedemo trebalo bi toga da se pridržavamo u svom daljem korišćenju zato što se time izbegavaju razni problemi koji mogu nastati kod složenijeg razvoja aplikacija u VB-u (ali i bilo kom drugom softverskom projektu).
- Visible – da li je kontrola vidljiva ili ne (u zavisnosti od nekih izbora možemo je sakriti ili pokazati)
- Enabled – da li se korisniku dozvoljava da radi sa kontrolom ili ne
- TabIndex – kada korisnik pritiska Tab on će ići redom od kontrole do kontrole i to prateći ovaj indeks od manjeg ka većem broju.
- ControlTipText – kada se kursor miša pozicionira iznad kontrole neko vreme, prikazaće se ovaj tekst (možemo koristiti da bliže objasnimo neku opciju korisniku)
- Locked – ova opcija nema uticaja na korisnika – koristimo je dok dizajniramo prozor da bi osigurali kontrolu od slučajnog pomeranja.

Sve elemente smoređali na UserForm koji sam po sebi sadrži neke osobine od kojih ćemo samo izdvojiti Caption (naslov prozora), i StartUpPosition – mesto gde se forma pojavljuje na ekranu prilikom prikazivanja. Identifikator prozora naveden sa "(Name)" za sada stoji podešen na UserForm1, mada bi mogli da ga promenimo npr. na frmUnos, ili slično.

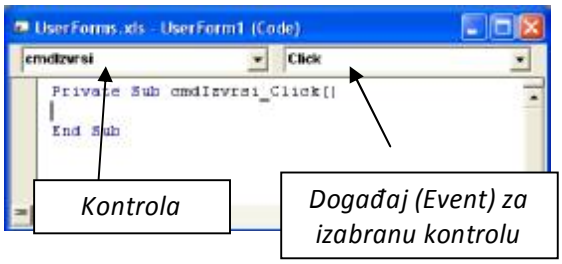
Kodiranje ponašanja

Prvo ćemo napisati kod koji će prikazivati formu koju smo prethodno napravili. Taj kod pišemo u modulu gde smo i ranije pisali makroe. Za prikaz ekrana koristi se njegova Show metoda, na sledeći način:

```
Sub prikazi()
    UserForm1.Show
End Sub
```

Ovaj makro možemo da pokrenemo na standardni način, kao i bilo koji makro koji smo pokretali ranije (npr, pozicioniranjem bilo gde na kod i pritiskom na F5 (odnosno klikom na "play")). Kad se pokrene, prikazaće se forma koju smo malopre napravili. Obzirom da nismo iskodirali nikakve akcije za dugmiće, oni neće reagovati na klik. Naš sledeći zadatak je da napišemo kod kojim ćemo obraditi ponašanja dugmića.

U VBE-u otvoriti UserForms, i duplim klikom na dugme koje je tu postavljeno otvorićemo kod koji će se izvršavati kada korisnik klikne na dugme.



Automatski će se ispisati format koda koji odgovara kliku na dugme. Sve što napišemo ovde će se izvršiti kada korisnik klikne na dugme.

Ako želimo ručno da dodamo neko ponašanje za bilo koju kontrolu – prvo izaberemo identifikator kontrole (definisan u Property prozoru sa "(Name)"), a zatim događaj u kome želimo da izvršimo akciju. Ako u UserForms prozoru u dizajnu duplim klikom kliknemo na bilo koju kontrolu VBA će automatski otvoriti kod za *default* događaj (npr, za dugme će biti Click, za TextBox će biti Change, itd…).

Dodaćemo sada kod za "Odustani" dugme koji samo treba da zatvori prozor bez ikakve obrade:

```
Private Sub cmdOdustani_Click()
    Unload UserForm1
End Sub
```

Unload je metoda kojom se uništava objekat koji je specificiran – u ovom slučaju navodimo identifikator našeg prozora – UserForm1.

Očitavanje osobina elemenata u kodu

Sve osobine kontrola koje smo koristili prilikom dizajna mogu da se pročitaju (i promene) i u kodu. Pristup do osobina je potpuno isti kao i za bilo koje druge Excel objekte – npr. da bi dobili tekst koji je napisan u TextBox-u kome je "(Name)" podešen na txtBroj, koristićemo njegov *property* – *Text* koji, kako smo ranije naveli, definiše šta je ispisano (bilo od ranije, bilo da je korisnik uneo u toku izvršavanja koda):

```
msgBox "Uneli ste " & txtBroj.Text
```

Ovde smo tekstu "Uneli ste" dodali tekst koji je napisan u txtBroj.Text osobini, i sve to ispisali kao poruku korisniku. Na sličan način možemo očitati bilo koju osobinu bilo koje kontrole.

