Milan Čistý

CAD SYSTÉMY V KRAJINNOM INŽINIERSTVE

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
2007

Recenzenti: Ing. Katarína Tóthová, PhD., Ing. Ľuboš Jurík, PhD.

Schválilo vedenie Stavebnej fakulty STU v Bratislave dňa 2. 3. 2007 pre 4. ročník – študijné programy VHK, ZDI, VSVH, HT.

ISBN 978-80-227-2749-5

EDÍCIA SKRÍPT

Doc. Ing. Milan Čistý, PhD.

CAD SYSTÉMY V KRAJINNOM INŽINIERSTVE

Vydala Slovenská technická univerzita v Bratislave vo Vydavateľstve STU, Bratislava, Vazovova 5, v roku 2007.

Rozsah 148 strán, 44 obrázkov, 22 tabuliek, 14,777 AH, 15,020 VH, 1. vydanie, edičné číslo 5329, vydané v elektronickej forme;
umiestnenie na http://www.svf.stuba.sk

85 – 242 – 2007

ISBN 978-80-227-2749-5

ÚVOD

Návrh objektov projektovaných v rámci krajinného inžinierstva (ako aj v iných stavbárskych odboroch) je inžinierska činnosť s určitým špecifickým obsahom, postupmi, a znalostnou bázou, ale zároveň vyžaduje aj vhodnú technológiu. Pokiaľ sa zaoberáme inžinierskymi činnosťami v rámci výuky, primárna úloha je dať predovšetkým odpoveď na otázku "ako" projektovať. Je to logicky prvotné, lebo študent ešte projektovať nevie, a preto sa takto orientované predmety vyučujú v skorších ročníkoch. Pri projektovaní v jej rutinnej a komerčnej podobe sa však táto otázka rozširuje na "Ako projektovať efektívne". Otázka správnosti návrhu, kvality a bezpečnosti projektovaných objektov by mala samozrejme zostať prvoradá, avšak keďže je inžinierska práca, tak ako každý iný druh práce aj ekonomická aktivita, čiže činnosť zameraná na zisk, je otázka jej efektívnosti nemenej dôležitá. Z tohto dôvodu sa nachádzajú v študijnom pláne aj niektoré predmety z výpočtovej techniky, orientované práve na túto stránku projekčnej činnosti. Na výučbu využitia CAD systémov v krajinnom inžinierstve (ako aj pre potreby inžinierskej praxe a iných odborov) má slúžiť predložený učebný text.

Autor sa nesnažil duplikovať to, čo je už na trhu počítačovej literatúry k dispozícii, ale skôr vyplniť informačné medzery, zvlášť tam, kde nachádza svoju podporu projektovanie krajinárskych a vodohospodárskych stavieb. Zvláštna pozornosť je venovaná užívateľskému prispôsobovaniu AutoCADu, najmä programovaniu makier. Nie sú preberané základy práce s AutoCADom, ktoré možno nájsť v rôznych knihách aj v preloženej časti helpového systému AutoCADu, ale skôr vybrané kapitoly z niektorých pokročilých techník jeho využitia.

Bratislava, máj 2007 Autor

1 VISUAL BASIC FOR APPLICATION V AUTOCADE

Užívatelia komerčných programových balíkov ako je Microsoft Office alebo AutoCAD – medzi nimi aj študenti alebo absolventi stavebnej fakulty, sa pri svojej práci často stretávajú s potrebou rozšírenia možností týchto programových prostriedkov, ktoré by umožnilo automatizovať ich každodenné rutinné činnosti. Tieto sa líšia často iba v číselných hodnotách vstupných údajov. Môže ísť o rôzne výpočty, ale aj grafické práce, ako je pri líniových stavbách krajinného inžinierstva napríklad vykreslenie pozdĺžneho profilu riešeného objektu (závlaha, úprava toku, hrádza, vodovod...).

Na riešenie komplexných projekčných úloh sa môžu použiť jednak nadstavbové, špecializované produkty pre danú oblasť projektovania – to je zvlášť vhodné (vzhľadom na cenu a potrebný čas na zvládnutie zložitých produktov ako napríklad Autodesk Civil 3D) pri väčšom objeme objednávok podobného typu a samozrejme vtedy, keď programový produkt pre danú inžiniersku činnosť existuje a je vyhovujúci (napr. z hľadiska u nás používaných technických noriem). Druhou možnosťou, hlavne pri potrebe menej rozsiahlych softwarových systémov, resp. pri absencii vyhovujúceho programu je využitie potenciálu vlastnej programátorskej tvorivosti. Takáto činnosť určite vyžaduje istý entuziazmus, avšak úlohou tejto časti skrípt je ukázať, že nejde o možnosť natoľko vzdialenú bežnému absolventovi technickej univerzity, ako by sa mohlo na prvý pohľad zdať – zvlášť preto, že existuje viac úrovní programovania, z ktorých je možné vybrať si na základe svojich potrieb a možností.

V tejto kapitole skrípt ukážeme možnosť uplatnenia a zefektívnenia si technickej práce pre občasného programátora, ktorá je v súčasnosti dostupná vďaka existencii tzv. objektovej OLE Automation technológie. Ide o vlastnosť mnohých súčasných softwarových prostriedkov na platforme Microsoft Windows využiteľnú programátorským nástrojom Visual Basic for Application, ktorý umožňuje túto technológiu na programové uspôsobovanie veľmi efektívne a pomerne jednoducho využívať.

Uvedená objektová technológia je nástroj, pomocou ktorého bežné, všeobecne používané programy, ako je Excel, Word, AutoCAD, ArcGis, CorelDRAW, ACCESS atď. zdieľajú svoje možnosti v prostredí Windows a umožňujú ich využitie z vonkajších aplikácií. Visual Basic je nástroj, ktorý vlastnosti uvedených programov integruje, prípadne pridáva celkom nové možnosti tak, aby sa vytvoril komplexný systém na automatizáciu špecifickej činnosti, napr. stavebnej vodohospodárskej projekcie ako je návrh závlahy, úpravy toku, vodovodu, či drenáže. V jednom softwarovom komplexe potom môžu byť synergicky využívané výpočtové možnosti Excelu aj grafické možnosti AutoCADu.

Visual Basic je objektovo orientovaný jazyk. Objekt je napríklad formulár (okno aké poznáme vo Windows) alebo ovládací prvok – tlačidlo na okne, rozvinovací zoznam. Niektoré objekty nemajú vizuálne rozhranie. Používaním objektov sa dá vytvoriť veľmi prehľadný a logicky usporiadaný kód. V nasledujúcich úvahách si zobrieme ako príklad objektu vec všeobecne známu z nášho života – videorekordér. –

Každý objekt je definovaný svojou triedou. Trieda predstavuje akýsi abstraktný všeobecný objekt, z ktorého vznikajú už konkrétne objekty, tzv. inštancie. Podobá sa to na formu (trieda), z ktorej vznikajú koláčiky (inštancie). V našom príklade si môžeme predstaviť abstraktný videorekordér – maketu alebo šablónu, z ktorého budeme vyrábať (predstavme si to ako kopírovanie) konkrétne projektory.

Každý objekt má svoje vlastnosti, metódy a udalosti. Vlastnosti sú charakteristiky objektu. Objekt videorekordér môže mať napr. vlastnosť Color, ktorá hovorí, akú má farbu alebo vlastnosť Enabled, ktorá má hodnotu True (pravda), ak je zapnutý a False, keď nie.

Niektoré vlastnosti môžeme aj čítať aj meniť, niektoré môžeme len čítať. Väčšinu vlastností je možné nastaviť počas návrhu – napríklad v okne Properties, s ktorým sa zoznámite pri popisovaní editora Visual Basicu. Niektoré vlastnosti je možné nastavovať len, keď aplikácia beží.

Metódy sú akcie, ktoré objekt môže vykonať. Videorekordér má napríklad metódu Play alebo Record.

Väčšina objektov má schopnosť rozpoznávať rôzne udalosti, pre ktoré môžeme napísať nejakú odpoveď. Videorekordér sa môže napríklad po určitom čase vypnúť.

Keď si teraz premietneme tento príklad do objektu formulár (okno Windows), tak zistíme, že:

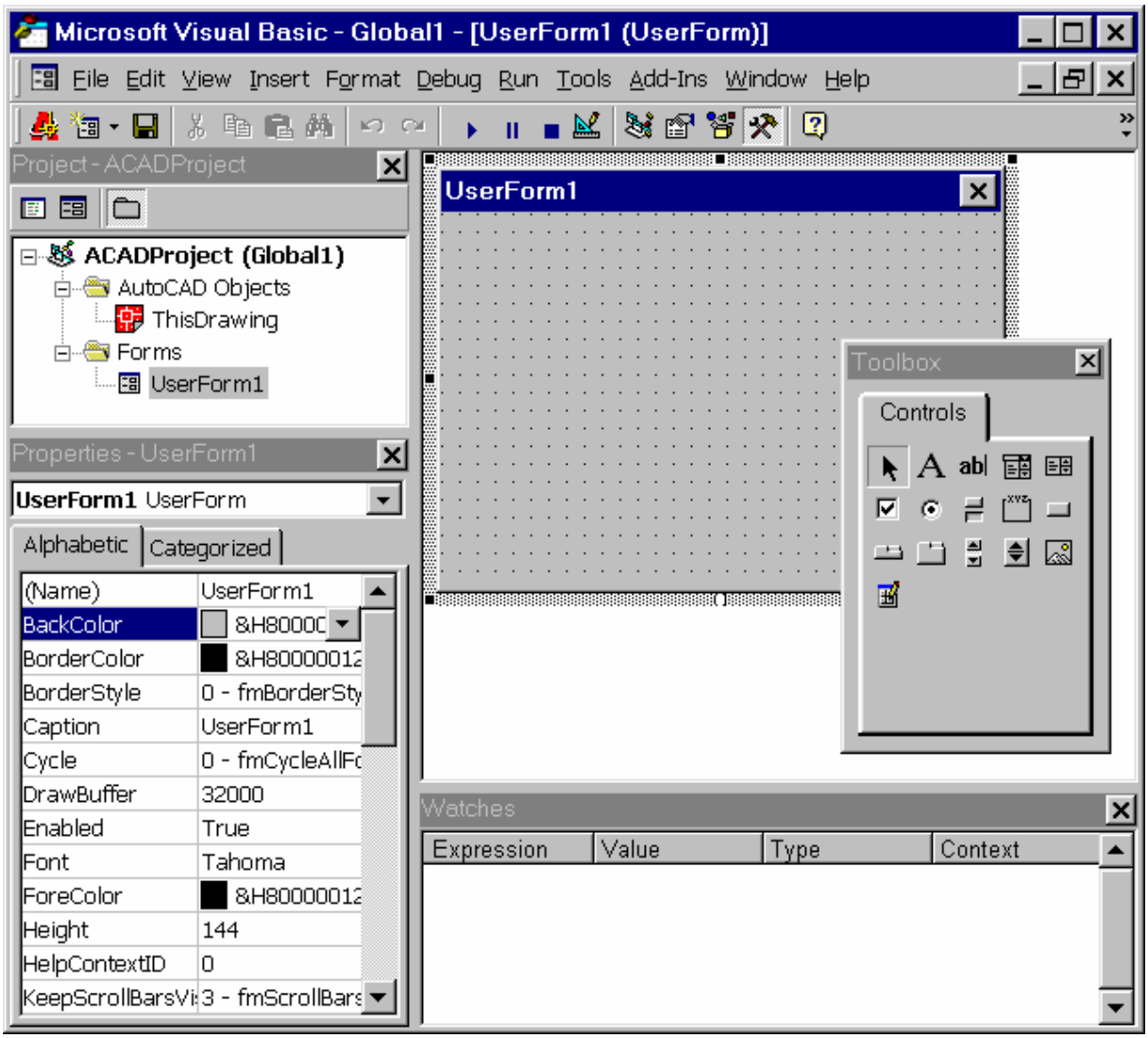
- jeho trieda sa volá Form a veľké okno, ktoré vidíme po štarte VB s názvom Form1 je inštancia triedy Form,
- má vlastosti ako Caption, ktorá určuje, aký text bude mať v hornom titulkovom pruhu alebo MaxButton, ktorá má hodnotu True, ak formulár má maximalizačné tlačidlo a False, keď nie,
- má metódu Show, ktorá ho zviditeľní a metódu Move, ktorá ho presunie na zadané súradnice,
- vie rozpoznať udalosť Click, ktorú vygeneruje, keď používateľ klikne naňho myšou.

Každý objekt (=inštancia) má vo Visual Basicu vlastnosť Name, ktorá ho jednoznačne identifikuje (túto vlastnosť nemôže mať teda viacero objektov rovnakú).

Pri ojektovo orientovanom programovaní sa dajú veľmi ľahko písať pre Windows charakteristické tzv. udalostne riadené (event-driven) aplikácie. Tie sa skladajú hlavne z kódu, ktorý odpovedá na udalosti. Napríklad ak sa aplikácia skladá z množstva príkazových tlačidiel, tak sa napíšu príslušné odpovede na udalosti Click na tlačidlo a aplikácia je hotová.

Majte tento rámec napamäti pri čítaní ďalšieho textu – zrejme až konkrétne inštancie použitia týchto princípov osvetlia presnejšie o čo ide.

Kníh týkajúcich sa programovania pomocou prostriedku Visual Basic for Application (VBA) je na trhu pomerne dostatok. Mnohé sú však veľmi obsiahle a väčšinou sa týkajú využitia VBA k úpravám jednotlivých súčastí Microsoft Office. V rámci týchto skrípt sa obmedzíme na stručný úvod do používania VBA v kontexte jeho používania neprofesionálnym programátorom – stavebným inžinierom. Ďalšie podrobnosti bude potrebné získavať z referenčných príručiek a helpových systémov programov. Zámerom bolo podať dostatok materiálu na pochopenie princípov a pre možnosť reálneho začiatku programátorskej práce. Druhým hlavným dôrazom bude uvedenie do problematiky programovania v prostredí Computer Aided Design (CAD) systému AutoCAD, keďže je pre stavebnú projekciu typický a príslušná literatúra pre túto stránku využitia AutoCADu sa na našom knižnom trhu prakticky nevyskytuje.



Obr. 1.1 Prostredie na tvorbu programu – stlačte v AutoCADe alt-F11

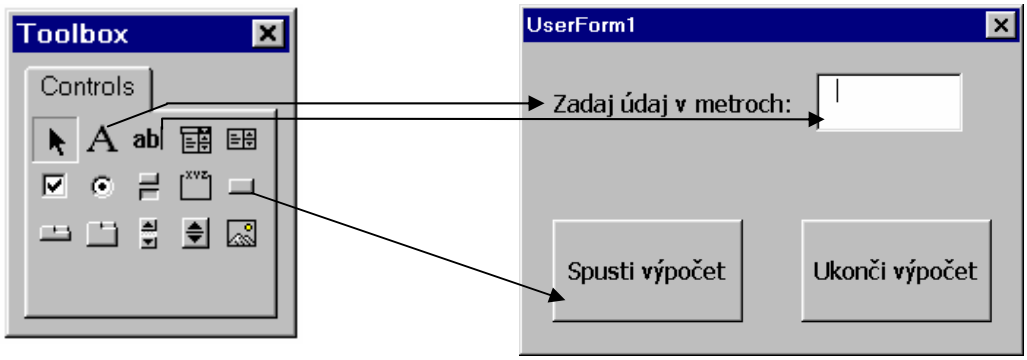
1.1 Rýchly štart

VBA je súčasťou AutoCADu od verzie AutoCAD 14.0. V tejto verzii bol pripojený ako oficiálne nepodporovaný tak, že sa nenahral pri základnej inštalácii, ale bolo ho potrebné doinštalovať špeciálnym Setup programom prítomným na inštalačnom CD. Vo vyšších verziách (od R 14.01) je už VBA oficiálnou súčasťou inštalácie AutoCADu.

Po inštalácii môžeme otvoriť integrované vývojové prostredia VBA, čo znamená zvolenie položky *Visual Basic Editor* z menu Tools ⇒ Macro. Okno, ktoré sa otvorí, je priestor, v ktorom sa vytvára alebo modifikuje program (obr. 1.1).

Po otvorení editora Visual Basicu je ďalšia činnosť, ktorú robíme pri vývoji programu vytvorenie užívateľského interface, t. j. grafického okna, ktoré sa po spustení programu ponúka na komunikáciu s užívateľom programu. Na obr. 1.1 je už zobrazená situácia po vykonaní prvého kroku – vložení UserForm1 z menu *Vložit (Insert)*. Takto vzniknuté prázdne okno je potrebné osadiť **ovládacími prvkami** zo súpravy nástrojov (Toolbox) – obr. 1.2,

ktoré umožňujú používateľovi vytvoreného programu komunikáciu s počítačom – spustenie výpočtu, zadanie vstupných údajov atď. Na obrázku je šipkami naznačené aký nástroj na Toolboxe vybrať pre vytvorenie istého prvku ovládacieho okna.



Obr. 1.2 Nástrojový panel pre pridávanie ovládacích prvkov do okna programu

Príkladom takéhoto ovládacieho prvku je *Popis (Label)*. Iným je *Textové pole (TextBox)* alebo *Príkazové tlačidlo (CommandButton)*. Pomocou ovládacieho prvku *Popis* získa užívateľ vytvoreného programu predstavu napríklad o tom, na čo slúžia jednotlivé oblasti grafického okna. Pred textovým poľom, do ktorého je potrebné zadať nejakú hodnotu, bude napríklad pomocou ovládacieho prvku *Popis* vhodné napísať akú hodnotu a v akých jednotkách má užívateľ zadať. Po zadaní vstupných údajov sa spustí výpočet pomocou príkazového tlačítka. Iným príkazovým tlačidlom sa program ukončí (obr. 1.2). Pokiaľ ide o umiestnenie a stanovenie rozmerov ovládacích prvkov, tieto sa na formulári kreslia myšou.

Každý ovládací prvok – objekt – má **vlastnosti**. Okno s vlastnosťami (Properties) vybraného ovládacieho prvku (prípadne samotnej UserForm) je v integrovanom vývojovom prostredí na obr. 1.1 vľavo dole. Vlastnosti hovoria hlavne o tom, ako má ovládací prvok vyzerať. "Width" je vlastnosť, ktorá určuje, ako má byť ovládací prvok široký. "Top" je vlastnosť, ktorá určuje umiestnenie ovládacieho prvku vzhľadom na vrch UserForm. Okrem uvedeného kreslenia ovládacích prvkov môžeme teda tieto rozmerové vlastnosti nastavovať v paneli *Vlastností* aj číselne. "Caption" je popis, ktorý sa zobrazuje na ovládacom prvku, "Font" určuje akým typom písma. Zatiaľ uvádzame iba tieto príklady. V kóde VBA sa na vlastnosti objektov v programe odkazuje formou zápisu s bodkou, pričom najprv sa napíše názov objektu, potom názov vlastnosti a obidva tieto prvky sa oddelia bodkou. Ak napríklad chcete zmeniť názov UserForm, použijete vlastnosť UserForm1.Caption:

UserForm1.Caption = "Nový popis okna"

S každým ovládacím prvkom sú spojené možné **udalosti** – užívateľ programu klikne myšou, raz alebo dvakrát, prejde s myšou ponad prvok bez kliknutia, alebo zadá z klávesnice do textového okna hodnotu. Toto všetko a mnoho iných akcií sú pre ovládacie prvky udalosti. Ovládací prvok vie, že sa stali. Na udalosti reaguje program pomocou udalostných procedúr, ktorých obsah zostavuje autor programu. Udalostné procedúry sú v názve charakterizované tak, že je jasné, ku ktorému ovládaciemu prvku a k akej udalosti na tomto prvku sú viazané. Napríklad nasledovný zdrojový text (text programu vo Visual Basicu) označuje udalostnú procedúru, ktorá reaguje na kliknutie myšou na príkazové tlačidlo č. 1:

Sub *CommandButton1_Click()*
.
.
End Sub

Kľúčové slová Sub a End Sub vyznačujú začiatok a koniec procedúry. Priestor medzi názvom a koncom procedúry je zatiaľ prázdny, jeho obsah musí napísať autor programu, inak sa po kliknutí na príkazové tlačidlo č. 1 (CommandButton1) nič nestane. Na to aby sme vedeli vytvoriť tieto časti programov sú potrebné dva základné okruhy vedomostí, z ktorých každý neskôr preberieme:

- Je potrebná znalosť syntaxe jazyka Visual Basicu (t. j. ako vyjadrovať matematické či logické postupy vo výkonnej časti programu a mnoho ďalších vecí)
- V prípade, ak chceme využívať možnosti programov, ktoré sú na našom počítači nainštalované (napr. využiť AutoCAD aby automaticky nakreslil výkres), musíme byť zoznámení so štruktúrou a pravidlami využívania tzv. objektových modelov týchto aplikácií.

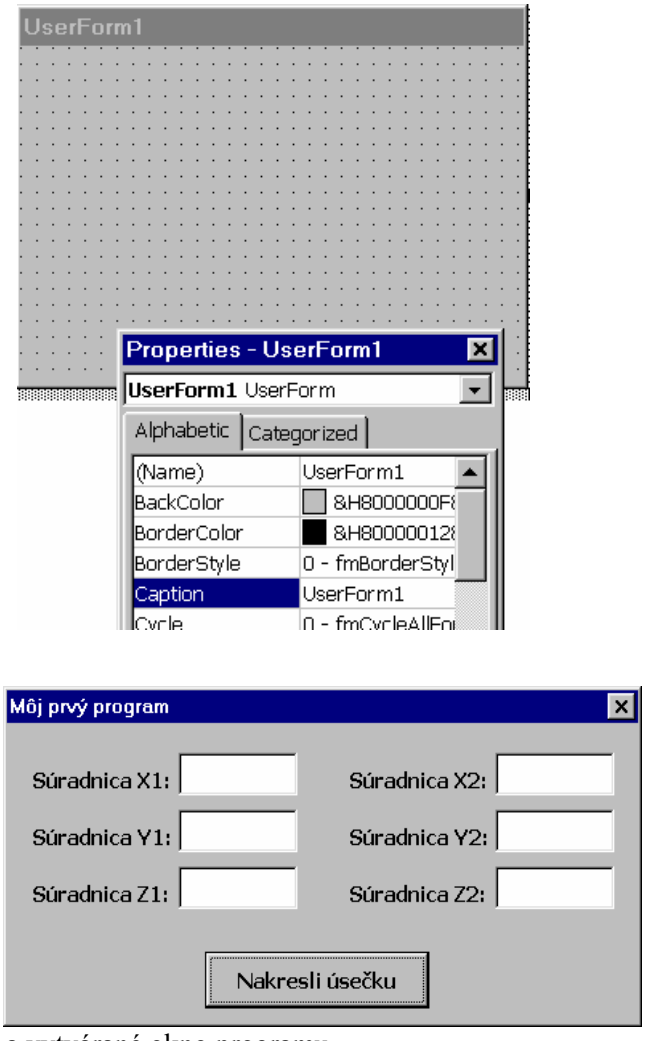
Ešte jeden základný pojem v tejto časti – ovládací prvok má **metódy**. Metóda je akcia, ktorú objekt vie vykonať. Napríklad pomocou metódy *Copy* kopírujete z TextBoxu jeho obsah do schránky Windows (odtiaľ ho môžete niekam nalepiť ako to každý užívateľ WORDu dobre pozná). Pravda, musíte mu to povedať rečou, ktorej rozumie – Visual Basicom (*TextBox1.Copy*). Iný príklad na metódu: K ovládacím prvkom okien Windows patria aj rozvinovacie zoznamy, z ktorých užívateľ programu vyberá preddefinovanú hodnotu, známu napr. z typických dialógových okien pre uloženie súboru – vyberáme pomocou nich z niekoľkých preddefinovaných hodnôt typ súboru v akom chceme svoj dokument uložiť (textový ASCI súbor, dokument WORDu…). Tento ovládací prvok sa nazýva Combo Box. Ak chce programátor pridať položku do Combo Boxu používa metódu *AddItem*, t. j. pridaj bod. V AutoCADe poznáte napríklad metódu *Explode*, ktorá rozloží napríklad krivku na úsečky. Podobne, ako v prípade vlastností, pri volaní metód sa používa zápis s bodkou. Ak chcete napríklad explodovať objekt krivka napíšete *Krivka.Explode*.

V ďalšom texte ukážeme postup tvorby jednoduchého programu – zatiaľ sa uspokojíme s nakreslením úsečky medzi dvoma zadefinovanými bodmi. Hlavné zadávané vlastnosti budú súradnice jej počiatočného a koncového bodu a potom použijeme metódu *AddLine*, čo je vlastne metóda vznikajúceho výkresu AutoCADu, presnejšie jeho modelového priestoru.

1. **Otvoríme AutoCAD a v ňom editor Visual Basicu** (Alt – F11 alebo z menu Nástroje → Makro)**.** Zatiaľ obsahuje iba rôzne vyššie uvedené okná – základné je okno Projektu, ktoré bude navigátorom po vytváranom programe.
2. **Z menu Vložit alebo Insert vložíme do projektu VBA UserForm.** Jej konkrétna inštancia sa bude volať UserForm1. Vo vývojovom prostredí pribudnú dve ďalšie okná: Ovládacie prvky (Toolbox) a Vlastnosti (Properties). Keby sa tak nestalo, zobrazíte ich z menu View. Pomocou Toolboxu pridávame na UserForm ovládacie prvky. V okne Vlastnosti nastavujeme jednotlivé vlastnosti aktívneho objektu – či už to je samotná UserForm1, alebo niektorý ovládací prvok. V okne Projektu, môžeme (podobne ako vo Windows Exploreri) pomocou štvorčekov, v ktorých je znamienko + alebo – rozvinúť, resp. stiahnuť (kliknutím) zložku Forms, v ktorej je aj naša UserForm1. Stlačením pravého tlačítka myši na nej si môžeme z ponúknutého kontext-senzitívneho menu napr.

zvoliť, či chceme vidieť programový kód UserForm1, ju samotnú ako grafický objekt alebo vykonať iné veci.

3. **S vlastnosťami môžeme trochu experimentovať.** Titulok UserForm1 v modrom pruhu je vlastnosť. Zmeníme ju tak, že v okne vlastností (Properties) vyhľadáme v ľavom stĺpci vlastnosť s názvom Caption – to je titulok – a v pravom stĺpci nastavím hodnotu tejto vlastnosti napr. na text: "Môj prvý program" (obr. 1.3). Podobnou stratégiou meníme aj iné vlastnosti. Niektoré vlastnosti ponúkajú pre svoje hodnoty rozvinovacie zoznamy s možnosťami aké môžu nadobúdať (napr. BackColor – farba pozadia UserForm) iné majú v okne hodnoty tlačidlo s troma bodmi (objaví sa až po kliknutí na príslušné okienko) – napr. Picture alebo Font. Takéto tlačidlo vyvolá dialógový panel, ktorý umožní vybrať pre vlastnosť jej hodnotu ako obrázok alebo typ písma (vlastnosť Font).

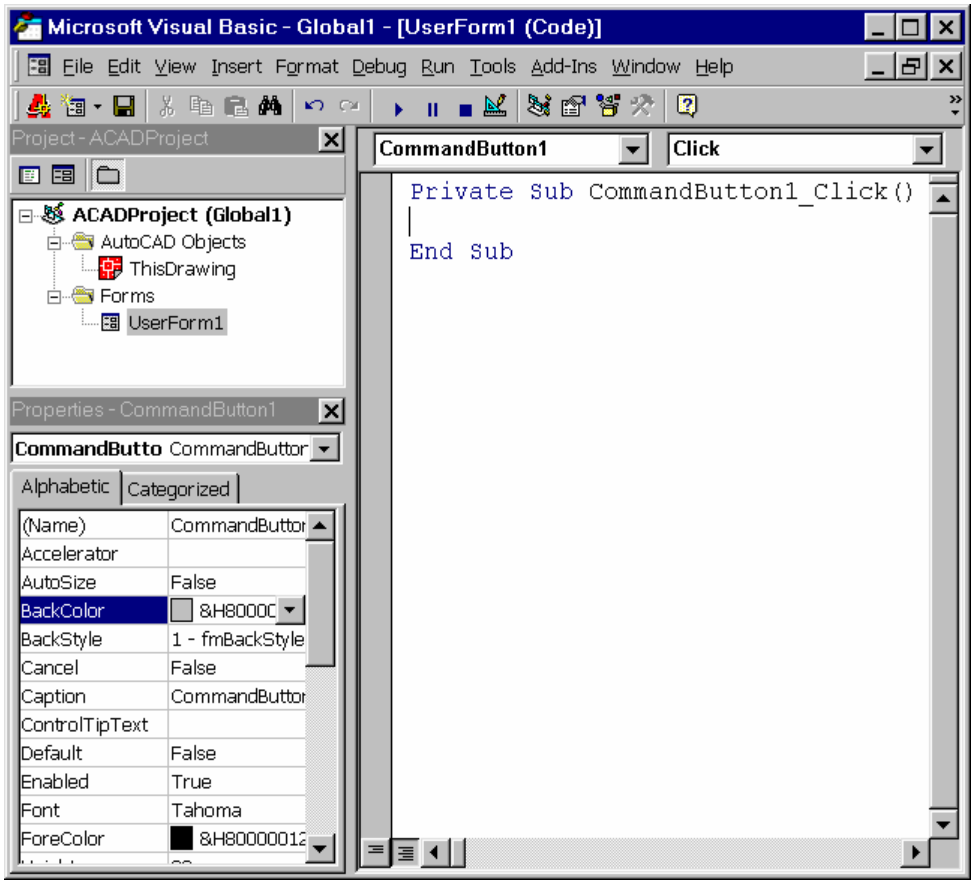


Obr. 1.3 Panel vlastností a vytvárané okno programu

4. **Teraz pridáme na UserForm1 ovládacie prvky podľa obrázka 1.3.** Uskutočníme to kliknutím na príslušný ovládací prvok na Toolboxe a následným zakreslením jeho polohy a veľkosti na UserForm1. Vložíme 6 popisných prvkov (Label) symbolizovaných na Toolboxe veľkým písmenom A –  . Obsah textu zadáme buď editovaním vlastnosti Caption Label v okne vlastností, alebo editovaním tohto textu priamo na UserForm1.

Ďalej vložíme 6 prvkov typu TextBox – ovládacích prvkov do ktorých bude používateľ programu zadávať príslušné hodnoty súradníc bodov úsečky, ktorú chceme týmto programom nakresliť. TextBox sa vkladá pomocou tlačítka  . Napokon vložíme CommandButton (). Nápis, ktorý je na ňom ("Nakresli úsečku") sa opäť nastavuje pomocou vlastnosti Caption. Nezabudnite: meníte vlastnosti toho prvku, ktorý je aktívny – zvýraznený štvorčekmi. Všimnite si, že ovládacie prvky sú číslované. Ak potrebujete zväčšiť nejaký ovládací prvok, vrátane UserForm, môžete to vykonať po jeho vyselektovaní ťahaním za úchytné body, ktoré sú zvýraznené bielymi štvorčekmi. Vyselektovaný objekt môžete tiež vymazať pomocou klávesy Delete, premiestniť ťahaním myšou pri stlačenom ľavom tlačidle myši a dokonca kopírovať, ak pri ťahaní myšou držíte zároveň stlačenú klávesu Ctrl. Môžete naraz vyselektovať viac objektov a robiť tieto úkony hromadne. Samotná selekcia sa uskutoční nakreslením obdĺžnika okolo príslušného prvku v režime, keď je na Nástrojovom paneli (Toolboxe) vybratá šípka v prvom rade vľavo. Mimochodom – ak chcete mať nejaké hodnoty súradníc v textboxoch preddefinované, t. j. po spustení programu nebudú prázdne textboxy, ale bude v nich nejaká hodnota (napr. v z-ových súradniciach bude 0) nastavíte ich v príslušnom TextBoxe pomocou vlastnosti Value v okne vlastností.

Udalosti. Program, ako sme ho zatiaľ vytvorili, je síce možné spustiť – z prostredia editoru Visual Basicu to môžeme urobiť klávesou F5, ikonkou na hlavnom paneli nástrojov v tvare  , alebo z menu voľbou príkazu Run. Môžeme zadať do TextBoxov v spustenom programe nejaké hodnoty, ale po stlačení príkazového tlačidla "Nakresli úsečku" sa nič nedeje. Je to preto, že stlačenie (jednoduché kliknutie na príkazové tlačidlo) obsluhuje udalostná procedúra *CommandButton1_Click(),* o ktorej už bola v predchádzajúcom texte zmienka. Táto zatiaľ nebola vytvorená. Aby sme s ňou mohli začať robiť, je potrebné na UserForm1 zobrazenej v editori Visual Basicu dvakrát kliknúť ľavé tlačidlo myši. To je jeden zo spôsobov ako sa dostaneme do okna kódu – obr. 1.4 a zároveň v ňom vytvoríme hlavičku (začiatok a koniec) potrebnej procedúry. Po uvedenom dvojkliku na CommandButton sa teda dostávame na miesto podľa obr. 1.4, čo je vlastne telo procedúry na obsluhu kliknutia myšou na CommandButton (jednoduchého, nie dvojitého) už počas behu funkčného programu. Program vykoná všetky príkazy medzi hlavičkou s názvom procedúry (ktorý indikuje na akú udalosť sa táto procedúra vyvolá) a koncom procedúry. Úlohou je teraz tieto príkazy napísať.



Obr. 1.4 Okno kódu programu

5. **Písanie programového kódu.** V prvom rade napíšeme do tela procedúry príkazy na tzv. deklaráciu premenných. K tejto téme sa bližšie vrátime, zatiaľ iba toľko, že vo vzorcoch a iných príkazoch programu používame premenné, ktoré sa naplnia konkrétnymi hodnotami počas behu programu (napríklad tak, že užívateľ programu zadá hodnoty súradníc do TextBoxov a odtiaľ si ich program prečíta). Tieto premenné je vhodné deklarovať, čo znamená zadefinovať, aký druh hodnoty bude určitá premenná nadobúdať. Môže to byť napríklad celé číslo (Integer), desatinné číslo (typ Double, Single) ale aj text alebo *objekt* z AutoCADu – úsečka a podobne. V našom prípade na ďalší riadok za hlavičku procedúry *CommandButton1_Click()* napíšeme:

Dim usecka As Object
Dim Bod1(0 To 2) As Double
Dim Bod2(0 To 2) As Double

AutoCAD ako aplikácia je v objektovom modeli základom, z ktorého sa nám sprístupňujú v stromovej štruktúre ďalšie objekty – napríklad objekt úsečky v AutoCADe, ktorý je deklarovaný v prvom riadku (*usecka* – v názvoch premenných nepoužívajte interpunkciu). Ďalšie dve deklarácie sa týkajú bodov, ktoré budeme pre zadanie úsečky potrebovať, tieto dve premenné sú trojprvkové vektory s indexmi 0 až 2, t. j. Bod1(0) je x-ová súradnica,

Bod1(1) y-ová súradnica prvého bodu atď. Indexovanie vektorov sa implicitne vo Visual Basicu začína nulou a nie jednotkou – možno to však zmeniť, ale to je zatiaľ nuansa.
Ďalším krokom bude načítanie užívateľom zadaných údajov o súradniciach bodu z príslušných textboxov do týchto deklarovaných premenných, keďže to je miesto kam ich zadá užívateľ programu. Využijeme vlastnosť Value, ktorá vyjadruje hodnotu zadanú v textboxe, do premenných podľa tohto kódu:

```
Bod1(0) = TextBox1.Value
Bod1(1) = TextBox2.Value
Bod1(2) = TextBox3.Value
Bod2(0) = TextBox4.Value
Bod2(1) = TextBox5.Value
Bod2(2) = TextBox6.Value
```

Vykreslenie úsečky vyžaduje trochu dlhší výraz, ktorý vyzerá takto:

```
Set usecka = ThisDrawing.ModelSpace.AddLine(Bod1, Bod2)
```

Zmysel tohto výrazu je: pridávame úsečku (*AddLine*) definovanú bodmi *Bod1* a *Bod2* do modelového priestor (*ModelSpace*) aktívneho dokumentu (*ThisDrawing*), ktorý je práve v AutoCADe otvorený. Ako vidieť, keď chceme narábať s nejakým objektom, je potrebné pri odvolávaní sa naň definovať prístupovú cestu v stromovej štruktúre objektového modelu. Z tohto dôvodu helpový systém AutoCADu obsahuje grafickú reprezentáciu objektového modelu, ktorá všetky tieto vzťahy zobrazuje (Obr. 1.17). Aby sa samotný objekt úsečky hneď zviditeľnil na obrazovke, je potrebné doplniť výraz:

```
usecka.Update
```

Celá časť programu (subrutina) obsluhujúca stlačenie príkazového tlačidla na vykreslenie úsečky vyzerá takto:

```
Private Sub CommandButton1_Click()
'Definovanie typu údajov:
   Dim usecka As Object
   Dim Bod1(0 To 2) As Double
   Dim Bod2(0 To 2) As Double

   'Načítanie údajovo súradniciach dvoch bodov  z textboxov:
   Bod1(0) = TextBox1.Value
   Bod1(1) = TextBox2.Value
   Bod1(2) = TextBox3.Value

   Bod2(0) = TextBox4.Value
   Bod2(1) = TextBox5.Value
   Bod2(2) = TextBox6.Value

   'Kreslenie úsečky:
   Set usecka = ThisDrawing.ModelSpace.AddLine(Bod1, Bod2)
```

```
    usecka.Update
End Sub
```

Riadky s úvodzovkami na začiatku sú komentáre, pomocou ktorých si píše programátor do programu poznámky, aby sa v ňom vyznal – nemajú nijaký vplyv na samotný program a editor Visual Basicu ich zvýrazňuje zelenou farbou. Po spustení vytvoreného makra a zadaní hodnôt súradníc (ak budete zadávať desatinné čísla použite ako oddeľovač desatín bodku) vás už tlačidlo *Nakresli úsečku* nesklame a úsečka sa naozaj nakreslí. Nezabudnite váš prvý program uložiť. Keď budete chcieť vyvolávať program kliknutím na ikonu použite postup popísaný v kapitole o makrách, pričom je potrebné zabezpečiť, aby bol vo výkrese, v ktorom ho použijete načítaný (príkaz menu Nástroje ⇒ Načíst aplikaci ⇒ Tlačidlo *Obsah...* v rámčeku *Při spuštení*. Príkaz, ktorý napíšete do položky Makro v CUI je *–VBARun NázovPrograma*, pričom majte program uložený v niektorej z prehľadávaných ciest AutoCADu alebo zadajte do *NázovPrograma* celú cestu.
Môžete si trochu zaexperimentovať, aby ste videli že VBA môže urobiť pri tých istých vstupoch aj niečo iné než príkaz úsečka. Nasledujúca sekvencia príkazov napíše v strede úsečky jej dĺžku:

```
'Výpočet stredu úsečky
Bod1(0) = (Bod1(0) + Bod2(0)) / 2
Bod1(1) = (Bod1(1) + Bod2(1)) / 2
Dim dlzkaText As Object
'Parametre v pravej zátvorke sú AutoCADu známa dĺžka objektu, vkladací bod textu
'(pôvodný Bod1 sme predefinovali) a výška textu –2  nemusí byť vždy vhodná, môžete
'ju zmeniť
Set dlzkaText = ThisDrawing.ModelSpace.AddText(usecka.Length, Bod1, 2)
DlzkaText.Update
```

1.2 Práca s editorom Visual Basicu

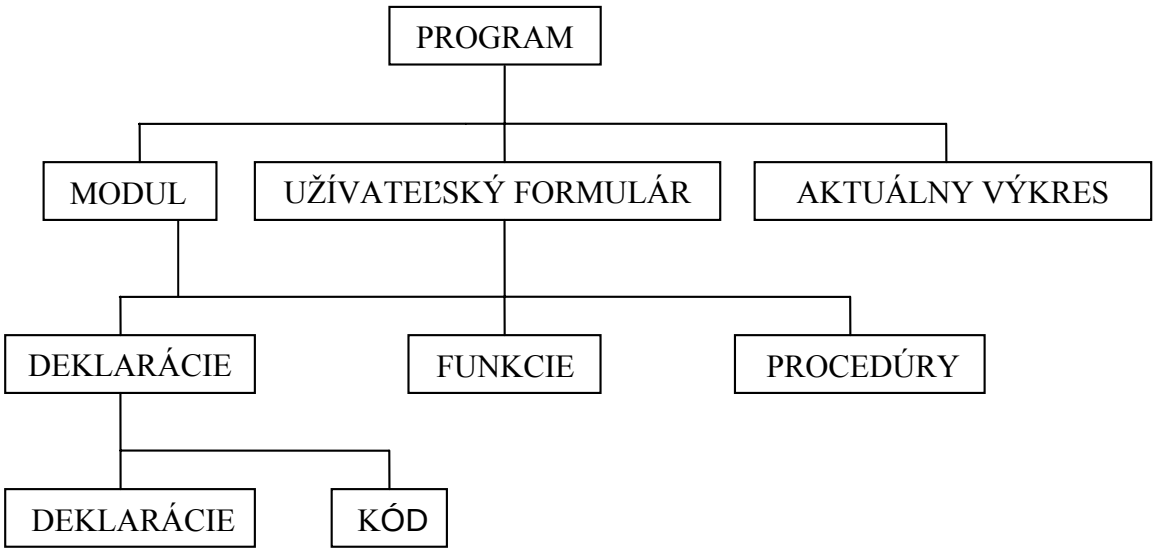
Ako sme ukázali v predchádzajúcej kapitole, kód programu VBA vytvárame v editori Visual Basicu. Ide o komplexné vývojové prostredie (IDE – Integrated Development Environment), obsahujúce nástroje, ktoré umožňujú vytvárať a spravovať projekty VBA. V tejto časti sa s týmto vývojovým prostredím zoznámime bližšie.

Často sa stane, že do projektu budeme nahrávať už existujúci program vo VBA, keďže existuje veľa zdrojov, napríklad na internete, kde možno nájsť užitočné zadarmo ponúkané programy AutoCADu vo VBA. Na tento účel používame príkaz *VBALoad*, ktorý môžeme napísať na príkazový riadok a následne zadať názov súboru (s príponou .dba) alebo ho vyvoláme z menu Tools (Nástroje) ⇒ Macro (Makro) ⇒ Load project (Načíst projekt). Tu máme dostupný aj VBA Manager (Správce VBA...), ktorý je možné využiť aj na odstránenie projektu z AutoCADu. Príkaz *RunVBA* slúži k spusteniu projektu VBA, avšak tento už musí byť načítaný – otvorí sa dialógový panel *Makra...*, ktorý inak tiež vyvoláme z menu Tools (Nástroje) ⇒ Macro (Makro).

Jednotlivé hlavné prvky používané v projektoch VBA, ktoré po nahratí existujúceho projektu môžeme modifikovať alebo vytvárať od začiatku sú:

- Objekt dokumentu AutoCADu – *ThisDrawing*
- moduly, ktoré obsahujú kód – deklarácie, funkcie, procedúry,

- používateľské formuláre, t. j. zobrazené okno windows, do ktorého môžete umiestniť grafické ovládacie prvky, ako sú tlačidlá, obrázky a oblasti textu a ktoré zároveň obsahuje aj programový kód.
- moduly tried, t. j. definície používateľom definovaných objektov, ktoré ste pre projekt vytvorili alebo prebrali z iných projektov (pokročilejšia téma, v rámci skrípt nebude preberaná)



Obr. 1.4 Hlavné časti projektu VBA v AutoCADe

Editor Visual Basicu sa skladá z menu, nástrojových líšt a z niekoľkých okien.

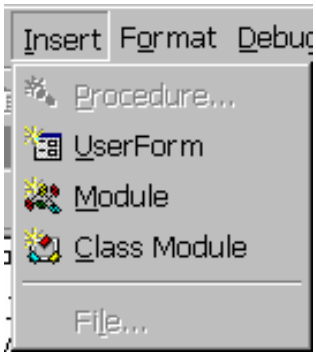
Základná obsluha z menu je podobná ako pri iných aplikáciách Windows. V menu File môžeme uložiť projekt, importovať alebo exportovať prácu, tlačiť a podobne. Keď sa vám niečo stratí z vývojového prostredia (nejaké okno a pod.), je zvyčajne možné zobrazenie danej položky z menu View – napríklad ho využijete ak nebudete mať zobrazenú paletu s nástrojmi. Debugovanie z príslušného menu umožňuje napríklad krokovať (postupne počítať) program po riadkoch. Zároveň môžete sledovať aké hodnoty nadobúdajú premenné napríklad tak, že kliknete (pred debugovaním) na premennú pravým tlačidlom myši a pomocou voľby Add watch ju pridáte do okna Watches, kde sa zobrazujú aktuálne informácie o nej. Posledné štyri riadky sú dôležité – nezabudnite na ne, keď budete skúšať písať programy! Z menu Tools (Nástroje) je možné zvoliť položku Project Properties (vlastnosti projektu), kde môžeme zmeniť napríklad meno projektu. Položky menu Window a Help sú pre užívateľov Windows známe.

Niektoré z možností menu máte tiež dostupné z paliet nástrojov, ako tomu zväčša vo Windows programoch býva.

Okno projektu sprístupňuje formou podobnou Windows Exploreru všetky hore uvedené súčasti projektu – vytvorený programový kód v jednotlivých moduloch, okná pre vstup údajov a interakciu s užívateľom (UserForms) a objekty, ktoré sú súčasťou aplikácie, ktorá je pre VBA hostiteľská (ThisDrawing). Slúži k celkovému prehľadu a správe projektu. Obsahuje svoj vlastný nástrojový panel s troma ikonkami, ktorý umožňuje otvárať jednotlivé komponenty. Tlačidlo *View Code* umožňuje zobraziť modul kódu, ktorý je pripojený k objektu; tlačidlo View Object umožňuje zobraziť objekt samotný, t. j. napr. User Form.

Podobné funkcie sprostredkúva aj kontext senzitívne menu ukazujúce sa po stlačení pravého tlačidla myši na príslušnom komponente. Toto menu umožňuje tiež pridávať a odoberať komponenty projektu.