Takođe, dok pišemo kod za UserForm mi imamo i kompletan pristup standardnim objektima – Worksheet, ActiveCell, Selection itd… Ako bi hteli da menjamo trenutno selektovane ćelije, od korisnika možemo zahtevati da prvo selektuje neki opseg pre nego što nastavimo dalje, i slično.

Događaji na worksheet-u / workbook-u

Ranije smo videli da neke kontrole imaju default događaje – npr. za CommandButton događaj Click, za TextBox – Change, itd… Pored ovih *default* mogli smo da definišemo kod i za bilo koji drugi događaj (dupli klik miša itd…). Slično tome, i workbook i worksheet imaju svoje događaje.

Worksheet Events

Otvorite Visual Basic Editor (VBE) i u *Project* prozoru duplim klikom na ime *sheet*-a otvorite njegov kod (npr. Sheet1). Prikazuje se standardni prozor za kod, gde možemo da selektujemo objekat i određeni događaj u njemu. Po *default*-u, u vrhu *Code* prozora prikazuje se *(General)* u *Object box*-u (levo), i *(Declarations)* u *Procedures/Events box*-u (desno). Dok je izabran General mi pišemo naše procedure i funkcije. Međutim, u Object box-u pored General možemo izabrati *Worksheet* kao objekat i tom prilikom automatski se bira default događaj za njega – *SelectionChange* (tj. promena trenutne selekcije, odnosno trenutak kad korisnik klikne na drugu ćeliju npr. Promena selekcije podrazumeva da se radi sa ćelijama – kada se selektuju grafički elementi – linija, kvadrat i sl. događaj se ne "pali"). U ovom kodu imamo sledeću definiciju procedure, koju principijelno nikad ne bi trebali da menjamo:

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
```

Ključna reč "Private" ograničava vidljivost ove procedure na *Sheet* u kome trenutno radimo, tj. za koji pišemo kod – ovo je tipično za bilo koji *Event* kod koji pišemo. Posle ključne reči Sub ide ime objekta za koji je događaj definisan, koji je donjom crtom odvojen od naziva događaja. Ono zbog čega je ovaj događaj različit od nečeg što smo do sada radili je u tome što poseduje svoj *parametar procedure* naveden u zagradama. Kada procedura za događaj ima parametar onda će taj parametar uvek dobiti vrednost koja je relevantna za događaj – u slučaju *SelectionChange* događaja, relevantna informacija je novi opseg koji je selektovan. Koristeći taj parametar, možemo na primer napisati sledeći kod koji će ispisivati adresu trenutno selektovanog opsega u ćeliju A1:

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    Range("A1").Value = Target.Address
End Sub
```

Primer promena u ćeliji A1: `$A$1`, `$S$32`, `$F$8:$J$20` i sl…

Još jedan primer za Sheet događaje je Change event. Ovaj događaj se aktivira kada nastane promena na nekom opsegu (bilo jednoj ćeliji ili više njih). Ako na našem *Sheet*-u imamo računanje nekih vrednosti u makrou, a taj makro koristi vrednosti određenog opsega (npr. nazvaćemo opseg "Podaci"), mogli bi da pratimo izmene koje su izvršene na tom opsegu i svaki put kad nastane promena da ponovo pokrenemo makro za izračunavanje. Da nismo koristili makro već funkcije, Excel bi automatski ponovo izračunao sve vrednosti, ali u našem slučaju mi moramo sami da vodimo računa. Procedura je data sledećim kodom:

```
Private Sub Worksheet_Change(ByVal Target As Range)
    Application.EnableEvents = False
    If Not (Intersect(Target, Range("Podaci")) Is Nothing) Then
        Call Proracunaj 'poziv procedure
    End If
    Application.EnableEvents = True
End Sub
```

Prvi i poslednji red procedure uključuju i isključuju osobinu Excel-a da reaguje na događaje pomoću osobine *Application.EnableEvents*. Ovo je potrebno ukoliko u proceduri "Proracunaj" menjamo neku ćeliju iz opsega "Podaci" – ako ne bi isključivali događaje, u tom trenutku bi se ponovo aktivirao *Change* događaj, koji opet poziva proceduru koja opet menja sadržaj, i tako u krug – dobijamo beskonačnu petlju (ukoliko vam se ovo dogodi, izvršavanje možete prekinuti sa Esc tasterom).