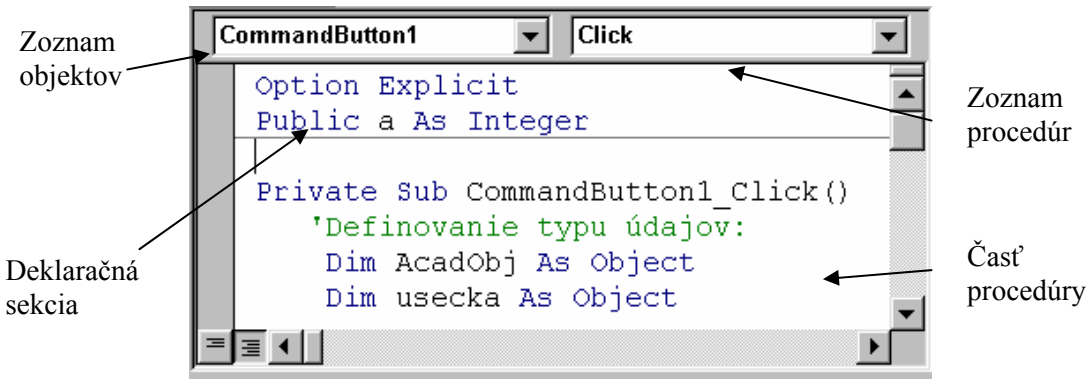
Jednou zo základných činností v tomto okne je pridávanie nových komponentov do programu cez kliknutie pravým tlačidlom na názov projektu a zvolení Insert. Podobnú činnosť umožňuje aj základné menu editora. Po výbere položky Insert z menu môžeme pridávať užívateľský formulár, modul alebo tzv. Class modul, o ktorom sme zatiaľ nehovorili.



Obr. 1.5 Používanie menu Insert pre vkladanie nového formuláru alebo modulu

Okno vlastností (Properties Window) je nástrojom IDE, ktorý umožňuje zobrazenie a nastavenie vlastností jednotlivých objektov počas vytvárania aplikácie (programu). Jedná sa predovšetkým o objekty užívateľského interface programu. Vlastnosť je charakteristika objektu ako napr. jeho meno, farba veľkosť atď. Ak toto okno nie je viditeľné v IDE, zvolíme z menu *View* položku *Properties Window* alebo stlačíme *F4*. Pod hlavičkou okna je rozvinovací zoznam, ktorý umožňuje zvoliť objekt, ktorého vlastnosti nás zaujímajú. Záložkami si môžeme zvoliť typ usporiadania vlastnosti tohto objektu podľa abecedy alebo kategórií. V ľavom stĺpci sú názvy vlastností a v pravom je hodnota príslušnej vlastnosti. Klávesa *F1* umožňuje získať Help k danej vlastnosti, na ktorej sme nastavení (je zvýraznená modrým pruhom).

Okno kódu je priestorom, do ktorého vpisujeme príkazy Visual Basicu. Poskytuje bohaté možnosti pri editovaní a vytváraní programového kódu. Ak chceme vidieť programový kód, ktorý je priradený ku objektu, klikneme na tento objekt v *Project Exploreri* pravým tlačidlom myši a zvolíme *View Code*. Na obr. 1.7 vidíme jednotlivé časti tohto okna. V hlavnej časti okna rozoznávame deklaračnú sekciu, v ktorej sa deklarujú premenné spoločné pre celý modul (resp. celý projekt, ale to nie je námetom tejto kapitoly), zvyšnú časť tvorí zoznam procedúr. Pod hlavičkou tohto okna sú dva dôležité rozvinovacie zoznamy – zoznam objektov a zoznam procedúr. Umožňujú navigáciu vo väčších programových moduloch. Ak hľadáme procedúru viazanú na udalosť (napr. kliknutie myšou na príkazové tlačidlo č. 1), nastavíme v ľavom okne objekt *CommandButton1* a v pravom udalosť – *Click*.

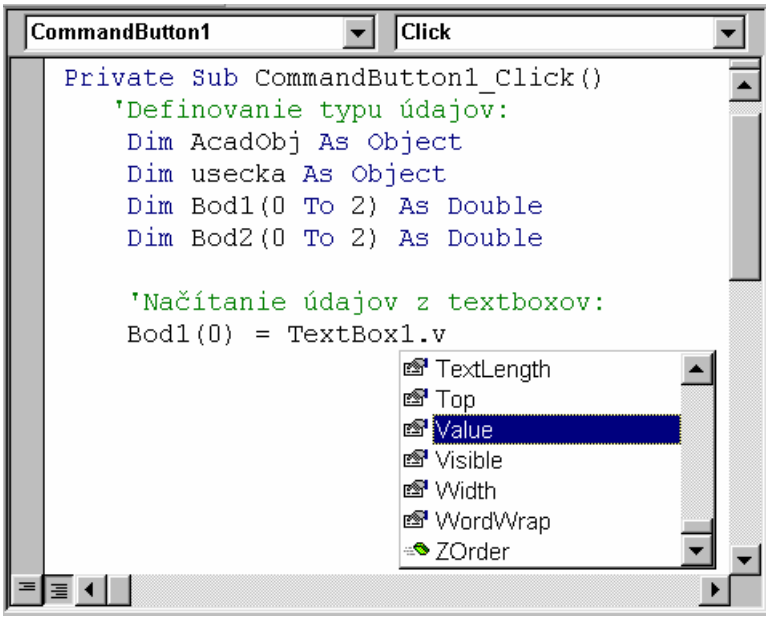


Obr. 1.6 Časti modulu kódu

V prípade, že v texte programového kódu píšeme odkaz na metódu alebo vlastnosť nejakého objektu, syntax je takáto:

MenoObjektu.Vlastnosť

Namiesto vlastnosti môže byť metóda. K týmto pojmom sa ešte vrátime, zatiaľ iba toľko, že vo chvíli, ako napíšeme bodku, ktorá indikuje editoru, že chceme použiť vlastnosť alebo metódu objektu, editor ponúkne rozvinovací zoznam s vlastnosťami a metódami prislúchajúcimi danému objektu. Keď píšeme prvé písmená zamýšľanej vlastnosti, zo zoznamu sa automaticky vyberá vlastnosť s týmito začiatočnými písmenami. Nemusíme ju dopísať, stlačením klávesy SPACEBAR (medzerník) dokončí editor túto činnosť automaticky. Tým sa šetrí čas a zabezpečuje správna syntax. Podobné správanie sa editoru je aj pri vkladaní konštánt a iných parametrov do funkcií. Mnoho slov, bude editor vedieť dokončiť po stlačení klávesovej kombinácie *CTRL+SPACEBAR*!



Obr. 1.7 Zoznam vlastností objektu TextBox ponúknutý počas editácie v okne kódu

Editor Visual Basicu ďalej pomáha rozlišovať nevýkonné, popisné časti programu, ktoré si píše autor pre vlastnú potrebu zelenou farbou. Aby ste si takéto poznámky mohli do programu vpisovať, treba pred príslušný text napísať jednoduchu úvodzovku (je dostupná v anglickej klávesnici, alebo pomocou stlačenia klávesy alt a napísania čísla 39). V prípade chyby zvýrazňuje editor chybnú časť červenou farbou.

1.3 Tvorba užívateľského prostredia programu

Užívateľské formuláre (Microsoft Forms) používame na tvorbu užívateľského interface programu, t. j. dialógových okien, pomocou ktorých užívateľ ovláda program. Sú miestom na ktoré vkladáme ovládacie prvky ako sú príkazové tlačidlá, polia na zadávanie údajov (TextBoxy), rozvinovacie zoznamy atď – tzv. ActiveX controls. Už sme ukázali, že novú UserForm vytvoríme pomocou menu Insert v editori VBA, zvolením položky UserForm.

1.3.1 *Pridanie ovládacích prvkov (Controls) do formulára*

Ovládacie prvky majú rôzne funkcie, počnúc od jednoduchších, ako je zobrazenie textu na UserForm, ale aj zložitejšie, ako napr. nakreslenie kompletného kalendára. Pridávanie ovládacích prvkov umožňuje Nástrojový panel – Control Toolbox. Na vyžadovanom prvku na Toolboxe klikneme myšou, potiahneme ho na formulár a uvoľníme tlačidlo myši. Prípadne iba klikneme na prvok a nakreslíme ovládací prvok na UserForm v požadovanom mieste a tvare. Ďalším krokom je prispôsobiť tvar a správanie tohto ovládacieho prvku, čo znamená nastaviť jeho vlastnosti.

1.3.2 *Nastavenie vlastností*

Ak napríklad vytvoríme objekt Človek, musíme ho nejakým spôsobom opísať. Napríklad vek, farba očí, výška atď. Môžeme povedať, že to sú jeho vlastnosti. To isté majú aj objekty vo VB. Napríklad objekt formulár má nejakú farbu, titulok, veľkosť atď. Vlastnosti každého objektu môžeme meniť *za behu programu* (kódom) napríklad takto: titulok formulára: UserForm1.Caption="Môj formulár" a zisťovať napríklad *Debug.Print UserForm1.Caption*. Takto vypisujeme rôzne hodnoty do okna Immediate – zobrazte si ho z menu View, ak ho nevidíte vo vývojovom prostredí.

K nastavovaniu vlastností *pri tvorbe programu* slúži okno Properties. Vo vrchnej časti okna je rozbaľovacie menu, v ktorom si vyberiete objekt, ktorého vlastnosti chcete zmeniť. To je možné vykonávať aj tak, že kliknete na prvok, ktorý chcete meniť. Pod týmto menu sa nachádzajú v ľavom stĺpci vlastnosti, ktoré možno nastavovať a v pravom ich hodnoty – na prvej záložke zoradené podľa abecedy, na druhej podľa kategórie. Ak kliknete na vlastnosť, v spodnej časti sa zobrazí jej stručný popis. Vlastnosť môžete zmeniť tak, že na ňu kliknete a buď hodnotu napíšete, alebo si vyberiete zo zoznamu. Niektoré vlastnosti je možné nastavovať iba počas behu aplikácie (v kóde, v okne Properties sa ani neobjavujú), niektoré zase iba pri návrhu, väčšinu vlastností však môžete nastaviť aj pri návrhu aj počas behu aplikácie, teda v obslužnom programovom kóde.

1.3.3 *Práca s udalosťami*

Udalosť nastane po nejakej špecifickej akcii užívateľa alebo systému. Potom ako nastane, spustí sa kód, ktorý napíšete do procedúry zviazanej s touto udalosťou. Každý objekt vo VBA môže mať mnoho prislúchajúcich udalostí. Napr. udalosť CLICK, ak niekto klikne myšou na objekt, KEYPRESS, ak bola stlačená klávesa apod. Pretože je VBA udalosťami riadený programovací jazyk, všetky jeho reakcie sú iniciované udalosťami. Napr. ak nastavíte spúšťanie projektu z formulára, spustí sa hneď niekoľko udalostí, napr. UserForm_Initialize (pokiaľ do nej napíšete nejaký kód). Ako bolo vysvetlené pri opise Editora VBA, písanie kódu sa uskutočňuje v okne kódu a pri písaní udalostných procedúr sa využívajú dva rozvinovacie zoznamy na vrchu tohto okna, z ktorých prvý identifikuje objekt, na ktorý sa má daná udalosť viazať a druhý identifikuje udalosť, nakoľko pre každý objekt ich je niekoľko.

1.3.4 *Stručný prehľad niektorých ovládacích prvkov*

V tejto kapitole budú uvedené, najbežnejšie ovládacie prvky, ktoré sa používajú vo Windows. Ide napríklad o Label (popisné pole), TextBox (textové pole), CommandButton (tlačidlo), ComboBox – rozvinovací zoznam, z ktorého užívateľ volí niektorú možnosť a ďalšie. So všetkými sa stretávame vo väčšine programov Windows, takže ich význam poznáte.

Samotný formulár – okno, na ktorom sú ovládacie prvky vždy umiestnené, programovo spúšťame zväčša volaním procedúry umiestnenej v module (ak váš projekt obsahuje modul – niekedy máme v projekte iba jeden formulár, ktorý sa automaticky otvorí pri spustení programu). V prípade volania z modulu najčastejšie aplikujeme na formulár metódu *Show*:

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

Formulár je zatváraný buď tlačidlom s príslušným kódom, alebo kliknutím na krížik v jeho záhlaví. Programové ukončenie ukazuje príklad. Pamätajte si, že na formulár sa v jeho vlastnej kódovej časti môžeme skrátene odkazovať slovom *Me*.

```
  'skrytie formulára (zostane v pamäti):
  Me.Hide
  zatvorenie formulára:
  Unload Me
```

Ak zavriete formulár kliknutím na krížik v pravom hornom rohu, spustí sa udalosť *Terminate*, ktorú je vo väčšine prípadov vhodné obslúžiť tým istým kódom ako pri tlačidle, ktoré normálne uzatvára program (voláte ho Koniec, End, Cancel a podobne).

Väčšinu ovládacích prvkov uvedených ďalej ovládame kliknutím myšou, čím sa vyvolá udalosť CLICK, ktorú môže obsluhovať príslušný kód. Pri formulári nás však často bude viac zaujímať udalosť *Initialize*, pomocou ktorej nastavujeme stav formulára po spustení, ale ešte pred jeho zobrazením. V tejto udalosti môžeme nastavovať napríklad položky do ComboBoxu, ako bude uvedené ďalej.

Label

Slúži na zobrazenie textu, ktorý nie je určený na priame menenie užívateľom. Hodí sa predovšetkým ako popis, napr. pred textové pole alebo tam, kde má byť text užívateľom nemenný. Tam kde to je možné, používajte ho prednostne pred TextBoxom, ktorý je trochu náročnejším prvkom pokiaľ sa týka systémových zdrojov. Ak potrebujete aby mal vzhľad TextBoxu, dosiahnete to jednoduchou úpravou vlastností (nastavíte SpecialEffect na fmSpecialEffectSunken a BackColor na systémovú farbu Window BackGround).

Pokiaľ používate Label iba pri návrhu, teda v kóde s ním nepracujete, sú pre Vás dôležité iba vlastnosti, ktorej sa týkajú vzhľadu. Napr. Caption, čo je text, ktorý sa zobrazí, Font, ktorá určuje typ použitého fontu, BackColor pre farbu pozadia apod. (nemá zmysel všetky vymenovávať, lepšie bude si všetko vyskúšať).

Pokiaľ meníte text Labelu v kóde, napr. na zobrazovanie počtu spracovaných záznamov, sú dôležité okrem toho, že opäť použijete vlastnosť Caption ďalšie dve vlastnosti. AutoSize, ktorá mení automaticky veľkosť prvku v závislosti od dĺžky textu a WordWrap, ktorá určuje, či sa bude alebo nebude text zalamovať na viac riadkov.

TextBox

Názov tohto prvku by sa dal do slovenčiny preložiť ako textové pole. Slúži nato, aby mohol užívateľ interaktívne meniť alebo zadávať hodnoty. Rozdiel od Label je v tom, že TextBox nielen zobrazuje, ale je možné ho i meniť užívateľom. Tam kde potrebujete získať ľubovoľné údaje, či už ich používate v kóde alebo ich napríklad ukladáte do databázy (pre čo má VB priamu podporu). Preto používame najčastejšie TextBox.

Ak vložíme na formulár TextBox na vstup údajov od užívateľa, chceme mať možnosť tieto údaje nejako prečítať. To umožňuje vlastnosť Text. V nej je obsiahnutý všetok text v TextBoxe (umožňuje text užívateľovi nielen čítať, ale aj zapisovať).

```
If TextBox1.Text="" Then
  MsgBox "Zadajte správny text"
Else
  MsgBox TextBox1.Text
End If
```

Ak chceme zabrániť, aby užívateľ po zadaní zlej hodnoty opustil TextBox, môžeme nato využiť udalosť Validate. Táto nastane pred stratou ohniska, čo je aktuálne miesto užívateľského formulára, pre ktoré platia nasledujúce akcie úžívateľa ako je kliknutie myšou, alebo stlačenie klávesy (napr. v TextBoxe, ktorý má ohnisko sa nachádza kurzor). Má jeden parameter, Cancel typu Boolean. Jeho nastavením na True môžete stratu ohniska zakázať. Užívateľ tak stále zostane na danom mieste užívateľského prostredia a vy ho môžete prinútiť hodnotu opraviť.

```
Private Sub Text1_Validate(Cancel As Boolean)
  If Not IsDate(TextBox1.Text) Then
    MsgBox "Chybne zadaný dátum", vbCritical, "Chyba"
    Cancel = True
  End If
End Sub
```

Alebo sa môžete chcieť poistiť, aby užívateľ nezadal namiesto desatinnej bodky desatinnú čiarku pomocou udalosti Change, ktorá reaguje na každú zmenu obsahu TextBoxu:

```
Private Sub Text1_Change
TextBox1.Text = Replace(TextBox1.Text, ",", ".")
End Sub
```

Štandardne je možné v TextBoxe písať text vždy iba na jeden riadok. To môžeme zmeniť nastavením vlastnosti MultiLine na True. Dĺžku užívateľom vkladaného textu obmedzíte pomocou vlastnosti MaxLength, ktorú nastavíte na číslo zodpovedajúce maximálnemu počtu znakov. Pri zadávaní čísiel do TextBoxu je potrebné si uvedomiť, že hodnota, ktorá sa z neho prečíta má charakter textového reťazca, čo je potrebné v texte programu prípadne ošetriť.

CommandButton

CommandButton, čiže tlačidlo, je používaný iba na jednu vec a to je vždy potvrdenie (spustenie) nejakej akcie. Užívateľ napríklad vyplní formulár so vstupnými údajmi a musí stlačiť tlačidlo, inak sa nedozvieme, že skončil a možno napr. spustiť výpočet. Preto je pri ňom najdôležitejšou udalosť Click. Nastáva, ako je zrejmé už z jej názvu, akonáhle užívateľ na tlačidlo klikne myšou. Napr.:

```
'Na stlačenie tlačidla Command1 sa ukáže formulár Form2
Private Sub Command1_Click()
  Form2.Show
End Sub
```

Dôležitými vlastnosťami pri tlačidlách sú Cancel a Default. Ak nastavíme pre niektoré tlačidlo Cancel = True, a užívateľ stlačí klávesu Esc (samozrejme iba pokiaľ je aktívny formulár, na ktorom sa tlačidlo nachádza), je vykonaný kód v udalosti Click toho tlačidla v ktorom je Cancel = True. Default je podobné, ale reaguje na klávesu Enter. Ak stlačí užívateľ Enter, spustí sa kód v udalosti Click príslušného tlačidla. Na formulári je možné nastaviť Cancel = True iba pre jedno tlačidlo (rovnako i Default).

CheckBox, OptionButton a Frame

CheckBox môžeme do slovenčiny preložiť ako zatrhávacie políčko. Ponúka užívateľovi na výber dve možnosti, zapnuté alebo vypnuté. Ak je zobrazený krížik, X, je zapnuté, bez X je vypnuté. Existuje ešte tretia možnosť, ktorá nebýva tak často používaná. Môžeme ho popísať ako zošednuté X a je možné ho bežne vidieť pri inštalácii, pri výbere užívateľskej konfigurácie a pri vybraní iba niekoľkých položiek z podmenu. Ak sa vrátite do hlavného menu, CheckBox je zatrhnutý, X, ale so sivým pozadím. Je možné ho využiť na rôzne ciele, napr. neznáma alebo nejednoznačná hodnota.

Vlastnosť, ktorá umožňuje zistiť, v akom stave sa CheckBox nachádza, je Value. Môže nadobúdať, ako vyplýva z vyššie uvedeného, troch hodnôt: 0, čiže nezatrhnuté (Unchecked), 1, čiže zatrhnuté (Checked) a 2, čiže Grayed. Ďalšou vlastnosťou je Style, pomocou ktorej nastavíte vzhľad na štandardný alebo ako tlačidlo. Inak je tento prvok

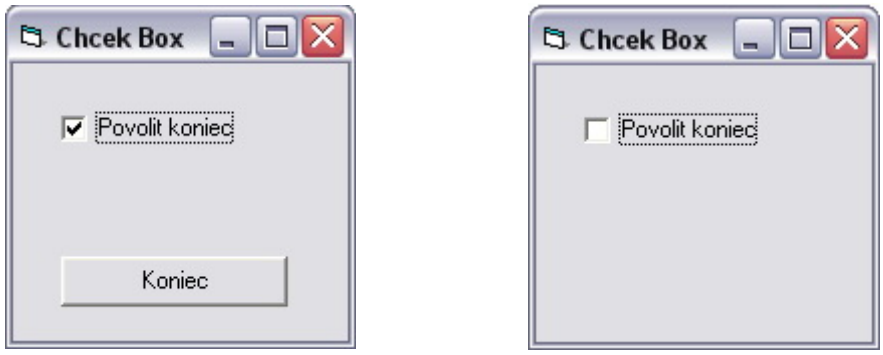
vybavený bežnými vlastnosťami, ako sú Visible, Enabled, Font atď. a udalosťami na prácu s myšou a klávesnicou.

Na opísanie OptionButtonu si budeme musieť vysvetliť pojem kontajner prvkov. Prvkom, ktorý má túto vlastnosť je napríklad formulár. Je možné do neho totiž vkladať iné prvky. Podobne však môžu fungovať i ďalšie prvky ako napríklad Frame, PictureBox atď. Nato je však najvhodnejší Frame. Ide o veľmi jednoduchý prvok pre ktorý môžete nastaviť titulok (Caption) a typ okraja (BorderSyle) ale i ďalšie bežné vlastnosti ako farby, font a pod. Čo sa udalostí týka, odlišuje sa Frame od CheckBoxu tým, že mu chýbajú udalosti na prácu s klávesnicou. Ďalej sa tento prvok hodí na optické delenie formuláru na menšie a prehľadnejšie časti. Viditeľnosť prvkov, ktoré do neho vložíte je vymedzená jeho veľkosťou. V prípade, že s ním pohnete, budete automaticky hýbať i so všetkými prvkami na ňom. Ďalej asi narazíte na problém, že nemôžete rámcom označiť niekoľko prvkov vo Frame. To vyriešite tým, že pred výberom rámčekom stlačíte klávesu Shift a budete ju držať. Pri práci v kóde sa však nič nemení. Neodkazujete sa na prvky cez patričný Frame, ale priamo cez meno, aké majú.

Ukážeme si jednoduchý príklad použitia prvku CheckBox. Vytvoríme formulár s jedným tlačidlom výberu a jeho Caption nastavíme na "Povoliť koniec" a jedno tlačidlo s popisom "Koniec". Visible tlačidla nastavíme na False. V handlery tlačidla výberu opäť použijeme funkciu Not na zjednodušenie zdrojového kódu. Procedúra Command1_Click() bude obsahovať iba príkaz na ukončenie "End".

```
Private Sub Check1_Click()
Command1.Visible = Not Command1.Visible
End Sub
```

Spustime teda program a pozrime sa ako bude fungovať. Po "zaškrtnutí" prvku CheckBox sa nám zobrazí tlačidlo.



a) b)

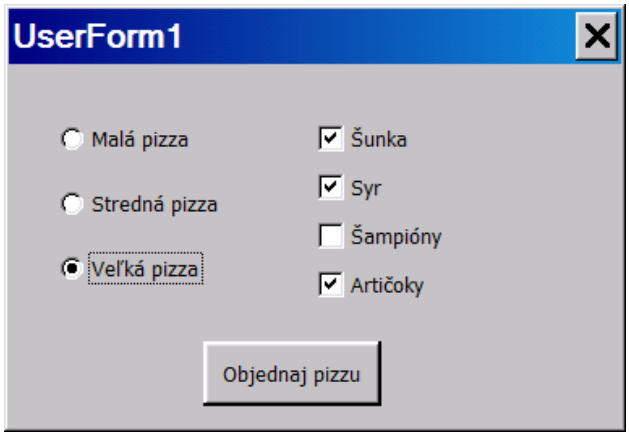
Obr. 1.8 a) Prvé spustenie (po "odškrtnutí" tlačidla výberu) b) Po "zaškrtnutí" tlačidla výberu

OptionButton pracuje podobne ako CheckBox. OptionButton má však iba dva stavy, zistiteľné tiež z vlastnosti Value a to True alebo False, t. j. či je vybraný alebo nie. V jednom kontajneri prvkov môže byť iba jeden OptionButton, ktorý má Value rovnú True. Ak chceme použiť viac skupín OptionButtonov musíme nato využiť ďalší prvok, ktorý je kontajner prvkov. Nato využijeme práve zmieňovaný Frame. Takto zaistíme, že je možné aby pre viac OptionButtonov bola nastavená vlastnosť Value na True. Avšak v rámci jedného kontajnera existuje takto nastavený vždy iba jeden. Akonáhle užívateľ na niektorý klikne, t. j. nastaví

Value rovnú True, pri ostatných sa automaticky nastaví False. Rovnako to prebieha aj pri nastavovaní z kódu. Inou možnosťou než použiť kontajner a mať v rámci jednoho kontajnera viac skupín súvisiacich CheckBoxov je nastaviť pre prvky typu CheckBox vlastnosť GroupName – prvky s rovnakým GroupName budú patriť spolu a vždy iba jeden z nich môže mať hodnotu True.

Z toho vyplýva, že OptionButton sa používa na výber jednej položky z viac ponúkaných. Rovnako ako CheckBox má aj OptionButton vlastnosť Style, ktorou nastavíme štandardný vzhľad alebo vzhľad tlačidla. Čo sa týka vlastností a udalostí, sú rovnaké ako pri CheckBoxe.

Použitie: keby ste si mali internetový obchod na objednávanie pizze, museli by ste urobiť užívateľské rozhranie podľa nasledujúceho obrázka, nakoľko pizza je buď veľká alebo malá ale môže obsahovať kombináciu ľubovoľného počtu príchutí:



Obr. 1.9 Rozdiel v použití ovládacích prvkov Checkbox a OptionButton

ComboBox

Hlavnou funkciou ComboBoxu je dať užívateľovi na výber jednu alebo viac položiek z rozbaľovacieho zoznamu. ComboBox obsahuje viac položiek, zobrazuje však iba jednu – tú, ktorá je aktívna. Kliknutím na jeho nadol smerujúcu šípku sa rozbalí zoznam, kde sa ukážu aj ostatné. Položky sa pridávajú a mažú pomocou udalostí AddItem a RemoveItem. Celý obsah je možné zmazať pomocou metódy Clear. Text vybranej položky zistíte z vlastnosti Text.

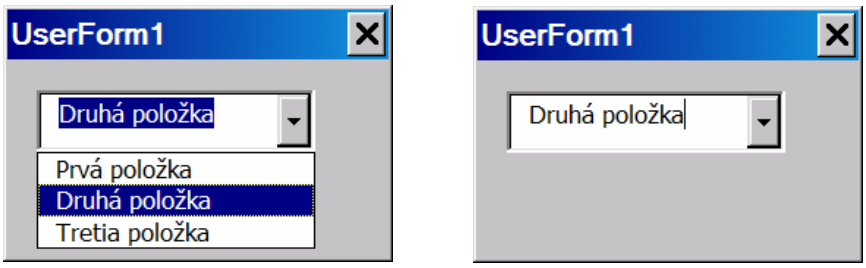
Naplnenie obsahu ComboBoxu môžeme napríklad v udalosti *Inicialize* formuláru, na ktorom sa Combox nachádza napríklad takto (obr. 1.10a):

```
ComboBox1.AddItem "Prvá položka"
ComboBox1.AddItem "Druhá položka"
ComboBox1.AddItem "Tretia položka"
```

V reťazcovej premennej *ZistenyText* bude po prebehnutí nasledujúceho príkazu reťazec "Prvá položka"(obr. 1.10b):

```
ZistenyText=Combo1.Text
```

a) b)



Obr. 1.10 Použitie ComboBoxu

ComboBox má tiež vlastnosť Style. Určuje chovanie ComboBoxu pri výbere položiek. Má tri hodnoty. Nastavením na 0 (default), môžete vyberať položky zo zoznamu alebo napísať akýkoľvek iný text. Nastavením na 1 (v tomto prípade musíte ComboBox trochu zväčšiť – na výšku) sa zmení v TextBox a ListBox spojené dohromady. V skutočnosti je to rovnaké ako pri nastavení na 0, iba s tým rozdielom, že je ComboBox stále rozbalený, t. j. ako keby užívateľ klikol na tlačidlo so šípkou. Nastavením na hodnotu 2 vyzerá ComboBox zase normálne, užívateľ si však môže vybrať iba zo zoznamu, nie je možné napísať iný text.

Ovládací prvok – Image

Image je grafický prvok, ktorý sa používa na načítanie obrázkov buď počas behu aplikácie, alebo pri jej návrhu. Image slúži hlavne na dekoráciu aplikácie a i keď je to grafický prvok, nie je do neho možné kresliť, na to slúži prvok PictureBox. Vytvorte si na formulári jeden prvok Image. Ihneď je viditeľné, že tento prvok nemá vlastné pozadie (farbu na pozadí, je teda transparentný – priesvitný). Pri vlastnosti Picture nájdete "...", kliknite na ne a vyberte si akýkoľvek obrázok z disku. Ako vidíte, po načítaní obrázku sa prvok Image nastaví na potrebnú veľkosť, aby sa do neho vošiel celý obrázok. Ak teraz začnete prvku Image meniť veľkosť, tak sa obrázok buď zreže, alebo zostane vedľa obrázku voľné miesto (v rámci prvku Image). Nájdite preto vo vlastnostiach položku Stretch (angl. roztiahnuť) a nastavte ju na hodnotu True. Od teraz sa bude veľkosť obrázku meniť s veľkosťou prvku Image. Jednoduchý príklad použitia prvku ako dekorácie aplikácie je Splash Screen. Splash Screen je okno, ktoré sa zobrazí pri spúšťaní niektorých aplikácií a pokým je toto okno viditeľné sa načítajú rôzne potrebné súčasti programu.

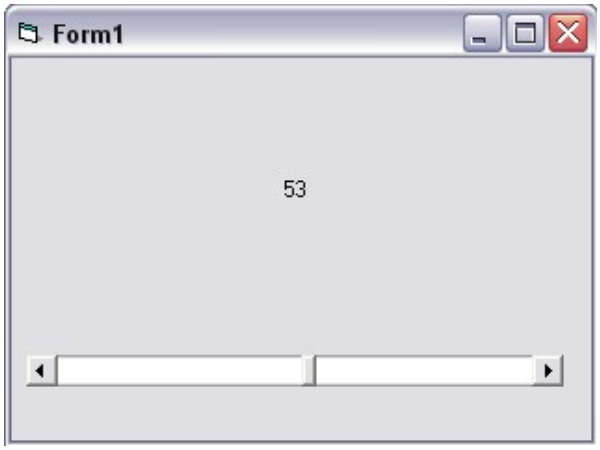
Ovládací prvok ScrollBar

Scrollbar slúži na posúvanie obrazovky (známy z Internet Exploreru, Microsoft Wordu, atď). Použitie prvkov ScrollBar si ukážeme hneď na jednoduchom príklade. Chcete aby užívateľ programu zadával hodnotu, ktorá bude celočíselná a z intervalu 0 až 100. Môžete navrhnúť grafické zadávanie pomocou posuvníku. Nakreslime na formulár jeden ľubovoľný ScrollBar a jeden prvok Label. V paneli vlastností vyhľadajte položku Max, tá udáva maximálnu hodnotu, ktorú môže prvok dosiahnuť. Rovnako Min určuje minimálnu hodnotu prvku ScrollBar(Min má najväčšie využite napr. pri textových editoroch, aby ste text nepísali na okraj papiera). Implicitne nastavená hodnota položky Max je 32767, čo je zároveň vrchná hranica premennej Integer. Nastavme max napr. na hodnotu 100. Ďalšia vlastnosť bez ktorej sa nezaobídete je vlastnosť Value (anglicky hodnota). Tá nám udáva aktuálnu hodnotu prvku

ScrollBar. Ak Value budete pokúšať meniť počas behu programu z kódu na hodnotu vyššiu ako je Max, tak to bude viesť k zrúteniu aplikácie.

Môžeme napísať obslužnú procedúru pre prvok Hscrollbar na Form1 s prvkom Label1 kde vlastnosti Caption prvku Label priradíme hodnotu Value prvku ScrollBar:

```
Private Sub ScrollBar1_Change()
Label1.Caption = ScrollBar1.Value
End Sub
```



Obr. 1.11 Do prvku Label sa vypíše hodnota prvku ScrollBar

Ovládací prvok – ListBox

ListBox (angl. "zoznamové pole") sa používa na vytváranie zoznamov, ako už vyplýva z jeho názvu. Vytvorme na formulári prvok ListBox. Ďalej je dôležité vedieť pridávať položky do zoznamu počas behu aplikácie. Ukážeme si to na konkrétnom príklade. Do udalosti UserForm_Initialize () napíšeme tento kód:

```
ListBox1.AddItem "Angličtina"
ListBox1.AddItem "Nemčina"
ListBox1.AddItem "Slovenčina"
```

Pre každú položku zoznamu je treba napísať takýto riadok kódu. Ak by teda mal zoznam obsahovať 100 položiek, bolo by nutné príkaz List1.AddItem (add angl. pridať, item angl. položka) napísať 100-krát.

Obr. 1.12 Prvok ListBox po pridaní položiek do zoznamu

Prvky sa zo zoznamu dajú aj vymazať. Pre názornú ukážku vymazávania prvkov zoznamu si najprv vytvorme zoznam, ktorý naplníme takto pomocou cyklu For:

```
Private Sub UserForm_Initialize ()
For i = 1 To 20
ListBox1.AddItem i
Next i
End Sub
```

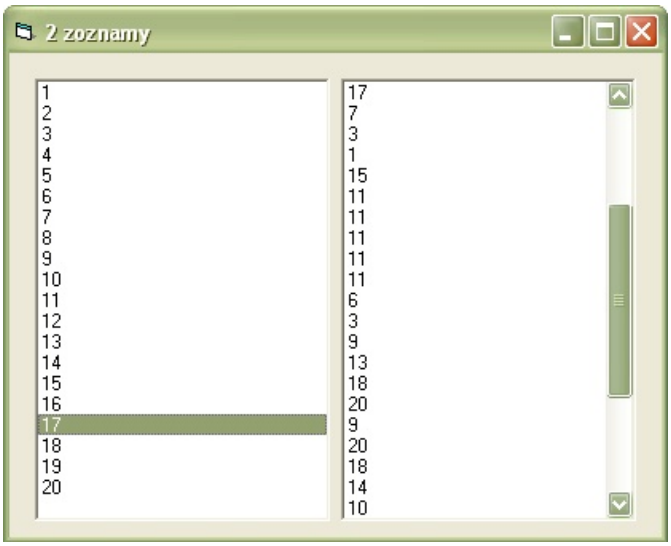
Prvky možno vymazať príkazom *ListBox1.RemoveItem index* (remove angl. odobrať), kde index je číselné vyjadrenie pozície prvku. Prvý prvok zoznamu má index 0, čiže ak chceme vymazať prvý prvok zoznamu musíme napísať príkaz ListBox1.RemoveItem 0. Ak chceme nasledovne vymazať ďalší prvok zoznamu, v našom prípade číslo 2, bolo by nesprávne písať ListBox1.RemoveItem 1, pretože po vymazaní čísla 1 sa prvým prvkom zoznamu automaticky stane číslo 2 a teda zároveň nadobudne jeho index hodnotu 0. Po dvojnásobnom použití príkazu List1.RemoveItem 0 by teda bolo prvým prvkom zoznamu číslo 3. Na vymazanie celého zoznamu má VBA osobitný príkaz a to *ListBox1.Clear* (clear angl. vyčistiť). Tento príkaz nájde svoje využitie hlavne pri väčšom množstve prvkov.

ListBox1.ListCount (count angl. počet) je ďalšou vlastnosťou prvku ListBox. Vo vlastnosti *ListCount* je zaznamenaný počet položiek zoznamu. Príkaz *Label1.Caption = ListBox1.ListCount*, by v našom prípade po spustení programu vypísal číslo 20. Ak je potrebné vymazať posledný prvok zoznamu, dá sa k tomu použiť príkaz *ListCount*. Nie je správne napísať to takto *ListBox1.RemoveItem ListBox1.ListCount* – je tu chyba, ktorá by pri použití daného príkazu viedla k zrúteniu aplikácie. Prvý prvok zoznamu má hodnotu indexu 0 a z toho vyplýva, že posledný prvok bude mať index o 1 menší ako je maximálny počet prvkov. V našom prípade teda má číslo 20 index 19. Preto príkaz na vymazanie posledného prvku zoznamu vyzerá takto *ListBox1.RemoveItem ListBox1.ListCount – 1*.

Vytvorte si teraz na formulár dva prvky ListBox a ukážeme si ako preniesť prvok jedného zoznamu do zoznamu druhého. Prvý ListBox môžete naplniť pomocou cyklu For rovnakým spôsob ako sme si ukazovali pred chvíľou. Druhý ListBox ponecháme prázdny. Udalosť Click prvého zoznamu bude vyzerať takto:

Private Sub ListBox1_Click()
ListBox2.AddItem ListBox1.Text
End Sub

Príkaz *List2.AddItem List1.Text* znamená: "Do zoznamu číslo 2 pridaj prvok s textom rovnakým, na aký bolo kliknuté v zozname číslo 1."



Obr. 1.13 Vkladanie položiek zoznamu jedného do zoznamu druhého

1.4 Stručný prehľad syntaxe jazyka Visual Basic for Application

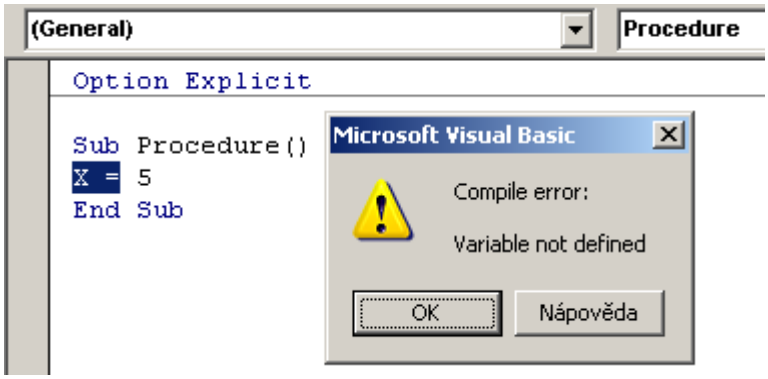
1.4.1 *Premenné*

Programy väčšinou spracovávajú nejaké údaje. Tieto musia byť niekde uložené. Za týmto účelom programátor pracuje s *premennými*. Premenná je úsek počítačovej pamäte, do ktorej si môže program ukladať vstupné údaje, výstupné údaje a rôzne medzivýsledky. Premenné teda slúžia na uchovanie hodnôt v pamäti. Premenná je sprístupnená svojim menom. Premenné sú zvyčajne určené pre nejaký *dátový typ*. Ten rozhoduje, aké údaje (a aký rozsah) možno do premennej uložiť. Aby sme mohli používať novú premennú, je vhodné ju *nadeklarovať*. Proces deklarácie zahŕňa stanovanie mena, dátového typu a iných informácií potrebných na prácu s premennou. Do premenných (alebo konštánt) si môžeme potom ukladať dáta tak, aby nám po zavolaní mena konkrétnej premennej alebo konštanty bola k dispozícii jej hodnota. Čo sa týka nuáns – premenná nemusí byť vo VBA nevyhbutne deklarovaná vopred, takže pokiaľ hocikde v programe napíšete *X=26* , VBA si vytvorí miesto v pamäti pre premennú X a ihneď do nej vloží hodnotu 26. I keď sa to zdá na prvý pohľad ako jednoduchšie riešenie, nie je to celkom tak. Okrem iného – pokiaľ si takto vytvoríme premennú X, VBA nemá informáciu, aký typ hodnoty bude premenná reprezentovať (číselnú hodnotu, text, dátum, objekt atď.). Na takto vzniknutú premennú si musí Visual Basic rezervovať v pamäti veľa miesta, čo nemá dobrý vplyv na efektívnu činnosť aplikácie.

V prípade, že premenná bude obsahovať počas celého behu iba jednu hodnotu, je lepšie použiť konštantu, lebo konštanta zaberie v pamäti iba toľko miesta, koľko ho potrebuje

hodnota priradená konštante, na rozdiel od premennej, pre ktorú musíme vo všeobecnosti počítať, že hodnota, ktorú nadobudne počas behu programu môže byť aj väčšia ako aktuálna. Konštanta sa líši od premennej tým, že jej hodnota sa zvyčajne nemení. Používajú sa na sprehľadnenie kódu a jeho ľahkú upraviteľnosť. Predstavte si, že používate vo svojej aplikácii dvadsaťkrát hodnotu 47 (napríklad počet nejakého materiálu pre ktorý robíte výkaz). Keď použijete priamo jej hodnotu, budete vedieť aj o mesiac, čo to číslo znamená (a budú to vedieť aj ľudia, ktorí budú po vás program čítať)? Ak použijete konštantu, napr. DruhMaterialu, hneď vám bude jasné, čo to číslo vlastne znamená. Predstavte si ešte jednu situáciu: rozhodnete sa zmeniť z nejakého dôvodu číslo 47 na 64. Budete musieť prejsť celý váš program, dvadsaťkrát nájsť hodnotu 47 a zmeniť ju. Keď použijete konštantu, stačí zmeniť len jej hodnotu, ktorú ste priradili kdesi na začiatku programu.

Ak bude premenná obsahovať premenlivú hodnotu, ale jedného typu, napr. číselnú hodnotu, text, dátum je vhodné ju pre tento typ deklarovať, nakoľko nie každý dátový typ zaberá v pamäti rovnaké množstvo miesta. Okrem toho deklarácia premenných uľahčuje ladenie a kontrolu napísaného kódu – v prípade, že do premennej sa kód snaží načítať nejaký nezmyselný údaj (napr. text do premennej deklarovanej ako číslo) program vyhlási chybu, ktorá by sa mohla inak ťažšie hľadať.



Obr. 1.14 Použitie voľby Option Explicit

Ak sa vyskytuje hneď na začiatku Okna kódu zápis *Option Explicit*, budete nútení každú premennú deklarovať explicitne. Inak sa dočkáte chybového hlásenia uvedeného vedľa.

Okrem zmienených výhod explicitne deklarovaných premenných ľahšie zistíte chyby z dôvodov preklepu v názve premennej – premenná s preklepom bude pre program neznáma.

Nastane situácia, keď nebudete vedieť, aký typ pre premennú zaviesť. V tom prípade jednoducho deklarujte premennú ako *Dim Neznama* bez uvedenia typu a po prvom priradení hodnoty použite v kóde funkciu *TypeName*. Uvedený príklad zobrazí okno s typom, ktorý priradil premennej program.

```
Sub NeznamyTyp()
  Dim Neznama
  Neznama = "Řetězec"
  MsgBox TypeName(Neznama)
End Sub
```

Deklarácia premenných sa teda skladá z týchto častí:

- **Typy viditeľnosti premenných.** Okrem určenia akého typu budú hodnoty v premennej (číslo, reťazec, dátum atď.), môžeme určiť aj kde všade bude premenná viditeľná (na ktorom mieste programu bude hodnota v nej uložená použiteľná). Ak si nadefinujeme, že premenná bude viditeľná iba v aktuálnom formulári (module) a budeme sa na ňu

odvolávať v inom formulári, bude tam premenná nulová (nebudete niesť žiadnu hodnotu), alebo program vyhlási, že nie je definovaná. Ukážme si typy viditeľnosti, ktoré môžete používať. Najviac je využívaný Dim.

- **Dim** – deklarácia lokálnych premenných vo funkcii, procedúre, alebo v module, formulári, či ClassModule.
- **Private** je typ, ktorý určuje, že premenná bude viditeľná iba v module, okne, ClassModule, v ktorom bola deklarovaná. Deklaruje sa mimo funkcií, procedúr, udalostí (v tzv. deklaračnej časti).
- **Public** je viditeľný v celom projekte. Rovnako musí byť deklarovaný v deklaračnej časti.

- **Názov premennej.** Väčšinou sa nazýva podľa toho, akú hodnotu nesie. Ak napr. nesie údaj o vašej výške nazvete si ho "Vyska". Nesmie však obsahovať národné znaky (ľ, š, č atď.), prvé písmeno nesmie byť číslica, nesmie sa zhodovať s pojmami, ktoré sú rezervované pre VB (napr. Open, Integer, If, názvy funkcií atď.).
- **Typ premennej**

Na začiatku sme si vysvetlili, že je pre rýchlosť aplikácie dôležité určovať aký typ premennej použijeme (číslo, reťazec, dátum). Ukážme si teraz, načo ktoré typy premennej fungujú:

- **Integer** je jedným z najviac používaných typov. Je 16-bitový a vyjadruje číselné hodnoty od –32 768 do 32 767. Často však potrebujeme pracovať ešte s väčšími číslami, na to sa hodí typ Double, ktorý až 64-bitový.
- **String** vyjadruje reťazec, pričom reťazec môže obsahovať ľubovoľné znaky. Aby si bol Visual Basic na istom pri zadávaní hodnôt do premenných typu String, celú hodnotu vložíme do úvodzoviek (napr. Meno = "Homér Simpson")
- **Boolean** pracuje ako 16-bitový dátový typ *Integer*, v skutočnosti však používa iba dve hodnoty –1 a 0. –1 zobrazí hodnotu TRUE (pravda) a 0 FALSE(nepravda). Mohli by sme ho prirovnať k vypínaču svetla. Ak ho stlačíte raz, svetlo sa zapne, ak ho stlačíte znova, svetlo sa vypne. Má teda iba dve polohy, jedna pre zapnuté svetlo, druhá pre vypnuté. Rovnako funguje aj **Boolean**. Ak sa napríklad užívateľa pýtate na pohlavie a odpovie Vám "muž", môžete to zaznačiť takto: Muz=True alebo Zena=False. Alebo iný príklad:

  Dim Stlacene As Boolean
  Private Sub CommandButton1_Click()
  If Stlacene = False Then
  Stlacene = True
  Command1.Caption = "Tlacitko je stlacene"
  Else
  Stlacene = False
  Command1.Caption = "Tlacitko nie je stlacene"
  End If
  End Sub

- **Date** sa bude týkať vyjadrenia dátumu a času. Dátum alebo čas môžete zadávať buď v úvodzovkách – vtedy uvedieme dátum alebo čas presne (napr.: "21.3. 2001";"16:00:00"), alebo obyčajným číselným výrazom. Táto druhá možnosť je trochu zložitejšia. Číslo 0 vyjadruje polonoc ("00:00:00") , 0,5 vyjadruje poludnie ("12:00:00"). Desatinné čísla vyjadrujú čas, celé čísla vyjadrujú dátum. Záporné čísla

vyjadrujú dátum pred 30. decembrom 1899, kladné vyjadrujú všetko po tomto dátume. Uveďme si jednoduchý príklad:

Dim Datum As Date
Datum = 1.5 31.12.1899 12:00:00
MsgBox Datum

- **Byte** je typ premenných, ktoré nesmú byť menšie ako nula a väčšie ako 255.
  Na záver ešte príklady na deklaráciu premenných:

Dim CeleCislo As Integer
Dim RacionalneCislo As Double
Public Datum As Date

Pokiaľ máme premennú, ktorá bude obsahovať tzv. textový reťazec, t. j. sled písmenných a iných znakov a nie číselnú hodnotu určenú pre výpočty, môžeme ju ako bolo uvedené deklarovať ako dátový typ String. Deklarácia vyzerá takto:

Private ZnakovyRetazec as String
*Private ZnakovyRetazec as String*20*

V príklade sú dve deklarácie reťazcov String1 a String2. Obidve deklarácie sú správne, ale je medzi nimi rozdiel. Prvá deklarácia si po vytvorení premennej String1 nechá v pamäti miesto pre 20 znakov reťazca. V druhom prípade premenná String2 nemá určené akú dĺžku bude mať a reťazec vložený do premennej môže mať maximálny počet znakov, čo je 1 bilión (32 núl).

Predpokladajme, že dátový typ String deklarujeme ako reťazec s pevnou dĺžkou. Ak vložíme reťazec kratší ako je definovaný počet znakov, zvyšné znaky nahradia medzery. Ak naopak vložíme reťazec, ktorý má viac znakov ako je definovaných Visual Basic ho skráti.

Dim Retazec as String * 10 *'deklarácia premennej s pevnou dĺžkou 10 znakov*

Do premennej deklarovanej ako String môžeme priraďovať ľubovoľné texty a pracovať s nimi pomocou reťazových funkcií, ktoré na inom mieste uvádzame. Môžu to byť mená osôb, identifikátory materiálov a mnoho iných vecí.

StringRetazec *= "Toto je text"*

Konštanty sú na deklaráciu oveľa jednoduchšie. Môžu síce obsahovať informáciu o viditeľnosti, ale žiadny typ konštanty sa neurčuje (teda, či ide o číslo, reťazec dátum atď.). Príklad:

Const Dlzka = 50

Údaje o jednotlivých typoch údajov sú uvedené v nasledovnej tab. 1.1.

Typy a rozsah premenných vo VBA Tabuľka 1.1

Dátový typ	**Veľkosť pamäte**	**Rozsah**
Byte	1 bajt	0 až 255
Boolean	2 bajty	True nebo False
Integer	2 bajty	-32 768 až 32 767
Long (dlhé celé číslo)	4 bajty	-2 147 483 648 až 2 147 483 647
Single (pohyblivá čiarka s jednoduchou presnosťou)	4 bajty	-3,402823E38 až -1,401298E-45 pre záporné hodnoty; 1,401298E-45 až 3,402823E38 pre kladné hodnoty
Double (pohyblivá čiarka s dvojitou presnosťou)	8 bajtov	-1,79769313486232E308 až -4,94065645841247E-324 pre záporné hodnoty 4,94065645841247E-324 až 1,79769313486232E308 pre kladné hodnoty
Currency (stupňované celé číslo)	8 bajtov	-922 337 203 685 477,5808 až 922 337 203 685 477,5807
Decimal	14 bajtov	+/- 79 228 162 514 264 337 593 543 950 335 bez desatinnej čiarky; +/-7,9228162514264337593543950335 s 28 miestami napravo od desatinnej čiarky; najmenšie číslo rôzne od nuly je +/-0,0000000000000000000000000001.
Date	8 bajtov	Od 1. januára 100 do 31. decembra 9999
Object	4 bajty	odkaz na Object
String (s premenlivou dĺžkou)	10 bajtov + dĺžka reťazca	Od 0 do približne 2 miliárd
String (s pevnou dĺžkou)	dĺžka reťazca	Od 1 do približne 65 400
Variant (s číslami)	16 bajtov	Ľubovoľná číselná hodnota až do rozsahu typu Double
Variant (so znakmi)	22 bajtov+ dĺžka reťazca	Rovnaký rozsah ako pre typ String s premenlivou dĺžkou
Užívateľský (používa sa Type)	Počet potrebný pre prvok	Rozsah každého prvku je rovnaký ako rozsah jeho dátového typu.

Ako možnosť navyše k deklarovaniu premenných explicitnými názvami uvedenými v tejto tabuľke, je možné deklarovať premenné aj pomocou zvláštnych znakov.

Napríklad namiesto použitia deklarácie:

Dim Retazec As String

Je možné napísať:

Dim Retazec$

Nasleduje kompletný zoznam príslušných dátových typov a korešpondujúcich znakov:

String ($)
Integer (%)
Long (&)
Single (!)
Double (#)
Currency (@)

Túto konvenciu je potrebné využívať s opatrnosťou, vzhľadom na čitateľnosť kódu. Určite ale využijete uvedenú informáciu v prípade, že sa vám dostane do rúk program, ktorý ju využíva.

Zvláštnym a dôležitým typom premennej je pole. Pole umožňuje odkazovať sa na množinu premenných pomocou jednotného mena a používať číslo (index) na ich rozlíšenie. Napríklad budete zadávať údaje o dĺžke potrubí vodovodu a vo vodovodnom systéme môže byť takýchto potrubí aj 1000. Nepoužijete teda premenné *DlzkaPotrubia1, DlzkaPotrubia2, DlzkaPotrubia3* ale pole *DlzkaPotrubia(i)*, kde *i* je premenlivý index v určitom zadefinovanom rozsahu. Toto pomáha tvoriť kratší a jednoduchší kód, pretože môžeme použiť cykly, ktoré budú účelne pracovať s položkami poľa pomocou ich indexov. Polia majú hornú a dolnú hranicu (maximálny a minimálny index) a jednotlivé prvky sa nachádzajú vo vnútri týchto hraníc. V tejto súvislosti si všimnite v časti skrípt o funkciách význam funkcií UBound a LBound.

Všetky prvky poľa sú rovnakého typu. Iba ak je nastavený typ Variant, môžu jednotlivé prvky obsahovať rôzne druhy dát (objekty, textové reťazce, čísla atď.) Môžeme deklarovať pole ktoréhokoľvek základného dátového prvku, vrátane užívateľských typov a objektových premenných.

Vo Visual Basicu existujú dva typy polí: polia s pevnou veľkosťou, ktoré majú stále rovnakú veľkosť (rozsah možných indexov) definovanú pri deklarácii premennej typu pole a dynamické polia, ktorých veľkosť sa môže meniť počas behu programu.

Dim StatPole (1 to 50) as Double – deklarácia poľa s pevnou veľkosťou typ prvkov bude double (reálne čísla s dvojitou presnosťou)

Dim DynPole ( ) – deklarácia dynamického poľa, nie je uvedený jeho rozsah
ReDim DynPole (1 to 50) – pomocou príkazu ReDim definujeme počet prvkov za behu programu

Často použijeme voľbu ReDim Preserve (pri zmene veľkosti hornej medze poľa so zachovaním pôvodných predchádzajúcich hodnôt poľa).

```
Sub DynamickePole()
  Dim Pole()
  M = 3
  'Redimenzovanie je nutné pred ďalším spracovaním
  'Redimenzovanie s výmazom obsahu poľa
  ReDim Pole(1 To M)
```

'Redimenzovanie so zachovaním predchádzajúceho obsahu
ReDim Preserve Pole(1 To M)
End Sub

Aj keď v prípade ReDim Preserve hovoríme o zachovaní pôvodného obsahu, pokiaľ pole zmenšíme, dôjde k odstráneniu časti obsahu. Príkaz ReDim Preserve neumožňuje zmenu počtu rozmerov poľa ani zmenu dolnej medze.

Ak chceme vynmazať obsah poľa nie je potrebné používať cyklus na nulovanie jeho prvkov ale lepší je príkaz:

Erase MojePole()

Niekedy potrebujeme uložiť do poľa sekciu súvisiacich informácií. Ak chceme napríklad uložiť informácie o nejakých bodoch v teréne, musíme uložiť ich súradnice X,Y a Z. V tom prípade je výhodné použiť *viacrozmerné pole*.

Dim Suradnice ( ) as Double

ReDim Suradnice (1 to 500, 1 to 3) – deklarácia viacrozmerného poľa pre zadanie X, Y, Z súradníc pre 500 bodov

Veľkosť dynamického poľa je možné počas behu programu upravovať a tiež je možné mu pridávať ďalšie rozmery. Akonáhle začneme pridávať rozmery poľa, celkové miesto na jeho uloženie dramaticky vzrastá. Je teda nutné používať viacrozmerné polia s rozmyslom. Zvlášť opatrní treba byť v prípadoch polí typu Variant, pretože sú náročnejšie na priestor ako ostatné.

Časť o premenných ešte ukončíme poznámkou o zmienenom type premennej variant. Variant je typ premennej, ktorý môže obsahovať ľubovoľný typ údajov s výnimkou reťazcov fixnej dĺžky a užívateľom definovaných typov. Variant môže obsahovať aj špeciálne hodnoty (*Empty, Error, Nothing* a *NULL*).

Prvky variant sa používajú na manipuláciu s poľami v prvkoch ActiveX AutoCADu. Ak potrebujete, aby pole bolo akceptované v modeli ActiveX metódami a vlastnosťami AutoCADu (bude vysvetlené neskôr), musí byť typu variant. Aj údaje vrátené rozhraním ActiveX AutoCADu, musia být spracované ako objekty typu variant. V AutoCADe, VBA automaticky prevádza pole na typ variant. Aj dáta vrátené AutoCADom sú typu variant a musia byť ako také spracované.

Rozhranie AutoCAD ActiveX automatizácie ponúka metódu – *CreateTypedArray* – na jednoduchý prevod všetkých polí údajov typu integer, float, double na prvok variant zodpovedajúceho typu (integer, float, double a pod.).

Metóda *CreateTypedArray* má dva argumenty. Prvým argumentom je typ konvertovaných údajov (typ údajov v poli, ktoré sa bude konvertovať), druhým parametrom je pole údajov. Metoda vracia pole hodnôt ako prvok variant.

Pole vrátené rozhraním AutoCAD ActiveX sú opäť typu variant. Ak je známy dátový typ vráteného poľa, je možné jednoduchým spôsobom pristupovať k premennej variant ako k poľu. V prípade, že nie je známy typ údajov obsiahnutých vo variante, musí sa použiť VBA funkcia VarType alebo Typename. Tieto funkcie vracajú dátový typ variantu. Na postupný prechod cez pole je nutné použiť VBA príkaz ForEach (pozri pri cykloch).

V nasledujúcom príklade (odčítanie dvoch bodov a určenie ich vzdialenosti) je dátový typ známy (súradnice sú typu double). 3D súradnice sú polia o troch prvkoch typu double a 2D súradnice sú polia o dvoch prvkoch, čo znamená, že premenné na získanie poľa

definujúceho bod deklarujeme ako variant, ale keďže poznáme jeho typ, ďalej s ním pracujeme ako s poľom s prvkami typu double.

```
Sub VypocetVzdialenosti()
  Dim bod1 As Variant          'Nutnosť deklarácie ako variant
  Dim bod2 As Variant
  ' Nasledujúci príkaz zistí body od uživateľa, ktorý bod napríklad odklikne myšou
  bod1 = ThisDrawing.Utility.GetPoint(, vbCrLf & "Prvý bod: ")
  bod2 = ThisDrawing.Utility.GetPoint(bod1, vbCrLf & "Druhý bod: ")
  ' Vypočítá vzdialenosť mezi bodom 1 a bodom 2
  Dim x As Double, y As Double, z As Double
  Dim dist As Double
  x = bod1(0) – bod2(0)
  y = bod1(1) – bod2(1)
  z = bod1(2) – bod2(2)
  dist = Sqr((Sqr((x ^ 2) + (y ^ 2)) ^ 2) + (z ^ 2))
  'Zobrazí výslednú vzdialenosť
  MsgBox "Výsledná vzdialenosť medzi bodmi je: " _
    & dist, , " vypočítaná vzdialenosť"
End Sub
```

1.4.2 *Príkazy a používanie operátorov*

Príkaz hovorí Visual Basicu, že má niečo urobiť. Každý príkaz je vo Visual Basicu umiestnený na samostatný riadok a môže sa nachádzať iba vo vnútri procedúr a funkcií:

```
prikaz1
prikaz2
```

V prípade potreby je možné umiestniť príkazy na jeden riadok a oddeliť ich dvojbodkou:

```
prikaz1 : prikaz2
```

Kvôli väčšej prehľadnosti kódu sa to ale neodporúča.

Asi najčastejšie používaný príkaz vo všetkých programovacích jazykoch je priraďovací príkaz. Jeho účelom je priradenie hodnoty do premennej. Jeho syntax je veľmi jednoduchá:

```
premenná = výraz
```

Predpokladajme, že máme nadeklarovanú premennú *vek*; potom do nej môžeme priradiť napr.:

```
vek = 17
vek = 15 + 10
vek = vek + 2
```

Prvý ani druhý príkad asi netreba objasňovať. Tretí je však zaujímavejší: keď sa spracováva priraďovací príkaz, vždy sa najprv vyhodnotí pravá strana, až potom sa priradí do ľavej. V tomto príklade keď má *vek* hodnotu napríklad 25 (vzhľadom na predchádzajúci beh programu), tak po vykonaní bude mať hodnotu 25 + 2, čiže 27.

Na konštrukciu príkazov VBA obsahuje niekoľko typov operátorov (aritmetické, logické, relačné a spojovacie). Okrem toho používa zátvorky, ktoré umožňujú predefinovať prioritu matematických operácií vo výraze, ktorá je inak podobná ako v matematike (najprv

umocňovanie, potom delenie a násobenie a nakoniec plus a mínus). V nasledujúcom výklade uvedieme prehľad operátorov.

Aritmetické operátory používané vo VBA

Aritmetické operátory sa používajú na vytváranie matematických výrazov, napr.:

*A = B*C –22/A*

Operátor ^ – umocnenie (*výsledok = číslo ^ exponent)*
Operátor ***** – násobenie (v*ýsledok = číslo1 * číslo2)*
Operátor / – delenie (*výsledok = číslo1 / číslo2)*
Operátor \ – celočíselné delenie (*výsledok = číslo1 \ číslo2)*
Operátor **Mod** – zvyšok po delení (celočíselnom)
výsledok = číslo1 ***Mod*** *číslo2*
Operátor + – Sčítanie (*výsledok = výraz1 + výraz2)*
Operátor – – Odčítanie (*výsledok = číslo1 – číslo2)*

Relačné operátory používané vo VBA

Ako uvidíme neskôr, program Visual Basicu obsahuje príkazy, ktoré umožňujú rozhodnúť sa programu akým spôsobom pokračovať (či vôbec vykonať isté príkazy, resp. zvoliť *aké* príkazy vykonávať z dvoch alebo viacerých možností). Pri takýchto príkazoch sa konštruujú logické podmienky, ktoré využívajú relačné a logické operátory.

Operátor < – menšie než
Operátor <= – menšie alebo rovné
Operátor > – väčší než
Operátor >= – väčšie alebo rovné
Operátor <> – nerovná sa
Operátor = – rovná sa

Pseudokódom môžeme naznačiť ich význam a použitie napríklad takto:

Ak je A<B Tak A = 10 Inak A = 20

Logické operátory používané v VBA

Využijú sa tiež pri budovaní logických podmienok, ale zväčša zložitejších než je uvedené vyššie. Všetky premenné a výrazy v dole uvedených príkladoch sú logického typu (Boolean):

Operátor **And** – používa sa na vytvorenie logickej kombinácie dvoch výrazov.
výsledok = výraz1 ***And*** *výraz2*
Operátor **Eqv** – používa sa na porovnanie logickej zhody dvoch výrazov.
výsledok = výraz1 ***Eqv*** *výraz2*
Operátor **Imp** – používa sa na vytvorenie logickej implikácie dvoch výrazov.
výsledok = výraz1 ***Imp*** *výraz2*
Operátor **Not** – používa sa na vytvorenie logickej negácie výrazu.

*výsledok = **Not** výraz*

Operátor **Or** – používa sa na vytvorenie logickej disjunkcie dvoch výrazov.
*výsledok = výraz1 **Or** výraz2*

Spojovacie operátory používané v VBA

Operátor **&** – používa sa na spojenie dvoch reťazcov. Výsledok je tiež reťazec.
výsledok = výraz1 & výraz2

Operátor + – spojenie dvoch výrazov (pokiaľ aspoň jeden je typu string)
výsledok = výraz1 + výraz2

1.4.3 *Logické výrazy a vetvenie programu*

V mnohých prípadoch je potrebné vetviť chod programu – povedzme postup výpočtu podľa určitej logickej podmienky, keď máme inak rátať nejakú hodnotu v prípade, že určitá premenná nadobúda hodnotu väčšiu alebo menšiu ako 0. VBA používa najmä dva druhy vetvenia programu: **If** a **Select**. Pri použití **If** sa predpokladá, že podmienka bude mať menej vetiev. Podmienku **Select** budeme používať pri podmienkach s mnohými možnými variantmi.

If *podmienka* Then
[*vykonať ak platí podmienka*]
[ElseIf *iná podmienka* Then
[vykonať ak platí ďalšia podmienka] . . .
[Else
[*ak neplatí ani jedno, vykonať toto*]]
End If

- **If podmienka** – tu uvedieme, akú podmienku budeme testovať (napr.: či premenná X sa rovná piatim). Pri jej zostavení využívame v predchádzajúcej kapitole uvedené logické a relačné operátory. Pokiaľ bola podmienka splnená, program pokračuje príkazmi za slovom Then, ak neplatí, program túto časť preskočí a pokračuje ďalej.
- **ElseIf iná podmienka** – túto časť program nemusí obsahovať. Ide o ďalšiu podmienku, ktorú chceme testovať (napríklad, či premenná X sa rovná šiestim) a bude sa testovať iba v tom prípade, že predošlá podmienka (If) nebola splnená. Využíva sa teda iba v tom prípade, ak testujeme rôzne hodnoty iba jednej premennej resp. funkcie.
- **Else** – táto časť tiež nie je povinná a príkazy sa slovom Else sa vykonajú iba v tom prípade, ak nebola splnená žiadna z predošlých podmienok (napr. hodnota X bude mať inú hodnotu ako je päť alebo šesť – to sme testovali v predošlých podmienkach)
- **End If** – týmito slovami ukončíme celý blok podmienky If

Štruktúra, ktorú sme si ukázali, platí v prípade, že príkazov, ktoré sa majú vykonať po splnení podmienky je viac alebo chceme použiť aj nepovinný prvok **ElseIf**. Pokiaľ však chceme vykonať iba jeden príkaz a nepovinný prvok **ElseIf** nepoužijeme, existuje ešte zjednodušená štruktúra:

If Podmienka **Then** Vykonať ak platí podmienka [**Else** Vykonať ak neplatí podmienka]

V takomto prípade slová **End If** nemusíme uvádzať, časť s Else je opäť nepovinná.

Napríklad absolútnu hodnotu by ste rátali takto (keby na to nebola vo VBA funkcia):
If a > b then absHodnota = a – b else absHodnota = b – a

SELECT

Select Case Premenná na testovanie
[**Case** hodnota
[vykonať ak premenná na testovanie obsahuje hodnotu po slove **Case**]] . . .
[**Case ...**
[**Case Else**
[ak neplatí ani jedno, vykonať toto]]
End Select

- **Premenná na testovanie** – uvedieme premennú, ktorej hodnotu budeme testovať
- **Hodnota** – Ak premenná obsahuje hodnotu uvedenú v tomto prvku, vykonajú sa príkazy pod ňou. Takýchto podmienok môže byť neobmedzene veľa, každú dávame za slovo Case
- **Case Else** – Pokiaľ neplatí ani jedna z podmienok za slovami Case, vykonajú sa príkazy pod týmto prvkom
- **End Select** – ukončenie celej podmienky Case

Napríklad:

```
Dim Cislo
Cislo = 8    ' Nastavenie hodnoty.
Select Case Cislo    ' Vyhodnoť Cislo.
Case 1 To 5    ' Cislo medzi 1 až 5, včetne.
   Debug.Print "Medzi 1 až 5"  'Debug.Print vypisuje do okna Immediate – súčasť editora VB
' Iba nasledujúci test bude vyhodnotený ako pravda.
Case 6, 7, 8    ' Cislo medzi 6 až 8. Pozor – čiarka značí alebo
   Debug.Print " medzi 6 až 8"
Case 9 To 10    ' Cislo je 9 alebo 10.
Debug.Print "Väčšie než 8"
Case Else    ' iné hodnoty.
   Debug.Print "Nie je medzi 1 až 10"
End Select
```

1.4.4 *Cykly*

Cyklus **For...Next,** ako aj ďalšie cykly Visual Basicu, slúžia na vykonanie príkazov viackrát po sebe, pričom sa pri každom opakovaní cyklu v premennej *Počítadlo* zvýši hodnota (prípadne zníži).

For *Počítadlo* = Začiatočná Hodnota **To** Konečná hodnota [**Step** *krok*]
[*príkazy*]
...
[**Exit For**]
...
Next [*počítadlo*]

- **Počítadlo –** ide o premennú, v ktorej mechanizmus cyklu automaticky zvyšuje (znižuje) hodnotu o jedna resp. o zadaný krok po prebehnutí príkazov cyklu medzy jeho hlavičkou a príkazom Next.
- **Začiatočná hodnota –** hodnota, ktorou budeme začínať pri počítaní.
- **Konečná hodnota –** ako dosiahne Premenná počítadla túto hodnotu, cyklus sa končí**.**
- **Krok –** tu uvedieme, akú hodnotu budeme pričítavať do premennej počítadla pri každom opakovaní. Nemusí byť udané – v takom prípade sa pri každom opakovaní pričíta hodnota +1.
- **Exit For –** predčasné ukončenie cyklu**.**

Takýto cyklus spôsobí, že počítač pípne 5 x krát po sebe:

```
For i = 1 To 5
Beep
Next i
```

Nasledovná subrutina vypočíta faktoriál z hodnoty, ktorú užívateľ zadá do TextBoxu1 (urobte si formulár s jedným TextBoxom a s jedným tlačidlom):

```
Private Sub CommandButton1_Click()
Dim I, A As Integer
Dim Faktorial As Long

A = TextBox1.Value
Faktorial = 1

For I = 2 To A
Faktorial = Faktorial * I
Next I

MsgBox "Faktoriál je " & Faktorial

End Sub
```

Cyklus typu For...Next môže definovať aj prírastok (Step, krok):

```
'pre i od 1 do 10 s krokom 2 (tj. čísla 1, 3, 5, 7, 9)
For i = 1 To 10 Step 2
  x = x + i
  r = r & i & ", "
Next i
Debug.Print Left(r, Len(r) – 2)        'tlač do debugovacieho okna v VBA editori
```

Krok môže byť aj záporný (pri zmene parametra cyklu dochádza k odpočítaniu):

```
'pre i od 5 do 1 s krokom –1 (5, 4, 3, 2, 1)
For i = 5 To 1 Step –1
  x = 2 * i
  r = r & i & ", "
Next i
```

```
Debug.Print Left(r, Len(r) – 2)
```

A nakoniec, krok sa nemusí nutne rovnať jednej...

```
'pre i od 5 do 1 s krokom –1.5 (5, 3.5, 2)
For i = 5 To 1 Step –1.5
  x = 2 * i
  r = r & i & ", "
Next i
Debug.Print Left(r, Len(r) – 2)
```

Do ... Loop

Cyklus Do...Loop funguje ako podmienkou ukončený cyklus. V praxi to znamená, že cyklus bude prebiehať dovtedy, dokiaľ bude platiť zadaná podmienka.

Syntax:

Do {**While** | **Until**} *podmienka*]
[*príkazy*]
[**Exit Do**]
[*príkazy*]
Loop
ALEBO:
Do
[*príkazy*]
[**Exit Do**]
[*príkazy*]
Loop [{**While** | **Until**} *condition*]

- **Do** – začiatok cyklu
- **{While | Until}** ***podmienka*****]** – Cyklus pokračuje, pokiaľ je splnená podmienka uvedená za While, alebo *nie* je splnená podmienka za Until.
- **[Exit Do]** – Predčasné ukončenie cyklu
- **Loop** – Koniec cyklu

Pozn.: While a **Until** nemusia byť vôbec použité – v tom prípade sa cyklus bude dať ukončiť iba pomocou **Exit Do**. Pri vytváraní cyklov **Do...Loop** treba dať pozor – pri chybe môžete vytvoriť nekonečný cyklus, ktorý môže prebiehať nekonečne, čo povedie k totálnemu zamrznutiu systému a teda aj strate neuložených dát. Napríklad:

```
x = 5
Do Until x > 5
Beep
x = x + 1
Loop
```

While...Wend

Cyklus **While...Wend** je zjednodušenou podobou cyklu Loop...While.
Syntax:
While podmienka
[príkazy]
Wend
Pokiaľ platí podmienka, vykonajú sa príkazy medzi **While** a **Wend**

Môžeme si tento cyklus vysvetliť na príklade, ktorým bude zisťovanie či je dané číslo prvočíslo. Prvočíslo je také číslo, ktoré má len dva delitele a to seba samé a jednotku.

Na formulár nakreslite jedno tlačidlo, jeden popis – Label a jedno textové pole. Do textového poľa zadáme zisťované číslo, po kliknutí na tlačidlo prebehne kód, ktorý zistí či to číslo prvočíslo je, alebo nie a výsledok sa vypíše do popisky – Label.

Kód pre tlačidlo bude vyzerať takto:

```
Private Sub CommandButton1_Click()
cislo = Val(Text1.Text)
delitel = 2
While cislo Mod delitel <> 0
delitel = delitel + 1
Wend
If delitel = cislo Then
Label1.Caption = "Zvolené číslo je prvočíslo"
Else
Label1.Caption = "Zvolené číslo nie je prvočíslo."
End If
End Sub
```

Rozbor: Načítame si číslo z textového poľa a zadefinujeme si deliteľ na 2, pretože číslo 2 je najmenšie prvočíslo (program nie je zabezpečený proti vloženiu 0,1 alebo písmena). Mod je funkcia VBA, ktorá vracia zvyšok po delení. Podmienený príkaz vypíše výsledok skúmania. Ak chcete, aby Vám program vypísal akého deliteľa našiel, stačí za druhé priraďovanie textu popisu dopísať napr. príkaz ... & "Našiel sa deliteľ " & delitel ...

For Each..Next

Tento typ cyklu má význam "pre každý prvok kolekcie opakuj". Kolekciu predstavuje napríklad množina hladín vo výkrese, objekty vyselektované vo výberovej množine a podobne. Podstatné je, že u tohto typu cyklu nemusíme poznať počet prvkov, pre ktoré sled príkazov opakujeme, napríklad nasledovný schematický kód (reálne príklady nájdete ďalej v skriptách):

```
'...pre všetky entity vo výberovej množine nazvanej Vyber
For Each Entity In Vyber
  'zmeň farbu
  Entity.Color = acRed
```

'ďalšia entita
Next Entity

1.4.5 *Funkcie a procedúry*

Funkcie a procedúry slúžia na rozdelenie programu na menšie logické časti. Zdrojový kód programu okrem niektorých deklarácií sa vždy nachádza vo vnútri nejakej funkcie alebo procedúry. Je vhodné ich čo najviac používať napríklad v takých častiach zdrojového kódu, ktoré sa často opakujú a sú volané z viacerých miest programu. Zdrojový kód vtedy nemusíme umiestňovať do každej časti osobitne ale vytvoríme si funkciu alebo procedúru s príslušným zdrojovým kódom a túto funkciu budeme na týchto miestach vyvolávať, čo nám ušetrí veľa práce.

V praxi to znamená, že ak si vytvoríme funkciu, ktorá sa bude volať *VypocitajVzorec()* a do nej vložíme príslušný kód pre výpočet určitého vzorca, stačí aby sme v nejakej časti zdrojového kódu napísali *VypocitajVzorec ()* a program hneď skočí na funkciu a jej príkazy vykoná. Po ukončení behu funkcie sa vráti na svoje miesto a pokračuje ďalej. V zátvorkách za menom funkcie môžu byť prípadné parametre pre výpočet vzorca.

Rozdiel medzi funkciou a procedúrou je veľmi jednoduchý – funkcia vracia hodnotu, ale procedúra nie, aj keď sa hodnoty v nej vypočítané môžu prenášať do iných častí kódu pomocou premenných.

Ďalej uvádzame deklaráciu funkcií. Údaje v hranatých zátvorkách sú nepovinné (to znamená, že ak ich nepotrebujete použiť, nemusia byť uvedené).

[**Public** | **Private** [**Static**] **Function** meno [(argumenty)] [**As** dátový typ]
… kód funkcie
[**Exit Function**]
… kód funkcie
End Function

Public | **Private** – Tieto kľúčové slová, ako aj pri premenných, aj tu označujú dostupnosť funkcie z rôznych miest kódu:

Sub Procedura1()
'procedura dostupná pro celý projekt
'ľubovoľný programový kód
End Sub

Public Sub Procedura2()
'to isté ako Procedura1
'ľubovoľný programový kód
End Sub

Private Sub Procedura3()
'procedura dostupná iba pre modul, v ktorom sa vyskytuje
'ľubovoľný programový kód
End Sub

[**Static**] – Ak uvediete do deklarácie aj **Static**, všetky hodnoty premenných nadobudnuté počas behu programu budú uchované aj v dobe, keď funkcia nie je aktívna.
[(argumenty)] – ide o údaje, ktoré musí užívateľ zadať pri volaní funkcie.
[As dátový typ] – Určuje typ hodnoty, ktorú funkcia vracia.
[Exit Function] – Používa sa ak chcete predčasne ukončiť funkciu.
[End Function] – Koniec funkcie.

Z procedúry na výpočet faktoriálu môžeme urobiť funkciu týmto spôsobom:

```
Public Function Faktorial(A as Integer) As Long
Dim I As Integer
Faktorial = 1
For I = 2 To A
Faktorial = Faktorial * I
Next I
End Function
```

Túto funkciu môžeme kdekoľvek v kóde volať napríklad týmto príkazom:

```
B=Faktorial(12)
```

Funkcie je najlepšie zapisovať do modulov. Z modulov ich môžete používať odkiaľkoľvek. Pokiaľ ale budete chcieť túto procedúru používať iba v rámci jedného modulu, kde ju budú používať ostatné funkcie práve tohto modulu, a pre iný modul budete chcieť použiť funkciu s rovnakým názvom, ale s iným obsahom kódu, tak sa z nej musí urobiť funkcia privátna:

```
Private Function NázovFunkcie() As Double
.
.
End Function
```

Takto zapísaná funkcia bude platná iba v rámci jedného modulu.
Deklarácia procedúr vyzerá takto:
[Private|Public] [Static] Sub *meno* ***(****argumenty****)***
[Exit Sub]
End Sub

Volanie procedúr záleží od ich použitia.

1. Pre funkcie, u ktorých nezáleží na návratovej hodnote a pre procedúry používame tieto dve formy:
   meno_funkcie_alebo_procedury(hodnoty_argumentov)
   Call meno_funkcie_alebo_procedury(hodnoty_argumentov)
2. Pre funkcie, u ktorých záleží na návratovej hodnote:
   meno_premennej = meno_funkcie(hodnoty_argumentov)
   Set meno_premennej = meno_funkcie(hodnoty_argumentov)

Príklad subrutiny pre výpočet plochy trojuholníka:

```
Sub VypocetPlochyTrojuholnika(OdvesnaA as Double, OdvesnaB as Double, Prepona as
Double)
  Dim Plocha As Double      ' Deklarácia premennej
  Dim S As Double           ' Deklarácia premennej
  If OdvesnaA = 0 Or OdvesnaB = 0 Or Prepona = 0 Then  ' Ak je jeden z argumentov 0:
    Exit Sub                                           ' Ukončenie procedúry Sub.
  End If
  S = (OdvesnaA + OdvesnaB + Prepona) / 2
  Plocha = Sqr (S * (S – OdvesnaA) * _
    (S – OdvesnaB) * (S – Prepona))
                    ' Výpočet plochy trojuholníka.
  Debug.Print Plocha                                   ' Tlač obsahu promennej
                                                       ' Plocha do okna Immediate.
End Sub
```

Zavolanie tejto subrutiny sa uskutoční takto:
VypocetPlochyTrojuholniku 10 10 15

Okrem možnosti napísania vlastných funkcií a procedúr obsahuje Visual Basic mnoho zabudovaných štandardných funkcií pre rôzne ciele. Uvedieme si ako príklad niektoré dialógové funkcie, matematické funkcie, funkcie pre prácu s textovými reťazcami a iné.

Funkcia MsgBox

Funkcia MsgBox je zabudovaná funkcia VBA a slúži na zobrazenie rôznych typov okien s textom a s tlačidlami podľa Vášho výberu. Funkcia vracia číslo **Integer**, ktoré určuje, ktoré tlačidlo bolo stlačené. To je odpoveď užívateľa a program môže na túto odpoveď v ďalšej časti reagovať. Pokiaľ nezáleží na tom, ktoré tlačidlo bolo stlačené, možno funkciu **MsgBox** použiť aj ako procedúru (t. j. bez udania premennej, do ktorej vložíme hodnotu funkcie).

Syntax:

MsgBox(Text[, tlačidlá] [, titulok] [, *HelpSúbor*, *context*])

- **Text** – ide o samotný text, ktorý bude zobrazený v dialógu. Môže mať maximálne 1024 znakov.
- **[Tlačidlá]** – konštanty, alebo hodnoty typu Integer, ktoré reprezentujú použité tlačidlá. Pokiaľ vynecháme tento parameter, bude implicitne priradené tlačidlo OK. Tu sú vysvetlivky k niektorých konštantám, ktoré sa Vám ponúknu v PopupMenu pri písaní zdrojového kódu
  - vbOKOnly – Iba tlačidlo OK.
  - vbOKCancel – Tlačidlá OK and Cancel.
  - vbAbortRetryIgnore – Tlačidlá Abort, Retry, a Ignore.
  - vbYesNoCancel – Tlačidlá Yes, No, a Cancel.
  - vbYesNo – Tlačidlá Yes a No.
  - vbRetryCancel – Tlačidlá Retry a Cancel.
  - vbYes – Tlačidlo Yes.
  - vbNo – Tlačidlo No.

- – vbOK – Tlačidlo OK.
- **[Titulok]** – Titulok dialógu. Pokiaľ nie je uvedený, ako titulok bude zvolený názov projektu t. j. aplikácie
- **[*HelpSúbor*, *context*]** – *HelpSúbor* je cesta k súboru Pomocníka (Helpu), ktorý sa má zobraziť. *Context* je číslo, ktoré špecifikuje časť Pomocníka, ktorý sa má zobraziť.

Príklad:

x = MsgBox("Dialóg s tlačidlom OK a CANCEL", vbOKCancel, "Titulok môjho dialógu")
If x = vbOK Then
MsgBox ("Klikol si na OK")
Else
MsgBox ("Klikol si na CANCEL")
End If

Iný spôsob zadania vzhľadu prvku MsgBox je ten, že zadáte číslo, ktoré určuje aký obrázok a aké tlačidlá sa majú zobraziť. Toto číslo je vždy súčtom troch čísiel: prvé určuje aké tlačidlá majú byť zobrazené, druhé určuje obrázok, a tretie určuje ktoré tlačidlo má byť prednastavené, t. j. že nie je nutné na dané tlačidlo kliknúť, ale stačí stlačiť klávesu Enter (tento spôsob zápisu umožňuje širšie určenie výzoru okna).

Nasledujúce tabuľky ukazujú význam všetkých používaných čísiel:

Prvé podčíslo	
Číslo	**Tlačidlo**
0	Ok
1	Ok, Storno
2	Prerušiť, Opakovať, Ignorovať
3	Áno, Nie, Storno
4	Áno, Nie
5	Opakovať, Storno

Druhé podčíslo	
Číslo	**Obrázok**
16	
32	
48	
64	

Tretie podčíslo	
Číslo	**Tlačidlo**
0	Predvolené je prvé tlačidlo
256	Predvolené je druhé tlačidlo
512	Predvolené je tretie tlačidlo

Funkcia InputBox

Funkcia **InputBox** slúži na zobrazenie dialógu, tak ako aj funkcia **MsgBox**. Rozdiel je však v tom, že dialóg zobrazený pomocou funkcie **InputBox**, obsahuje aj políčko na zadávanie hodnôt, takže nemá užívateľa informovať ale má ho požiadať o zadanie určitej hodnoty.

Tu je syntax:

Premenna = InputBox([text, titulok] [, default] [, x] [, y] [, pomocník, kontext])

- **Premenna** – Ak v dialógu bolo stlačené OK, do tejto premennej sa vloží hodnota vyplnená v kolónke, ak bolo stlačené Cancel, premenná bude obsahovať "" – prázdnu hodnotu.
- **Default** – hodnota, ktorá má byť vyplnená pri spustení dialógu
- **x,y** – súradnice horného rohu dialógu na obrazovke

Príklad:

```
Hodnota = InputBox("Zadajte hodnotu", "Hodnota", "Hodnota", 150, 150)
If Hodnota <> "" Then
MsgBox ("Hodnota = " & Hodnota)
Else
MsgBox ("Žiadna hodnota")
End If
```

Funkcia ABS

Vráti absolútnu hodnotu čísla, ktoré bolo zadané.

Syntax Abs(číslo)

Číslo – číselný argument môže byť hocijaký platný matematický výraz. Ak číslo obsahuje Null, Null je aj výsledná hodnota. Ak je číslo prázdna premenná, výsledná hodnota je 0.

Absolútna hodnota čísla je jeho hodnota bez znamienka.

Príklad: ABS(–1) a ABS(1) ‘výsledná hodnota pre obidve je 1.

Funkcia Sin

Vráti sínus zadaného uhla

Syntax Sin(číslo)

Číslo – potrebný argument je dátový typ Double alebo hocijaký platný číselný výraz ktorý vyjadruje stupne v radiánoch.

Funkcia **Sin** počíta s uhlom a vracia podiel strán protiľahlej strany uhla k prepone pravouhlého trojuholníka. Výsledné hodnoty sa nachádzajú v rozpätí od –1 do 1.

Prevod stupňov na radiány: stupne * pi /180. Prevod radiánov na stupne: radiány*180/pi.

Funkcia Cos

Vráti kosínus zadaného uhla.
Syntax – **Cos**(číslo)

Číslo – potrebný argument je dátový typ Double alebo hocijaký platný číselný výraz ktorý vyjadruje stupne v radiánoch.

Funkcia **Cos** počíta s uhlom a vracia podiel strán – priľahlej strany uhla k dĺžke prepony pravouhlého trojuholníka.
Výsledné hodnoty sa nachádzajú v rozpätí od –1 do 1.

Funkcia Tan

Vráti tangens daného uhla v radiánoch.
Syntax **Tan**(číslo)

Číslo – potrebný argument je dátový typ Double alebo hocijaký platný číselný výraz ktorý vyjadruje stupne v radiánoch.

Funkcia **Tan** počíta s uhlom a vracia podiel strán pravouhlého trojuholníka. Podiel je dĺžka protiľahlej strany uhla k priľahlej strane.

Funkcia Exp

Vracia číslo v tvare **Double,** ktoré predstavuje číslo **e** (opak prirodzeného logaritmu) umocnené mocninou x (e^x).
Syntax **Exp**(number)

Číslo – potrebný argument je dátový typ Double alebo hocijaký platný číselný výraz.

Ak hodnota čísla prekročí 709.782712893, nastane chyba. Konštanta **e** je približne 2.718282.

Funkcia Log

Vráti prirodzený logaritmus čísla.
Syntax **Log**(číslo)

Číslo – potrebný argument je dátový typ Double alebo hocijaký platný číselný výraz väčší ako nula.

Prirodzený logaritmus je logaritmus zo základom **e**. Konštanta **e** je približne 2.718282.

Môžeme počítať logaritmy zo základom n pre hocijaké číslo x vydelením prirodzeného logaritmu x prirodzeným logaritmom n takto:

Log n (x) = **Log**(x) / **Log**(n)

Nasledujúci príklad uvádza ako sa počítajú logaritmy zo základom 10:

```
Static Function Log10(X)
    Log10 = Log(X) / Log(10#)
End Function
```

Funkcia Sqr

Vráti druhú odmocninu daného čísla.
Syntax **Sqr**(číslo)

Číslo – Potrebný číselný argument je číslo v tvare Double, alebo hocijaký platný číselný výraz väčší alebo rovný nule.

```
Príklad:    A = Sqr(4)     ' Výsledok 2.
            A = Sqr(23)    ' Výsledok 4.79583152331272.
            A = Sqr(0)     ' Výsledok 0.
            A = Sqr(–4)    ' Pri zadaní záporného čísla nastane run-time error
```

Funkcia Int, funkcia Fix

Vracajú zaokrúhlené celé číslo.
Syntax **Int**(číslo)
Fix(číslo)

Číslo – potrebný argument je dátový typ Double alebo hocijaký platný číselný výraz. Ak číslo obsahuje Null, hodnota Null je vrátená.

Obidve funkcie **Int** a **Fix** odstraňujú zlomkovú časť z čísla a vracajú vyplývajúce celé číslo. Rozdiel medzi **Int** a **Fix** je vtedy, ak je číslo záporné. **Int** vracia prvé záporné číslo menšie alebo rovnajúce sa číslu, zatiaľ čo funkcia **Fix** vracia prvé záporné číslo väčšie alebo rovné číslu. Napríklad **Int** konvertuje -8,4 na -9, a **Fix** konvertuje -8,4 na -8.

Fix(číslo) je rovné ako: **Sgn**() * **Int**(**Abs**(číslo))

```
Príklad:
    A = Int(99.8)      ' Vráti 99.
    A = Fix(99.2)      ' Vráti 99.
```

```
A = Int(–99.8)        ' Vráti –100.
A = Fix(–99.8)        ' Vráti –99.

A = Int(–99.2)        ' Vráti –100.
A = Fix(–99.2)        ' Vráti –99.
```

Rnd Function

Vracia dátový typ Single obsahujúci náhodné číslo.
Syntax Rnd[(číslo)]

Číslo – potrebný argument je dátový typ Single, alebo hocijaký platný číselný výraz.

Výsledky:

Ak je číslo:	Rnd generuje:
Menšie ak ako nula	Vždy to isté číslo, použitím čísla ako počiatočné číslo.
Väčšie ako nula	Náhodné číslo v rade.
Rovné nule	Najčastejšie posledné generované číslo.
Nie je zadané	Náhodné číslo v rade.

Výsledkami funkcie Rnd sú hodnoty menšie ako 1 ale väčšie alebo rovnajúce sa nule.
Hodnota čísla určuje, ako funkcia Rnd generuje náhodné čísla:
Pre každé zadané vstupné číslo, je generovaná tá istá číselná postupnosť pretože každá postupne volaná funkcia Rnd používa predchádzajúce číslo ako počiatočné číslo pre ďalšie číslo v poradí.
Pred volaním funkcie **Rnd** sa používa príkaz the **Randomize** bez argumentov na iniciovanie náhodného číselného generátora s počiatočným číslom založeným na systémovom časovači.
Na tvorbu náhodných čísel v danom rozsahu *lowerbound* až *upperbound* použite tento vzorec:

*Int((upperbound – lowerbound + 1) * Rnd + lowerbound)*

Kde: Upperbound – je navyššie číslo v rozsahu
Lowerbound – najnižšie číslo v rozsahu

Na zopakovanie postupnosti náhodných čísel, treba volať funkciu Rnd zo záporným argumentom ihneď pred použitím **Randomize** s číselným agrumentom. Použitie **Randomize** s tou istou hodnotou pre číslo nezopakuje predchádzajúcu postupnosť.

Sgn Function

Vracia Variant (Integer) indikujúci druh čísla – kladné záporné, nula.
Syntax **Sgn**(číslo)

Číslo – Potrebný číselný argument môže byť hocijaký platný číslený výraz.

Výsledky:

Ak je číslo	Funkcia Sgn vráti
<0	1
0	0
>0	–1

Príklad:
```
A = Sgn(12)        ' Výsledok  1
A = Sgn(–2.4)      ' Výsledok –1
A = Sgn(0)         ' Výsledok  0
```

S premennými typu string, ktoré obsahujú textové reťazce môžeme manipulovať pomocou ďalej uvedených funkcií na prácu s reťazcami.

Funkcia Left, Right:

Vráti z premennej deklarovanej ako String prípadne Variant zodpovedajúci počet znakov z ľavej strany reťazca. Na zistenie počtu znakov v textovom reťazci sa používa funkcia **Len**.

Syntax: ***Left****(string, dĺžka)*

String – názov premennej, musí byť definovaná ako dátovy typ – String (prípadne Variant)
Dĺžka – je počet znakov, ktoré chceme aby boli vrátené.

Príklad:
StringRetazec = "Toto je text"
StringRetazec = ***Left*** *(StringRetazec, 4)*

V našom prípade by StringRetazec obsahoval hodnotu "Toto".

Funkcia **Right** je obdobou funkcie **Left** s rozdielom, že vracia zodpovedajúci počet znakov z pravej strany.