Funkcija *Intersect* vraća presek dva opsega. Ako se opsezi ne presecaju, vraća poseban Excel tip *Nothing* koji je zapravo specijalna vrednost koja se koristi za poređenje sa objektima (slično kao što se nula koristi u poređenju sa brojevima, tako i *Nothing* može da se koristi pri poređenju sa objektima ali se umesto znaka jednakosti koristi ključna reč `Is`). Ukoliko presek nije prazan, pozivamo proceduru za obradu – `Proracunaj`. Obzirom da smo već isključili reagovanja na događaje, ne moramo da vodimo računa koje ćelije menjamo u toj proceduri, i da li će nastupiti beskonačna petlja.

Treba voditi računa o tome da ako obrišemo neki *worksheet*, ujedno brišemo i sve događaje koje smo za njega pisali.

Workbook Events

Workbook je opštiji od *worksheet*-a zato što on sadrži sve *worksheet*-ove. U njemu možemo uhvatiti događaje kao što su na primer otvaranje dokumenta (*Open*), dodavanje novog Sheet-a (*NewSheet*), itd… Jedan primer za događaj *BeforePrint* bi bio sledeći:

```
Private Sub Workbook_BeforePrint(Cancel As Boolean)
    vr = MsgBox("Da li ste promenili footer da sadrži brojeve stranica?", _
                vbYesNo, "Podsetnik")
    If vr = VbMsgBoxResult.vbNo Then
        Cancel = True
    End If
End Sub
```

Ovaj kod se izvršava svaki put pre štampe (bilo *PrintPreview*, bilo "pravi" *Print*). Ovaj primer je uzet zbog parametra koji sadrži – `Cancel As Boolean`. Ovaj parametar je u vezi sa događajem i ima default vrednost `True` koja određuje da štampanje treba da se izvrši. Ako ga promenimo na `False` unutar našeg koda, prekinućemo izvršavanje štampe. Slično korišćenje ćete verovatno sretati i kod drugih događaja.

Dodavanje kontrola na Worksheet

Ako želimo da iskoristimo worksheet da nam bude prostor za unos podataka, ne moramo praviti posebne UserForms ekrane – možemo sve kontrole smeštati na worksheet umesto na poseban prozor. Ovo se najčešće radi kada imamo neku jednostavnu interakciju sa korisnikom, koju možemo da ostvarimo i na samom worksheet-u.

Dodavanje kontrola se vrši iz toolbar-a *Control Toolbox* (ako imate aktiviran toolbar VisualBasic, možete ga aktivirati i odatle klikom na odgovarajuće dugme). Pored ovog postoji još jedan toolbar sa setom kontrola – Forms. Međutim, kontrole koje Forms pruža na korišćenje imaju manje opcija i podešavanja i generalno se ne koriste. Control Toolbox ima sledeći izgled:

U njemu su izlistane skoro sve kontrole koje smo koristili i za pravljenje UserForms ekrana. Pored njih imamo još nekoliko:

- Design Mode: prebacivanje i mod za editovanje, i za izlazak iz tog moda. Dok je ovo aktivno. dugme izgleda utisnuto i možemo menjati sve kontrolne elemente koje smo dodali na *sheet*. Kada je deaktivirano onda kontrole izvršavaju funkciju koju smo im definisali u *design* modu.
- Properties: Koristi se kada smo u *design* modu za promenu osobina kontrola (slično kao i pri dizajnu UserForms ekrana)
- ViewCode: Može se kliknuti samo dok smo u *design* modu. Prikazuje kod koji se nalazi „iza“ dugmeta, ili bilo koje druge kontrole. Slično kao i u UserForms, kod možemo videti i duplim klikom na kontrolu.
- More Controls: Izlistava sve ActiveX kontrole koje su instalirane u Windows-u. Na ovaj način uz malo poznavanja kontrola unutar worksheet-a možete dodati video klip, otvoriti pdf dokument itd...