Funkcia Len

Funkcia **Len** vracia počet znakov, ktoré sa nachádzajú v stringovej premennej.

Syntax: ***Len****(string)*

Príklad: *A =* ***Len*** *(StringRetazec)*

Premenná A bude obsahovať číselnú hodnotu 12, ktorá vyjadruje počet znakov v premennej StringRetazec.

V prípade, že si potrebujeme uložiť súbor pod iným menom tak môžeme použiť tento príklad. Premenná OtvSubor obsahuje cestu k momentálne otvoreného súboru. Pomocou

príkazu Len zistíme počet znakov, ktorá táto cesta obsahuje a odčítame z neho 4 znaky. Príkazom Left zapíšeme tieto znaky a následne k nim pripočítame.

OtvSubor = "C:\Skola\diplomova praca.doc"
Subor = Left(OtvSubor, (Len(OtvSubor) – 4))
Premenna Subor obsahuje hodnotu "C:\Skola\diplomova praca"

Subor = Subor + "1.doc"
Premenna Subor obsahuje hodnotu "C:\Skola\diplomova praca1.doc"

Funkcie – LTrim, RTrim, Trim

Reťazce môžu byť zadané s medzerami, ktoré sú zbytočné, prípadne mätúce v rôznych porovnávacích testoch (napríklad či sa zadaný reťazec s menom nachádza v databáze zamestnancov) a preto sú k dispozícii funkcie Trim, LTrim a RTrim, ktoré prípadné zbytočné medzery odstránia.

Funkcie LTrim, RTrim, Trim vracajú kópiu textového reťazca bez medzier z ľavej strany (LTrim), z pravej strany (RTrim), alebo z oboch strán (Trim).

Syntax:
LTrim(string)
RTrim(string)
Trim(string)
Príklad:
Retazec = **LTrim**(" udaj1 ") 'Retazec obsahuje "udaj1 ".
Retazec = **RTrim**(" udaj1 ") 'Retazec obsahuje " udaj1".
Retazec = **Trim** (" udaj1 ") 'Retazec obsahuje "udaj1".

LCase, UCase – konverzia na malé alebo veľké písmená

Funkcia LCase – vracia textový reťazec skonvertovaný na malé písmená.
Funkcia UCase – vracia textový reťazec skonvertovaný na veľké písmená.

Syntax: **LCase**(string)
UCase(string)

String – Hocijaký platný textový reťazec. Ak je reťazec prázdny, vrátená hodnota nie je žiadny znak.

Príklad: retazec = "ToTo je ReTaZec 151!!"
A = **LCase(retazec)**
Výsledkom je hodnota premennej A "toto je retazec 151!!"

V prípade LCase sú veľké písmena skonvertované na malé. Všetky malé písmená a nepísmenové znaky zostávajú bezo zmeny. To iste platí v prípade UCase.

Často sa stane, že potrebujeme porovnať dva reťazce, napr. pri zadávaní hesla. Užívateľ má ako korektné heslo napísať napr. "heslo", lenže nie každý užívateľ ho napíše s

malými písmenami. Niekto napíše "Heslo", iný "HESLO" a pár expertov napíše "HeSlO". My, ako ľudia, by sme všetky tieto varianty odpovede uznali, lebo ide o tie isté slová, ale každé je inak napísané. Lenže ak bude počítač porovnávať tieto reťazce s pôvodným reťazcom "heslo", určí, že reťazce sa nezhodujú. Aby sme dokázali korektne porovnať tieto dva údaje, musíme zadávaný upraviť.

```
'upravenie reťazca zmenením jeho písmen
Dim Retazec as String
Dim  UpravenyRetazec as String
Dim  SpravneHeslo as String
SpravneHeslo ="heslo"
Retazec ="HeSlO"
UpravenyRetazec = LCase(Retazec)
If  UpravenyRetazec = SpravneHeslo  then MsgBox "Správne zadané heslo"
```

Tento príklad, použil funkciu LCase a zmenil všetky znaky reťazca na malé písmená. Opak tejto funkcie je funkcia UCase, ktorá funguje úplne rovnako, len všetky znaky zmení na veľké. Napr.: UCase("Jozef Mak") = "JOZEF MAK"

Ak chcete zistiť, či písmeno, ktoré zadal užívateľ je malé alebo veľké použite takýto kód:

```
Dim Retazec as  String
Retazec  = "toto"
If Lcase(Retazec) = Retazec then MsgBox "reťazec  je napísaný  malým písmenom"
```

Funkcia InStr

Vráti pozíciu prvého vyskytu textového reťazca vo vnútri druhého.

Syntax: InStr([start,]string1, string2[, compare])

Funkcia InStr má tieto argumenty:

Start – Voliteľná možnosť. Číselný výraz ktorým nastavujem štartovaciu pozíciu pre každé hľadanie. Ak sa na pozícii nenachádza znak, hľadanie začne na prvom nájdenom znaku. Ak je hodnota Start nula, nastane chyba. Hodnota Start je potrebná ak je nastavené **porovnávanie (compare)**.

String1 – Povinný parameter. Textový reťazec v ktorom prebieha hľadanie.

String2 – Povinný parameter. Textový reťazec ktorý je hľadaný.

Compare – Voliteľná možnosť. Špecifikuje typ porovnávacieho reťazca. Ak je hodnota start vynechaná, nastavenie *Možnosti Porovnania* určí typ porovnania.

Nastavenia pre porovnanie sú:

Konštanta	Hodnota	Popis
vbUseCompareOption	–1	Uskutočňuje porovnanie použitím nastavenia z možnosti porovnania **(Option Compare)**.
vbBinaryCompare	0	Uskutočňuje binárne porovnanie.
vbTextCompare	1	Uskutočňuje textové porovnanie.
vbDatabaseCompare	2	Len pre Microsoft Access. Uskutočňuje porovnávanie založené na informáciách v databáze.

Vrátené hodnoty:

Ak:	Funkcia InStr vráti hodnotu:
reťazec1 je nulový	0
reťazec1 je prázdny	Prázdnu
reťazec2 je nulový	Hodnota Start
reťazec2 je prázdny	Prázdnu
reťazec2 nie je nájdený	0
reťazec2 je nájdený v reťazci1	Pozícia na ktorej je zhoda nájdená
start > reťazec2	0

Príklad – Určenie správnosti e-mailovej adresy

```
Dim  Email As String
Dim Zavinac  As String
Email = "thomas@ulej.sk"
Zavinac = "@"
If InStr(Email,  Zavinac) <> 0 Then  vráti hodnotu 7
MsgBox "Správne zadaná e-mailová adresa"
End If
End Sub
```

Mid Function

Vráti Variant (String) obsahujúci znak(y) z konkrétneho miesta v textovom reťazci.

Syntax: **Mid** (reťazec, štart, [length])

Reťazec – názov premennej typu string z ktorého znaky funkcia vracia. Ak neobsahuje žiadnu hodnotu, nie je vrátený žiadny znak.

Štart – vyžaduje premennú typu Long. Číslo, ktoré vyjadruje pozíciu znaku v stringu od ktorého začína. Ak *štart* je väčší ako počet znakov v stringu, Mid vráti prázdny string ("").

Length – počet znakov, ktoré ma funkcia Mid vrátiť. Ak je počet znakov väčší ako dĺžka stringu od štartovacieho znaku (*štart*) do konca, funkcia vráti všetky znaky od štartovacieho znaku až do konca stringu (vrátane štartovacieho znaku).

Príklad:

```
Dim vektor as String
Vektor = "abcdefgh123"
```

Mid (Vektor, 3, 1)

Mid vráti jeden znak z textového reťazca *Vektor* na pozícii 3. Výsledkom je znak "c".

Iný príklad – niekedy potrebujete vybrať zo známeho reťazca (známeho v danom mieste kódu) jeho časť.

```
'Vyberanie časti reťazca
Dim Retazec  as String
Dim VynatyRetazec
Retazec = "Jozef as usualy Mak"
VynatyRetazec = Mid(Retazec, 1, 5)  VynatyRetazec="Jozef"
```

Často však nepomôže iba vybratie pomocou presných pozícii znakov, lebo napr. neviete čo užívateľ programu zadá do TextBoxu – napríklad chceme vybrať z e-mailovej adresy text pred "@", nemôžte vedieť dopredu kde sa máte začať vyberať z textu. Potom je možné spojiť funkciu Mid s funkciou InStr a dosiahne sa požadovaný výsledok. Príklad s e-mailovou adresou by vyzeral takto:

```
'Vybranie textu pred "@" z e-mailovej adresy
Dim Email  As String
Dim  Zavinac As String
Dim VyslednyRetazec As  String
Email =  "Jozef.Mak@guglmail.sk"
Zavinac = "@"
VyslednyRetazec  = Mid(Email, 1,  InStr(Email, Zavinac) – 1)
MsgBox VyslednyRetazec
```

Funkcia StrComp

Vracia Variant (Integer) indikuje výsledok reťazového porovnávania.

Syntax: StrComp(string1, string2[, compare])

Parameter	**Popis**
Reťazec1	Hocijaký textový reťazec
Reťazec2	Hocijaký textový reťazec
compare	Voliteľná možnosť. Špecifikuje druh reťazového porovnávania. Ak porovnávací argument je prázdny, nastane chyba. Ak je porovnávanie vynechané, nastavenie *compare* určí typ porovnávania.

Nastavenia pre porovnanie (*compare*) sú tieto:

Konštanta	Hodnota	Popis
vbUseCompareOption	–1	Uskutočňuje porovnanie použitím nastavenia z *Možnosti Porovnania* . **(Option Compare)**.
vbBinaryCompare	0	Nastane binárne porovnávanie.
vbTextCompare	1	Nastane textové porovnávanie.
vbDatabaseCompare	2	Len pre Microsoft Access. Uskutočňuje porovnávanie založené na informáciách v databáze.

Funkcia StrComp vracia tieto hodnoty:

Ak	StrComp vracia:
reťazec1 je menší ako reťazec2	–1
reťazec1 je rovnako veľký ako reťazec2	0
reťazec1 je väčší ako reťazec2	1
reťazec1 alebo reťazec2 je prázdny	Null

Príklad:
```
Retazec1 = "ABCD"
Retazec2 = "abcd"
Vysledok1 = StrComp(Retazec1, Retazec2, 1)   ' Vráti 0.
Vysledok2 = StrComp(Retazec1, Retazec2, 0)   ' Vráti –1.
Vysledok3 = StrComp(Retazec2, Retazec1)      ' Vráti 1.
```

Ak je tretí argument rovný 1, nastáva textové porovnávanie. Ak je tretí argument 0 alebo vynechaný nastáva binárne porovnávanie.

Prevod reťazca na číslo a naopak. Funkcie Val, Str.

Ak textový reťazec reprezentuje číselnú hodnotu (čo sa môže stať napríklad pri čítaní z TextBoxu, kde napr. TextBox1.Text vráti text aj keď je to číslo), je možné ju priradiť do číselnej premennej. Je tiež možné priradiť číselnú hodnotu do reťazovej premennej.

Príklad:
```
Dim SingleX As Single
Dim StrY As String
StrY = "100,23"
SingleX = StrYy      'prevod stringu na číslo
```

Visual Basic dokáže automaticky prevádzať premenné na príslušné dátové typy. Pri prevode reťazcov na čísla a naopak je nutné si dávať pozor. Snaha o priradenie nečíselnej hodnoty v reťazci do číselnej premennej spôsobí pri behu programu chybu (run-time error). Veľmi častou chybou je snaha o prevod reťazca tam, kde sú desatinné čísla oddelené čiarkou namiesto bodkou. Používanie desatinnej čiarky je namiesto bodky V prípade, že sú v reťazci oddelené desatinné čísla bodkou, prevod na číslo má túto syntax: **Val**(string)

```
Dim CisloX as Single
StrY = "100.23"
CisloX = Val(StrY)
```

Prevod z čísla na string sa dá jednoducho vykonať pomocou príkazu **Str.**
Syntax: **Str**(číslo)

Príklad: A = Str(–459.65) ' Vráti String hodnotu "–459.65"

Ešte uvedieme niekoľko funkcií na prácu s poliami:

Funkcia LBound

Výsledkom je najmenší dostupný index pre indikovanú dimenziu poľa. Dimenzia je nepovinný údaj, nakoľko je dôležitý iba pri viacrozmerných poliach.

Syntax **LBound**(Meno_pola[, dimenzia])

Syntax funkcie **LBound** má tieto časti:

Časť	Popis
Meno_pola	Povinný parameter. Meno premennej typu pole, riadi sa štandardmi pre označovanie premenných.
dimenzia	Voliteľný parameter. Variant (Long). Celé číslo, ktoré znázorňuje dimenziu, ktorej najmenší index chceme získať. Pre prvú dimenziu je to 1, pre druhú 2 atď. Keď je tento parameter vynechaný zisťuje sa pre prvú dimenziu.

LBound vracia hodnoty v nasledujúcej tabuľke pre pole s týmito dimenziami.
Dim A(1 To 100, 0 To 3, 3 To 4)

Príkaz	Výsledok
LBound(A, 1)	1
LBound(A, 2)	0
LBound(A, 3)	3

Na okraj – prednastavený spodný okraj pre hocijakú dimenziu je 0 alebo 1, závisí to na nastavení v príkaze Option Base. Pre pole vytvorené s fukciou Array je nula nezávisle od Option Base.

Polia pre ktoré sú dimenzie nastavené pomocou Dim, Private, Public, ReDim alebo Static môžu mať ako spodný okraj hocijaké celočíselné číslo.

Príklad

```
Dim MojePole(1 To 10, 5 To 15, 10 To 20)  ' deklarácia poľa
    Dim InePole(10)
    A = Lbound(MojePole, 1)          ' vráti 1.
    A = Lbound(MojePole, 3)          ' vráti  10.
    A = Lbound(InePole)              ' vráti 0 alebo 1, závisí na nastavení
                                     Option Base
```

Funkcia UBound

Funkcia **LBound** sa používa s podobnou funkciou **UBound** na zistenie veľkosti poľa. Na zistenie horného okraja poľa sa používa funkcia **UBound**.

Výsledkom je najväčší dostupný index pre indikovanú dimenziu poľa.

Syntax **UBound**(menopola[, dimenzia])

Syntax funkcie **LBound** má tieto časti:

Časť	Popis
menopolla	Potrebný parameter. Meno premennej poľa, riadi sa štandardmi na označovanie premenných.
dimenzia	Voliteľný parameter.Variant (Long). Celé číslo, ktoré znázorňuje dimenziu, ktorej najväčší index chceme získať. Pre prvú dimenziu je to 1, pre druhú 2 atď. Keď je tento parameter vynechaný zisťuje sa pre prvú dimenziu.

UBound vracia hodnoty pre pole s týmito dimenziami uvedené nižšie
Dim A(1 To 100, 0 To 3, 3 To 4)

Príkaz	Výsledok
UBound(A, 1)	100
UBound(A, 2)	3
UBound(A, 3)	4

Algoritmy, ktoré pracujú s poľami, sú niekedy postavené na poliach s indexovaním od nuly, niekdy od jednej. Implicitne ak zadeklarujete pole takto:

Dim Pole(3) As String,

tak Vám musí byť jasné, ako je to s dolným indexom – bez zadefinovania tzv. *Option Base* bude dolný index 0, pri zadefinovaní (na začiatku modulu) *Option Base 1* bude 1 (Option Base 0 by bol zbytočný). S ohľadom na toto obmedzenie je vhodné robiť do určitej miery aj z tohto pohľadu všeobecné algoritmy. Medze analyzujeme pomocou funkcií *LBound(Pole)* (dolná medza) a *Ubound(Pole)* (horná medza) ako už bolo uvedené. A pretože polia bohužiaľ nedisponujú vlastnosťou *Count* pre zistenie počtu položiek, zistíme ich nezávisle na nastavení Option Base takto:

PocetPoloziek = UBound(Pole) – LBound(Pole) + 1

Ako sa odkazovať vždy na n-tú položku poľa bez ohľadu na Option Base?

n = 2
OdkazNaEntuPolozku = Pole(n – 1 + LBound(Pole))

Funkcia Split, Join

V súvislosti s poliami sú ešte zaujímavé funkcie Split a Join. Prvá z nich rozdelí textový reťazec na základe zadaného oddelovača údajov (napr. bodkočiarka, čiarka, medzera, tabelátor) a druhá naopak spojí údaje z poľa do reťazca.

Syntax:
Split(Reťazec [**,** Oddelovač [**,** počet]]**)**
Join(pole[,Oddelovač]**)**

Napríklad:

```
  Dim myArray
   Dim iLoop
   myArray = Split("Mišo, Jano, Georgína, Jozef", ", ")
   For iLoop = 0 To UBound(myArray)
      MsgBox "myArray(" & iLoop & ") = " & myArray(iLoop), vbInformation, "Príklad na
Split"
   Next
```

Funkcia Asc

Funkcia Asc umožňuje získať asci kód (hodnotu integer) pre určitý znak.
Syntax:
Asc(*string***)**
Pokiaľ má string viac znakov, vráti hodnotu prvého znaku. Môžeme si ukázať príklad využitia takejto funkcie na to, aby sme mohli obmedziť užívateľa programu v tom, čo napíše do TextBoxu. Chceme napríklad (čo môže byť častá požiadavka) aby zadal iba čísla a desatinnú bodku. Umožní nám to udalosť *Change* pre TextBox:

```
Private Sub TextBox1_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
'Umožníme zadať iba jednu desatinnú bodku:
 If InStr(1, TextBox1.Text, ".") > 0 And KeyAscii = Asc(".") Then
      KeyAscii = 0
      Exit Sub
   End If
'Ak nebola zadaná desatinná bodka pokračujeme ďalej:
   Select Case KeyAscii
      Case Asc("0") To Asc("9"), Asc(".")
      Case Else
         KeyAscii = 0
   End Select
End Sub
```

Táto subrutina sa vyvolá vždy keď sa mení obsah text boxu a v prípade zadávania iného než povolených znakov (čísla a desatinná bodka), zmení zadávaný znak na znak s kódom 0, čo je žiaden znak.

1.4.6 *Práca so súbormi*

V mnohých aplikáciách sú potrebné viac alebo menej rozsiahle vstupné údaje na základe ktorých aplikácia spracováva konkrétny problém, alebo naopak poskytuje výstupné údaje, ktoré je potrebné niekam zaznamenať. Na tento účel slúžia napríklad dokumenty Wordu **.doc*, Excelu **.xls*, AutoCADu **.dwg* a podobne. Tieto uvedené aplikácie majú svoje špecifické, tzv. binárne typy súborov, ktoré uchovávajú Vašu prácu. Pri vlastných – hlavne pri menších aplikáciách, väčšinou vystačíme z tzv. textovými súbormi, prácu s ktorými v ďalšom popíšeme.

Ďalej uvedené príkazy jazyku Visual Basic môžeme použiť na základnú manipuláciu so súbormi; pre ich jednoduchosť ich uvedieme iba názvami:

Kill	Zmaže súbor z disku.
Name	Premenuje súbor alebo zložku.
FileCopy	Kopíruje súbor.
FileDateTime	Funkcia vracia dátum a čas vytvorenia alebo poslednej zmeny súboru.
FileLen	Funkcia vracia dĺžku súboru v bytoch.
GetAttr	Vracia atribúty súboru alebo zložky.
SetAttr	Nastavuje atribúty súboru alebo zložky.

Trochu zložitejšia je funkcia *Dir*. Vracia reťazec reprezentujúci súbor alebo zložku podľa zadaného vzoru, atribútov, disku alebo názvu disku.

Dir[(cesta[, atributy])]

Funkcia sa často používa pre zistenie existencie súboru.

```
Dim subor As String
If Dir("C:\autoexec.bat") = "" Then
  MsgBox "Súbor autoexec.bat nebol nájdený"
End If
```

Pokiaľ súbor nie je nájdený, je vrátený prázdny reťazec, v opačnom prípade názov súboru.

Výpis všetkých súborov v zložke je možný týmto spôsobom:

```
Dim subor As String
subor = Dir("C:\WINDOWS\TEMP\*.*")
Do While subor <> ""
  Debug.Print subor
  subor = Dir
Loop
```

Kód vypíše do okna editora VBA Immediate všetky súbory zo zložky C:\WINDOWS\TEMP. Prvý príkaz *Dir* aj s parametrami určí, čo sa bude vyhľadávať, druhý v cykle *Do .. Loop*, vracia ďalšie súbory zložky podľa skôr zadaného vzoru. Po vrátení mena posledného súboru je vrátený prázdny reťazec.

Druhým parametrom funkcie *atributy*, môžeme určiť, aké súbory vyhľadávať. Je možné zadať tieto kombinácie parametrov:

vbNormal	0	Súbory bez atribútov (default hodnota).
vbReadOnly	1	Súbory iba pre čítanie plus súbory bez atribútov.
vbHidden	2	Skryté súbory plus súbory bez atribútov.

vbSystem	4	Systémové súbory plus súbory bez atribútov.
vbVolume	8	Názov disku.
vbDirectory	16	Výpis všetkých zložiek.

Výpis všetkých zložiek koreňového adresáru disku C: a výpis všetkých súborov vrátane skrytých, systémových a iba pre čítanie na disku C: vyzerá takto:

```
Dim subor As String
subor = Dir("C:\", vbDirectory)
Do While subor <> ""
  Debug.Print subor
  subor = Dir
Loop

subor = Dir("C:\*.*", vbSystem + vbHidden + vbReadOnly)
Do While subor <> ""
  Debug.Print subor
  subor = Dir
Loop
```

Príkaz OPEN

Aby sme mohli do súboru zapisovať alebo z neho čítať, musíme ho najskôr otvoriť. Nato slúži príkaz *Open*.

Open cesta For mód [Access prístup] [lock] As [#]číslo [Len = dĺžka_záznamu]

cesta	Názov súboru s cestou.
mód	Mód, v akom bude súbor otvorený.
prístup	Prístup k súboru, čítanie, zápis alebo čítanie i zápis.
lock	Umožnenie alebo naopak zakázanie niektorých operácii na súbore iným procesom.
číslo	Číslo súboru, pomocou ktorého sa k súboru ďalej pristupuje. Pohybuje sa v rozmedzí 1 – 511. Pokiaľ zadáte číslo z rozmedzia 1 – 255, nebude súbor prístupný iným aplikáciám, číslo nad 255 umožní prácu so súborom i iným aplikáciám.
dĺžka záznamu	Pri súboroch s náhodným prístupom udáva dĺžku záznamu v rozmedzí 1–32767 bytov. Pri textových súboroch určuje veľkosť bufferu (udáva sa v počte znakov).

Parameter *prístup* určuje, či bude súbor otvorený pre čítanie (*Read*), zápis (*Write*) alebo na čítanie i zápis (*ReadWrite*). Parameter *lock* určuje prístup pre iné procesy. *Shared* znamená zdieľaný prístup, ostatné procesy môžu do súboru zapisovať aj z neho čítať. *Lock Read* znamená zákaz čítania, *Lock Write* zákaz zápisu pre iné procesy. *Lock Read Write* znamená zákaz zápisu i čítania.

Aby sme si boli istý, že nepoužijeme pre súbor číslo už použité, využijeme služieb funkcie *FreeFile*. Vracia najbližšie voľné číslo pre súbor. Má jediný (nepovinný) parameter, pomocou ktorého môžete rozhodnúť, z ktorého intervalu sa bude číslo vyberať. 0 znamená interval <1 – 255>, 1 interval <256 – 511>.

Textový súbor môžete otvoriť v móde *Append* (zápis od konca súboru), *Input* (čítanie) alebo *Output* (zápis). Pokiaľ súbor existuje, otvorenie v móde *Output* spôsobí výmaz súboru.

```
Dim subor As Integer
subor = FreeFile(1)
Open "C:\prvý.txt" For Output As #subor
```

V tomto kóde najskôr definujeme premennú *subor*, ktorá bude reprezentovať otvorený súbor. Ďalej zistíme prvé voľne číslo z intervalu <256 – 511>, t. j. nechávame možnosť práce so súborom aj iným procesom. Nakoniec otvoríme súbor C:\prvy.txt pre zápis (prípona txt nemá žiadny význam pre kód, iba pre nás, aby sme vedeli, že vytvárame textový súbor). Pokiaľ súbor existoval, je jeho obsah "vynulovaný", v prípade že neexistoval, je vytvorený.

Pokiaľ súbor existuje a my chceme pridávať dáta od jeho konca, otvoríme ho takto (v prípade, že neexistuje, je vytvorený):

```
Open "C:\prvy.txt" For Append As #subor
```

Zápis. Do textového súboru môžeme dáta zapisovať dvoma príkazmi *Print* a *Write*. Syntax oboch je rovnaká. Prvý parameter je vždy číslo súboru, druhý je zapisovaný text. Obidva k zadanému textu pridajú znak konca riadku. Prvý (Print) zapisuje do súboru dáta tak, ako ich vidíte. Druhý (Write) zapisuje dáta do súboru podľa typu zapisovanej premennej.

```
Dim subor As Integer

subor = FreeFile(1)
Open "C:\prvy.txt" For Output As #subor

Print #subor, "zapis print"
Write #subor, "zapis write"

Close #subor
```

Vytvorený súbor bude vyzerať takto:

```
zapis print
"zapis write"
```

Príkaz *Close* uzavrie súbor a uvoľní použité číslo pre ďalšie použitie. Pokiaľ by sme súbor neuzavreli, nebol by v ňom žiadny text. Bol by vytvorený, ale mal by nulovú dĺžku.

Druhý parameter príkazov *Print* a *Write* nie je vlastne jeden parameter, ale pole parametrov, ktoré môže obsahovať viac premenných naformátovaných pomocou ďalších príkazov.

```
Dim subor As Integer

subor = FreeFile(1)
Open "C:\prvy.txt" For Output As #subor

Print #subor, "prvy riadok"
Print #subor,
Print #subor, "Ahoj" ; " " ; "svet"
Print #subor, Spc(20) ; "20 medzier"

Close #subor
```

Prvý riadok kódu zapíše do súboru text *prvy riadok*, druhý zapíše prázdny riadok, tretí slová *Ahoj svet* (bodkočiarka znamená, že jednotlivé premenné sú zaradené hneď za sebou). Štvrtý zapíše 20 medzier a za ne text *20 medzier*.

Čítanie. Z textového súboru sa údaje získavajú tromi príkazmi: *Input*, *Input #* a *Line Input*.

Príkaz *Input* číta zo súboru zadaný počet znakov a vracia premennú typu *String*. Číta všetky znaky, vrátane konca riadku, úvodzovky, čiarky atď., preto sa používa väčšinou na čítanie súborov, ktoré boli zapisované príkazom *Print*.

```
Dim subor As Integer

subor = FreeFile(1)
Open "C:\prvy.txt" For Input As #subor

MsgBox(Input(26, #subor))
Close #subor
```

Vytvorte súbor prvy.txt pomocou prvého kódu, čo sme si uviedli, t. j. aby výsledný súbor vyzeral takto:

```
zapis print
"zapis write"
```

Vyššie uvedený kód zobrazí rovnaký text v okne. 26 znakov čítame preto (viditeľných je iba 24), že Funkcia *Input* počíta medzi znaky aj koniec riadku, t. j. znaky z Ascii kódom 10 a 13.

Príkaz *Input #* číta zo súboru dáta do zadaných premenných.

```
Dim subor As Integer
Dim s1 As String, s2 As String

subor = FreeFile(1)
Open "C:\prvy.txt" For Input As #subor

Input #subor, s1, s2
MsgBox s1 & vbCrLf & s2

Close #subor
```

V prípade, že použijeme rovnaký súbor prvy.txt ako v predchádzajúcom prípade, dostaneme výstup:

```
zapis print
zapis write
```

Z toho je zrejmé, že sa príkaz *Input #* používa na čítanie dát zo súboru, do ktorého bolo zapisované príkazom *Write*. Keby sme použili súbor s jedným riadkom, ktorý by obsahoval napríklad hodnoty 24 53 41, premenné s1 a s2 by boli typu Long, dostali by sme výstup:

```
24
53
```

Posledný z príkazov na čítanie textových súborov je *Line Input*. Už podľa názvu je zrejmé, že príkaz číta jeden riadok súboru a ukladá ho do premennej typu String.

```
Dim subor As Integer
Dim s1 As String, s2 As String
```

```
subor = FreeFile(1)
Open "C:\prvy.txt" For Input As #subor

Line Input #subor, s1
Line Input #subor, s2

MsgBox s1 & vbCrLf & s2

Close #subor
```

Opäť použijeme rovnaký súbor prvy.txt ako predtým a dostaneme výstup:

```
zapis print
"zapis write"
```

Príkaz číta dáta opäť priamo tak, ako boli zapísané, preto sa používa na čítanie súborov, do ktorých bolo zapisované pomocou *Print*. Na rozdiel od *Input* je z reťazca (predtým, než je uložený do premennej) odstránený znak konca riadku.

Skoro vždy, keď postupne čítate všetky dáta z nejakého súboru potrebujete zistiť, či sa už nenachádzate na konci súboru. To umožňuje Funkcia EOF (end of file). Má jediný parameter, číslo súboru. Vracia buď hodnotu True, pokiaľ ste na konci súboru, alebo False, pokiaľ nie:

```
Dim InputData
Open "C:\subor.txt" For Input As #1
Do While Not EOF(1)    ' Rob nasledujúce kým nie si na konci súboru
   Line Input #1, InputData    ' Čítaj jeden riadok zo súboru
   Debug.Print InputData    ' Vytlač ho v okne Immediate
Loop
Close #1    ' Zavri súbor
```

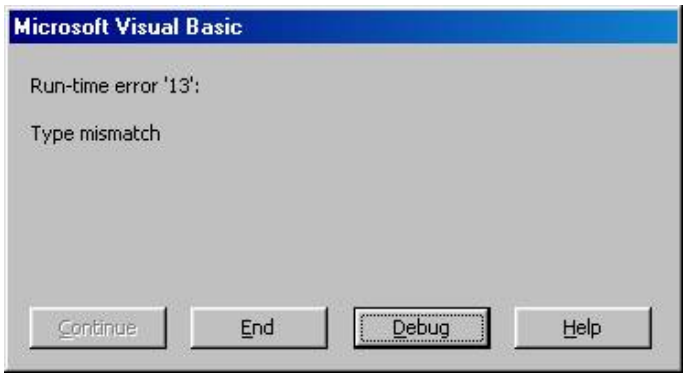
1.4.7 *Ošetrovanie chýb*

Aj keď sa budeme snažiť napísať kód čo najpozornejšie, môžu sa vyskytnúť (a pravdepodobne sa aj vyskytnú) chyby. Pri ideálnych podmienkach by žiadne ošetrenie chýb nebolo nutné. Niektoré súbory však mohli byť omylom zmazané, na disku nie je voľný priestor, odpojí sa sieťová jednotka, užívateľ zadá nevhodný údaj, resp. vôbec žiaden údaj nezadá tam, kde to je potrebné atď. Aby sme mohli ošetriť podobné chyby musíme preto napísať kód, ktorý to prevedie.

V niektorých prípadoch môže tiež dôjsť k chybe priamo v samotnom kóde, tomuto typu chýb sa hovorí skryté chyby (bug). Určitá časť kódu môže byť jednoducho chybne naprogramovaná a program počíta avšak nesprávne. Pokiaľ ide o drobné chyby (napríklad kurzor sa nechová podľa očakávania), môžu užívateľa znechutiť, alebo mu znepríjemniť prácu. Vážnejšie chyby môžu spôsobiť, že aplikácia prestane reagovať na príkazy (napríklad sa zacykluje v nekonečnom cykle) a užívateľ bude možno musieť aplikáciu spustiť znovu a stratí všetko, čo doposiaľ urobil.

Odhaľovaniu a oprave chýb sa hovorí ladenie. Vo VBA je niekoľko nástrojov, ktoré pomáhajú analyzovať činnosť aplikácii. Sú zvlášť užitočné na odhaľovanie miesta, kde chyba vzniká, ale môžeme ju tiež použiť na experimentovanie so zmenami v aplikácii alebo na zistenie, ako pracujú iné aplikácie. Pre tento účel už boli spomenuté možnosti editora VBA ako je krokovanie programu pomocou klávesy F8, sledovanie premenných pomocou okien

Watches a Immediate. Beh programu môžete na niektorom mieste zastaviť pomocou definovania tzv. Breapoint, pričom definovanie spočíva v kliknutí naľavo vedľa príkazu, kde chceme program zastaviť na šedý okraj editora. Odstránenie Breakpoints je v menu Debug.



Obr. 1.15 Chyba vo Visual Basicu

Ako ošetrovať chyby

Pokiaľ v tele procedúry alebo funkcie príde za behu programu k chybe, záleží ďalšia udalosť od toho, či volajúca procedúra (funkcia) mala nastavené zachytávanie chýb. Ak áno, program pokračuje kódom, ktorý vzniknuté chyby ošetruje. Ošetrenie chyby môže znamenať veľa vecí – napríklad ak bola zadaná nesprávna hodnota, kód nepokračuje, ale vráti sa na zadávací formulár, alebo sa program ukončí avšak s presným oznámením ku akej chybe a kde došlo, čo ho umožňuje spustiť znova a správne. Pokiaľ ani volajúca procedúra (funkcia) nemá nastavené ošetrovanie chýb, hľadá program vyššie a vyššie, či je nejaké ošetrenie chýb nastavené. Pokiaľ nič nenájde, potom vyvolá bežné chybové hlásenie.

Režim ošetrenia chýb sa dá nastaviť trojakým spôsobom – príkazmi On Error:

On Error Resume Next – v tomto prípade je vzniknutá chyba ignorovaná a program pokračuje ďalej. .

On Error GoTo error_label – Od tohto miesta v prípade výskytu chyby za behu programu pokračuje príkazom, ktorý nasleduje za uvedeným odkazom. To platí až do konca procedúry alebo funkcie, prípadne do ďalšieho výskytu príkazu zachytenia chyby. Napríklad:

```
Private Sub Form_Load ()
On Error Goto FileOpenError
    Open "C:\NejakySubor.TXT" For Input As #1
    Line Input #1, sData
    Exit Sub

FileOpenError:
    MsgBox "Vyskytol sa problém pri otváraní súboru"
    End
End Sub
```

On Error GoTo 0 – pokiaľ bolo už prv v tele procedúry alebo funkcie nastavené zachytenie chýb, tak od riadku, ktorý nasleduje za týmto príkazom, už chyby ošetrené nie sú a v prípade ich výskytu príde iba k vyvolaniu štandardného chybového hlásenia.

V prípade, ak chceme vymazať vo výkrese nejakú výberovú množinu a nie sme si istí, že či v ňom naozaj existuje, možno použiť nasledujúcu kombináciu:

```
On Error Resume Next        'V prípade chyby choď na ďalší riadok
Application.ActiveDocument.SelectionSets("Obluky").Delete
On Error GoTo 0             'Zrušenie nastaveného ošetrovania chýb
```

Pokiaľ chceme naprogramovať ošetrenie chýb, musíme najskôr zistiť k akej chybe došlo. K tomu slúži objekt Err, ktorý je súčasťou základnej knižnice VBA vývojového prostredia. Na bežnú prácu nám postačia dve vlastnosti tohto objektu. Najčastejšie budeme pracovať s vlastnosťou Number (východzia vlastnosť), ktorá udáva číslo chyby za chodu programu. Ak je hodnota tejto vlastnosti 0, potom k žiadnej chybe nedošlo. Ďalej budeme potrebovať vlastnosť Description čo je vlastnosť typu String, ktorá vracia opis vzniknutej chyby. Text opisu sa môže v rôznych verziách Visual Basicu líšiť. Často je dobré pre identifikáciu chyby používať Err.Number a nie Err.Description a opis chyby upresniť na základe Err.Number a ďalšieho kontextu daného miesta programu (napr. ako sa presne volá súbor, ktorý sa nepodarilo otvoriť)

Návrh chybovej rutiny

Prvým krokom pri písaní chybovej rutiny je určenie odkazu, kde bude chybová rutina začínať. Odkaz na riadok by mal obsahovať opisný názov rutiny a musí byť ukončený dvojbodkou. Zvykom je umiestňovať odkaz na koniec procedúry a pred ňu sa pridáva príkaz Exit Sub, Exit Funkction alebo Exit Property. To umožňuje ukončiť procedúru, bez toho aby bolo prevádzaná chybová rutina, pokiaľ k žiadnej chybe nedošlo.

Telo chybovej rutiny obsahuje kód ošetrujúci chyby, obvykle vo forme príkazov Select Case, alebo If...Then...Else. Musíme určiť, ku ktorým chybám môže dôjsť a stanoviť zodpovedajúci spôsob ošetrenia. Rovnako by vždy malo byť určené, ako budú ošetrené neočakávané chyby, väčšinou v klauzuli Case Else alebo Else (väčšinou príkazom Stop). Vlastnosť number objektu Err obsahuje číslo predstavujúce najbežnejšie chyby za behu programu. Použitím objektu Err v kombinácii s príkazmi Select Case alebo If...Then..Else môže vyriešiť výskyt akejkoľvek chyby.

Chybová rutina je teda časť procedúry určená pre zachytávanie chýb a reakcií na ne. Chybové rutiny by mala obsahovať každá procedúra, kde sa očakávať výskyt chýb (správne by sa malo predpokladať, že každý príkaz Visual Basicu môže spôsobiť chybu). Postup pri navrhovaní chybovej rutiny obsahuje tri kroky:

1. Zapnúť zachytávanie chýb tak, že určíme ktorá časť kódu sa bude prevádzať pri výskytu chyby za behu programu (ktorá chybová rutina bude aktivovaná). Príkaz On Error zapne zachytávanie chýb a prevedie odskok na návestie označujúce začiatok chybovej rutiny. V ďalej uvedenom príklade sa nachádza vzorová chybová rutina začínajúca sa odkazom CheckError. Volanie tejto rutiny v programe bude vyzerať nasledovne:

   ```
   On Error GoTo CheckError
   ```

2. Treba napísať chybovú rutiny, ktorá vyrieši všetky chyby, ktorých výskyt predpokladáte. Chybová rutina CheckError ošetruje chyby použitím príkazov If...Then...Else, kde skontroluje číslo chyby, obsiahnuté vo vlastnosti Number objektu Err. V tomto príklade, ak dôjde k chybe "Disk not ready", zobrazí sa správa upozorňujúca užívateľa na nutnosť zásahu. Iné varovanie sa zobrazí, ak príde k chybe "Device unavailable". Pokiaľ sa vyskytne iná chyba, zobrazí sa popis a postup programu sa preruší.
3. Opustenie chybovej rutiny. Ak príde k chybe "Disk not ready", príkaz Resume vykoná odskok na riadok, v ktorom došlo k chybe. Visual Basic následne skúsi previesť riadok znovu. V prípade ak sa situácia nezmenila, príde k ďalšej chybe a znova je spustená chybová rutina. V prípade výskytu chyby "Device unavailable" vykoná príkaz Resume Next odskok na riadok za riadkom, v ktorom došlo k chybe.

Príklad chybovej rutiny CheckError:

```
CheckError:                    ' Sem je presmerovaný program ak sa vyskytne chyba
   'definícia konštánt, ktoré predstavujú čísla chýb
   Const mnErrDiskNotReady = 71, mnErrDeviceUnavailable = 68
If (Err.Number = mnErrDiskNotReady) Then
      Msg = "Zasuňte disketu správne do mechaniky."   'Zobrazí okno hlásenia s voľbami OK
alebo Cancel
      If MsgBox(Msg, vbExclamation & vbOKCancel) = vbOK Then
         Resume            'ak je stlačené tlačidlo OK, program sa pokúsi previesť príkaz znovu
      Else
         Resume Next       'ak je stlačené iné tlačidlo, program bude pokračovať na ďalšom
riadku
      End If
   ElseIf Err.Number = mnErrDeviceUnavailable Then
      Msg = "Zadaný disk alebo cesta neexistujú " & filename
      MsgBox Msg, vbExclamation
      Resume Next
   Else
      Msg = "Došlo k neočakávanej chybe číslo" & Str(Err.Number)
      MsgBox Msg, vbCritical                'ohlási hlásenie s ikonou STOP
      Stop
   End If
   Resume
End Function
```

Opustenie chybovej rutiny

V ukážkovom príklade CheckError sa používa príkaz Resume, aby bol znova zopakovaný riadok kódu, v ktorom došlo ku chybe a príkaz Resume Next, ktorý prevedie odskok na riadok nasledujúci za tým, v ktorom došlo ku chybe. Existujú ešte ďalšie spôsoby, ako opustiť chybovú rutinu. V závislosti od aktuálneho stavu môžeme použiť ktorýkoľvek z týchto príkazov:

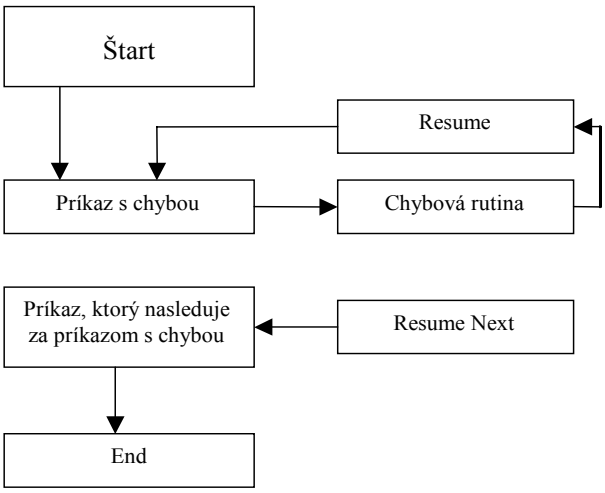
Resume – Pokiaľ je príkaz uvedený v tele ošetrení chyby, pokračuje program v mieste, kde chyba vznikla. Pokiaľ zadáte tento príkaz do kódu, ktorý neošetruje chybový stav, dôjde

k chybe programu. Využíva sa tam, kde sa dá chyba napraviť, napríklad pri zadaní písmena miesto čísla.

Resume Next – Pokiaľ došlo k chybe v procedúre, ktorá chybu spracovala, program bude pokračovať príkazom, ktorý bezprostredne nasleduje za príkazom, ktorý chybu vyvolal .

Resume Line – Program bude pokračovať na uvedenom čísle riadku alebo na uvedenom odkaze. Číslo riadku alebo odkazu musí byť definované v tej istej procedúre, ktorá chybu spracovala.

Err.Raise Number: = číslo Vyvolá chybu za behu programu. Pokiaľ je tento príkaz spustený vnútri chybovej rutiny, hľadá Visual Basic inú možnú chybovú rutinu v zozname volaní procedúr.



Obr. 1.16 Beh programu pri použití príkazov Resume a Resume Next

Rozdiel je v tom, že Resume znova spôsobí prevedenie riadku, v ktorom došlo ku chybe, zatiaľ čo po príkaze Resume Next kód pokračuje až riadkom nasledujúcim za ním, v ktorom došlo ku chybe. Obvykle sa Resume používa vtedy, pokiaľ je chybová rutina schopná chybu opraviť a Resume Next pokiaľ to nie je možné. Je možné napísať chybové rutiny, kde sa výskyt chyby užívateľovi neohlasuje, alebo sa naopak zobrazí chybové hlásenie a užívateľ môže vykonať nutné opravy.

1.4.8 *Využitie Windows API funkcií vo VBA*

Microsoft® Visual Basic® for Applications (VBA) je výkonným nástrojom na vytváranie aplikácií pre Microsoft® Windows®. Pomocou VBA však možno kontrolovať iba relatívne malú časť systému — teda iba tú časť, ktorej objekty a funkcie sú ponúkané hostiteľskou aplikáciou VBA, v našom prípade AutoCADu. V kapitole 1.8 si ukážeme ešte ako možno ovládať aj iné aplikácie (Excel) a využívať ich k vytvoreniu komplexného systému. V tejto kapitole naznačíme využívanie tzv. Windows API funkcií, ktoré nám sprístupňujú funkcie operačného systému.

Ich používanie vyžaduje deklaráciu, ktorá má informovať vami vytváraný program, v akej knižnici systému možno funkciu alebo procedúru, ktorú chcete použiť nájsť. Väčšina funkcií API sa nachádza v súboroch kernel32.dll, user32.dll, gdi32.dll, shell32.dll, ktoré

tvoria jadro systému Windows. Okrem toho sa špecifikujú parametre tejto funkcie a hodnota, ktorú vracia. Syntax bude teda vyzerať takto:

[Public|Private] Declare Sub globalname Lib libname$ [Alias aliasname$] [(argument list)]
alebo:
[Public|Private] Declare Function globalname Lib libname$ [Alias aliasname$] [(argument list)] [As type]

Upozornenie – pokiaľ budete tieto funkcie deklarovať v UserForm, musia byť deklarované ako *Private*.

Celkom jednoduchý príklad – chcete vymazať súbor. Musíte vedieť, že k tomu potrebujete funkciu DeleteFile, ktorá sa nachádza v knižnici Windows Kernel32.dll. Deklarácia potom vyzerá takto:

Private Declare Function DeleteFile Lib "kernel32" Alias "DeleteFileA" (ByVal lpFileName As String) As Long

Funkcia sa použije takto:

DeleteFile ("D:\useky.txt")

Alebo môžete chcieť zistiť či nebeží pri spustení vášho programu nejaká aplikácia (napríklad kalkulátor) a ak áno potrebujete ju zavrieť. Môžete použiť funkciu FindWindow, ktorá nájde danú aplikáciu na základe jej titulku v okne a potom funkciu PostMessage, ktorá jej pošle správu, že sa má zavrieť.

```
Private Declare Function FindWindow Lib "C:\WINDOWS\system32\user32" Alias
"FindWindowA" (ByVal lpClassName As String, ByVal lpWindowName As String) As Long
Private Declare Function PostMessage Lib "C:\WINDOWS\system32\user32" Alias
"PostMessageA" (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long,
lParam As Any) As Long
Const WM_CLOSE = &H10
```

Pri otváraní okna vašej aplikácie sa aktivuje udalosť inicialize, ktorú obslúžite nasledujúcim kódom:

```
Private Sub UserForm_Initialize()
Dim winHwnd As Long
Dim RetVal As Long
winHwnd = FindWindow(vbNullString, "Calculator")
If winHwnd <> 0 Then
PostMessage winHwnd, WM_CLOSE, 0&, 0&
Else
MsgBox "Kalkulátor nie je otvorený"
End If

End Sub
```

Nevýhodou je, že musíte poznať presný titulok okna.

Kľúčom k používaniu API funkcií Windows je teda ich deklarácia. Deklarácie môžete zistiť pomocou aplikácie APIviewer, ktorý je súčasťou inštalácie samostatného Visual Basicu. Jedným z najlepších zdrojov o Windows API funkciách pre Visual Basic je internetová stránka *http://www.allapi.net*.

Na záver ešte uvedieme funkciu pre otváranie súborov, ktorá je trochu zložitejšia, avšak takmer v každej aplikácii potrebná – keby ste teda aj niečomu nerozumeli, stačí si tento príklad prekopírovať do vášho programu. Postavená je na využívaní API funkcií Windows, keďže VBA nemá priamy prístup k tzv. CommonDialog komponentu (ako napríklad Visual Basic 6.0).

```
Private Declare Function GetOpenFileName Lib "comdlg32.dll" Alias _
"GetOpenFileNameA" (pOpenfilename As OPENFILENAME) As Long
Private Type OPENFILENAME
lStructSize As Long
hwndOwner As Long
hInstance As Long
lpstrFilter As String
lpstrCustomFilter As String
nMaxCustFilter As Long
nFilterIndex As Long
lpstrFile As String
nMaxFile As Long
lpstrFileTitle As String
nMaxFileTitle As Long
lpstrInitialDir As String
lpstrTitle As String
flags As Long
nFileOffset As Integer
nFileExtension As Integer
lpstrDefExt As String
lCustData As Long
lpfnHook As Long
lpTemplateName As String
End Type

Public Function ShowOpen(Filter As String, _
InitialDir As String, _
DialogTitle As String) As String
Dim OFName As OPENFILENAME
OFName.lStructSize = Len(OFName)
OFName.hwndOwner = 0
OFName.lpstrFilter = Filter
OFName.nMaxFile = 255
OFName.lpstrFile = Space(254)
OFName.lpstrFileTitle = Space$(254)
OFName.nMaxFileTitle = 255
OFName.lpstrInitialDir = InitialDir
OFName.lpstrTitle = DialogTitle
```

```
OFName.flags = 0
If GetOpenFileName(OFName) Then
ShowOpen = Trim(OFName.lpstrFile)
Else
ShowOpen = ""
End If
End Function
```

Teraz zadefinujte na UserForm príkazové tlačidlo a do neho nasledujúci kód:

```
Private Sub CommandButton1_Click()
Dim Filter As String
Dim InitialDir As String
Dim DialogTitle As String
Dim OutputStr As String

Filter = "Drawing Files (*.dwg)" + Chr$(0) + "*.dwg" + Chr$(0) + _
"All Files (*.*)" + Chr$(0) + "*.*" + Chr$(0)
InitialDir = "C:\Program Files\AutoCAD 2006\Sample"
DialogTitle = "Open a DWG file"
OutputStr = ShowOpen(Filter, InitialDir, DialogTitle)
MsgBox OutputStr
End Sub
```

Reťazec, ktorý vypíše MsgBox sa normálne použije na príkaz na otvorenie súboru, ktorý sme už preberali.

1.5 Vytváranie entít (objektov) AutoCADu – programovo riadené kreslenie

Vedomosti, ktoré boli vysvetľované doteraz, sa týkajú Visual Basicu všeobecne – možno ich využiť na vytváranie programov vo všetkých verziách VBA, teda nielen v tej verzii, ktorá je súčasťou AutoCADu, ale napr. aj v Exceli, Worde, ArcGis atď. Uvedené boli síce iba stručné, ale v princípe dostatočné vstupné informácie na vytváranie programov, ktoré *nevyužívajú* špecifické vlastnosti hostiteľskej aplikácie. To znamená, že s nimi môžete robiť v rámci AutoCADu napríklad ľubovoľné, prípadne aj dosť zložité výpočty, ale s týmito informáciami nemôžete vyvolať z vnútra AutoCADu napríklad príkaz na kreslenie úsečky, alebo iný príkaz AutoCADu. Nato je potrebné získať ešte vedomosti o tzv. objektovom modeli AutoCADu a o spôsobe ako ho používať.

1.5.1 *Objektový model – spojenie s AutoCADom*

Prostredníkom na využitie špecifických vlastností určitej profesionálnej programovej aplikácie v špecifických užívateľských programoch, sú tzv. objektové modely niektorých Windows programov. Nie každý program je týmto spôsobom postavený. Takto je naprogramovaný napr. Excel, Word, ACCESS, AutoCAD, Autodesk Civil 3D, ArcGis, CorelDraw. Vytvárajú sa aj samostatné ActiveX komponenty poskytujúce objektové modely, ktoré nemajú využitie ako samostatne fungujúce užívateľské programy, je však možné ich

využiť v kontexte určitého vytváraného programu (programátorskými nástrojmi ako je VBA ale aj iné), kde potom rieši nejakú špecifickú úlohu: prácu s grafikou, výpočet hydrauliky rúrovej siete, rôzne matematické výpočty – napr. optimalizačné atď. Toto má ten zmysel, že určité opakujúce sa alebo naopak veľmi špecifické úlohy vyžadujúce zvláštne vedomosti, nemusí programátor vždy riešiť a preberá potrebné funkcie z takýchto objektových modelov – tzv. ActiveX komponentov.

Objektovosť znamená, že aplikácia exponuje (poskytuje) svoje jednotlivé súčasti – objekty –"vonkajšiemu prostrediu (Windows)" v podobe tohto modelu, takže sú prístupné a je možné s nimi manipulovať pomocou ich metód a vlastností akýmkoľvek tzv. Automation controllerom, čo je napríklad aj VBA. Manipulovať s objektom AutoCADu úsečka znamená nakresliť jej konkrétnu inštanciu, zmeniť jeden bod už existujúcej úsečky, zmeniť jej farbu alebo posunúť ju vo výkrese. Keď sme menili farbu alebo bod, menili sme **vlastnosť** objektu. **Metóda** je akcia, ktorú je objekt schopný absolvovať – úsečka sa vie premiestniť, keď na ňu aplikujeme metódu *Posun* (*Move*). Výsledkom viacerých takýchto akcií bude napríklad automaticky nakreslený výkres pozdĺžneho profilu na základe číselných údajov o jeho jednotlivých súčastiach, ktoré budeme zadávať ako vlastnosti.

Objekty sú teda časti aplikácie; v prípade AutoCADu:

- grafické objekty – čiary, oblúky, texty, kóty
- štýly čiar, textov, kót sú tiež objekty (konkrétne hodnoty nastavení – napr. font – je potom vlastnosť objektu štýl
- organizačné štruktúry ako hladiny, bloky, modelový priestor (opäť napr. farba hladiny je vlastnosť objektu hladina, vytvorenie hladiny vyžaduje aplikáciu príslušnej metódy)
- zobrazenie kresby (pohľad)
- dokument výkresu a AutoCAD samotný sú tiež objektami

Objekty sú usporiadané v hierarchickej štruktúre, v ktorej koreň alebo základ tvorí aplikácia AutoCADu ako taká. Zobrazenie tejto hierarchickej štruktúry je objektový model – pozri obr. 1.17. Objektový model zobrazuje, ktorý objekt sprostredkuje prístup k ďalšej hierarchicky nasledujúcej úrovni objektov. Prístup k objektom je potrebný na to aby sme vedeli zadefinovať, čo to je s čím chceme manipulovať pomocou vlastností a metód.

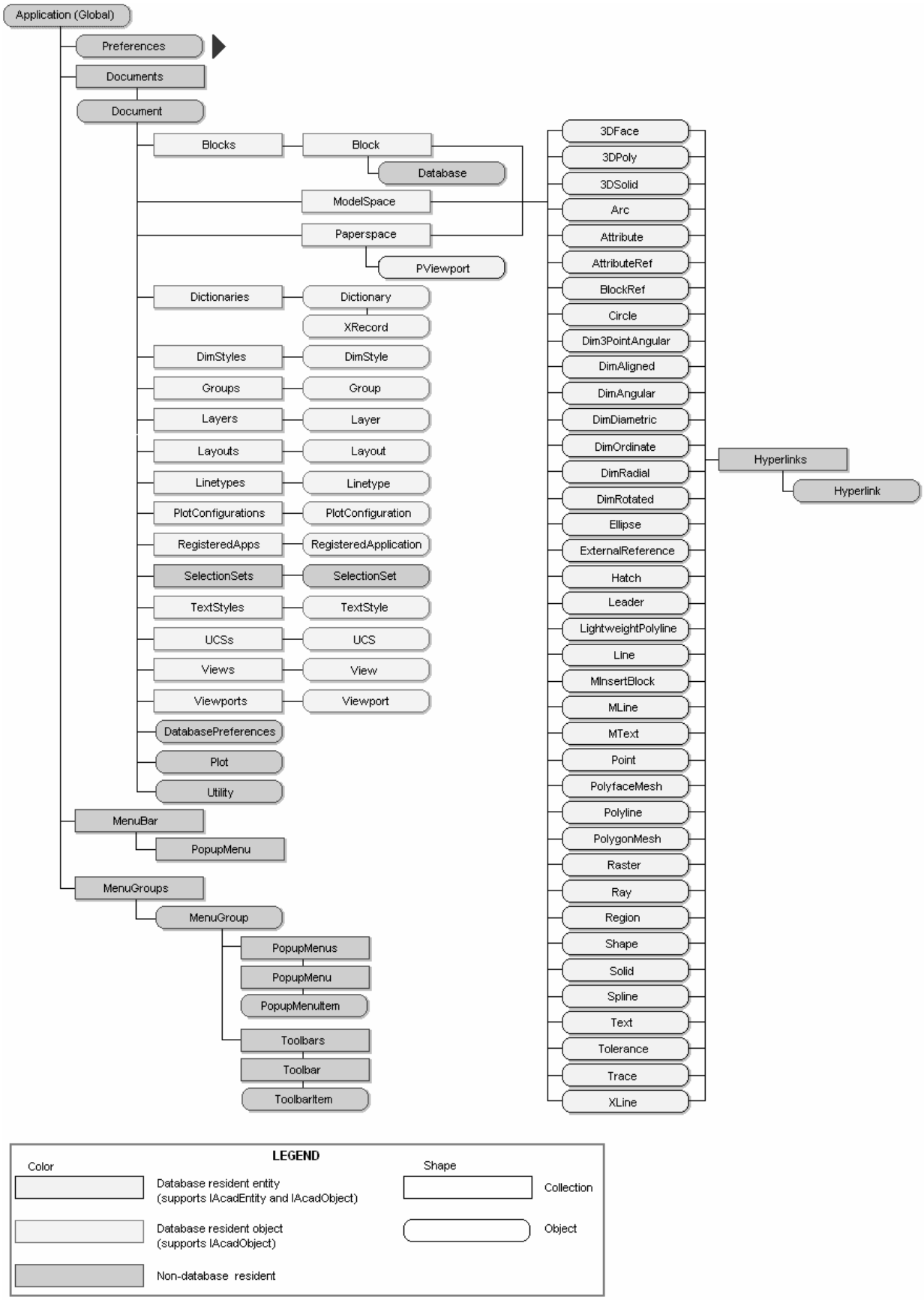
Pomocou Automation môžete vytvárať a manipulovať s objektami AutoCADu v rámci ľubovoľnej aplikácie, ktorá potom slúži ako nástroj Automation. Automation umožňuje vývoj makier medzi rôznymi aplikáciami, čo je schopnosť, ktorá nie je u iných programových rozhraní AutoCADu – napríklad AutoLISPu dostupná. Pomocou Automation možno kombinovať vlastnosti viacerých aplikácií do jednej aplikácie.

Ak chcete využívať objektový systém a AutoCAD ActiveX Automation efektívne, je potrebné byťdôkladne oboznámený so všetkými prvkami práce s AutoCADom ako takým – entitami, ich tvorbou, prepínačmi príkazov aj editáciou. Čím lepšie je programátor oboznámený s grafickými aj negrafickými objektmi AutoCADu, tým efektívnejšia bude aj jeho manipulácia s nimi prostredníctvom ActiveX Automation.

Základným, alebo koreňovým objektom objektového modelu, ktorý sprístupňuje všetky ostatné objekty, je samotná aplikácia AutoCADu. Aby sme v prípade volania z externého program mohli objektový model AutoCADu používať, bude potrebné pripojiť sa na objekt AutoCADu, resp. tento jeho koreňový objekt, pomocou tohto kódu:

```
Dim acad as Object
Set acad = GetObject(, "AutoCAD.Application")
```

Tento kód deklaruje v prvom kroku "acad" ako premennú typu Object. Potom nastavuje hodnotu tejto premennej pomocou príkazu Set ako "autocad.application", t. j. ako aplikáciu AutoCADu. Od tejto chvíle, kdekoľvek sa použije premenná "acad", je možné pomocou nej manipulovať s objektom AutoCADu ako takého, resp. získať prístup k ďalším objektom.



Obr. 1.17 Objektový model AutoCADu

Objekt aktívny dokument

Objekt aktívneho dokumentu má niekoľko užitočných metód a vlastností. V exernom programe (napr. VB) sa zadefinuje pomocou tohto kódu :

```
Dim acad as Object
Dim DOC as Object
Set acad = GetObject(, "AutoCAD.Application")
Set DOC = acad.ActiveDocument
```

"DOC" bude názov premennej pre ActiveDocument, ktorý bude používaný v niektorých nasledujúcich príkladoch. Všimnite si inú možnosť aktivácie aktuálnej kresby pomocou ThisDrawing v tu uvádzaných príkladoch, v ktprých nepredpokladáme volanie AutoCADu zvonka, ale využívanie jeho vlastného VBA.

Ďalej uvedieme niekoľko metód s týmto objektom.

Export
Metóda pre export aktívneho dokumentu.
Export(MenoSuboru, Pripona, Selekcia)

Menosuboru a Pripona sú reťazce, Selekcia je objekt. Meno súboru definuje cestu kde uložíme súbor aj s jeho menom, Prípona je niektorý trojznakový reťazec z WMF, SAT, EPS, DXF alebo BMP. Pre WMF, SAT a BMP formáty, selekcia špecifikuje objecty, ktoré majú byť exportované. Pre EPS a DXF format je tento údaj ignorovaný a exportuje sa celá kresba.
Použitie:

```
Dim FName as String
Dim FExt as String
Dim NewSS as Object
FName = "c:\novydxf"
FExt = "DXF"
Set NewSS = DOC.SelectionSets.Add("AVB")
ThisDrawing.Export FName, FExt, NewSS
```

Tento kód exportuje všetky entity kresby do súboru novydxf.

Import
Metóda pre importovanie súboru do AutoCADu.
Import(MenoSub, Bod, Mierka)

MenoSub je string udávajúci cestu k vkladanému súboru, Bod je trojrozmerné pole čísiel typu double reprezentujúce bod vloženia, Mierka je číslo typu double definujúce zväčšenie alebo zmenšenie vkladaného súboru. S týmto príkazom je možné importovať SAT, EPS, DXF, alebo BMP formát súborov.

Použitie:

```
Dim MenoSub As String
Dim Bod(0 To 2) As Double
Dim Meritko As Double
MenoSub = "c:\Subor.dxf"
Bod (0) = 0#
```

```
Bod (1) = 0#
Bod (2) = 0#
Meritko = 1.5
ThisDrawing.Import MenoSub, Bod, Meritko
```

Podobne môžeme používať v rôznych súvislostiach ďalšie nižšie uvedené príkazy – metódy, ktorých zmysel je jasný podľa názvu, pričom v prvom prípade je v zátvorke cesta k šablóne výkresu (asi bude iná) a v ďalších dvoch prípadoch je meno súboru:

```
ThisDrawing.New ("c:\acad.dwt")
ThisDrawing.Open ("C:\test.dwg")
ThisDrawing.SaveAs ("C:\test.dwg")
ThisDrawing.Close([UložZmeny][, FileName])
```

Ukážme si použitie posledného príkazu. V AutoCADe chýba funkcia *Návrat k uloženej verzii*, ktorá sa môže hodiť pokiaľ pri rozpracovanom uloženom výkrese urobíte nejakú chybu a vrátenie sa pomocou Undo by bolo zdĺhavé. Pomocou Alt-F11 spustite editor VBA a zadajte tento jednoduchý kód – ide iba o zistenie mena výkresu, jeho zavretie bez uloženia a následné otvorenie:

```
Sub Navrat()
 MenoVykresu = ThisDrawing.FullName
 ThisDrawing.Close (False)    'Zmeny vo výkrese sa neuložia
 ThisDrawing.Application.Documents.Open MenoVykresu
End Sub
```

Pozor, kód bude pracovať iba pri nastavení systémovej premennej SDI na 0.

Tento kód si môžeme uložiť napíklad do .DVB súboru Navrat.DVB. Aby bola funkcia k dispozícii vždy, pridajme súbor NAVRAT.DVB do sady *"Při spuštění" (Startup Suite)* – pozri príkaz *APLČTI (_APPLOAD)* alebo z menu *Nástroje – Načíst aplikaci*.

Teraz už iba musíme vyvolať novo vytvorenú funkciu z menu alebo nástrojového panelu. Vytváranie nových nástrojov v nástrojových paneloch je identické s tým, čo je uvedené v kapitole o makrách. Ako makro dopíšeme v paneli CUI:

```
_–VBARUN Navrat
```

alebo

```
(command "_VBARUN" "Navrat")
```

1.5.2 *Kolekcie objektov a práca s hladinami*

Dôležitý prvok objektovo orientovaného programovania, ktorý by ste mali poznať, je kolekcia. Ako naznačuje názov, kolekcia je skupina objektov rovnakého typu. Objekty typu *Collection* sú preddefinované objekty, které obsahujú (sú rodičovským objektom pre) súbor inštancií podobných objektov. V AutoCADe existujú mnohé objekty typu *Collection*

napríklad: dokumenty, hladiny, typy čiar, textové štýly, dimenzačné štýly, užívateľské súradné systémy, nástrojové panely, menu, pohľady atď.

Väčšina objektov Collection je prístupných z objektu Document. Objekt Document obsahuje vlastnosti pre každý objekt Collection. Nasledujúci príklad definuje premennú a nastaví ju na Layer collection aktuálneho výkresu:

```
Dim Hladiny as AcadLayers
Set Hladiny = ThisDrawing.Layers
```

Na pridanie nového člena do kolekcie sa použije metoda Add s parametrom. Príklad vytvorenia novej hladiny a jej pridanie do Layer collection:

```
Dim novaHladina as AcadLayer
Set novaHladina = ThisDrawing.Layers.Add("Nová hladina")
```

Alebo pomocou predchádzajúceho môžeme skrátiť na:

```
Set novaHladina = Hladiny.Add("Nová hladina")
```

Pri práci s objektmi kolekcie často potrebujeme vybrať určitý jej prvok a ďalej s ním manipulovať (napr. potrebujeme nájsť určitú hladinu a zmeniť jej farbu, alebo nastaviť ju ako aktuálnu). Výber prvku sa uskutoční pomocou metódy Item. Metóda Item potrebuje pre svoju činnosť použitie identifikátoru pre špecifikáciu hľadanej položky kolekcie. Ako identifikátor sa používa:

- index špecifikujúci umiestnenie položky v súbore,
- reťazec reprezentujúci meno položky.

Metóda Item je defaultná metóda pre kolekcie. Ak sa nešpecifikuje meno metódy pri referencii kolekcie, predpokladá sa Item. Preto sú aj nasledovné dva príkazy ekvivalentné:

```
ThisDrawing.Layers.Item("ABC")
ThisDrawing.Layers("ABC")
```

Ak ste už nejaké programy vytvorili, pravdepodobne ste narazili na slučku For…Next, ktorá umožňuje niekoľkokrát opakovať množinu inštrukcií použitím podobnej sekvencie ako je nasledovná. Príklad demonštruje prechádzanie kolekcie Layer a zobrazenie mien všetkých hladín:

```
Sub PrechadzenieHladin()
  ' prechadzanie cez kolekciu
  On Error Resume Next
  Dim I As Integer
  Dim msg As String
  msg = ""
  For I = 0 To ThisDrawing.Layers.count – 1          'indexovanie v kolekcie začína nulou
    msg = msg + ThisDrawing.Layers.Item(I).Name + vbCrLf
  Next
  MsgBox msg
```

```
End Sub
```

Tento kód obsahuje časť, ktorá zistiťuje počet hladín vo výkrese, ale existuje jednoduchší spôsob: použiť namiesto toho slučku For Each…Next. Slučka For Each…Next vyhľadá počet objektov v kolekcii, napríklad hladín a krokuje každý výskyt. Nasledujúci príklad ozrejmuje tento spôsob a jeho úlohou je vyhodnotiť, či zadaná hladina existuje.

```
Public Function HladinaExist(HladinaMeno As String) As Boolean
Dim Hladina As AcadLayer
For Each Hladina In ThisDrawing.Layers
If Hladina.Name = HladinaMeno  Then
HladinaExist = True
Exit Function
End If
Next Hladina
HladinaExist = False
End Function
```

Ešte elegantnejšie riešenie zistenia existencie určitého prvku v kolekcii ilustruje nasledovný príklad. Výraz *(Err.Number = 0)* vracia logickú hodnotu True alebo False podľa toho aký bol vrátený chybový kód v predchádzajúcom riadku:

```
Public Function HladinaExist(MenoHladiny As String) As Boolean
Dim hladina As AcadLayer
On Error Resume Next
Set hladina = ThisDrawing.Layers(MenoHladiny)
HladinaExist = (Err.Number = 0)
End Function
```

Príklad nájdenia hladiny s menom "NovyStav" a nastavenie vlastností pre túto hladinu:

```
Sub NajdenieHladiny ()
  ' použitie metódy Item pre nájdenie hladiny s menom " NovyStav"
  On Error Resume Next
  Dim NovaHladina As AcadLayer
  Set NovaHladina = ThisDrawing.Layers.Item("NovyStav")
  If Err <> 0 Then
    MsgBox "Hladina NovyStav neexistuje."
    Exit Sub
  End If
With NovaHladina
.Linetype = "CONTINUOUS"
.Lineweight = acLnWt030
.Color = acRed
End With
ThisDrawing.ActiveLayer = NovaHladina
End Sub
```

Vysvetlenie kódu: Prvý riadok kódu ošetruje výskyt chyby pri behu funkcie. Ďalej je definovaná premenná NovaHladina (typu AcadLayer). Ďalší riadok kódu sa pokúsi nastaviť premennú NovaHladina, aby ukazovala na hladinu s menom " NovyStav". Nasleduje zistenie či nedošlo k chybe (ak už hladina " NovyStav" vo výkresu existuje). V prípade neexistencie hladiny " NovyStav" vo výkrese, je zobrazená chybová správa "Hladina NovyStav neexistuje". Inak jej kód priradí červenú farbu a ďalšie vlastnosti a urobí túto hladinu aktívnou. Všimnite si použitie *With...End With* pri nastavovaní viacerých vlastností objektu NovaHladina.

Ďalší príklad preformátuje všetky názvy hladín tak, aby boli písané veľkými písmenami:

```
Sub Premenuj_UpperCase()
  Dim DwgLayer As AcadLayer
  For Each DwgLayer In ThisDrawing.Layers
    DwgLayer.Name = UCase(DwgLayer.Name)
  Next DwgLayer
End Sub
```

Hladinám je potrebné vedieť priradiť určité vlastnosti, napr. farbu, typ čiary, treba vedieť nastaviť aktívnu hladinu. Tieto činnosti ilustruje nasledujúci príklad.

```
Private Sub VlastnostiHladiny()
Dim kruznica As AcadCircle
Dim stred(0 To 2) As Double
Dim Radius As Double
Dim layerObj As AcadLayer
   Set layerObj = ThisDrawing.Layers.Add("ModraHladina")
   layerObj.color = acBlue
   ThisDrawing.ActiveLayer = layerObj
' nakreslenie kružnice v tejto hladine:
stred(0) = 2#
stred(1) = 0#
stred(2) = 0#
Radius = 2.4
Set kruznica = ThisDrawing.ModelSpace.AddCircle(stred, Radius)
kruznica.Update
End Sub
```

1.5.3 *Tvorba grafických objektov*

Modelový a papierový priestor (Model Space, Paper Space) slúžia ako kontajnery najdôležitejších objektov o ktoré pri práci s AutoCADom ide, t. j. obsahujú entity, ktoré sa v týchto priestoroch vytvárajú pomocou metód. Pri vytváraní výkresu sa väčšinou používa modelový priestor, preto sa budeme v ďalšom odkazovať hlavne na premennú MS, ktorú môžeme predpokladať ako deklarovanú v tomto tvare:

```
Dim MS as AcadModelSpace
Set MS = ThisDrawing.ModelSpace
```

Týchto metód je pomerne veľa, ako pre dvojrozmerné, tak aj pre trojrozmerné objekty (entity), preto bude potrebné používať pri ich aplikácii helpový systém AutoCADu. Každá metóda vyžaduje parametre v závislosti od druhu vytváranej entity v modelovom priestore – napríklad kružnica stred a polomer:

AddCircle(Center, Radius)

Použije sa takto:

```
Dim kruznica as Object
Dim stred(0 to 2) as Double
Dim Radius as Double
stred (0) = 2#
stred (1) = 0#
stred (2) = 0#
Radius = 2.4
‘ aplikácia metódy na modelový priestor:
Set kruznica = ThisDrawing.ModelSpace.AddCircle(stred, Radius)
kruznica.Update
```

Tento kód nakreslí kružnicu so stredom (2,0,0) a polomerom 2.4. Ďalšie entity vytvárame podobne, napríklad úsečku:

```
Sub Priklad_AddLine()
   Dim lineObj As AcadLine
   Dim startPoint(0 To 2) As Double
   Dim endPoint(0 To 2) As Double

   ' Zadefinujeme začiatočný a koncový bod úsečky
   startPoint(0) = 1#: startPoint(1) = 1#: startPoint(2) = 0#
   endPoint(0) = 5#: endPoint(1) = 5#: endPoint(2) = 0#

   ' Vytvoríme úsečku v modelovom priestore
   Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)
   ZoomAll

End Sub
```

Objekt typu polyline (krivka) môžeme vytvoriť týmto kódom:

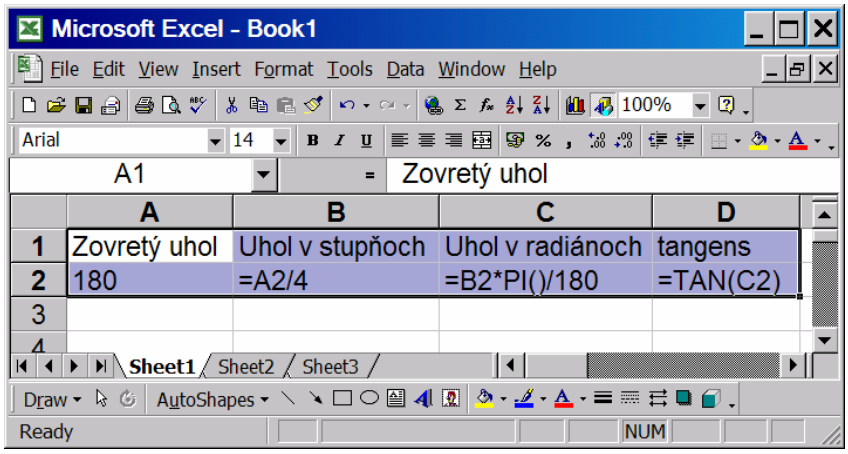
```
Sub Priklad_AddPolyline()
   Dim plineObj As AcadPolyline
   Dim points(0 To 14) As Double

   ' definujeme body pre 2-rozmernú krivku:
   points(0) = 1: points(1) = 1: points(2) = 0
   points(3) = 1: points(4) = 2: points(5) = 0
```

```
    points(6) = 2: points(7) = 2: points(8) = 0
    points(9) = 3: points(10) = 2: points(11) = 0
    points(12) = 4: points(13) = 4: points(14) = 0

    ' vytvoríme krivku v modelovom priestore:
    Set plineObj = ThisDrawing.ModelSpace.AddPolyline(points)
    ZoomAll

End Sub
```

V AutoCADe existuje aj tzv. odľahčená krivka, používaná v 2D. V ďalšom príklade je ukázané ako vytvoriť takúto krivku a následne ako zmeniť niektorý jej segment na oblúkový.

```
Private Sub CommandButton1_Click()
    Dim LahkaKrivka As AcadLWPolyline
    Dim BodyKrivky(0 To 11) As Double

    ' Definícia 2D krivky
    BodyKrivky(0) = 1: BodyKrivky(1) = 1
    BodyKrivky(2) = 1: BodyKrivky(3) = 2
    BodyKrivky(4) = 2: BodyKrivky(5) = 2
    BodyKrivky(6) = 3: BodyKrivky(7) = 2
    BodyKrivky(8) = 4: BodyKrivky(9) = 4
    BodyKrivky(10) = 4: BodyKrivky(11) = 1

    Set LahkaKrivka = ThisDrawing.ModelSpace.AddLightWeightPolyline(BodyKrivky)
    LahkaKrivka.Update
    ZoomExtents

    ' Zistenie zaoblenia (bulge) 3-tieho segmentu
    Dim currentBulge As Double
    currentBulge = LahkaKrivka.GetBulge(3)
    MsgBox "Zaoblenie tretieho segmentu " & LahkaKrivka.GetBulge(3), , ""

    ' Zmena zaoblenia (bulge) 3-tieho segmentu
    LahkaKrivka.SetBulge 3, –0.5
    LahkaKrivka.Update
    MsgBox "Nove zaoblenie je " & LahkaKrivka.GetBulge(3), , ""
End Sub
```

Hodnota bulge je tangens ¼ zovretého uhla oblúka. Záporné číslo znamená, že uhol oblúka je myslený v smere hodinových ručičiek. Bulge rovný 0 ponechá segment priamy, pri bulge = 1 pôjde o polkruh. Nie je to celkom "userfriendly" vymyslené, ale dá sa to pochopiť ak budete chvíľku experimentovať. Prakticky ide o to, že oblúkový segment je zadaný ako oblúk definovaný dvoma bodmi a zovretým uhlom. Na experimenty si môžete pomôcť aj s Excelom a napísať do neho nasledujúce vzorce, ktoré vám pomôžu pochopiť o čo ide:



Obr. 1.18 Matematický zmysel hodnoty "Bulge"

Text pridáme do výkresu takto:

```
Sub Priklad_AddText()
   Dim textObj As AcadText
   Dim textString As String
   Dim insertionPoint(0 To 2) As Double
   Dim vyska As Double

   ' definujeme obsah vloženého textu a bod vloženia
   textString = " Toto je text"
   insertionPoint(0) = 2: insertionPoint(1) = 2: insertionPoint(2) = 0
   vyska = 0.5

   ' vytvoríme textový objekt v modelovom priestore
   Set textObj = ThisDrawing.ModelSpace.AddText(textString, insertionPoint, vyska)
   ZoomAll

End Sub
```

Dôležité – ako vidíme pri tvorbe textu nie su k dispozícii niektoré možnosti ako v prípade obvyklého písania textu v AutoCADe: prvé čo je zrejmé je to, že nikde nemáme možnosť zadať uhol textu. Tieto ďalšie vlastnosti nastavujeme textu až následne po jeho vytvorení, zmenou vlastností. Špecifické vlastnosti textového objektu, ktoré môžeme po jeho vytvorení (pre)definovať sú:

Alignment	Špecifikuje zarovnanie textu (na stred, na pravý bod a podobne)
InsertionPoint	Umožňuje predefinovať bod vloženia textu

ObliqueAngle	Uhol zošikmenia textu
Rotation	Špecifikuje uhol textu v radiánoch
TextAlignmentPoint	Špecifikuje bod pre definíciu zarovnania textu
TextString	Umožňuje zmeniť obsah textu

Príklad na zarovnanie textu:

```
Private Sub UserForm_Click()
  ' Tento príklad vytvorí v modelovom priestore text a nastaví zarovnanie (alignment)

   Dim textObj As AcadText
   Dim textString As String
   Dim insertionPoint(0 To 2) As Double, alignmentPoint(0 To 2) As Double
   Dim VyskaTextu As Double
   Dim Bod As AcadPoint

   Me.Hide 'Skryjeme UserForm

   ' Definujeme text
   textString = "Pokusy s textom"
   insertionPoint(0) = 3: insertionPoint(1) = 3: insertionPoint(2) = 0
   alignmentPoint(0) = 3: alignmentPoint(1) = 3: alignmentPoint(2) = 0
   VyskaTextu = 0.5
   Set textObj = ThisDrawing.ModelSpace.AddText(textString, insertionPoint, VyskaTextu)

   Set Bod = ThisDrawing.ModelSpace.AddPoint(alignmentPoint)

   'zoom na text
   ThisDrawing.Application.ZoomExtents

   'Nasledovné dva príkazy nastavia zarovnanie na pravý bod,
   'všimnite si v AutoCADe, že text na bode 3,3 (alignmentPoint)končí
   textObj.Alignment = acAlignmentRight
   textObj.TextAlignmentPoint = alignmentPoint

   textObj.Update

End Sub
```

Otočenie textu je jednoduchšie, ak ho chceme otočiť okolo podobne zadaného bodu vloženia ako v predchádzajúcom príklade a o 45°, po jeho vytvorení napíšeme:

```
textObj.rotation insertionPoint, 3.14/4
```

Ďalšie informácie na kreslenie entít je potrebné vyhľadávať v helpovom systéme AutoCADu. Po vyvolaní helpu pre užívateľské úpravy AutoCADu zvolíme v ľavej časti referenčnú príručku ActiveX a VBA (ActiveX and VBA Reference). V nej vyhľadáme položku Object model. Pomocou tejto zobrazíme na obrazovke objektový model v grafickej forme ako to je na obr. 1.17. Grafické objekty modelového priestoru, kde zvyčajne kreslíme, sú v pravej časti obrázka (modrou). Ak nás zaujíma kružnica, klikneme na Circle. V stránke

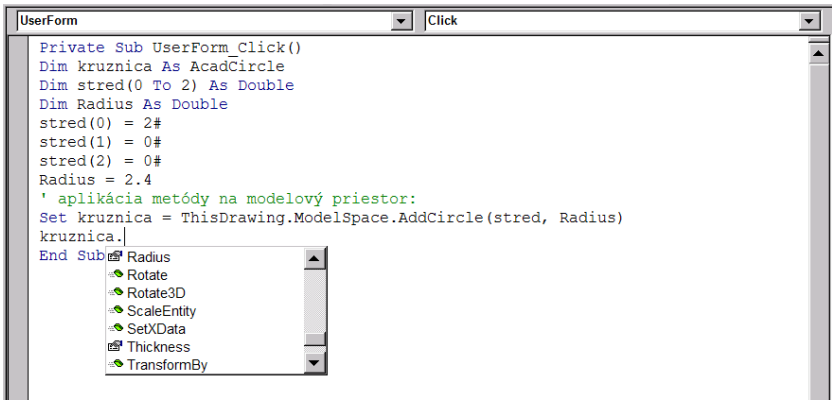
menu venujúcej sa tomuto objektu, ktorá sa objaví sú údaje o názve tohto objektu – AcadCircle – ktorý je vhodný používať pri jeho deklarácii:

Dim MojKruh As AcadCircle

Aj keď je možné používať aj všeobecnú deklaráciu Object:

Dim MojKruh As Object

Prvá deklarácia je výhodnejšia v tom, že pri napísaní názvu premennej v editori VBA a pokračovaním s bodkou, ktorá deklaruje, že budeme chcieť použiť, priadiť a pod. nejakú metódu alebo vlastnosť daného objektu, nám editor ponúka zoznam vlastností a metód dostupný práve pre objekt tohto typu. Napríklad, ak by sme chceli po vytvorení kružnice zmeniť jej polomer, napíšeme názov objektu, bodku a po objavení sa rozvinovacieho zoznamu napíšeme r, bude situácia v editori podľa obr. 1.19 – stačí potom kliknúť na Radius.



Obr. 1.19 Editor VBA

Ďalej sa v helpe nachádza opis spôsobu tvorby tohto objektu, t. j. pomocou metódy AddCircle v modelovom alebo papierovom priestore, prípadne v bloku. Ďalej nasleduje schematický obrázok pre vysvetlenie parametrov a informácia o tom, že už existujúcu kružnicu môžeme editovať, prípadne zisťovať jej vlastnosti pomocou nasledujúceho zoznamu vlastností a metód, ktoré sú ďalej vymenované a umožňujú odkazy do ďalších úrovní menu hypertextovým spôsobom. Po kliknutí na hypertextový odkaz (napr. na AddCircle) sa objaví ďalšia stránka menu, ktorá vysvetľuje príslušnú položku, v tomto prípade účel metódy AddCircle, jej syntax, popis parametrov. Dôležitý je odkaz na príklad (EXAMPLE), nakoľko mnohé veci môžeme z príkladov priamo kopírovať do svojich programov, prípadne nepatrne upraviť a tak programovať rýchlejšie a efektívnejšie.

Vytváraným objektom bude potrebné nastavovať vlastnosti ako je farba, typ čiary, hrúbka. Môžeme to vykonať kresbou v hladinách s príslušnými atribútmi, alebo vlastnosti priamo priraďovať jednotlivým objektom – podobne ako pri normálnej práci v AutoCADe. Pri popise práce s hladinami bol uvedený príklad na nastavenie farby hladiny, tu uvedieme príklad na nastavenie typu čiary jednotlivému objektu. V príklade najprv prebehne test, či žiadaný typ čiary už je vo výkrese načítaný, ak nie načíta ho zo súboru definícií čiar acad.lin, potom sa nakreslí úsečka a priradí sa jej tento typ čiary.

Private Sub UserForm_Click()

```
  Dim entry As AcadLineType
  Dim found As Boolean
  found = False

'Hľadá typ čiary Čárkovaná
  For Each entry In ThisDrawing.Linetypes
     If StrComp(entry.Name, "Čárkovaná", 1) = 0 Then
        found = True
        Exit For
     End If
  Next

 'Ak nenájde načíta ju do výkresu
  If Not (found) Then ThisDrawing.Linetypes.Load "Čárkovaná", "acad.lin"

  ' Vytvorenie čiary:
  Dim lineObj As AcadLine
  Dim startPoint(0 To 2) As Double
  Dim endPoint(0 To 2) As Double
  startPoint(0) = 1#: startPoint(1) = 1#: startPoint(2) = 0#
  endPoint(0) = 4#: endPoint(1) = 4#: endPoint(2) = 0#
  Set lineObj = ThisDrawing.ModelSpace.AddLine(startPoint, endPoint)

  ' Zmena typu čiary
  lineObj.Linetype = "Čárkovaná"
  ZoomAll

End Sub
```

Ak chceme nakresliť niekoľko objektov istým druhom čiary, je možné urobiť vyžadovaný typ čiary aktívny týmto príkazom:

```
ThisDrawing.ActiveLinetype = ThisDrawing.Linetypes.Item("Čárkovaná")
```

1.5.4 *Editácia grafických objektov*

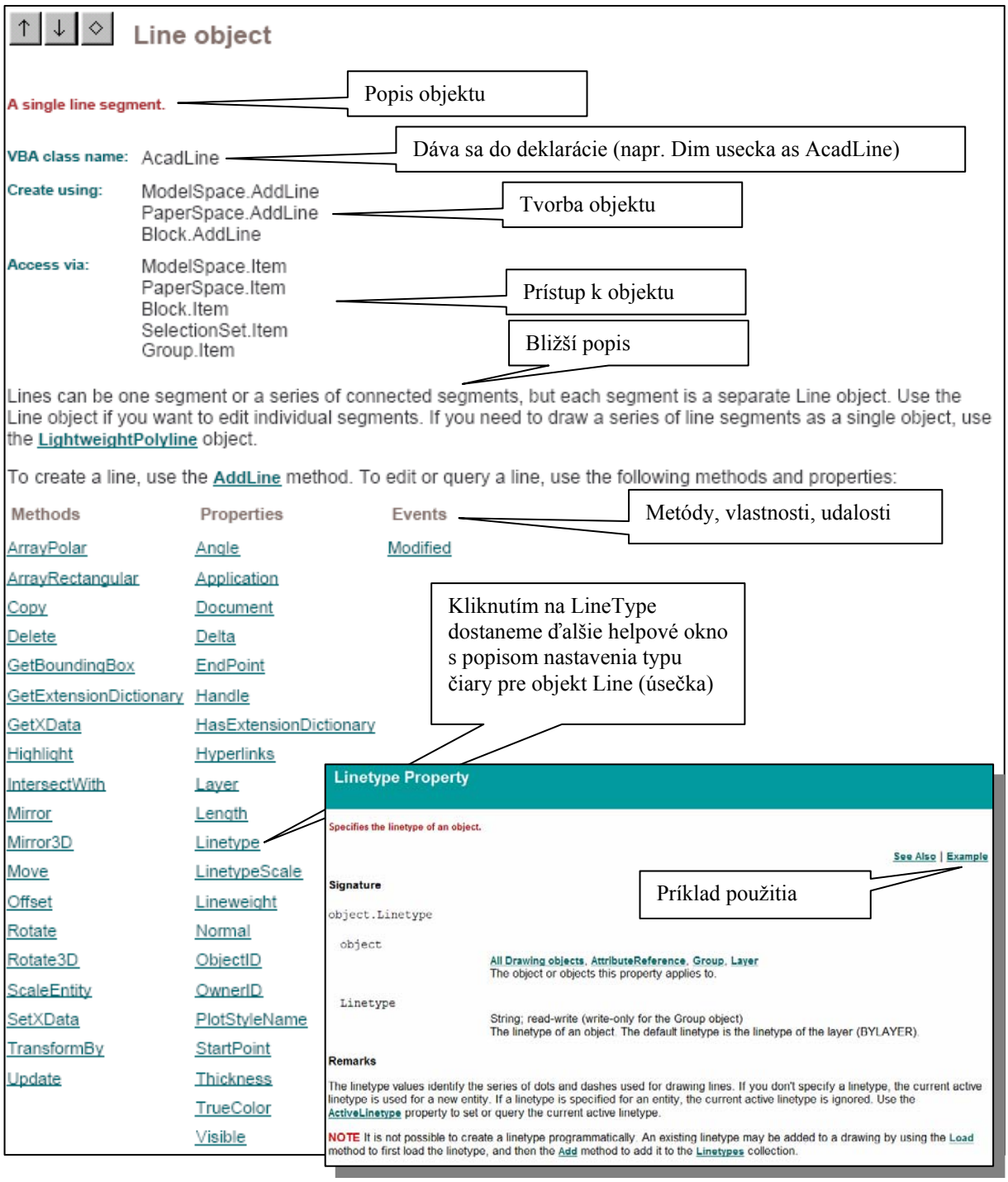
Entity ako každý iný objekt majú priradené vlastnosti (Properties) a metódy (Methods). Vlastnosti popisujú individuálne črty konkrétnej inštancie objektu. Metódy sú akcie prevedené nad objektom. Akonáhle je objekt vytvorený, môžete sa na neho informovať alebo ho editovať pomocou jeho vlastností a metód. Pomocou vlastností a metód teda editujete aj entity.