Kao primer, probajte da dodate jedan CommandButton, i promenite njegov property Caption tako da na njemu piše: „Pokreni obradu“. Sada, duplim klikom na dugme (u *design* modu) otvoriće se kod i to u Sheet-u u kome se nalazi dugme (a ne u modulu), i napraviće se standardna procedura za obradu događaja. Ovde možemo sada unositi neki makro kao što smo i ranije pisali u „običnim“ makroima. Na primer, možemo i prikazati neku UserFormu:

```
Private Sub CommandButton1_Click()
    UserForms1.Show
End Sub
```

Da bi ste isprobali kako dugme funkcioniše, isključite *design* mod klikom na odgovarajuće dugme u toolbar-u. Ako želite ponovo da vršite neke izmene na kontroli, morate ponovo ući u *design mode.*

Povezivanje kontrola sa worksheet-om

Neke kontrole su automatski programirane tako da mogu da se povežu sa ćelijama worksheet-a pomoću svog *property*-a *LinkedCell*. Npr. *Spin Button* možemo povezati sa nekom ćelijom unosom reference na ćeliju u polje *LinkedCell,* tako da kada korisnik klikće dugmiće gore-dole vrednost u ćeliji se automatski menja.

ListBox i *ComboBox* imaju dodatne opcije – kod njih možemo definisati opseg odakle se popunjava lista sa *ListFillRange property*-jem, a šta je korisnik izabrao se smešta u ćeliju definisanu sa *LinkedCell*.

Za sve kontrole koje smo dodali možemo definisati dodatne obrade događaja u kodu za Sheet u kom se nalaze – kada otvorite kod za Sheet, u Object box-u, gde smo ranije imali samo Worksheet, sada imamo izlistane sve kontrole koje smo dodavali (sa onim imenima koje smo definisali u njihovom property-u "(Name)").

Prilikom štampe worksheet-a koji na sebi ima kontrole za unos odštampaće se ujedno i sve kontrole. Međutim, imamo mogućnost da ovo isključimo – dovoljno je promeniti *Print object* u *property* prozoru i setovati na *False* za svaki objekat koji želimo da isključimo iz štampe.

Obrada grešaka

Kada nastupi greška u VBA kodu automatski se zaustavlja izvršavanje i prikazuje se deo koji je problematičan. Postoje situacije kada znamo da neka komanda može izazvati grešku, i umesto da prekidamo izvršavanje želimo da obradimo grešku na neki način i da nastavimo dalje. Da bi ovo ostvarili postoji više načina. Upotrebu ćemo objasniti na sledećem primeru. U ovom primeru greška može nastupiti u više slučajeva: npr. prilikom pokretanja makroa nije selektovana ćelija nego neki grafički element, ili u ćeliji A1 piše tekst umesto broja.

```
Private Sub CommandButton1_Click()
On Error GoTo greska
    Selection.Value = 5
    Range("A1").Value = Range("A1").Value + 1
    Exit Sub 'da nismo stavili ovo nastavili bi izvršavanje i sledećih redova
greska:
    MsgBox "Doslo je do greske: " & Error()
    On Error GoTo 0
End Sub
```

`On Error GoTo` *`linija`* – ako dođe do greške prebaci se na liniju koja je označena labelom *linija*. Labele se pišu tako što stoje same u liniji, a iza njih je znak dvotačke (kao u gornjem primeru "greska:"). U ovom primeru, čim nastupi greška prekidamo izvršavanje koda gde god da je nastupila greška i prebacujemo se na liniju označenu sa "greska:". Npr, ako je greška nastupila u prvoj naredbi "`Selection.Value = 5`" sledeća naredba koja se izvršava će biti ona iza "greska:", dok se `Range("A1")`… neće izvršiti.

`On Error GoTo 0` – resetuje prethodno ponašanje, tako da kada nastupi greška negde drugde u kodu, ne idemo stalno na liniju "greska:", već se primenjuje standardni način rada sa greškama.

Bitno je primetiti da pre labele za obradu greške stoji `Exit Sub` – to znači da ako smo došli do te linije prekidamo dalje izvršavanje procedure. Da kojim slučajem nismo stavili `Exit Sub`, procedura bi nastavila sa izvršavanjem, i izvršio bi se kod koji ispisuje "Došlo je do greške", bez obzira što greške nema.