Napríklad Objekt Circle (kružnica) má vlastnosť Center (stred). Táto vlastnosť reprezentuje bod, ktorý je stredom tejto kružnice. Pri zmene stredu kružnice stačí jednoducho nastaviť stred na nové súradnice. Objekt Circle má tiež metódu Offset. Táto metóda vytvorí nový objekt v špecifickej vzdialenosti od existujúcej kružnice. Nasledujúci programový kód tieto možnosti bližšie vysvetľuje:

```
Sub Priklad_Editacia()

   Dim circObj As AcadCircle
   Dim currCenterPt(0 To 2) As Double
   Dim newCenterPt(0 To 2) As Double
   Dim radius As Double

   ' Definovanie stredového bodu pre kružnicu:
   currCenterPt(0) = 20: currCenterPt(1) = 30: currCenterPt(2) = 0
   radius = 3

   ' Vytvorenie kružnice:
   Set circObj = ThisDrawing.ModelSpace.AddCircle(currCenterPt, radius)
   ZoomAll
   MsgBox "Stred kružnice je " & currCenterPt(0) & ", " & currCenterPt(1) & ", " &
currCenterPt(2), vbInformation, "Príklad na editáciu"

 ' Zmena stredu kružnice:
   newCenterPt(0) = 25: newCenterPt(1) = 25: newCenterPt(2) = 0
   circObj.center = newCenterPt
   circObj.Update

   ' Zistenie novej pozície stredu:
   ' Všimnite si, že CenterPoint získame ako premennú typu Variant
   Dim centerPoint As Variant
   centerPoint = circObj.center
   MsgBox "Nový stred kružnice je " & centerPoint(0) & ", " & centerPoint(1) & ", " &
centerPoint(2), vbInformation, "Príklad na editáciu"
End Sub
```

Pri práci s vlastnosťami a metódami entít je potrebné využívať helpový systém AutoCADu, tu si iba ako príklad uveďme niektoré metódy objektu Circle (kružnica):

ArrayPolar	Vytvorí z objektu polárne pole
ArrayRectangular	Vytvorí z objektu obdĺžnikové pole
Copy	Kopírovanie objektu
Delete	Vymazanie objektu
GetBoundingBox	Získa 2 body ohraničujúceho obdĺžnika
IntersectWith	Nájde priesečník(y) s iným objektom
Mirror	Zrkadlenie
Move	Posúvanie
Offset	Ofset
Rotate	Otáčanie
ScaleEntity	Zväčšovanie/zmenšovanie objektu
Update	Prekreslenie objektu s aktuálne nastavenými vlastnosťami



Obr. 1.20 Helpový systém AutoCADu

Niektoré vlastnosti a metódy entít sú spoločné, napr. vlastnosti hladina (Layer), farba (Color), metódy ako napríklad premiestni (Move), vymaž (Erase), kopíruj (Copy), otoč (Rotate), obnov (Update) atď. Vytvorenie novej hladiny hladinaA červenej farby a kopírovanie kružnice do novej pozície sa uskutoční pomocou tohto kódu:

```
Sub Priklad_Editacie2()
    ' Vytvorenie novej hladiny hladinaA
```

```
Dim layerObj As AcadLayer
Set layerObj = ThisDrawing.Layers.Add("hladinaA ")
layerObj.Color = acRed

' Vytvorenie Kružnice
Dim circleObj As AcadCircle
Dim center(0 To 2) As Double
Dim radius As Double
center(0) = 2: center(1) = 2: center(2) = 0
radius = 0.5
Set circleObj = ThisDrawing.ModelSpace.AddCircle(center, radius)
ZoomAll

' Nastavenie hladiny pre kružnicu " hladinaA "
circleObj.Layer = " hladinaA "      '(hladinaA musi existovať)
' Obnovenie pohľadu
ThisDrawing.Regen (True)
MsgBox "Kružnica je v hladine  " & circleObj.Layer, , "Príklad"

' Definovanie bodov vektoru posunutia
Dim point1(0 To 2) As Double
Dim point2(0 To 2) As Double
point1(0) = 0: point1(1) = 0: point1(2) = 0
point2(0) = 2: point2(1) = 0: point2(2) = 0

' Premiestnenie kružnice
circleObj.Move point1, point2

ZoomAll
MsgBox "Posun kružnice vykonaný.", , "Príklad"

End Sub
```

Niekedy je potrebné vykonať editačnú operáciu na viacerých objektoch súčasne. V tom prípade musíme vytvoriť najprv výberovú množinu (SelectionSet), naplniť ju objektmi, ktoré nás zaujímajú, napr. výberom pomocou okna a potom aplikovať na všetky vybrané entity editačný príkaz, napríklad zmenu farby:

```
Sub Priklad_Editacia()

' Zadefinovanie výberovej množiny:
Dim ssetObj As AcadSelectionSet
Set ssetObj = ThisDrawing.SelectionSets.Add("StSET")


' Pridanie všetkých entít, ktoré ležia v okne (28,17,0) a
' (–3.3, –3.6,0) do výberovej množiny
Dim roh1(0 To 2) As Double
```

```
Dim roh2(0 To 2) As Double

roh1(0) = 28: roh1(1) = 30: roh1(2) = 0
roh2(0) = –3.3: roh2(1) = –3.6: roh2(2) = 0
ssetObj.Select acSelectionSetWindow, roh1, roh2

' Zmena farby entít vo výberovej množine:
For Each entity In ssetObj
entity.Color = acRed
entity.Update
Next

End Sub
```

Výraz acSelectionSetWindow je kľúčové slovo pre spôsob selekcie objektov. Ďalšie možnosti sú:

acSelectionSetCrossing:
Vyselektuje všetky entity, ktoré krížia hranicu okna so zadanými rohmi alebo sa nachádzajú v jeho vnútri

acSelectionSetPrevious:
Zvolí naposledy použitú selekciu.

acSelectionSetLast:
Vyselektuje naposledy vytvorenú entitu.

acSelectionSetAll:
Vyselektuje všetky entity.

V ďalšom príklade na výberové množiny vytvoríme funkciu ArcDump, ktorá ilustruje použitie filtrovaných výberových množín AutoCADu vo VBA a prácu s vlastnosťami entít zahrnutých do výberovej množiny. V príklade ide o selekciu entít typu "Arc (oblúk)" vo hladine "Steny". Uvedený príklad je možné jednoducho prispôsobiť aj pre podobné potreby a situácie, keď je treba vybrať a spracovať určité typy entít (objektov) výkresu.

```
Sub ArcDump()

 Dim oSS As AcadSelectionSet
 Dim oArc As AcadArc
 Dim iFilterCode(0) As Integer
 Dim vFilterValue(0) As Variant

 'Errorhandler (vymaže prípadne už existujúcu výberovú množinu)
 On Error Resume Next
 Application.ActiveDocument.SelectionSets("Obluky").Delete
 On Error GoTo 0
```

```
 'vytvor výberovú množinu nazvanú "Obluky":
 Set oSS = Application.ActiveDocument.SelectionSets.Add("Obluky")

'V nasledujucom zadavame vždy dvojicu hodnôt – kód pre filtrovanú vlastnosť a jej hodnotu
'Kódy nájdete v Helpe AutoCADu ako Group codes
 iFilterCode(0) = 0: vFilterValue(0) = "Arc"
 iFilterCode(1) = 8: vFilterValue(0) = "Steny"

 'odfiltrovat všetko okrem entít typu "Arc" v hladine "Steny"
 oSS.SelectOnScreen iFilterCode, vFilterValue
 If oSS.Count Then
   For Each oArc In oSS 'spracovať všetky entity výberovej množiny "Obluky"
     With oArc
       MsgBox "StartPoint: " & .StartPoint(0) & ", " & .StartPoint(1) & ", " & .StartPoint(2) &
vbCrLf & _
           "EndPoint  : " & .EndPoint(0) & ", " & .EndPoint(1) & ", " & .EndPoint(2)
     End With
   Next oArc
 End If

End Sub
```

Namiesto dialógového okna výpisu vlastností entity (MsgBox) je možné použiť ľubovoľnú užitočnejšiu funkciu na spracovanie entít výberovej množiny – výpisy do súboru, modifikácia vlastností, výpočty so získanými vlastnosťami atď. Môžeme si uviesť príklad, v ktorom bude jeho užívateľ vyzvaný na vyselektovanie ľubovoľného množstva úsečiek vo výkrese a program ku každej dopíše jej dĺžku. Zadefinujte si vo výkrese UserForm1 a funkcia sa spustí po kliknutí na ňu. Funkcia MidPnt (na konci príkladu) hľadá stred úsečky, ktorý bude vkladacím bodom pre text.

```
Private Sub UserForm_Click()
Dim selekcia As AcadSelectionSet
Dim dlzka As AcadText
Dim usecka As AcadLine
Set selekcia = ThisDrawing.SelectionSets.Add("USECKY")
Me.Hide                 'skryje UserForm pre umožnenie selekcie na obrazovke
selekcia.SelectOnScreen

For Each entity In selekcia
If TypeOf entity Is AcadLine Then
Set usecka = entity
Set dlzka = ThisDrawing.ModelSpace.AddText(Str(Round(entity.Length, 2)), MidPnt(usecka),
5)
End If
Next

ThisDrawing.SelectionSets("USECKY").Delete
End Sub
```

```
Public Function MidPnt(objLine As AcadLine) As Variant
Dim Bod1 As Variant
Dim Bod2 As Variant
Dim varMidPnt(0 To 2) As Double
Bod1 = objLine.StartPoint
Bod2 = objLine.EndPoint
varMidPnt(0) = (Bod1(0) + Bod2(0)) / 2
varMidPnt(1) = (Bod1(1) + Bod2(1)) / 2
varMidPnt(2) = (Bod1(2) + Bod2(2)) / 2
MidPnt = varMidPnt
End Function
```

Trochu zložitejším a dôležitým objektom v AutoCADe je krivka (polyline). Keď s ňou budeme chcieť manipulovať, môžeme použiť aj jej ďalšie vlastnosti než sú uvedené farba alebo hladina. V časti o tvorbe entít bola už pri popise vytvárania kriviek spomenutá napríklad vlastnosť bulge, určená k nastavovaniu niektorých častí krivky ako oblúkov.

Nasledujúci príklad zistí počet vrcholov krivky. *TypeName* je funkcia, ktorá vracia reťazec s názvom objektu, ktorý sme napríklad zadali v AutoCADe počas predchádzajúceho behu programu (tento objekt je argumentom ďalej uvedenej funkcie). Rozhodovacia štruktúra *Case* slúži na odlíšenie správania sa vytvorenej funkcie pre 2D a 3D krivky. Vlastnosť *Coordinates*, ako z jej názvu vyplýva, získa do premennej typu variant *ZoznamBodov* súradnice. Súradnice vytvoria v tejto premennej pole, ktorého indexovanie sa začína nulou.

```
Public Function PocetVrcholovKrivky(Krivka) As Integer
Dim ZoznamBodov As Variant
On Error Resume Next
Select Case TypeName(Krivka)
  Case "IAcadLWPolyline"
     ZoznamBodov = Krivka.Coordinates
     GetVertexCount = (UBound(ZoznamBodov) + 1) / 2
  Case "IAcadPolyline", "IAcad3DPolyline"
     ZoznamBodov = Krivka.Coordinates
     PocetVrcholovKrivky = (UBound(ZoznamBodov) + 1) / 3
End Select
End Function
```

V ďalšom kroku môžeme napríklad využiť túto funkciu na napísanie programu, ktorý bude fungovať tak, že užívateľ vyberie v AutoCADe krivky a program vypíše do súboru ExportSuradnic.txt súradnice bodov vybraných kriviek.

```
Public Sub ExportCoords()
 Dim AcSSet As AcadSelectionSet
 Dim pt As Variant
 On Local Error Resume Next
 If TypeName(ThisDrawing.SelectionSets("ExportCoords")) = "Nothing" Then
    ThisDrawing.SelectionSets.Add "ExportCoords"
 End If
```

```
 Set AcSSet = ThisDrawing.SelectionSets("ExportCoords")
 AcSSet.Clear
 AcSSet.SelectOnScreen
 Open "C:\ExportSuradnic.txt" For Output As #1
 If AcSSet.Count > 0 Then
    For X = 0 To AcSSet.Count – 1
       Set Object = AcSSet.Item(X)
            For i = 0 To PocetVrcholovKrivky(Object) – 1
              OutStr = ThisDrawing.Utility.RealToString(Object.Coordinate(i)(0),
acDefaultUnits, 3)
              OutStr = OutStr & " " &
ThisDrawing.Utility.RealToString(Object.Coordinate(i)(1), acDefaultUnits, 3)
              If TypeName(Object) = "IAcad3DPolyline" Then
                 OutStr = OutStr & " " &
ThisDrawing.Utility.RealToString(Object.Coordinate(i)(2), acDefaultUnits, 3)
              Else
                 OutStr = OutStr & " " & ThisDrawing.Utility.RealToString(Object.Elevation,
acDefaultUnits, 3)
              End If
              Print #1, OutStr
            Next
    Next
 End If
 Close
 AcSSet.Delete
End Sub
```

1.6 Spustenie príkazu AutoCADu z VBA

Niektoré funkcie AutoCADu, ktoré je možné využívať pri normálnom kreslení, nie sú dostupné prostredníctvom objektového modelu. Nenájdete napríklad kreslenie prstenca, hraničnú krivku a podobne. V tomto prípade možno z VBA poslať na príkazový riadok ľubovoľný textový reťazec, ktorý môže obsahovať aj názov príkazu AutoCADu a pomocou neho sa príkaz spustí. Využiť môžete príkaz *ThisDrawing.SendCommand*, alebo ďalej uvedenú funkciu API.

Ak si zadefinujete vo svojom programe (v module, nie v UserForm) nasledujúcu funkciu, môžete z vášho programu vo VBA poslať na príkazový riadok AutoCADu ľubovoľný príkaz. Funkcia používa pokročilejšie programátorské techniky (tzv. Windows API knižnicu, ale stačí si kód programu opísať tak ako je uvedený):

```
Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
   (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, _
   lParam As Any) As Long

Declare Function GetForegroundWindow Lib "user32" () As Long

Public Const WM_COPYDATA = &H4A
```

```
Type COPYDATASTRUCT
dwData As Long
cbData As Long
lpData As String
End Type

Public Sub SendToCommandPrompt(strMessage As String)
   Dim DataStruct As COPYDATASTRUCT
   DataStruct.dwData = 1
   DataStruct.lpData = strMessage
   DataStruct.cbData = Len(strMessage) + 2
   AppActivate ThisDrawing.Application.Caption
   SendMessage GetForegroundWindow, WM_COPYDATA, 0, DataStruct
End Sub
```

Napríklad spustenie príkazu ZOOM pomocou okna vykoná nasledujúca sekvencia príkazov:

```
Sub Okno()
 'vbCr zastupuje klávesu Enter, ZOOM je príkaz pre zoomovanie a
' volíme prepínač pre okno (Window):
SendToCommandPrompt "ZOOM" & vbCr & "Window" & vbCr

'Okno vyžaduje zadať súradnice dvoch bodov:
SendToCommandPrompt "10,10" & vbCr
SendToCommandPrompt "30,30" & vbCr

End Sub
```

1.7 Získanie vstupu od užívateľa programu

Bežnými prostriedkami Visual Basicu môžeme získať od užívateľa programu číselný alebo textový údaj, a to napríklad tak, že ho zapíše do TextBoxu. V prípade, že je potrebné aby užívateľ zadal súradnice určitého bodu z výkresu AutoCADu napríklad odkliknutím, môžeme použiť metódu GetPoint z objektu Utility:

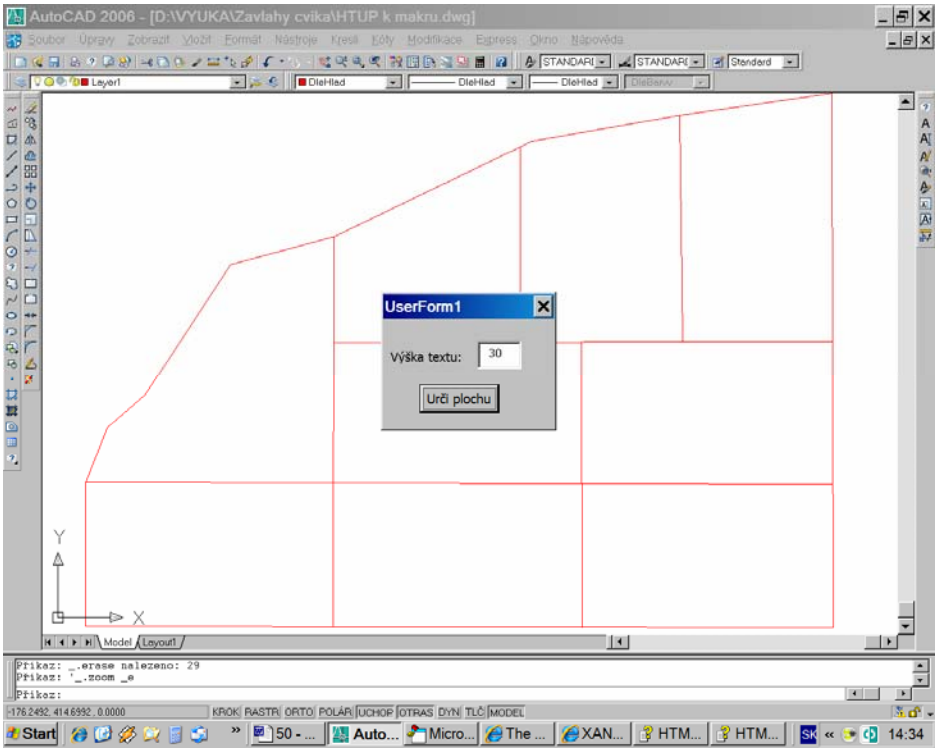
```
Sub Priklad_GetPoint()
Dim returnPnt As Variant      'Variant – premenná, korá. môže obsahovať ľuboovoľný typ
                              'údajov – v tomto prípade to bude 3-prvkové pole súradníc
   returnPnt = ThisDrawing.Utility.GetPoint(, "Zadaj Bod: ")
   MsgBox "Súradnice zadaného bodu sú:" _
   & returnPnt(0) & ", " & returnPnt(1) & ", " & returnPnt(2)
End Sub
```

Na tomto mieste si môžeme napísať aj funkciu pre odmeranie plochy uzavretej oblasti ohraničenej entitami AutoCADu. Použijeme príkaz *Hranice (Boundary)*, ktorý nemá

ekvivalentnú metódu vo VBA a preto využijeme metódu SendCommand, ktorá umožňuje zaslanie príkazu do príkazového okna AutoCADu.

Pri normálnej práci v AutoCADe by sme odmeranie plochy pomocou tohto príkazu vykonali nasledovne:

1. Nakreslíme nejaký útvar so známou plochou, ktorého hranica sa neskladá iba z jednej entity – napríklad kružnicu so známym polomerom a preseknutú napoli s úsečkou.
2. Príkazom *HKŘIVKA*, kde si zvolíme "vybrať body" a kliknutím do plochy tejto plochy vytvoríme jej hraničnú krivku.
3. Následne pomocou príkazu *plocha* (*area*) odmeriame plochu takto vzniknutého objektu.
4. Použitím príkazu *text* napíšeme do objektu/plochy jeho plochu.
5. Vymažeme pomocnú hraničnú krivku príkazom vymaž (erase).



Obr. 1.21 Program pre zistenie plochy a jej zápis do zvoleného miesta výkresu

Ak budeme robiť vo VBA vytvoríme si užívateľský formulár podľa obrázku, pričom v nasledujúcom kóde predpokladáme, že TextBox je TextBox1 a tlačidlo *Urči plochu* je CommandButton1:

```
Private Sub CommandButton1_Click()

Dim bod As String
Dim ssetObj As AcadSelectionSet
Dim returnPnt As Variant
Dim textObj As AcadText
```

```
Dim textString As String
Dim insertionPoint(0 To 2) As Double
Dim height As Double

'Errorhandler (vymaže prípadne už existujúcu výberovú množinu)
On Error Resume Next
Application.ActiveDocument.SelectionSets("plocha").Delete
On Error GoTo 0

UserForm1.Hide 'ukrytie formulára aby sa dal odkliknúť bod myšou

'Získanie bodu od užívateľa vo vnútri meranej plochy:
returnPnt = ThisDrawing.Utility.GetPoint(, "Zadaj Bod: ")
bod = Str(returnPnt(0)) & "," & Str(returnPnt(1))
bod = Replace(bod, " ", "")

'Spustenie príkazu pre vytvorenie hraničnej krivky:
ThisDrawing.SendCommand "_–Boundary" & vbCr & bod & vbCr & vbCr
‘(vbCr je Enter)

Set ssetObj = ThisDrawing.SelectionSets.Add("plocha")
ssetObj.Select acSelectionSetLast
For Each entity In ssetObj
plocha = entity.Area
entity.Delete
Next     ‘je síce iba jedna

plocha = Round(plocha, 0)
ssetObj.Delete

'Nadefinovanie vlastností textového objektu:
textString = Str(plocha)
insertionPoint(0) = returnPnt(0)
insertionPoint(1) = returnPnt(1)
insertionPoint(2) = 0
height = TextBox1.Value

'Napíš text s veľkosťou plochy:
Set textObj = ThisDrawing.ModelSpace.AddText(textString, insertionPoint, height)
textObj.Update

UserForm1.Show

End Sub
```

Môžeme tiež využiť funkciu GetEntity – výber entity pre rôzne ďalšie spracovanie v kóde. Nasledujúci príklad vyzve užívateľa programu k vyselektovaniu úsečky v AutoCADe a nastaví nitkový kríž v uhle, ktorý má táto úsečka.

```
Sub UholNK()
  Dim ZdrojovyObj As AcadObject
  Dim SelBod As Variant
  Dim NovyUhol As String

  On Error Resume Next
  With ThisDrawing.Utility
    .GetEntity ZdrojovyObj, SelBod, vbCr & "Vyber úsečku definujúcu uhol>> "
  End With
  If ZdrojovyObj.ObjectName = "AcDbLine" Then
    ThisDrawing.SetVariable "snapang", ZdrojovyObj.Angle
  Else: MsgBox "Nevhodný objekt"
    Exit Sub
  End If
End Sub
```

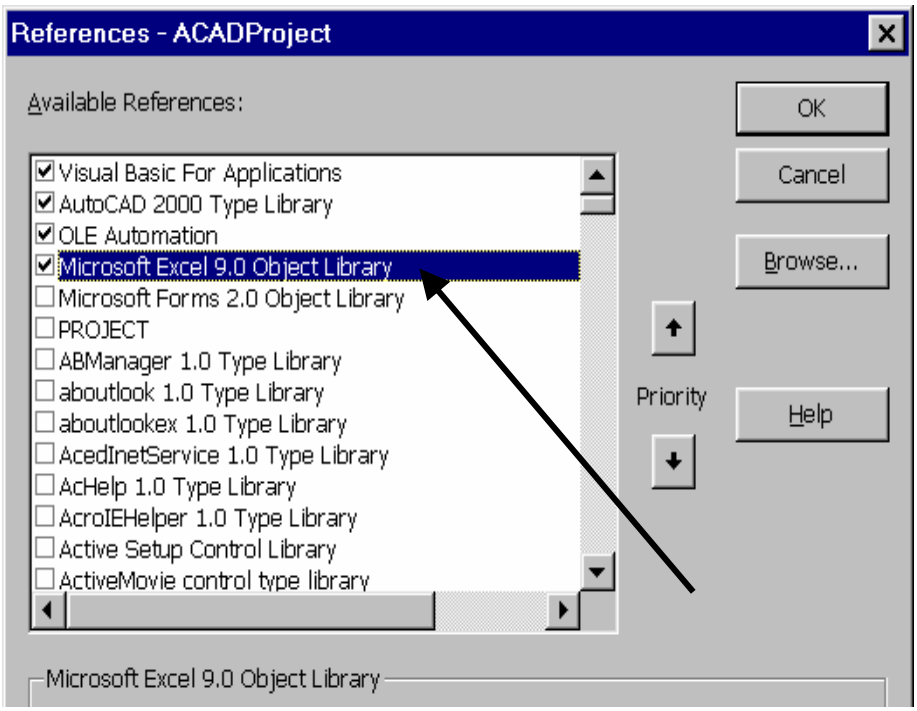
Ako je vidno z uvedených príkladov tieto funkcie prináležia rodičovskému objektu *Utility* v objektovom modeli. Objekt *Utility* umožňuje používať aj mnohé ďalšie funkcie na vstup údajov od užívateľa napr. uhlov (Get Angle, AngleFromXAxis), čísiel (GetReal, GetInteger), vzdialenosť (GetDistance), textových reťazcov (GetString) atď. Umožňuje tiež prístup k rôznym konverzným funkciám (napr. RealToString) – pozri helpové okno AutoCADu, ktoré sa objaví po kliknutí na objekt Utility v grafickom zobrazení objektového modelu AutoCADu.

1.8 Vykreslenie pozdĺžneho profilu líniovej stavby – spolupráca s Excelom

Technológia ActiveX a objektových modelov aplikácií zdieľaných prostrediu Windows je vlastnosťou mnohých ďalších programov. Prvou výhodou tejto skutočnosti je to, že čo sa naučíte v rámci VBA pre AutoCAD využijete aj pri iných programoch (napr. Excel, Word, ArcInfo, ArcView, CorelDraw, ACCESS). Výhodou využívania ActiveX je tiež možnosť kombinácie schopností viacerých programov do komplexného celku a tak znásobovanie ich schopností pre špecifických užívateľov.

Jednou z najbežnejších aplikácií Windows, s ktorou sa stretne technik je Excel. Jeho objektový model nebudeme preberať podrobne, nakoľko je pre túto problematiku dostupná literatúra v našom jazyku. Ukážeme si iba najbežnejšie úkony, a potom na príklade zjednodušeného vykresľovania pozdĺžneho profilu líniovej stavby (vodovod, závlaha, úprava toku) naznačíme možnosti prepojenia a súčasného využitia objektových modelov AutoCADu a Excelu v rámci jednoho užívateľského programu.

Ak chceme program vyvíjať v prostredí VBA AutoCADu a z neho volať Excel a využívať jeho možnosti, musíme si v rámci daného projektu VBA *referencovať* objektový model Excelu. Toto je úkon, ktorý otvorí možnosť používať iný objektový model, čiže možnosti ešte inej aplikácie než hostiteľského AutoCADu vo vašom programe VBA. Z menu Tools editora Visual Basicu si zvolíme príkaz References. Zobrazí sa okno podľa obr. 1.22, v ktorom zaškrtneme Microsoft Excel Object Library a dáme OK. Teraz ak poznáte syntax príslušných príkazov, môžete čítať údaje z Excelu – čo môžu byť samozrejme aj výsledky výpočtov, zapisovať údaje do buniek Excelu, využívať ho ako databázu pre výpis materiálu a podobne.



Obr. 1.22 Dialóg pre referencovanie Objektových modelov

Náš príklad s kreslením pozdĺžneho profilu predpokladá, že budeme mať na disku C v koreňovom adresári vytvorený súbor Excelu. V tomto súbore budú uložené údaje o pozdĺžnom profile, ktorý chceme vykresliť – teda predovšetkým súradnice vrcholov jednotlivých bodov pozdĺžneho profilu – staničenia a nadmorské výšky a niektoré pomocné údaje – mierky pre pozdĺžny profil. porovnávacia rovina a pod.

V prvom rade preto potrebujeme vedieť tento súbor za behu nášho programu otvoriť a prečítať z neho príslušné údaje do projektu VBA. Pomocou objektového modelu Excelu, ktorý sme si práve v svojom programe referencovali, môžeme samozrejme robiť mnoho ďalších a pokročilejších vecí, ktoré Excel dokáže, pre potreby týchto skrípt si však ukážeme iba tieto dva základné úkony.

V prvom rade musíme vytvoriť vo svojom projekte objekt Excelu ako takého – je to akýsi ekvivalent spustenia programu Excel, čo vykonávame napríklad pomocou ikonky na ploche, keby sme túto činnosť robili v normálnom užívateľskom režime. Vo VBA sa vykoná tento úkon zadaním nasledovnej deklarácie a príkazu v programe VBA AutoCADu:

Dim MojExcel As Excel.Application
Set MojExce1 = New Excel.Application

Excel sa nezobrazí na základe tohto príkazu viditeľne na obrazovke, bude však aktívny v pamäti, a preto ho musíme po skončení využívania pomocou svojho programu aj ukončiť:

MojExcel.Quit

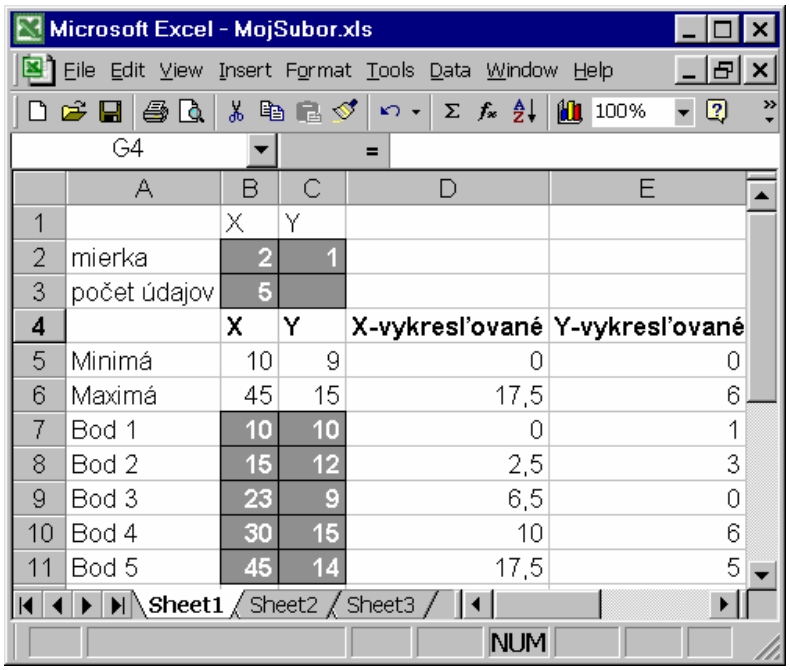
Pre naše potreby budeme chcieť otvoriť už existujúci súbor Excelu, v ktorom sú zapísané v predpokladanej forme údaje, ktoré budeme potrebovať pre vykreslenie pozdĺžneho profilu. Nasledujúci kód preto uvádza syntax pre otvorenie súboru "C:\MojSubor.xls", nastavenie sa na jeho prvý list (worksheet) a prečítanie údaju v 2 riadku a 3 stĺpci tohto listu (stĺpec "C") do premennej Cislo1. Ďalej je uvedená matematická operácia s týmto údajom, zápis výsledku do inej bunky daného listu excelovskej tabuľky a nakoniec uloženie takto upraveného súboru pod iným menom a uzavretie Excelu:

```
Dim MojExcel As Excel.Application
Dim ListUdajePozdlznyProfil As Worksheet
Dim MojWorkbook As Workbook

Set MojExce1 = New Excel.Application
Set MojWorkbook = MojExcel.Workbooks.Open("C:\MojSubor.xls")
Set ListExcelu = MojExcel.Worksheets(1)

Cislo1 = ListExcelu.Cells (2, 3)
Cislo2 = A+5
ListExcelu.Cells (2, 4) = Cislo2
MojWorkbook.SaveAs "C:\MojSubor1"
MojExcel.Quit
```

Toto je minimum, ktoré potrebujeme vedieť z práce s objektovým modelom Excelu pre potreby nášho zjednodušeného vykreslenia pozdĺžneho profilu. Teraz si popíšeme ako budú pripravené údaje v súbore Excelu.

Otvorte Excel a uložte prázdny súbor pod menom MojSubor na disk C do koreňového adresára. Môžte si samozrejme zvoliť akýkoľvek iný adresár, ale potom je potrebné vykonať zmenu na tom mieste programu, ktorý ideme písať, kde sa súbor vyvoláva.

Do súboru zapíšeme údaje podľa obr. 1.23.



	A	B	C	D	E
1		X	Y		
2	mierka	2	1		
3	počet údajov	5			
4		X	Y	X-vykresľované	Y-vykresľované
5	Minimá	10	9	0	0
6	Maximá	45	15	17,5	6
7	Bod 1	10	10	0	1
8	Bod 2	15	12	2,5	3
9	Bod 3	23	9	6,5	0
10	Bod 4	30	15	10	6
11	Bod 5	45	14	17,5	5

Obr. 1.23 Údaje pre vykresľovanie pozdĺžneho profilu zadávané v programe Excel

Predpokladáme, že všeobecné pojmy týkajúce sa vykresľovania pozdĺžnych profilov sú čitateľovi skript známe. Ako je z obrázku vidno, do buniek B2 a C2 sa zadajú mierky v smere osi X resp. Y. V bunke B3 je zapísané z koľkých bodov sa pozdĺžny profil skladá.

Všetky tieto aj nasledujúce údaje je potrebné zapisovať do tých buniek, ako je uvedené, nakoľko v týchto miestach ich bude očakávať vykresľovací program.

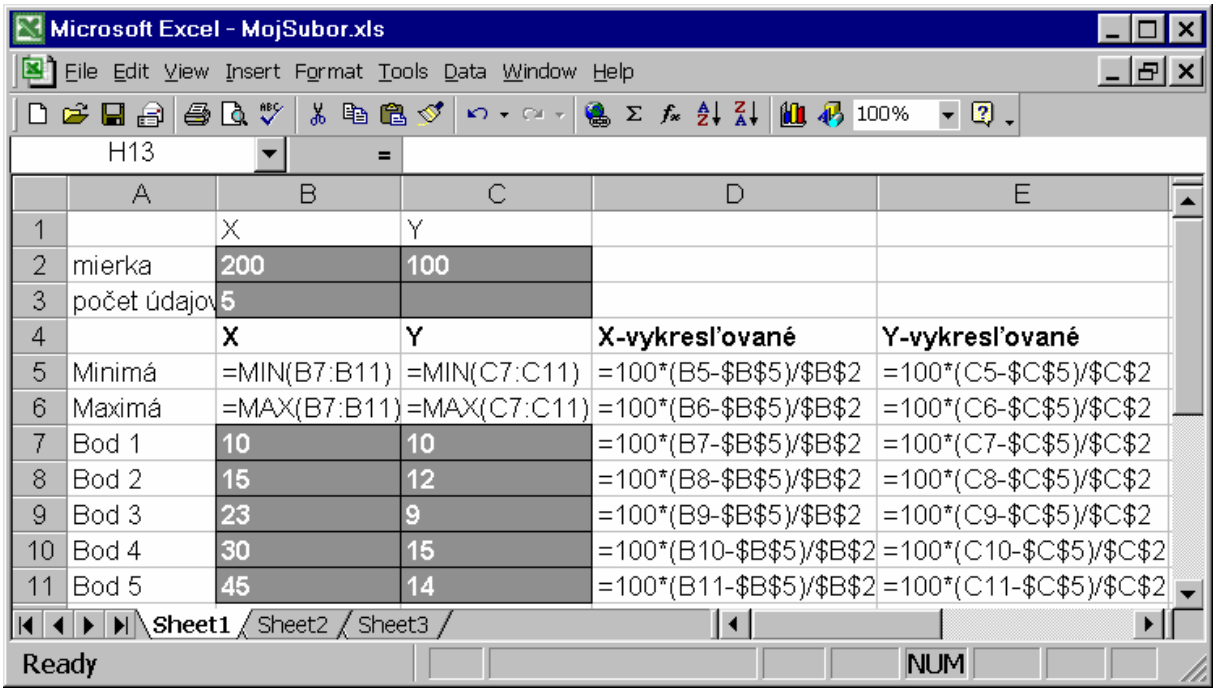
Riadok 4 obsahuje hlavičku vlastných zadávaných bodov uvedených nižšie. Okrem skutočných údajov zadávaných v metroch obsahuje hlavička aj kolónku X-vykresľované a Y-vykresľované, čo sú hodnoty súradníc prepočítané na potrebu samotného vykreslenia vzhľadom na zadanú mierku. Z toho dôvodu sú v stĺpci D a E vzorce, ktoré je potrebné v prípade väčšieho počtu údajov potiahnuť myšou nadol.

Vzorce, ktoré v týchto miestach nachádzajú sú:

X-vykresľované = $100^* (X - X_{min})/Mierka_x$

Y-vykresľované = $100 * (Y - Y_{min})/Mierka_y$

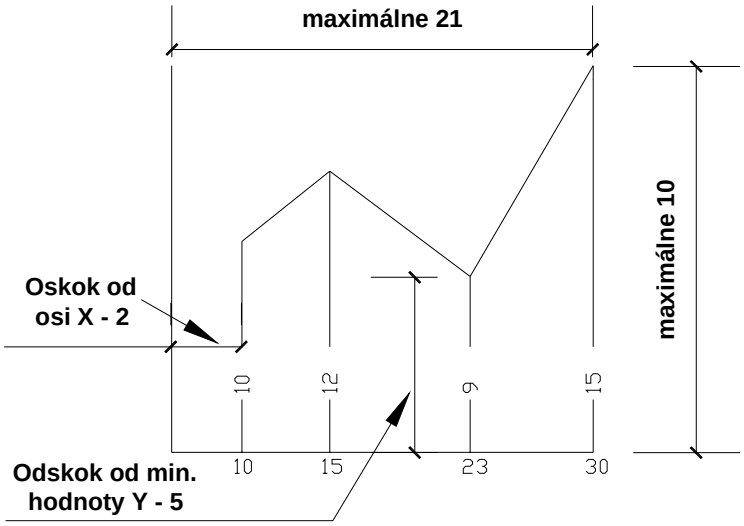
Okrem toho obsahuje tabuľka aj vzorce pre výpočet miním a maxím zo zadaných súradníc, ktoré sú potrebné pri vykresľovaní. Konkrétny tvar vzorcov je vidno z ďalšieho obrázku (obr. 1.24), na ktorom je obrazovka Excelu v modifikácii, keď zobrazuje vzorce (takáto voľba je dostupná z menu Tools ⇒ Options ⇒ View zaškrtnúť políčko Formulas (vzorce). V prípade zadávania iných údajov, ktorých je prípadne viac, je potrebné doplniť ďalšie vzorce pod riadkom 11 potiahnutím s myšou, aby sa vypočítali ďalšie hodnoty v stĺpcoch D a E.



	A	B	C	D	E
1		X	Y		
2	mierka	200	100		
3	počet údajov	5			
4		**X**	**Y**	**X-vykresľované**	**Y-vykresľované**
5	Minimá	=MIN(B7:B11)	=MIN(C7:C11)	=100*(B5-$B$5)/$B$2	=100*(C5-$C$5)/$C$2
6	Maximá	=MAX(B7:B11)	=MAX(C7:C11)	=100*(B6-$B$5)/$B$2	=100*(C6-$C$5)/$C$2
7	Bod 1	10	10	=100*(B7-$B$5)/$B$2	=100*(C7-$C$5)/$C$2
8	Bod 2	15	12	=100*(B8-$B$5)/$B$2	=100*(C8-$C$5)/$C$2
9	Bod 3	23	9	=100*(B9-$B$5)/$B$2	=100*(C9-$C$5)/$C$2
10	Bod 4	30	15	=100*(B10-$B$5)/$B$2	=100*(C10-$C$5)/$C$2
11	Bod 5	45	14	=100*(B11-$B$5)/$B$2	=100*(C11-$C$5)/$C$2

Obr. 1.24 Obrazovka Excelu zobrazujúca vzorce

Predpokladáme, že údaje sú zadávané v metroch a vykresľované na papier v centimetroch, a preto je pri iných údajoch než sú uvedené v tomto našom príklade (na obr. 1.23) potrebné zvoliť (inú) vhodnú mierku ich vykreslenia, podľa toho aký veľký chceme mať obrázok (resp. aký je k dispozícii formát papiera alebo tlačiarne). Tento príklad uvažuje s papierom A4 na šírku (obr. 1.25) , takže ak uvažujeme po všetkých stranách papiera aspoň 2 cm okraj a na odskok osí uvažujeme 2 cm od osi X a 5 cm od osi Y vzhľadom na popis zvislíc, nemal by byť maximálny kreslený rozmer v smere osi X väčší než 21 (podľa obr. 1.25 je to 17,5) a maximálny rozmer v smere osi Y 10 (podľa obr. 1.25 je to 6).

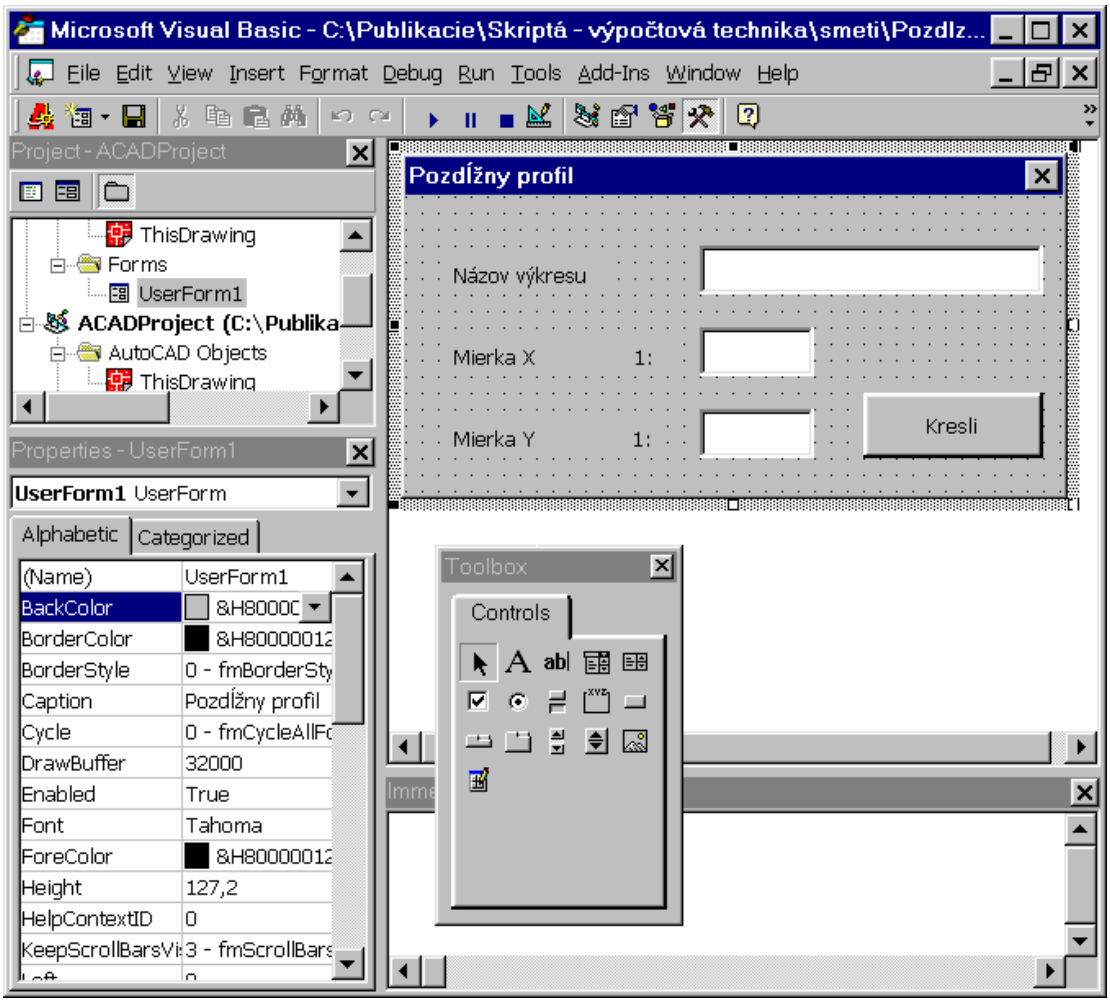


Obr. 1.25 Schéma pre nastavenie rozmerov pozdĺžneho profilu

Samotný program pre vykresľovanie pozdĺžneho profilu budeme písať v prostredí editora Visual Basicu v AutoCADe. Z menu Tools ⇒ Macro ⇒ Visual Basic Editor vyvoláme editor Visual Basicu. V programe budeme mať veľmi jednoduché užívateľské rozhranie, ktoré bude obsahovať iba tri textboxy pre zadanie údajov o názve pozdĺžneho profilu a o mierkach v smere osi X a Y.

Keďže budeme v programe využívať aj objektový model Excelu je potrebné referencovať ho z menu Tools ⇒ References, ako už bolo uvedené.

Vložme z menu Insert ⇒ UserForm do projektu nový užívateľský formulár UserForm1. Podľa obr. X3 umiestnime na formulár 3 popisy (Label) a k nim zodpovedajúce TextBoxy. V spodnej časti formuláru bude tlačidlo (CommandButton) na spustenie vykresľovania (obr. 1.26).



Obr. 1.26 Vytvorenie užívateľského rozhrania programu

Názov výkresu bude po spustení programu editovateľný údaj, ale údaje o mierkach by mal program iba prečítať zo súboru Excelu, keďže sme ich tam už nastavili vzhľadom na rozmery papiera. Na načítanie údajov z Excelu využijeme udalosť Initialize pre UserForm, ktorá sa automatický spúšťa vo chvíli, keď sa spustí program a v pamäti počítača sa UserForm inicializuje. Túto udalosť obslúžime potrebným kódom tak, že v editori Visual Basicu v okne kódu vyhľadáme v prvom hornom rozvinovacom zozname názov objektu –

UserForm a v druhom názov s ním spojenej udalosti – Initialize. V editori Visual basicu sa objaví prvý a posledný riadok procedúry, ktorá je vyvolaná touto udalosťou. Do tejto udalosti napíšeme kód, ktorý otvorí súbor Excelu, prečíta z neho potrebné údaje a zapíše ich do príslušných TextBoxov. Pred názov tejto procedúry je ešte potrebné napísať niektoré deklarácie premenných a výraz Option Explicit, ktorý znamená, že v programe je povinné deklarovať premenné (napíšeme už aj niektoré deklarácie pre premenné, ktoré budeme používať až neskôr:

```
Dim MojExcel As Excel.Application
Dim ListUdajePozdlznyProfil As Worksheet
Dim MojWorkbook As Workbook
Dim PocetBodov As Integer
Dim MinX, MaxX, MinY, MaxY As Double
Dim X, Y As Double
Dim i As Integer
Dim Body

Private Sub UserForm_Initialize()

'Otvorenie Excelu, súboru v ňom a nastavenie listu s údajmi:
Set MojExcel = New Excel.Application
Set MojWorkbook = MojExcel.Workbooks.Open("C:\MojSubor.xls")
Set ListUdajePozdlznyProfil = MojExcel.Worksheets(1)

'Prečítanie údajov o mierkach a ich zápis do TextBoxov:
TextBox2.Text = ListUdajePozdlznyProfil.Cells(2, 2)
TextBox3.Text = ListUdajePozdlznyProfil.Cells(2, 3)

End Sub
```

Ďalej je potrebné napísať procedúru pre vlastné vykreslenie pozdĺžneho profilu, ktorá sa spustí po stlačení tlačítka s popisom *Kresli*. Po dvojitom kliknutí na toto tlačidlo pri zobrazení grafického okna s užívateľským rozhraním ako je zobrazené na obr. 1.26 (určite ho nájdete v menu *View* editora *Visual Basicu*, ak ho práve nemáte na obrazovke) sa dostaneme priamo do okna kódu a do tela tejto procedúry.

Tu napíšeme nasledujúci kód:

```
Private Sub CommandButton1_Click()

KresliPopisPozdlznehoProfilu
KresliOsi
KresliKrivkuZExcelu
For i = 1 To PocetBodov
X = ListUdajePozdlznyProfil.Cells(i + 6, 4)
Y = ListUdajePozdlznyProfil.Cells(i + 6, 5)
KresliZvislicu
Next i
MojExcel.Quit
```

End Sub

V tomto kóde *KresliPopisPozdlznehoProfilu, KresliOsi, KresliKrivkuZExcelu, KresliZvislicu* znamená volanie procedúr, ktoré kreslia z názvu týchto procedúr zrejmé časti pozdĺžneho profilu. Vždy si je vhodné podobným spôsobom navrhnúť štruktúru procedúr, ktorá logicky rozdeľuje riešený problém. Okrem toho ako je z kódu vidieť procedúra *KresliZvislicu* sa v kóde opakuje toľkokrát, koľko program obsahuje bodov, a preto sa týmto spôsobom šetrí písanie toho istého kódu viackrát.

Hlavnou prácou je potom napísanie týchto procedúr. Túto časť už nebudeme uvádzať krok za krokom, ale ďalej je uvedený text celého programu, ktorý je potrebné preštudovať a zapísať do editoru Visual Basicu.

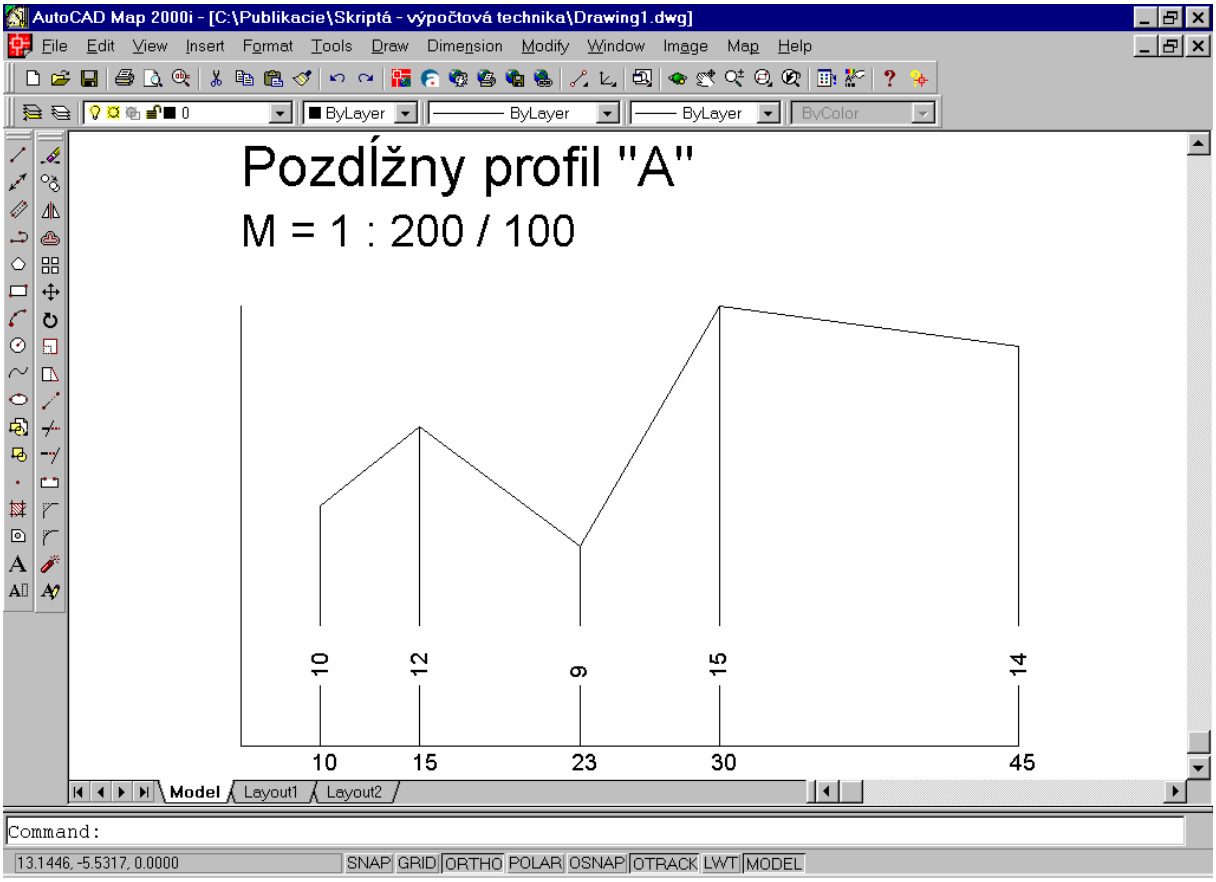
```
Option Explicit

Dim MojExcel As Excel.Application
Dim ListUdajePozdlznyProfil As Worksheet
Dim MojWorkbook As Workbook
Dim PocetBodov As Integer
Dim MinX, MaxX, MinY, MaxY As Double
Dim X, Y As Double
Dim i As Integer
Dim Body

Private Sub CommandButton1_Click()

KresliPopisPozdlznehoProfilu
KresliOsi
KresliKrivkuZExcelu
For i = 1 To PocetBodov
X = ListUdajePozdlznyProfil.Cells(i + 6, 4)
Y = ListUdajePozdlznyProfil.Cells(i + 6, 5)
KresliZvislicu
Next i
MojExcel.Quit
End Sub

Public Sub KresliPopisPozdlznehoProfilu()

Dim MojText As AcadText
Dim Bod(0 To 2) As Double

Bod(0) = –2
Bod(1) = 14
Set MojText = ThisDrawing.ModelSpace.AddText(TextBox1.Text, Bod, 1)
MojText.Update

Bod(1) = 12.5
Set MojText = ThisDrawing.ModelSpace.AddText _
```

```
("M = 1 : " & TextBox2.Text & " / " & TextBox3.Text, Bod, 0.8)
MojText.Update

End Sub

Public Sub KresliKrivkuZExcelu()

Dim Krivka As AcadLWPolyline
'Najprv zisti pocet vykreslovaných bodov
PocetBodov = ListUdajePozdlznyProfil.Cells(3, 2)
ReDim Body(0 To PocetBodov * 2 – 1) As Double

'Prečíta z Excelovskej tabuľky z 4 a 5 stĺpca počnúc
'8-mim riadkom v počte PocetBodov:
For i = 6 To 6 + PocetBodov – 1
      Body((i – 6) * 2) = ListUdajePozdlznyProfil.Cells(i + 1, 4)
      Body((i – 6) * 2 + 1) = ListUdajePozdlznyProfil.Cells(i + 1, 5)
Next
'Vykreslenie krivky:
Set Krivka = ThisDrawing.ModelSpace.AddLightWeightPolyline(Body)
Krivka.Update

End Sub

Public Sub KresliOsi()
'Vykresľuje os X a Y

Dim os As AcadLine
Dim Bod1(0 To 2) As Double
Dim Bod2(0 To 2) As Double

'Z EXCELu odčíta minimálne a maximálne údaje
MinX = ListUdajePozdlznyProfil.Cells(5, 4) – 2   'Posun z estetických dôvodov
MaxX = ListUdajePozdlznyProfil.Cells(6, 4)
MinY = ListUdajePozdlznyProfil.Cells(5, 5) – 5
MaxY = ListUdajePozdlznyProfil.Cells(6, 5)

'vykreslenie osi X:
Bod1(0) = MinX
Bod1(1) = MinY
Bod2(0) = MinX
Bod2(1) = MaxY
Set os = ThisDrawing.ModelSpace.AddLine(Bod1, Bod2)
os.Update

'vykreslenie osi Y:
Bod1(0) = MinX
Bod1(1) = MinY
```

```
Bod2(0) = MaxX
Bod2(1) = MinY
Set os = ThisDrawing.ModelSpace.AddLine(Bod1, Bod2)
os.Update
End Sub
Public Sub KresliZvislicu()
'Vykreslenie popisných zvislíc

Dim zvislica As Object
Dim Bod1(0 To 2) As Double
Dim Bod2(0 To 2) As Double
Dim Text As String

'prvá vertikálna čiarka dĺžky 1,5 (po vertikálny text)
Bod1(0) = X
Bod1(1) = MinY
Bod2(0) = X
Bod2(1) = MinY + 1.5

Set zvislica = ThisDrawing.ModelSpace.AddLine(Bod1, Bod2)
zvislica.Update

'Text s údajom o hodnote X (pri osi X)
Bod1(0) = Bod1(0) – 0.2
Bod1(1) = Bod1(1) – 0.6
Text = ListUdajePozdlznyProfil.Cells(i + 6, 2)
Set zvislica = ThisDrawing.ModelSpace.AddText(Text, Bod1, 0.4)

'Text s údajom o hodnote Y (vertikálny)
Bod2(0) = Bod2(0) + 0.2
Bod2(1) = Bod2(1) + 0.2
Text = ListUdajePozdlznyProfil.Cells(i + 6, 3)
Set zvislica = ThisDrawing.ModelSpace.AddText(Text, Bod2, 0.4)
zvislica.Rotate Bod2, 3.14 / 2
zvislica.Update

'druhá vertikálna čiarka dĺžky 1,5 (po krivku)
Bod1(0) = X
Bod1(1) = MinY + 3
Bod2(0) = X
Bod2(1) = Y
Set zvislica = ThisDrawing.ModelSpace.AddLine(Bod1, Bod2)
zvislica.Update

End Sub

Private Sub UserForm_Initialize()
```

'Otvorenie Excelu, súboru v ňom a nastavenie listu s údajmi:
Set MojExcel = New Excel.Application
Set MojWorkbook = MojExcel.Workbooks.Open("C:\MojSubor.xls")
Set ListUdajePozdlznyProfil = MojExcel.Worksheets(1)

'Prečítanie údajov o mierkach a ich zápis do TextBoxov:
TextBox2.Text = ListUdajePozdlznyProfil.Cells(2, 2)
TextBox3.Text = ListUdajePozdlznyProfil.Cells(2, 3)
End Sub



Obr. 1.27 Výsledný obrázok pozdĺžneho profilu

Keďže sme nepreberali nastavovanie fontov, je potrebné v prípade, že sa použije v okienku pre popis pozdĺžneho profilu interpunkcia, nastaviť v AutoCADe štýl písma s našimi (CZ) fontami. Všetky takéto veci je samozrejme možné ošetriť priamo z kódu a pri vykresľovaní zvážiť mnohé ďalšie veci, ktoré sme pri tejto zjednodušenej verzii pozdĺžneho profilu zanedbali – napríklad vykresľovanie viacerých položiek v rámci pozdĺžneho profilu ako je pri rúrových sieťach napr. terén, niveleta ap., detailnejší popis pozdĺžneho profilu v zvisliciach či na jeho ľavej strane, vykresľovanie rôznych častí pozdĺžneho profilu do rôznych hladín aby bolo možné navoliť farby a hrúbky čiar atď. Treba však povedať, že je možné aj kombinovať práce – niektoré veci robiť automaticky pomocou makra Visual Basicu a niečo dorábať manuálne normálnymi príkazmi AutoCADu. Pokiaľ zadáte všetko tak ako bolo uvedené v tomto príklade a spustíte program, dostanete obrázok podľa obr. 1.27.

2 MAKRÁ V DEFINÍCIÁCH VLASTNÝCH PRÍKAZOV

Od vzniku AutoCADu bol jeden z dôležitých dôrazov jeho tvorcov kladený na možnosť jeho užívateľského prispôsobenia. Okrem viacerých programových rozhraní, z ktorých v rámci týchto skrípt podrobnejšie rozpisujeme ActiveX® Automation a VBA (Visual Basic® for Applications), poskytuje ešte na tvorbu jednoduchších vlastných príkazov makro jazyk, ktorého zásady bližšie popíšeme v tejto kapitole. Makro definuje akciu, ku ktorej dôjde pri výbere určitého prvku užívateľského rozhrania (napríklad ikony z palety nástrojov alebo položky menu) a ktorá vykoná určitú úlohu – napríklad vykreslenie nejakého zložitejšieho objektu, pričom táto úloha by inak vyžadovala viacero akcií od užívateľa. Makro môže obsahovať príkazy AutoCADu, niektoré špeciálne znaky a vo svojej najdokonalejšej podobe aj programový kód v jazyku DIESEL (Direct Interpretively Evaluated String Expression Language).

Využitie makier je aktuálne z viacerých dôvodov. Jednak pre skutočnosť, že niektoré jednoduché, ale efektívne úpravy sa urobia týmto spôsobom jednoduchšie ako pomocou zložitejších jazykov (C, VBA a pod.). Ďalej je dôležité, že Visual Basic® for Applications nie je dostupný v odľahčenej verzii AutoCADu – v AutoCADe LT. Makrá popisované v tejto kapitole budú fungovať aj v tomto systéme, ktorý je pre mnohé firmy nevyužívajúce napríklad 3D v AutoCADe cenovo ďaleko prijateľnejší než plný AutoCAD, a pritom je úplne dostačujúci a plne kompatibilný.

Makrá, ako bolo uvedené, možno spúšťať rôznym spôsobom. V týchto skriptách budeme vysvetľovať možnosť pripájať makrá k novodefinovaným tlačidlám na paneloch nástrojov, a preto na úvod stručne preberieme prácu s prispôsobovaním nástrojových panelov. To neznamená, že makrá nemožno využiť napríklad pri prispôsobovaní položiek ponúk (menu) AutoCADu alebo stavového riadku, avšak vzhľadom na rozsah skrípt sa sústredíme iba na nástrojové panely.

Do panelov nástrojov je možné pridať nové tlačidlá, ktorým priraďujeme buď existujúce príkazy AutoCADu alebo makrá. Môžeme tiež niektoré nepoužívané tlačidlá z panelu odstrániť (za účelom zväčšenia pracovnej plochy). V rámci takýchto úprav môžeme vytvoriť aj vlastné panely nástrojov, ktorým priradíme ľubovoľné príkazy podľa svojej potreby. Na prispôsobovanie máme preto tieto tri úrovne:

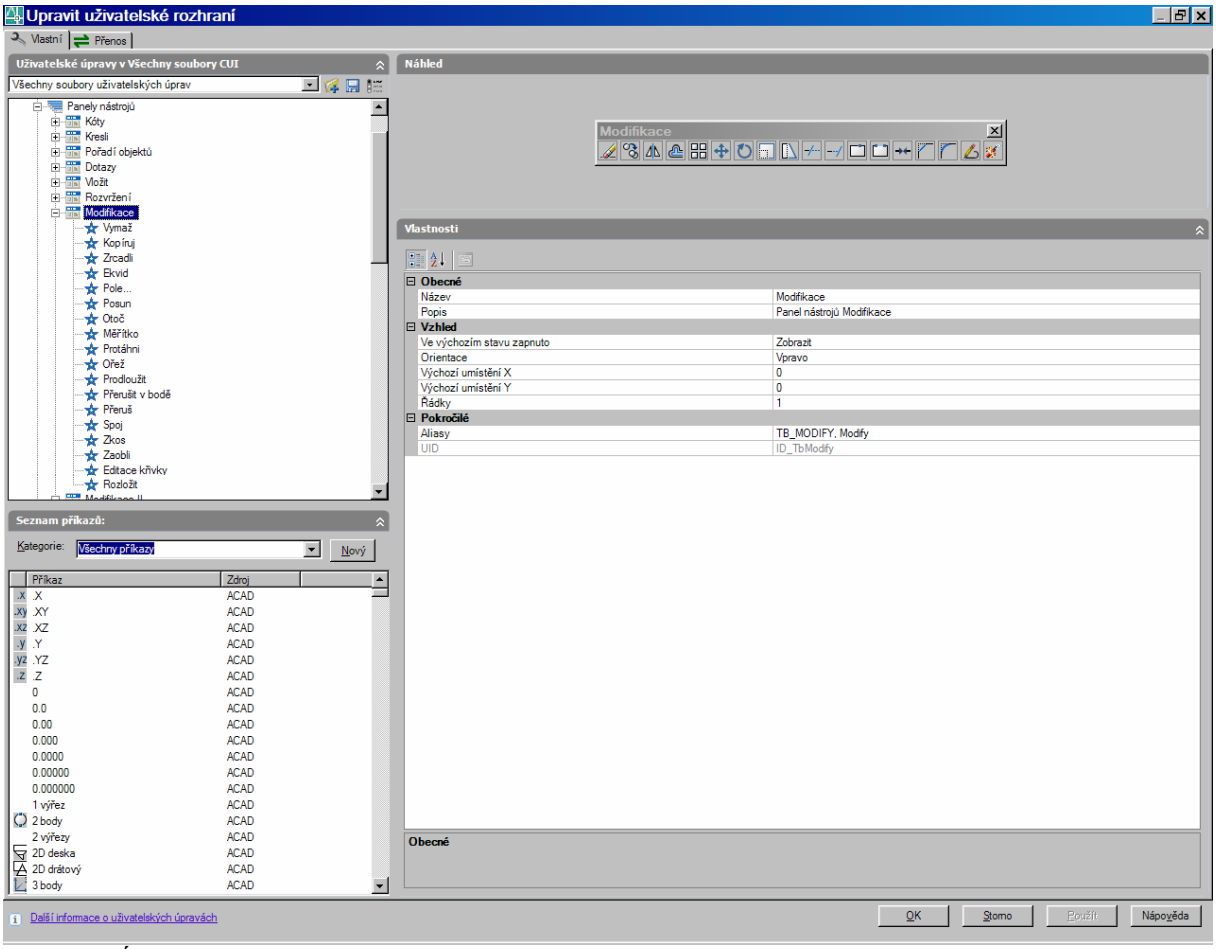
3. Prvý a najjednoduchší zásah do užívateľského prostredia AutoCADu spočíva vo výbere a umiestnení na obrazovku len vybraných nástrojových panelov (prípadne roletových menu) AutoCADu. Je to vcelku jednoduchá a nič v AutoCADe neovplyvňujúca úprava, ktorú bežne používa každý užívateľ AutoCADu. Vykonáva sa napr. kliknutím pravým tlačidlom myši na niektorý zo zobrazených panelov nástrojov, selekciou zo zobrazeného zoznamu a následným umiestnením novozobrazeného panelu nástrojov na pracovnej ploche.
4. Druhá jednoduchá úprava spočíva v úprave existujúcich nástrojových panelov AutoCADu (t. j. niektoré ikonky v paneloch premiestnime, vymažeme, alebo pridáme), alebo tvorba nových panelov so štandardnými príkazmi AutoCADu.
5. Tretia úprava spočíva v tvorbe vlastných nástrojových panelov s vlastnými príkazmi v prostredí AutoCADu.

Na úpravy užívateľského rozhrania je od verzie AutoCAD 2006 zavedený nový príkaz CUI (Customize User Interface), ktorý je možné vyvolať jeho menom z príkazového riadku, kliknutím pravým tlačidlom na niektorý zo zobrazených panelov nástrojov a výberom položky Vlastní (Custom) zo zobrazeného zoznamu alebo z menu

Nástroje→Vlastní→Rozhraní. Ďalšie podrobnosti o dialógovom paneli sú vcelku intuitívne pochopiteľné a dostatočne opísané v helpe AutoCADu.

Nový panel nástrojov vytvoríme na karte Vlastní CUI dialógového okna kliknutím pravým tlačidlom myši na položku Panely nástrojov v ľavom hornom okne *Uživatelské úpravy v...* Ďalej sa klikne na príkaz *Nový Panel nástrojov*, prepíše sa text Panel nástrojov 1(2..) novým názvom a aktualizujú sa údaje v okne *Vlastnosti dialógu CUI* (popis a niektoré východiskové hodnoty ako umiestnenie a pod.).

Pokiaľ by sme chceli umiestniť niektorý príkaz do tohto alebo niektorého iného z príkazových panelov, využijeme okno CUI vľavo dole so zoznamom príkazov, z ktorého premiestňujeme príkazy tak, že v ňom označíme príkaz a ľavým tlačidlom myši ho prenesieme do zvoleného panelu nástrojov vľavo hore (okno Uživatelské úpravy v...). Po pridaní príkazu do panelu nástrojov stlačíme OK alebo pokračujeme v ďalších úpravách.



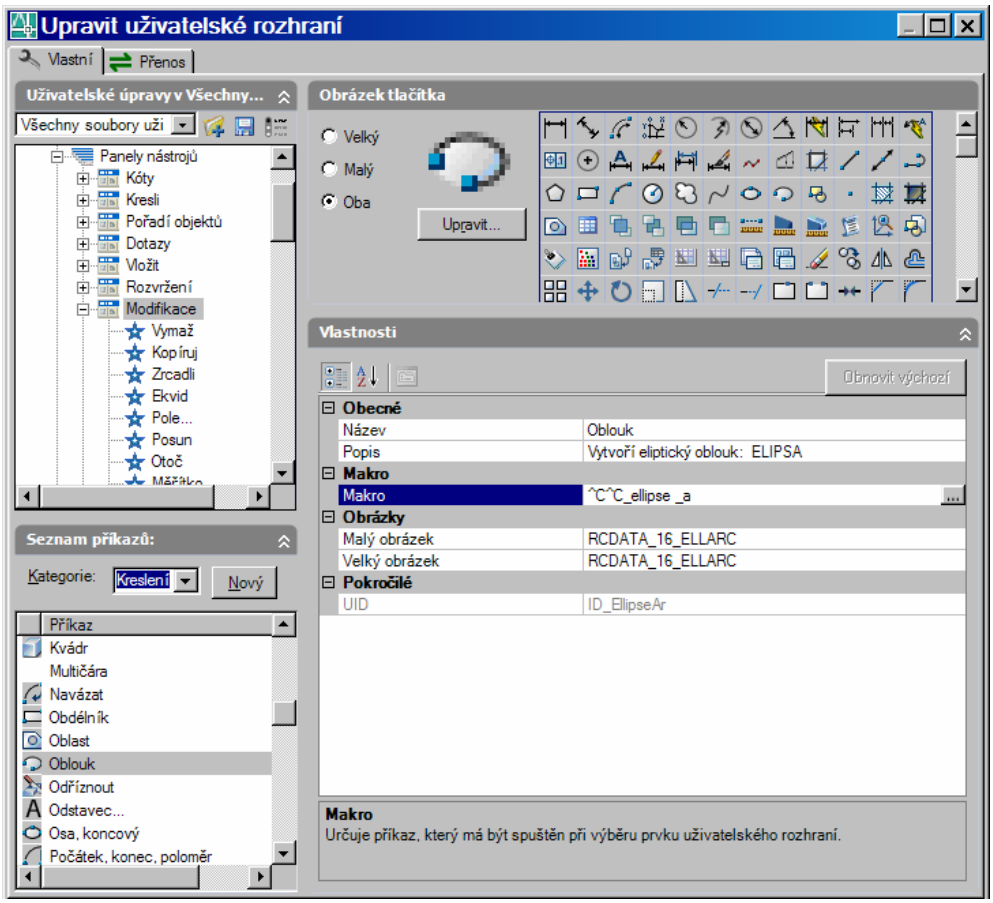
Obr. 2.1 Úpravy užívateľského rozhrania

Ak chceme vytvoriť a pridať nový príkaz, vytvára sa v sekcii *Zoznam príkazov* v CUI. Tu po stlačení *Nový* v pravej časti tejto sekcie nastavíme v pravom okne *Informace* zobrazené požadované parametre vrátane ikon. Následne pomocou ľavého tlačítka myši prenesieme príkaz do zvoleného nástrojového panelu, či už pôvodne existujúceho v AutoCADe, alebo nového.

Vhodné zásady pre zadávanie ikon:

- Obrázky vlastných ikoniek si ukladajte mimo adresára AutoCADu (napríklad do C:/MojeIkony.
- Adresár, do ktorého si ukladáte ikonky musí byť pridaný do – *Nástroje/Možnosti/Soubory/Cesta* k *podpurnym souborum*.
- Názvy obrázkov ikoniek si dávajte zrozumiteľné. Vami vytvorené obrázky s názvami slúžia na to, aby ste sa v nich vyznali a mohli ich aj neskôr pohodlne použiť

Hlavnou vecou, ktorú je potrebné podrobnejšie vysvetliť je vytvorenie vlastného príkazu, čiže makra v okne *Vlastnosti*.



Obr. 2.2 Tvorba vlastného príkazu

2.1 Úvod do makier – príkazy AutoCADu a špeciálne znaky v makrách

Makro definuje akciu, ku ktorej dôjde napríklad po kliknutí na ikonu, ktorú sme vytvorili postupom opísaným v predchádzajúcom texte, samozrejme s tým, že v okne *Vlastnosti* bude zadefinovaný príkaz resp. sled príkazov, čiže nejaké makro, ktorého vytváranie bude bližšie popísané v tejto časti.

Makro je potrebné do tohto miesta zadať vždy. Niekedy ide iba o jednoduchý príkaz – napríklad kreslenie úsečky – takéto makrá sú normálnou súčasťou AutoCADu a môžete si ich

pozrieť v paneli CUI. Predsa však ide o makro, lebo keď si spomenutý príkaz trochu pozriete, vidíte, že v okienku makro je predsa len niečo viac než spustenie úsečky (^C^C_line). Užívateľsky tvorené makro však zvyčajne spraví úlohu, ktorá by inak vyžadovala viac krokov. Jeho zmyslom je dosiahnuť väčšiu efektívnosť pri práci s AutoCADom. Napríklad chcete vytvoriť príkaz na písanie textu, ktorý pred vyvolaním samotného príkazu *Text* nastaví aktuálnu hladinu ako *PopisneTexty*. Alebo často zaobľujete s istým polomerom a makro ho pred spustením príkazu *Zaobli* nastaví, alebo si môžete predstaviť akýkoľvek iný sled príkazov, ktoré opakovane robíte v AutoCADe.

Do sekcie *Makro* v pravej časti panelu CUI sa zadá text makra. Dĺžka makier nie je nijako obmedzená. Je však treba vedieť, ako sa v makrách používajú určité znaky a poznať ďalšie aspekty a obmedzenia.

Makro v prvku užívateľského rozhrania môže byť veľmi jednoduché a obsahovať iba jeden príkaz (napríklad spomenutá *úsečka* – aj keď z ďalej uvedených dôvodov je anglický ekvivalent) a niektoré špeciálne znaky, ktorých zmysel o chvíľu objasníme.

Napríklad makro ^C^C_circle \2 nakreslí kružnicu o polomere 2 (polomer potom nemusí užívateľ zadávať). Komponenty, ktoré definujú toto makro, sú tieto:

Tabuľka 2.1 Komponenty makra KRUŽNICA2

Komponent	Typ komponentu	Výsledok
^C^C	Špeciálny riadiaci znak	Zruší všetky predchádzajúce spustené príkazy.
_	Špeciálny riadiaci znak	Umožňuje použiť anglické príkazy v národných verziách.
_circle	Príkaz	Spustí príkaz KRUŽNICA.
medzera	znak	Slúži ako enter po zadaní príkazu circle
\	Špeciálny riadiaci znak	Pozastaví makro, aby užívateľ mohol zadať stred kružnice.
2	údaj	Zadanie odpovede na výzvu k zadaniu polomeru (2).

V prípade potreby automaticky zrušiť pred prevedením makra každý príkaz, ktorý prípadne v AutoCADe beží, je potrebné zadať na začiatok makra znak ^C^C (čo je to isté, ako dvojité stlačenie klávesy ESC). Takto sa začína aj makro KRUŽNICA2.

Uvedené makro dokumentuje tiež zásadu, že je účelné zvykať si v makrách používať vždy len anglické príkazy (aj keď tvorca makra vlastní český AutoCAD). Je to z toho dôvodu, aby úpravy bežali na všetkých jazykových mutáciách príslušnej verzie AutoCADu, ktoré môžu byť nainštalované na inom počítači v ktorom sa bude makro tiež využívať. Anglické príkazy je potrebné písať s podtržítkom aj keď je vytvárané v anglickom AutoCADe – tiež z už spomenutého dôvodu.

Ak teda chceme napísať príkaz na kreslenie kružnice, nepíše sa *kružnice* (v českom AutoCADe), ani *circle* (v anglickom AutoCADe) ale vždy: *_circle*.

Keďže sa vyžaduje uprostred makra zadanie stredu kružnice od užívateľa (z klávesnice alebo pomocou ukazovacieho zariadenia), použije sa spätné lomítko (\). Touto konvenciou signalizujeme užívateľský vstup. V našom prípade umožnila táto možnosť zadať stred kružnice.

Niektoré makrá sú už v AutoCADe priamo vytvorené. Je vhodné si ich pri štúdiu tvorby makier prezrieť. Napríklad si spustite príkaz CUI a zobrazte si v panely nástrojov Modifikácie vlastnosti príkazu Preruš v bode. Vo vlastnosti makro je zapísaná textová

sekvencia ^C^C_break _f \@. *Break* je anglický ekvivalent príkazu *Preruš*, lomítko umožní zvoliť objekt ktorý sa má prerušiť, _f slúži k voľbe vyber prvý bod prerušenia a znakom @ sa špecifikuje druhý bod prerušenia a to ten istý ako prvý, pretože zadaním zaviináča zadávame znova naposledy zadaný bod, podobne ako pri normálnej práci v AutoCADe – môžete si to vyskúšať ak ste túto skratku nepoznali.

Špeciálne znaky používané v makrách sú zosumarizované v nasledujúcej tabuľke.

Tabuľka 2.2 Špeciálne znaky používané v makrách

Znak	Popis
;	Zadá ENTER
^M	Zadá ENTER
^I	Zadá TAB
[medzera]	Zadá medzeru; medzera medzi sekvenciami príkazov v príkaze je rovnaká ako stlačením klávesy MEDZERNÍK
\	Pozastaví makro a čaká na vstup od užívateľa (nemožno použiť s prístupovými klávesmi)
_	Preloží anglický názov príkazu alebo kľúčového slova na aktuálnu jazykovú verziu (t. j. anglický príkaz s podtržítkom je možné použiť v každej jazykovaj verzii AutoCADu)
=*	Zobrazí najvyššiu úroveň roletové, miestnej alebo obrázkovej ponuky
*^C^C	Opakuje príkaz, dokiaľ nie je zvolený iný príkaz
$	Vloží podmienený výraz jazyka DIESEL ($M=)
^B	Zapne alebo vypne krok (rovnaké ako CTRL+B)
^C	Zruší príkaz (rovnaké ako ESC)
^D	Zapne alebo vypne súradnice (rovnaké ako CTRL+D)
^E	Nastaví ďalšiu izometrickou rovinu (rovnaké ako CTRL+E)
^G	Zapne nebo vypne raster (rovnaké ako CTRL+G)
^H	Zadá BACKSPACE
^O	Zapne alebo vypne režim Ortho
^P	Zapne alebo vypne MENUECHO
^Q	Vytlačí odozvy všetkých výziev, zoznamov stavov a zadania pre tlačiareň (rovnaké ako CTRL+Q)
^T	Zapne alebo vypne tablet (rovnaké ako CTRL+T)
^V	Zmení aktuálny výrez
^Z	Prázdny znak, ktorý potlačí automatické pridanie klávesy MEZERNÍK na koniec príkazu

Ako je v tabuľke uvedené, v makrách možno použiť viaceré špeciálne znaky vrátane riadiacich znakov. Strieška (^) v makre je rovnocenná stlačeniu klávesy CTRL na klávesnici. Kombinovaním striešky s iným znakom možno vytvoriť makrá, ktoré napríklad zapnú nebo vypnú raster (^G) alebo zrušia príkaz (^C).

V makre je dôležitý každý znak a medzera. Pokiaľ vložíte medzeru na koniec makra, AutoCAD spracuje makro rovnako, ako keby ste zadali normálnym postupom príkaz (napríklad kreslenie kružnice) a potom ho dokončili stisnutím klávesy medzerník.

Niektoré makrá vyžadujú špeciálne ukončovacie členy. Určité príkazy (napríklad TEXT) vyžadujú k ukončeniu príkazu stlačenie klávesy ENTER, nie klávesy medzerník. Vtedy je potrebné zadať bodkočiarku(;), ktorá v makre automaticky zadá do príkazového riadku príkaz ENTER. Určité príkazy vyžadujú k dokončeniu viacej medzier alebo príkazov ENTER.

Napríklad:

Makro *_Zoom;_a;* aj *_Zoom;_a* prevedie *Zoom Všetkého* (*All*) nakoľko AutoCAD dá na koniec makra automaticky Enter (pokiaľ nie je posledným znakom bodkočiarka alebo medzera). Takže poslednú bodkočiarku nie je potrebné ani zadávať.

Väčšinou je potrebné pred napísaním makra prebehnúť si príkaz manuálne v AutoCADe. Predpokladajme napríklad, že chceme vymazať posledný nakreslený objekt v AutoCADe. Manuálny postup je tento:

Příkaz: _erase (anglický ekvivalent "vymaž" s podtržítkom)
Vyberte objekty: _l (anglický ekvivalent "poslední" s podtržítkom)
nalezeno: 1 (oznam na príkazovom riadku)
Vyberte objekty: enter

Makro bude potom vyzerať takto:

_Erase;_L;;

Niekedy je potrebné naučiť sa vzhľadom na tvorbu makier aj iné možnosti spustenia niektorých príkazov ako zvyčajne používame pri štandardnej práci. Napríklad chceme vytvoriť príkaz pre písanie textu, ktorý pred vyvolaním samotného príkazu *Text* nastaví aktuálnu hladinu na *PopisneTexty*. Pre nastavenie aktuálnej hladiny použijete najpravdepodobnejšie roletku pre hladiny zo štandardného nástrojového panelu. Aktuálnu hladinu však nastavuje aj príkaz *CLAYER*, ktorý sa vyvoláva aj obsluhuje z príkazového riadku. Makro potom bude vyzerať takto:

_Clayer;PopisneTexty;_text

V nasledujúcom makre si ukážeme ešte dve črty, o ktorých je potrebné vedieť pri vytváraní makier. Mnoho príkazov v AutoCADe komunikuje s užívateľom prostredníctvom dialógových panelov. V prípade, že nechcete zastaviť beh makra tým, že sa zobrazí dialógový panel a dopredu poznáte parametre, ktoré by bolo potrebné v dialógu nastaviť, môžete využiť verzie príkazov komunikujúce s užívateľom prostredníctvom príkazového riadku. Tieto verzie príkazov sa vyvolávajú tak, že pred názov príkazu sa zadá mínus. Napríklad makro pre vloženie bloku s názvom hydrant sa vytvorí tak, ako vyplýva z jeho manuálneho použitia v AutoCADe. Manuálny postup:

Příkaz: _–insert (anglický ekvivalent "vlož" s podtržítkom a mínusom)
Zadejte jméno bloku nebo [?] <kalnik>: hydrant

Jednotky: Milimetry Převod: 1.0000
Určete bod vložení nebo [Refbod/Měřítko/X/Y/Z/Otoč/PMěřítko/PX/PY/PZ/POtoč]: enter
Zadejte měřítko v ose X, určete protější roh nebo [Obdélník/XYZ] <1>: enter
Zadejte měřítko Y <použijte měřítko X>: enter
Určete úhel otočení <0>: enter

Makro bude mať potom tvar:

^C^C_–insert;hydrant;\;;;

Niekedy chceme opakovať určitý príkaz pokiaľ sa nestlačí Esc. V tomto prípade sa iba vloží na začiatok makra hviezdička. V prípade, že chceme do výkresu vložiť viac hydrantov vyzerá makro takto:

**^C^C_–insert;hydrant;\;;;*

Ďalej môžeme použiť makrá na nastavenie niektorých kresliacich pomôcok. Napríklad máte nejakú obľúbenú kombináciu uchopovacích módov, ktoré keď počas práce prestavíte chcete znovu nastaviť. Môže sa využiť nasledovná syntax makra dajme tomu pre nastavenie uchopenia koncového bodu (ENDpoint), priesečníku (INTersection) a polovice (MIDpoint):

^C^C_–OSNAP;_END,_INT,_MID

Iná možnosť je nastavenie systémovej premennej OSMODE, ktorá nastavenie uchopovacích módov uchováva. Nastavenie je definované pomocou kódu z nasledujúcej tabuľky:

Tabuľka 2.3 Kódy pre uchopenie objektov

Kód	Český výraz pre uchopovací mód	Anglický výraz pre uchopovací mód
0	Žádné	NONe
1	koncový	ENDpoint
2	polovina	MIDpoint
4	střed	CENter
8	bod	NODe
16	kvadrant	QUAdrant
32	průsečík	INTersection
64	referenční	INSertion
128	kolmo	PERpendicular
256	tečna	TANgent
512	nejblíže	NEArest
1024	Vymaže všetky uchopenia objektov	Clears all object snaps
2048	zdánlivý průsečík	APParent Intersection
4096	vynášecí	EXTension
8192	rovnoběžně	PARallel

Ak bude potrebné použiť viac ako jeden režim uchopenia, uvedie sa súčet hodnôt kódov. Zadaním hodnoty 35 potom určíme koncové uchopení objektu (bit 1), uchopenie objektu v polovici (bit 2) a v priesečníku (32) $\Rightarrow$ 1+2+32 = 33 tak ako v predchádzajúcom prípade. Zadaním 16383 sa nastavia všetky uchopenia objektov. V normálnom režime práce nastavujeme systémové premenné zadaním ich názvu (iba anglické názvy, nemusíme dávať podtržítko) do príkazového riadku AutoCADu a následným zadaním jej hodnoty. Pri vytvorení makra môžeme preto použiť celkom jednoduchý textový reťazec:

'OSMODE;35

V tomto príklade je nové ešte použitie apostrofu na začiatku makra, ktoré umožňuje aby bol vytvorený príkaz transparentný, t. j. aby sa mohol vyvolať počas behu iného príkazu.

Podobne je transparentným príkazom aj nasledujúci príklad, v ktorom nastavíme novú hodnotu systémovej premennej SNAPANG uchopením dvoch bodov a následné nastavenie ortho režimu. SNAPANG určuje uhol kroku a rastru pre aktuálny výrez. Príkaz sa potom využije na kreslenie čiar rovnobežných so zadaným smerom a kolmíc na tento smer:

'SNAPANG;_NEAR;_NEAR; ^O

Na prepísanie niektorých vlastností jednotlivej kóty možno použiť príkaz KÓTYPŘEPIŠ (anglicky DIMOVERRIDE). Môžeme ho využiť k napísaniu napríklad nasledujúcich makro príkazov:

Pridať ku kóte označenie priemeru: *^C^C_dimoverride;dimpost;%%c<>;;\;*

Pridať ku kóte znak ~: *^C^C_dimoverride;dimpost;~<>;;\;*

Predstavme si situáciu, že kreslíme v AutoCADe schémy objektov, ktorých jednotlivé časti označujeme číslami v krúžkoch alebo nejakým iným kratším textom v krúžku. Ukážme si makro pre kreslenie takéhoto krúžku (s polomerom 100) a s textom vo vnútri. Pred spustením príkazu je treba nastaviť primerane štýl písma, napr. nech má výšku 50.

V nasledujúcom makre sa potom zadáva iba stred kružnice a text:

**^C^C_CIRCLE;\100;_TEXT;_j;_m;@;;*

V tomto makro príkaze je *CIRCLE* anglický príkaz na kreslenie kružnice, za ktorým je bodkočiarkou vyznačený enter a lomítkom vstup užívateľa, ktorý zadefinuje stred kružnice. Polomer kružnice bude v tomto príklade 100. Ďalej nasleduje príkaz na písanie riadkového textu, enter a voľba **J**ustify (Upravit) a **M**iddle (Stred), čo zabezpečí zacentrovanie textu na zadaný bod, ktorým je stred kružnice, nakoľko na výzvu zadania tohto bodu makro obsahuje odpoveď v podobe @, čo znamená naposledy zadaný bod. Potom sú dva entery a vstup užívateľa, ktorý zadá vyžadovaný text, ktorý sa má napísať do krúžku.

Nasledujúci príkaz je podobný ako príkaz *Srovnat* (*ALIGN*), ktorý poskytuje plný AutoCAD avšak nie AutoCAD LT, v ktorom môžeme makro príkazy využívať. Najprv sa vyberú objekty, ktoré sa majú zrovnať a potom prvý bod na objekte, ktorý sa má zrovnať a následne jeho nové umiestnenie. Ďalej sa vyberie druhý bod a jeho umiestnenie na cieľovom objekte. Ak sa vzdialenosti budú líšiť tento príkaz nevykoná zväčšenie/zmenšenie ako pôvodný príkaz Srovnat, použije sa iba definícia smeru definovanej hrany určenej týmito dvoma bodmi:

^C^C_select;_move;_p;;_rotate;_p;;@;_ref;@;

Makrá môžeme využiť aj na iné činnosti než kreslenie alebo editáciu. Pomocou nasledujúcej syntaxe možno dosiahnuť dynamickú výmenu nástrojových panelov, t. j. to, že stlačením tlačidla v nástrojovom paneli sa pôvodný panel vymení za iný panel, v ktorom keď sa stlačí určené tlačidlo, znova sa vymení za pôvodný nástrojový panel. Zmyslom je úspora miesta na pracovnej ploche.

Môžete si buď vytvoriť vlastné panely, alebo pridať tlačidlá slúžiace na výmenu nástrojových panelov pre existujúce panely. Napríklad môže byť pracovná plocha usporiadaná tak, že naľavo je ukotvený panel pre kreslenie a modifikáciu a napravo chceme mať buď panel *Text* alebo *Kóty*.

V paneli *Text* sa vytvorí tlačidlo, ktoré *Text* skryje (_hide) a *Kóty* zobrazí (_show).

^C^C_–toolbar;"Text " _h _–toolbar "Kóty " _s

V druhom paneli sa vytvorí tlačidlo, ktoré *Kóty* skryje (_hide) a *Text* zobrazí (_show).

^C^C_–toolbar;"Kóty " _h _–toolbar " Text " _s

Uvedený postup možno využiť nielen na výmenu nástrojových panelov, ale aj na zobrazenie nového nástrojového panela, ktorý bude potrebný len určitú dobu a jeho trvalé zobrazenie by zbytočne zaberalo miesto (napr. úchopy, ...).

Nasledujúce makro využíva premennú AutoCADu *Perimeter*, v ktorej je uchovaný naposledy určený obvod pomocou príkazu plocha (area). Výstupom príkazu *plocha* je ako ste si už určite všimli nielen plocha ale aj obvod, ktorý ako uvidíme sa nám hodí na odmeranie dĺžky. V prípade neuzavretých kriviek zistí príkaz *plocha* aplikovaný na krivku jej dĺžku, ktorá sa môže skladať z priamych častí aj oblúkov. Príkaz, ktorý teraz uvedieme slúži na odmeranie dĺžky nadväzujúcich entít (ktoré majú spoločné koncové body), t. j. napríklad

môže ísť o dĺžku líniovej stavby, osi toku a podobne avšak tieto entity nie su nakreslené súvislou krivkou – tam by sme dĺžku zistili jednoducho z panelu vlastnosti. Príkaz najprv vyzýva k selekcii týchto dielčích entít, z ktorých sa meraná vzdialenosť skladá (oblúky, úsečky, krivky – nie však iba jedna krivka), potom ich spojí do krivky, spustí príkaz plocha a vypíše premennú Perimeter, t. j. dĺžku krivky do príkazového riadku. Následne zruší spojenie jednotlivých entít do krivky pomocou undo aby sme nenarušili pôvodný stav výkresu, keďže má ísť iba o informačný príkaz. Inak údaj 0, ktorý sa v príkaze vyskytuje, je tolerancia vzdialenosti nadväzujúcich bodov – môže sa sem zadať aj nejaké malé číslo pre prípad nepresného kreslenia, keď by jednotlivé entity celkom nenadväzovali jedna na druhú. Výpis dĺžky o ktorý ide hľadajte v príkazovom riadku v tvare *PERIMETER = 649.08 (pouze pro čtení)*.

^C^C_select;_pedit;_m;_p;;_y;_j;0;;_area;_o;_l;perimeter;_undo;3

2.2 Využitie jazyku DIESEL v makrách

DIESEL je skratka z "Dumb Interpretively Evaluated String Expression Language", čo je jednoduchý programovací jazyk v AutoCADe a tiež v AutoCADe LT.

AutoCAD poskytuje viacero API rozhraní (Application Programming Interfaces) na programovanie a modifikáciu vnútorného správania sa AutoCADu ako je LISP, VBA alebo C++. Tento malý programovací jazyk sa objavil prvýkrát v AutoCADe Release 12.

DIESEL bol sprvu používaný programátormi AutoCADu najmä k úprave stavového riadku a niektorých vlastností menu. Po objavení sa AutoCADu pre Windows to vyzeralo tak, že bude ďalej nepotrebný, avšak keď Autodesk vyvinul AutoCAD LT a rozhodol sa neposkytovať v tejto odľahčenej verzii AutoLisp, VBA ani iné pokročilé rozhrania na užívateľské prispôsobovanie, zostal DIESEL jediným programovacím jazykom ponechaným v LT. Z toho dôvodu má zmysel venovať mu pozornosť. V tejto kapitole sa budeme zaoberať možnosťou využitia jazyku DIESEL v užívateľských makrách. Môžeme robiť makrá aj bez využitia DIESELu, tak ako to ilustrujú príklady z predchádzajúcej časti, ale jeho využitie zväčšuje možnosti, ktoré ako uvidíme nie su nezaujímavé.

DIESEL zostavil John Walker pre Autodesk, Inc. Autor aj Autodesk dali programu licenciu public domain – čiže je svojim spôsobom verejným vlastníctvom. Je možné stiahnuť ho z WWW Johna Walkera na adrese www.fourmilab.ch/diesel. Toto umožňuje zahrnúť DIESEL interpreter do akýchkoľvek programov, či už majú vzťah k AutoCADu alebo nie.

Základnou myšlienkou výstavby jazyku DIESEL je to, že je orientovaný na prácu s reťazcami znakov – písmen, číslic a pod. Interpreter jazyku DIESEL je program, ktorý berie ako vstup reťazec znakov, vyhodnotí ho a vráti opäť reťazec znakov. V svojej najjednoduchšej akcii zoberie vstupný reťazec a vráti ako výstup ten istý reťazec – čo však pochopiteľne nie je príliš zaujímavé. Z toho dôvodu rozpoznáva DIESEL isté charakteristické sekvencie znakov, ktoré volajú jeho funkcie a to čo vráti bude výsledok volanej funkcie.

Na pokusy a zoznámenie sa s DIESELom bude zaujímavé vytvoriť si jeho interpreter, nakoľko by sme museli inak robiť všetky ďalšie pokusy cez príkaz CUI, tak ako sme písali makrá v predchádzajúcej kapitole. Interpreter urobíme tak, že sa napíše na príkazovom riadku AutoCADu nasledovný kód:

```
(defun c:DIESEL (/ s)
 (while (/= "" (setq s (getstring "\nDIESEL: " T)))
  (princ (menucmd (strcat "m=" s)))
```

```
)
 (princ)
)
```

Toto implementuje nový príkaz do AutoCADu, ktorý sa bude volať DIESEL. Bude požadovať vstupný string, odovzdá ho interpreteru DIESELu na spracovanie a poskytne výsledok.