Još jedan način obrade grešaka je: `On Error Resume Next` – ako nastupi greška ignorišemo red u kome je nastala i nastavljamo izvršavanje sa sledećom linijom. Npr, u gornjem primeru, ako je nastupila greška pri radu sa *Selection* objektom, nastavićemo kod i pokušati da izmenimo ćeliju A1.

Pronalaženje grešaka - *Debugging*

Kao i u većini programskih alata, i VBA pruža usluge *debugger*-a. Debugger je poseban deo programa koji vam veoma olakšava da otklonite neke logičke greške tako što ćete zaustaviti izvršavanje koda u trenutku koji vama odgovara i izvršavati liniju po liniju, prateći način izvršavanja (u koje grane koda ulazimo, koliko puta, itd…) i koje vrednosti se pri tome menjaju.

Breakpoint – tačka zaustavljanja programa. Postavlja se klikom u sivi deo koda, u nivou reda u kome želimo da se zaustavimo (vidi sliku – treba kliknuti u sivi deo gde je velika tačka). Kada dodamo breakpoint, red se boji u crveno i dodaje se velika tačka sa leve strane. Kada pokrenemo kod, on će se zaustaviti na ovom mestu. Red koji će se sledeći izvršiti je obeležen žutom bojom.

Step Over – komanda koja se nalazi u meniju – *Debug->Step Over*, će izvršiti prvu sledeću naredbu u nizu izvršavanja (odnosno, red koji je obeležen žutom bojom). Najčešće postavimo *breakpoint*, i zatim se krećemo red po red sa *Step Over*. Ako želimo da nastavimo dalje do kraja, odnosno do sledećeg breakpoint-a, kliknemo *Run->Continue* (ili pritisnemo F5).

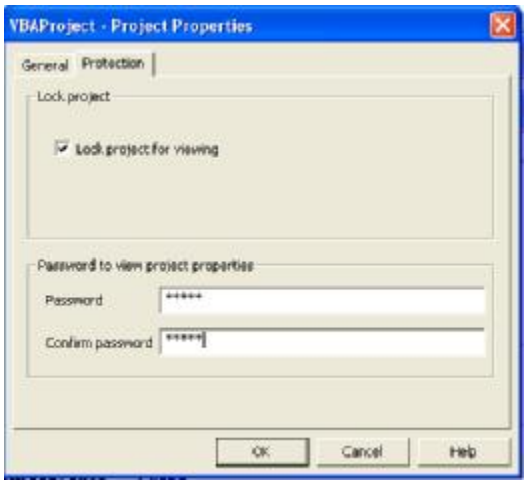
Watches – prozor u kome pratimo vrednosti promenljivih. Aktiviramo ga preko *View->Watch Window*, a promenljive dodajemo u *Debug->Add watch*. U Watch možemo dodavati bilo kakve vrednosti – npr. vrednosti opsega iz *worksheet*-a, osobine raznih kontrola u *UserForm*-i, vrednosti promenljivih itd… Vrednosti promenljivih koje koristimo u kodu možemo videti i prostim pozicioniranjem kursora miša iznad te promenljive – njena vrednost biće ispisana kao *Tooltip*.

Pomeranje sledećeg reda za izvršavanje – dok je izvršavanje pauzirano, žutu liniju koja predstavlja sledeći red koji se izvršava možete jednostavno pomeriti na neki drugi red tako što ćete levim dugmetom miša "uhvatiti" žutu strelicu sa strane i prevući je na željeni red. Na ovaj način možete preskakati redove, ili neke redove ponovo izvršavati.

Zaštita koda

Slično kao što možemo da zaštitimo worksheet, tako možemo da zaštitimo i sav kod koji pišemo tako da ne može da se vidi i menja bez ukucavanja šifre.

Kliknite desnim dugmetom na ime projekta u Visual Basic Editoru (npr. *VBAProject (Book1)*), i izaberite *Properties*. Dobićete sledeći ekran:



Štiklirajte "Lock Project For Viewing" i unesite šifru koja je obavezna. Da bi zaštita postala aktivna, sačuvajte dokument, zatvorite ga i probajte ponovo da ga otvorite. Dokument će se otvoriti bez problema, možete menjati worksheet, a čak je dozvoljeno i da se pokreću makroi. Međutim, ako neko pokuša da otvori VBA kod, od njega će se tražiti da ukuca šifru kojom je taj kod zaštićen.

- Kraj kursa -