V prípade, že používate AutoCAD LT vytvorte vlastné tlačidlo na niektorom nástrojovom paneli a ako makro vložte:

```
^C^C_SETENV DIESEL;\_SETENV DIESEL;$M=$(EVAL,"$(GETENV,DIESEL)");
```

Teraz môžme začať experimentovať. Najprv jednoduchšie – spustite príkaz DIESEL a pozdravte interpreter:

```
Příkaz: DIESEL
DIESEL: Hello,world!
Hello,world!
```

DIESEL práve absolvoval obľúbený základný test akéhokoľvek programovacieho jazyku a vytlačil (zobrazil) obligátny pozdrav. Ak sa Vám to nebude zdať ako dôkaz jeho činnosti dosť, môžete použiť trochu iný vstupný reťazec:

```
DIESEL: "Hello,world!"
Hello,world!
```

Tu už naozaj niečo spravil – odstránil úvodzovky uzatvárajúce textový reťazec. Keďže je všetko čo DIESEL pozná reťazec, niekedy mu musíme explicitne dať na známosť, že to čo sa mu poskytuje je naozaj iba reťazec, nie reťazec "meno-funkcie", ktorú má vyvolať. DIESEL interpreter teda akceptoval vstupný reťazec s úvodzovkami, ale keďže vedel, že tieto sú konvenciou, interpretoval tento vstup o dva znaky kratším reťazcom, t. j. odstránil úvodzovky.

Môžeme pokročiť o kúsok ďalej a vyskúšať schopnosť DIESELu rozpoznať a interpretovať nielen konvenciu (označovanie textového reťazca úvodzovkami) ale aj funkciu. Môžeme napríklad vypočítať dĺžku reťazca:

```
DIESEL: $(strlen,"Hello,world!")
12
```

Volania funkcií v DIESELi sú označované znamienkom doláru a uzavreté v zátvorke. V zátvorke oddeľuje čiarka názov funkcie a jej argumenty. Argumenty nie je potrebné dávať do úvodzoviek, pokiaľ neobsahujú čiarku, keďže čiarka je oddeľovač argumentov. Keby sme napríklad v predchádzajúcom príklade nedali úvodzovky, funkcia by vrátila hodnotu 5, pretože pri čiarke by jej argument končil.

V makrách využívajúcich DIESEL môžeme využívať aj jeho matematické funkcie. Pri tom je treba vždy myslieť na to, že keď chceme počítať, musia reťazce, ktoré DIESEL dostane vyzerať ako čísla. Toto je napríklad súčet dvoch čísiel, + je meno funkcie a obe jednotky sú jej argumenty:

DIESEL: $(+,1,1)
2

Matematické funkcie v DIESELi sú +, –, * a /. Ďalšie funkcie a konvencie jazyku DIESEL uvádzame v nasledujúcom zozname.

$M

Vnútri makra môžete použíť príkaz $M=, ktorý predchádza makru vytvorenému v jazyku DIESEL. AutoCAD vyhodnotí časť makra za $M= a výsledok je opäť spracovaný ako makro. Formát je: $M=výraz, kde $M= určuje AutoCADu, že má vyhodnotiť reťazec ako výraz DIESELu a *výraz* je výraz DIESELu.

^Z

Ak za výrazom nasleduje ^Z, AutoCAD nepridá za makro ENTER. AutoCAD totiž automaticky dopĺňa na koniec riadkov ENTER

+ (Súčet)

Formát: $(+, hodnota1 [, hodnota2, ..., hodnota9]). Vráti súčet čísel hodnota1, hodnota2, ..., hodnota9.

– (Rozdiel)

Formát: $(–, hodnota1 [, hodnota2, ..., hodnota9]). Vráti rozdiel čísel hodnota2 až hodnota9 od hodnoty1.

*** (Súčin)**

Formát: $(*, hodnota1 [, hodnota2, ..., hodnota9]). Vráti súčin čísel hodnota1, hodnota2, ..., hodnota9.

/ (Podiel)

Formát: $(/, hodnota1 [, hodnota2, ..., hodnota9]). Vráti výsledok delenia čísla hodnota1 číslami hodnota2, ..., hodnota9..

=

Formát: $(=, hodnota1, hodnota2). Ak je číslo hodnota1 rovné číslu hodnota2, funkcia vráti 1, inak vráti 0.

<

Formát: $(<, hodnota1, hodnota2). Ak je číslo hodnota1 menšie ako číslo hodnota2, funkcia vráti 1, inak vráti 0.

>

Formát: $(>, hodnota1, hodnota2). Ak je číslo hodnota1 väčšie ako číslo hodnota2, funkcia vráti 1, inak vráti 0.

!=

Formát: $(!=, hodnota1, hodnota2). Ak číslo hodnota1 nie je rovné číslu hodnota2, funkcia vráti 1, inak vráti 0.

<=

Formát: $(<=, hodnota1, hodnota2). Ak je číslo hodnota1 menšie, alebo rovné číslu hodnota2, funkcia vráti 1, inak vráti 0.

>=

Formát: $(>=, hodnota1, hodnota2). Ak je číslo hodnota1 väčšie, alebo rovné číslu hodnota2, funkcia vráti 1, inak vráti 0.

and

Formát: $(and, hodnota1 [, hodnota2, ..., hodnota9]). Vráti výsledok logickej operácie AND medzi argumentami hodnota1 až hodnota9.

angtos

Formát: $(angtos, hodnota [, režim, presnosť]. Vráti veľkosť uhlu v zadanom formáte a so zadanou presnosťou. Upraví zadanú hodnotu hodnota ako uhol vo formáte zadanom argumentami režim a presnosť podobne ako funkcia AutoLISPu. (Hodnoty pre argument režim sú uvedené v nasledujúcej tabuľke.) Ak hodnoty *režim* a *presnosť* nie sú uvedené, funkcia použije aktuálne nastavenie príkazu *_units*.

Hodnoty uhlových jednotiek:

Hodnota pre režim	**Formát reťazca**
0	Stupne
1	Stupne/Minúty/Sekundy
2	Gradiány
3	Radiány
4	Zemepisné jednotky

edtime

Formát: $(edtime, čas, vzor). Vráti dátum a čas v zadanom formáte podľa vzoru. Upraví hodnotu čas AutoCADu v juliánskom formáte (ktorú možno získať napr. pomocou výrazu $(getvar,date)) podľa formátu v argumente vzor. Vzor obsahuje formátovacie reťazce, ktoré sú nahradené zodpovedajúcimi hodnotami dátumu a času. Ostatné znaky vo formátovacom reťazci nie sú vyhodnotené a sú skopírované do výsledného reťazca. Formátovacie znaky sú uvedené v nasledujúcej tabuľke. Predpokladajme aktuálny dátum Pondelok, 7. augusta 2000 4:53:17.506.

Formátovacie znaky edtime:

Formát	**Výstup**	**Formát**	**Výstup**
D	7	H	4
DDD	Mon	MM	53
DDDD	Monday	SS	17
M	7	MSEC	506
MO	07	AM/PM	AM
MON	Aug	am/pm	am
MONTH	August	A/P	A
YY	00	a/p	a
YYYY	2000		

Zadajte celý výraz AM/PM, tak, ako je ukázané v predchádzajúcej tabuľke; ak použijete len AM, A bude vyhodnotené ako písmeno a M vráti aktuálny mesiac. Ak sa vo vzore objaví AM/PM, výrazy H a HH upravia čas v 12-ásť hodinovom formáte (12:00–12:59 1:00–11:59) namiesto 24 hodinového formátu (00:00–23:59). Ak čas sa rovná 0, použije sa dátum a čas vyhodnotenia výrazu. Tým sa vyhnete zbytočnému volaniu funkcie $(getvar,date) a je tým zaručené, že rôzne reťazce predané $(edtime) zobrazia rovnaký čas.

eq

Formát: $(eq, hodnota1, hodnota2). Ak sú reťazce hodnota1 a hodnota2 totožné, vráti táto funkcia 1, inak 0.

eval

Formát: $(eval, reťazec). Predá reťazec vyhodnocovaču DIESELu a vráti výsledok.

fix

Formát: $(fix, hodnota). Prevedie reálne číslo hodnota na celé číslo tak, že odreže časť za desatinnou čiarkou.

getenv

Formát: $(getenv, názov_premennej). Vráti hodnotu užívateľskej premennej so zadaným názvom názov_premennej. Ak premenná neexistuje, vráti prázdny reťazec.

getvar

Formát: $(getvar, názov_premennej). Vráti hodnotu systémovej premennej so zadaným názvom názov_premennej.

if

Formát: $(if, výraz, keď_pravda [,keď_nepravda]). Podľa podmienky vyhodnotí výraz. Ak je výraz nenulový, vyhodnotí výraz keď_pravda a vráti jeho výsledok. Inak vyhodnotí a vráti výsledok výrazu keď_nepravda. Všimnite si, že zátvorky sa nevyžadujú, keď sa výraz nevyhodnotí.

index

Formát: $(index1, ktorý, reťazec). Vráti zadaný prvok (*ktorý*) z *reťazca* oddeleného čiarkami. Predpokladá, že argument reťazec obsahuje jednu alebo viac hodnôt oddelených čiarkami. Argument ktorý určuje poradie vrátenej hodnoty je index1, pričom index prvej hodnoty je 0.

Táto funkcia sa najčastejšie používa pre získanie X, Y a Z súradníc z hodnoty vrátenej funkciou $(getvar) ak je týmto výsledkom bod (napr. Lastpoint).

linelen

Formát: $(linelen). Vráti dĺžku (v znakoch) najdlhšieho stavového riadku, ktorú je možné zobraziť. Túto funkciu môžete použiť pre formátovanie stavového riadku, v závislosti na kapacite displeja. Táto funkcia je užitočná len pre konfiguráciu stavového riadku pomocou premennej MODEMACRO. Priestor dostupný pre MODEMACRO je aktuálne obmedzený na 240 znakov. Funkcia $(linelen) vždy vráti 240. Táto funkcia nemusí byť dostupná v budúcich verziách AutoCADu.

nth

Formát: $(nth, ktorý, arg0 [, arg1,..., arg7]). Vyhodnotí a vráti argument určený argumentom ktorý. Ak je ktorý rovný 0, nth vráti arg0 atd. Všimnite si rozdielu mezi $(nth) a $(index). $(nth) vracia jeden z argumentov funkcie, zatiaľ čo $(index) získa hodnotu z čiarkami oddelených položiek v jedinom reťazci. Argumenty, ktoré nie sú vybrané argumentom *ktorý*, sa nevyhodnocujú.

or

Formát: $(or, val1 [, val2,..., val9]). Vráti logickú operáciu OR medzi celočíselnými hodnotami val1 až val9.

Rtos

Formát: $(rtos, hodnota [, formát, presnosť]). Vráti reálne číslo v zadanom formáte a presnosti. Upraví zadanú hodnotu ako reálne číslo v zadanom *formát*e a so zadanou *presnosť*ou, podobne ako funkcia AutoLISPu. Ak nie je zadaný *formát* a *presnosť*, použije aktuálne nastavenie podľa príkazu _units.

Strlen

Formát: $(strlen, reťazec). Vráti počet znakov v reťazci.

Substr

Formát: $(substr, reťazec, počiatok [, dĺžka]). Vráti časť reťazca určenú argumentami počiatok a dĺžka. Argument dĺžka určuje počet znakov, ktoré budú z reťazca prečítané. Znaky

v reťazci sú číslované od 1. Ak nie je zadaný parameter dĺžka, vracia reťazec od počiatku až do konca daného reťazca.

Upper

Formát: $(upper, reťazec). Vráti *reťazec* prevedený na veľké písmená podľa pravidel aktuálneho jazyka.

Xor

Formát: $(xor, hod1 [, hod2, ..., hod9]). Vrátí logickú operáciu XOR medzi celočíselnými hodnotami hod1 až hod9.

Pri písaní vlastných makier je treba brať do úvahy, že celé správanie AutoCADu sa riadi hodnotami tzv. systémových premenných AutoCADu. Teda aj všetky dialógové panely, slúžia len na to, aby nastavili hodnoty niektorých systémových premenných.

Pomocou príkazov DIESELu môžete tieto hodnoty čítať, používať vo svojich makrách a samozrejme meniť. Taktiež môžete používať svoje vlastné užívateľské premenné, do ktorých môžete tiež hodnoty ukladať a neskôr prečítať. Okrem toho obsahuje AutoCAD 15 špeciálnych premenných USERXX, ktoré tiež možno použiť na ukladanie hodnôt premenných ako bude uvedené v ďalšom. Tieto základné vlastnosti patria medzi najdôležitejšie a zároveň najpoužívanejšie.

Niekedy dá najviac práce práve zistiť tú premennú, ktorá riadi vlastnosť, ktorú chceme zmeniť, alebo použiť. Zoznam všetkých systémových premenných nájdete v helpe AutoCADu; najdôležitejšie systémové premenné spolu s popisom v prílohe skrípt.

Nastavenie hodnoty systémovej premennej ktorá sa dá meniť (niektoré sú určené len pre čítanie) vykonáte príkazom setvar. Napríklad nastavenie výšky kótovacieho textu:

setvar DIMTXT

Výška je uložená v premennej DIMTXT, odkiaľ ju môžeme aj čítať príkazom *getvar* (používa sa len v makrách) alebo meniť príkazom *setvar* (môže sa použiť aj priamo v príkazovom riadku). Systémové premenné však môžeme meniť aj priamo, t. j. bez príkazu setvar. Stačí napísať názov premennej a potvrdiť ENTERom.

Použitie hodnoty systémovej premennej v makrách demonštruje nasledujúci príklad. Chceme písať text s výškou ktorá sa nastaví v závislosti od výšky kótovacieho textu. Syntax príkazu getvar je: *$(getvar, názov_premennej),* kde *názov_premennej* je DIMTXT.

Pri použití v makrách musíme pred použitím aplikovať príkaz *$M=*, ktorý predchádza príkazu DIESELu. Tento príkaz sa nepoužíva pre ďalší (zahniezdený) príkaz DIESELu ak sa tento už nachádza v príkaze DIESELu.

```
^C^C_text;\$M=$(getvar,DIMTXT);;
```

Ak by sme chceli písať textom, ktorého veľkosť by bola 2x väčšia ako výška kótovacieho textu, syntax by vyzerala takto:

```
^C^C_text;\$M=$(*,2,$(getvar, DIMTXT));;
```

Všimnite si, že vnútri príkazu DIESLu na násobenie sme pri použití getvar nedávali príkaz *$M=*.

AutoCAD umožňuje definovať aj užívateľské premenné (t. j. premenné s ľubovoľným názvom), do ktorých môžete ukladať rôzne hodnoty pre ďalšie použitie.

Nastavenie hodnoty užívateľskej premennej sa nastavuje odlišne v plnom AutoCADe a v AutoCADe LT, v ktorom túto akciu vykonáte príkazom *setenv*.

Nastavenie počiatočnej hodnoty premennej bude vyzerať takto:

`setenv HODNOTA`

Potom zadáme nejaké číslo, ktoré sa uloží do premennej HODNOTA, odkiaľ ho môžeme čítať príkazom *getenv* (používa sa len v makrách) alebo meniť príkazom *setenv* (môže sa použiť aj priamo v príkazovom riadku)

Užívateľské premenné sa nedajú meniť priamo, t. j. bez príkazu setenv (napísaním názvu premennej a potvrdením ENTERom.)

Nastavenie užívateľskej premennej vo veľkom AutoCADe je žiaľ odlišné, čo nie je najšťastnejšie z hľadiska kompatibility makier pod oboma AutoCADmi:

^C^C(setenv"PREMENNA"(getstring"PREMENNA:"))

Nastavenie užívateľskej premennej napr.: PREMENNA2 podľa inej premennej napr. PREMENNA1:

^C^C(setenv"PREMENNA2"(getenv"PREMENNA1"))

Prečítanie užívateľskej premennej PREMENNA2:

^C^C(getenv"PREMENNA2");

Prečítanie systémovej premennej napríklad DIMTXT

^C^C(getvar"DIMTXT");

Hodnoty užívateľských premenných za zapisujú do registrov Windows, takže aj po vypnutí počítača, alebo pri práci v inom výkrese sú stále prístupné ich naposledy nastavené hodnoty.

Okrem užívateľských premenných sa môžu využiť aj systémové premenné. Tieto premenné sa ukladajú spolu s výkresom. Takže ich veľká výhoda je, že v každom výkrese budú mať nastavené svoje naposledy zadané hodnoty (vhodné napr. pri makrách týkajúcich sa mierky)

Premenné USERR1, USERR2, USERR3, USERR4 a USERR5 slúžia na ukladanie a obnovovanie reálnych hodnôt

Premenné USERI1, USERI2, USERI3, USERI4 a USERI5 slúžia na ukladanie a obnovovanie celočíselných hodnôt

Premenné USERS1, USERS2, USERS3, USERS4, a USERS5 slúžia na ukladanie a obnovovanie textových raťazcov – tieto sa však s výkresom neukladajú.

Zvláštnym druhom premenných v AutoCADe sú tie, ktoré vracajú bod. Na načítanie hodnoty premennej Viewctr využijeme funkciu *getvar*:

DIESEL: $(getvar,viewctr)
261.70086705,176.04582323,0

Premenná AutoCADu viewctr uchováva stred pohľadu v aktuálnom výreze, teda jej výsledok je bod. V takomto prípade vytvára DIESEL čiarkou rozdelený reťazec. Keby sa išlo o celočíselné alebo reálne premenné poskytne obyčajný reťazec. Pri tejto príležitosti sa môžeme zmieniť o šikovnej funkcii *index*, ak potrebujeme získať x-ovú alebo y-ovú súradnicu bodu, ako v nasledujúcom príklade:

DIESEL: $(rtos,$(index,1,$(getvar,viewctr)),2,3)
Eval: $(RTOS, $(index,1,$(getvar,viewctr)), 2, 3)
Eval: $(INDEX, 1, $(getvar,viewctr))

Eval: $(GETVAR, viewctr)
===> 261.70086705,176.04582323,0
===> 176.04582323
===> 176.046
176.046

Dôležitou vlastnosťou implementovanou v jazyku DIESEL je to, že podobne ako väčšie programovacie jazyky dokáže interpretovať a vyhodnocovať zahniezdené funkcie. To znamená že voláme funkciu, ktorá obsahuje argument, ktorý sa tiež vypočíta nejakou funkciou. Môžeme si to znova vyskúšať vo vyhodnocovači DIESELu, ktorý implementujeme postupom uvedeným v úvode kapitoly:

DIESEL: $(+,$(strlen,aDIESELteststring),$(strlen,anotherstring))
30

Pri zostavovaní zahniezdených funkcií pomocou DIESELu bude často dôležité preveriť v akom slede sa funkcie vyhodnocujú a s akým výsledkom. Aj v AutoCADe aj v AutoCADe LT získame záznam tohto vyhodnocovania tak, že nastavíme systémovú premennú MACROTRACE na 1 (napísaním tejto premennej na príkazový riadok a zadaním 1). Tu je takýto záznam z naposledy uvedeného príkladu:

DIESEL: $(+,$(strlen,aDIESELteststring),$(strlen,anotherstring))
Eval: $(+, $(strlen,aDIESELteststring), $(strlen,anotherstring))
Eval: $(strlen,aDIESELteststring)
===> 17
Eval: $(strlen,anotherstring)
===> 13
===> 30

Potrebné je zaoberať sa ešte otázkou postupnosti vyhodnocovania zložitejšieho makra, obsahujúceho výrazy jazyka DIESEL. Ide o pomerne ťažšie pochopiteľnú vec a keby ste nasledujúcu úvahu aj hneď nepochopili, určite sa k nej vráťte, ak Vám makro nebude akceptovať nejaké nastavené hodnoty na jeho začiatku.

Všimnite si nasledujúci príklad:

snapang;0;_text;$M=$(getvar,insbase);'snapang;45;20;$(getvar,snapang);Insbase;

Akým spôsobom bude tento sled príkazov vyhodnotený? Najprv je nastavená premenná SNAPANG (Nastavuje uhol kroku a rastru pre aktuálny výrez) na 0. Potom sa naštartuje príkaz text. Vyhodnotí sa sekvencia $M= a interpreter DIESELu vyhodnotí zvyšok makra. Vstupný reťazec DIESELu je (pre prehľadnosť budeme používať pokračovací znak + namiesto písania do jednoho riadku):

$(getvar,insbase);+
'snapang;45;+
20;+
$(getvar,snapang);+
Insbase;
Výstupný reťazec DIESELu je nasledovný:

0,0,0;+
'snapang;45;20;+
0;+
Insbase;

Všimnite si, že ako uhol textu je použitá hodnota SNAPANG, ktorá bola nastavená na začiatku, nie modifikovaná počas behu príkazu text transparentným príkazom. Ak chcete aby bola použitá modifikovaná hodnota, prvým pokusom môže byť použiť ďalšiu sekvenciu "$M=":
snapang;0;_text;$M=$(getvar,insbase);'snapang;45;20;$M=$(getvar,snapang);Insbase;

Vstupný reťazec DIESELu je teraz:
$(getvar,insbase);+
'snapang;45;+
20;+
$M=$(getvar,snapang);+
Insbase;
Toto sa však bohužiaľ preloží ako:
0,0,0;+
'snapang;45;+
20;+
$M=0;+
Insbase;
Čo opäť nie je dobre.
Správne riešenie je pridať ešte dvojicu úvodzoviek. Toto "zdrží" vnútorný výraz DIESELu vo vyhodnocovaní:

snapang;0;_text;$M=$(getvar,insbase);'snapang;45;20;"$M=$(getvar,snapang)";Insbase;

Vstupný reťazec DIESELu je teraz:

$(getvar,insbase);+
'snapang;45;+
20;+
"$M=$(getvar,snapang)";+
Insbase;

Výstupný reťazec DIESELu je takýto:

0,0,0;+
'snapang;45;+
20;+
$M=$(getvar,snapang);+
Insbase;

Ako je vidieť druhý výraz DIESELu nebol vyhodnotený, ale bol iba skopírovaný do výstupu makra. Interpreter potom pokračuje ďalej. Pošle bod 0,0,0 do príkazového okna, nastaví hodnotu SNAPANG na 45 (čo je 0.78539816 v radiánoch). Potom pokračuje a príkazový riadok obdrží výšku textu – hodnotu 20.

Znova je vyhodnotený "$M=" výraz a interpreter DIESELu je volaný opäť, tentokrát vďaka zdržaniu s výsledkom:

$(getvar,snapang);+
Insbase;
Čiže výstup je:
0.78539816;
INSBASE;

čiže je použitá správna hodnota nového nastavenia SNAPANG.

Nakoniec uvedieme niekoľko príkladov využívajúcich získané vedomosti.

Nasledujúci príklad kreslí číslo pri každom kliknutí a automaticky ho zvyšuje o 1. Používa štýl, výšku a otočenie naposledy vytvoreného textu. Pred jeho spustením nastavte východziu hodnotu číslovania do premennej USERI1. Na príkazovom riadku sa zadá setvar useri1 a napíše sa číslo od ktorého sa má číslovať (najčastejšie 1). Tiež treba nastaviť štýl písma.

**^C^Ctext;_m;\;;$M=$(getvar,USERI1);setvar;USERI1;$(+,1,$(getvar,USERI1))*

Pred kapitolou o využívaní príkazov jazyka DIESEL sme uviedli príkaz na kreslenie krúžku s daným polomerom a s textom vo vnútri. Princíp hore uvedeného makra môžeme rozšíriť a uviesť kreslenie krúžku, kde sa do krúžku vpisuje číslovanie a zväčšuje sa o 1. Prvý údaj číslovania sa musí opäť nastaviť pomocou systémovej premennej USERI1.

**^C^C_circle;\100;_text;_j;_m;@;50;0;$M=$(getvar,useri1);setvar;useri1;$M=$(+,1,$(getvar,useri1))*

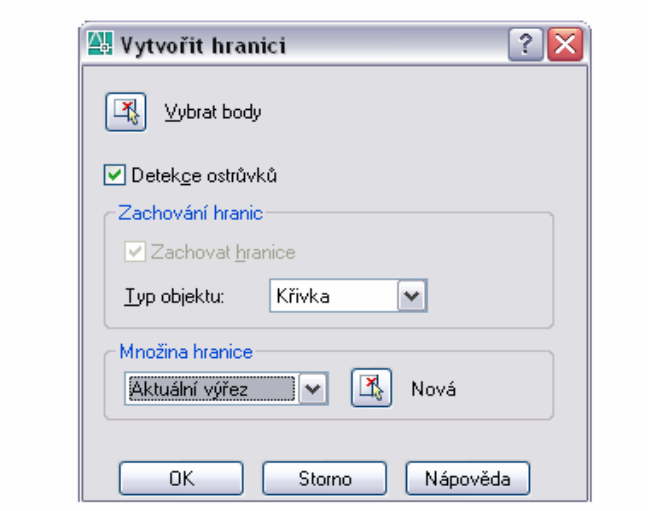
Použité príkazy:

*	– opakovanie
Circle	– kružnica
100	– polomer kružnice
_text	– riadkový text
j	– voľba upraviť (**J**ustify) v príkaze text
m	– Stred (**M**iddle) t. j. text upraviť na stred v kružnice (@– posledne zadaný bod)
50	– výška textu
0	– otočenie textu
;	– ENTER
$M=$	– vyvolá jazyk DIESEL
getvar, setvar	– príkazy jazyka DIESEL na uloženie a prečítanie premennej
$(+,1,$(getvar,useri1)	– operácia sčítania v DIESELi
USERI1	– premenná v ktorej je uložená celočíselná hodnota

Príklad teda predpokladá, že je nastavený textový štýl s výškou 0, ktorá sa v makre nastaví na 50 a kružnica má polomer 100 – obe tieto hodnoty možno jednoducho upraviť.

V ďalšom príklade ide o označenie bodu jeho súradnicami. Využíva niektoré poznatky o ovplyvňovaní postupnosti vyhodnocovacej sekcie príkazu, ktorý obsahuje výrazy jazyku DIESEL a nastavovanie premennej v rámci makra. Obsahuje tiež využitie funkcie *index* na rozkódovanie zoznamu oddeleného čiarkami akým je aj premenná AutoCADu *lastpoint* v ktorej je uložený posledný zadaný bod (pre príkaz id). Funkcia *rtos* formátuje výstup.

^C^C_id;_setvar;userr1;$M=$(rtos,$(index,0,$(getvar,lastpoint)));_setvar;userr2;$M=$(rtos,$(index,1,$(getvar,lastpoint)));_leader;@;\;"$M=$(getvar,userr1)","""$M=$(getvar,userr2)""";;



Obr. 2.3 Príkaz na tvorbu hraničnej krivky

Nakoniec si podrobnejšie popíšeme vytvorenie makra pre odmeranie plochy uzavretej ľubovoľnými entitami napr. úsečkami, krivkami, kružnicami oblúkmi atď. Na toto využijeme príkaz BOUNDARY (HKŘIVKA), ktorý umožňuje vytvorenie oblasti alebo krivky pomocou určeného bodu vo vnútri plochy, ktorá je ohraničená objektmi. Jej jednotlivé parametre zadávané na dialógovom paneli sú:

Vybrať body

Umožňuje výber bodu na základe ktorého príkaz následne určí hranicu z existujúcich objektov, ktoré tvoria uzatvorenú oblasť okolo určeného bodu.

Nájdenie ostrovov

Určuje, či príkaz *HKŘIVKA* zisťuje vnútorné uzatvorené hranice nazývané ostrovy.

Typ objektu

Riadi typ nového hraničného objektu. Príkaz *HKŘIVKA* vytvorí hranicu ako oblasť alebo objekt typu krivka.

Množina hraníc

Definuje množinu objektov, ktoré príkaz *HKŘIVKA* analyzuje pri definovaní hranice z určeného bodu.

Aktuálny výrez

Definuje množinu hraníc zo všetkých objektov v rozsahu aktuálneho výrezu. Pri výbere tejto možnosti budú zrušené všetky aktuálne množiny hraníc.

Nová

Vyzve vás na zadanie objektov, z ktorých chcete vytvoriť množinu hraníc. Pri vytvorení novej množiny hraníc zahrnie príkaz *HKŘIVKA* iba objekty, ktoré možno použiť na vytvorenie oblasti alebo uzavretej krivky.

Manuálne odmeranie plochy by sme pomocou tohto príkazu vykonali takto:

6. Pre jednoduchosť kontroly si nakreslíme nejaký jednoduchý útvar.

7. Príkazom *HKŘIVKA*, kde si zvolíme "vybrať body" a kliknutím do plochy vytvoríme hraničnú krivku.

8. Následne pomocou príkazu *plocha* (*area*) odmeriame plochu takto vzniknutého objektu.
9. Použitím príkazu *text* napíšeme do objektu/plochy jeho plochu.
10. Vymažeme pomocnú hraničnú krivku príkazom *vymaž* (*erase*).

Na odmeranie plochy môžeme použiť s výhodou aj makro, ktoré sa bude riadiť horeuvedeným postupom, iba s tým rozdielom, že príkaz *HKŘIVKA* (*boundary*) spustíme v jeho režime bez dialógového panelu použitím mínus pred príkazom.

V menu "Upraviť užívateľské rozhraní(Customize user interface)" v "Zozname príkazov(Command list)" zvolíme "Vlastný príkaz" (Custom commands) kde v pravej časti tabuľky vo Vlastnostiach (Properties) zadáme nové makro:

^C^C_id;\users1;$M=$(getvar,LASTPOINT);_–boundary;@;
_area;_O;_last;_erase;_last;;_text;"$M=$(getvar,USERS1)";15;0;"$M=$(rtos,$(getvar,area)",2,2);

Poznámky:

Príkaz je dlhší, pokračuje vždy bez medzier až po tretí riadok

Users1	– reťazcová premenná, do ktorej je uložená premenná AutoCADu *lastpoint*, nakoľko *lastpoint* bude prepísaná v rámci príkazu *boundary* inou – nechcenou hodnotou
Boundary	– hraničná krivka
\	– užívateľský vstup: kliknutím označíme vnútorný bod plochy, ktorú chceme ohraničiť
area	– príkaz na odmeranie plochy, ale aj systémová premenná AutoCADu v závere príkazu
O	– objekt
last	– posledný objekt (jeho plochu odmeria)
erase	– vymaže iba dočasne potrebnú ohraničujúcu krivku
last	– výber naposledy vytvorenej entity, v našom prípade krivky
text	– príkaz na písanie textu
15;0	– znamená výšku písma, enter a natočenie textu
$M=$(rtos,$(getvar,area),2,2)	– zadá ako obsah textu poslednú odmeranú plochu

3 UŽÍVATEĽSKÉ TYPY ČIAR

Pri technickom kreslení je zaužívanou a aj normovo stanovenou zásadou rozlišovanie rôznych typov čiar na kreslenie rozdielnych objektov alebo ich aspektov (os, hranica, líniové vedenia, viditeľná časť objektu, skrytá hrana, rôzne typy materiálov, rezy atď.). Používaním odlišných typov čiar sa stávajú technické výkresy prehľadnejšie a čitateľnejšie.

Definíciu typu čiary určuje konkrétna sekvenciu čiarok a bodiek, relatívna dĺžka čiarok a medzier a charakteristiky prípadných textu alebo tvarov (značiek) obsiahnutých v definíciách čiar. Pri tvorbe výkresu je možné použiť ľubovoľný zo štandardných typov čiar, ktorý ponúka AutoCAD, alebo možno vytvoriť svoje vlastné typy čiar.

Keďže pre rôzne možnosti použitia AutoCADu ako aj rôzne normové predpisy vzťahujúce sa na vzhľad čiar (podľa teritória alebo technického odboru) je veľmi veľa možností, nie su všetky do úvahy pripadajúce čiary integrované ako súčasť súboru výkresu. Typy čiar sú definované pre výkresy externe v jednom alebo viacerých súboroch definícií typov čiar, ktoré majú príponu LIN. Z týchto súborov sa tie čiary, ktoré sú pre vytváraný výkres aktuálne do výkresu načítavajú.

AutoCAD obsahuje od výrobcu nasledujúce súbory LIN: acad.lin a acadiso.lin. Zobrazením alebo tlačou týchto textových súborov (stačí použiť Word alebo Notepad) možno lepšie porozumieť konštrukcii vlastných typov čiar, o ktorých sa hovorí ďalej v tejto kapitole.

Súbor LIN môže obsahovať definíciu viacerých jednoduchých aj zložitých typov čiar. K existujúcemu súboru LIN je možné pridať nové typy čiar alebo možno vytvoriť vlastný súbor LIN. Ak chcete vytvoriť alebo upraviť definíciu typov čiar, dá sa použiť ľubovoľný textový editor – napr. Notepad (Poznámkový blok) alebo textový procesor, v ktorom sa textový ASCI súbor LIN otvorí, modifikuje alebo od začiatku vytvára. Toto je nutné predovšetkým pri tvorbe zložitejších čiar so značkami alebo textami. Pre tvorbu tzv. jednoduchých typov čiar (ide o čiary, ktoré obsahujú iba bodky, čiarky a medzery) je výhodné využitie príkazu TYPČ, ktorý sa spúšťa z príkazového riadku AutoCADu.

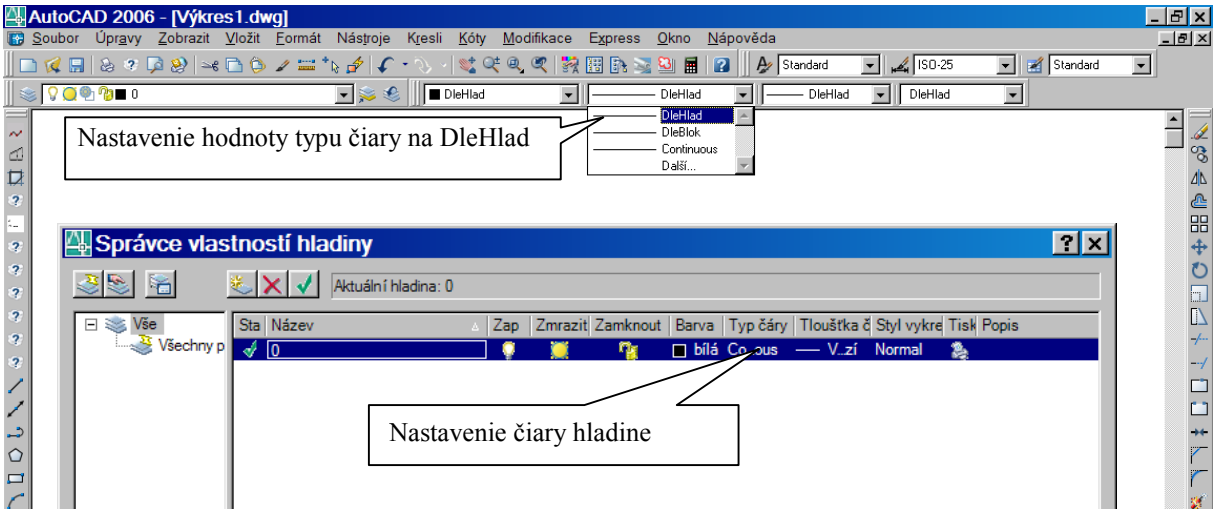
3.1 Práca s existujúcimi typmi čiar

V tejto časti iba stručne zhrnieme základy práce s aplikáciou rôznych typov čiar, nakoľko ide o bežnú problematiku, ktorej zvládnutie v týchto skriptách predpokladáme.

Primárnou úlohou s existujúcimi typmi čiar je ich priradenie rôznym typom objektov vo výkrese AutoCADu. Tento úkon je podobný ako pri priraďovaní iných vlastností objektom (napr. farba): buď priradíte typ čiary hladine a kreslením v nej pri súčasnom nastavení hodnoty vlastnosti typ čiary v štandardnom paneli nástrojov na DleHlad (ByLayer) dosiahneme požadovaný efekt (Obr. 3.1), alebo sa nastavuje typ čiary individuálne jednotlivým objektom. Prvý spôsob je z hľadiska organizácie práce vhodnejší. Pri priraďovaní typu čiary jednotlivým objektom použijeme buď panel Vlastnosti (Properties), alebo roletku na paneli *Vlastnosti objektu*. Keď pred použitím tejto roletky sú v pracovnom priestore AutoCADu vyselektované nejaké entity, roletka priradí vybraný typ čiary týmto entitám; keď nie sú, tak zvolený typ čiary bude aktuálny pre nasledujúce novovytvárané entity v AutoCADe. Toto nastavenie z panelu nástrojov *Vlastnosti objektu* má prednosť pred nastavením v paneli hladín, ktoré zostane aktívne iba pri zmienenom nastavení DleHlad.

Pred alebo počas priraďovania typu čiary objektu alebo hladine je potrebné požadované typy čiar do výkresu načítať – sú to bežné úkony obsluhované buď z menu

Formát → Typ čáry... alebo z roletky z panelu nástrojov *Vlastnosti objektu* (Další...), prípadne je príslušný dialóg dostupný aj zo Správcu hladín. Keď sa jednou z týchto ciest dopracujete cez dialóg Správce typu čáry a cez tlačidlo Načíst až k dialógovému panelu s názvom Načíst typy čár všimnite si, že tento má v hornej časti názov súboru, z ktorého sa čiary načítavajú, resp. tlačidlo s možnosťou výberu (iného) požadovaného súboru. Toto je už prv zmienený súbor LIN, ktorého modifikáciou získavame nové definície čiar v AutoCADe.



Obr. 3.1 Priradenie typu čiary hladine

Poslednými dvoma vecami, o ktorých sa v tejto úvodnej časti zmienime je merítko čiary a vlastnosť generovanie typu čiary. Po aplikovaní napr. čiarkovanej čiary sa niekedy stáva, že objekt je vizuálne stále akoby nakreslený obyčajnou súvislou čiarou Continuous. Môže to byť spôsobené buď príliš veľkým alebo malým merítkom čiary, čo je hodnota, ktorou sa prenásobujú dĺžky čiarok a medzier v definícii čiary tak, aby boli vizuálne adekvátne mierke a rozmerom daného výkresu. To znamená, že napr. pri mierke čiary 10 budú čiarky a medzery 10-krát väčšie než v pôvodnej definícii a táto definícia bude vhodnejšia pre rozmerovo relatívne väčšie výkresy (čo do výkresových jednotiek). Pre rozmerovo menšie výkresy môže byť naopak vhodné prenásobenie mierkou čiary menšou než jedna. Ak sú čiarky príliš malé zlievajú sa do súvislej čiary a naopak ak sú príliš veľké, tak jedna čiarka čiarkovanej čiary môže byť väčšia ako napr. úsečka, ktorá je týmto typom čiary nakreslená a taktiež bude konečný vizuálny efekt súvislá čiara. Obsluha tejto vlastnosti je opäť možná buď globálne pre celý výkres z menu Formát → Typ čáry... alebo individuálne pre jednotlivé objekty z panelu Vlastnosti.

Vlastnosť generovanie typu čiary, ktorú možno nájsť v spodnej časti panelu Vlastnosti pri výbere krivky, generuje typ čiary po celej krivke podľa vzoru bez ohľadu na vrcholy krivky. Ak je táto voľba vypnutá, generuje sa typ čiary tak, aby vzor vždy začínal a končil pri každom vrchole krivky čiarkou.

Vlastnosť generovanie typu čiary, ak je zapnutá, umožňuje jeden trik. Text v komplexných typoch čiar (môžete experimentovať napr. s typom čiary PLYN) je štandardne písaný v smere kreslenia čiary, ale oblúky sú AutoCADom vždy kreslené v rovnakom smere, a to tak, že text mieri vždy hornou časťou dovnútra oblúku. Ak zapnete generovanie typu čiar, bude text smerovaný jednotne v smere orientácie krivky.

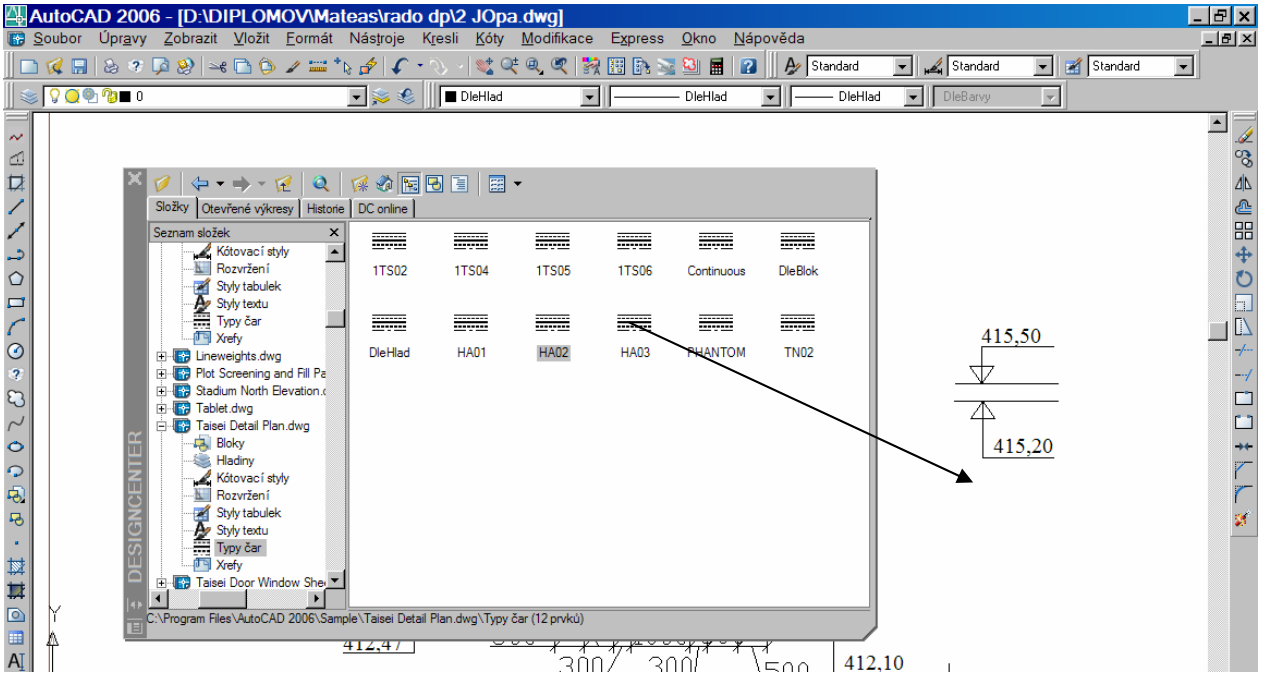
Ak potrebujete smer textu otočiť, môžete použiť autolisp utilitu Rvrsline.lsp z www.cadstudio.cz alebo použiť postup bez tejto utility, ktorý je uvedený nižšie.

Mimochodom niekedy vám vadí smerovanie textu v oblúku – ak oblúk prevediete na krivku (KEDIT, _PEDIT), môžete tiež jeho smer otočiť (RvrsLine, alebo podľa toho istého dole uvedeného postupu) a potom aj text zmení orientáciu (ak máte zapnutú vlastnosť entity ' generovanie typu čiary').

Zmena smeru krivky bez autolispu a iných utilít:

6. Nakreslite novú krivku odkiaľkoľvek na obrazovke ku koncu (s uchopením) krivky, ktorej smer chcete zmeniť. Tento koniec má byť vyžadovaným novým začiatkom krivky.
7. Spustite editáciu tejto novej krivky pomocou príkazu KEDIT (alebo _PEDIT)
8. Zvoľte voľbu pripoj a pripojte k tejto pomocnej krivke pôvodnú krivku
9. Pomocou príkazu OŘEŽ odrežte pomocný segment z pôvodnej krivky

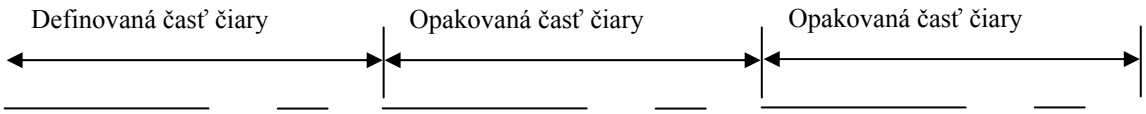
Keď potrebujete získať definíciu čiary, ktorá už existuje v inom výkrese na použitie v aktuálnom výkrese, možno použiť príkaz DesignCentrum z menu nástroje, v ktorého ľavom okne v štýle prieskumníka daný výkres nájdete, rozviniete jeho zdieľané objekty, zvolíte Typy čar a požadovanú čiaru pretiahnete myšou do pracovného okna aktuálneho výkresu (obr. 3.2)



Obr. 3.2 Získanie definície typu čiary pomocou DesighCenter

3.2 Tvorba vlastných jednoduchých užívateľských typov čiar

Aj keď AutoCAD ponúka najbežnejšie typy čiar používané na tvorbu grafickej projektovej dokumentácie, nemusia pre určitý prípad vyhovovať napríklad medzery v prerušovanej čiare, alebo potrebujeme sekvenciu iného počtu opakujúcich sa čiarok alebo bodiek. Typy čiar skladajúce sa iba z týchto prvkov (čiarky, medzery, bodky) sú tzv. jednoduché typy čiar (obr. 3.3). Pre ich vytvorenie možno použiť príkaz Typč (Linetype).



Obr. 3.3 Definícia čiary

Vytvorenie nového typu čiary pomocou príkazu Typč(český AutoCAD)

10. Zadajte na príkazovom riadku AutoCADu **–typč** (znamienko mínus spôsobí vyvolanie príkazu vo forme bez dialógového panelu, nakoľko len takto je dostupná voľba na tvorbu čiar)
11. Zadejte volbu [?/Definuj/Načti/naStav]: **d**
12. Zadejte jméno vytvářeného typu čáry: **Vlastna**
13. Ďalej je potrebné zadať buď názov nového súboru, alebo existujúceho súboru, do ktorého sa pridá nová definícia. Predpokladajme, že chcete vytvoriť nový súbor. Do políčka pre názov súboru zadajte **MojeCiary**.
14. Popisný text. Tu vložte popis typu čiary dlhý maximálne 47 znakov. Popis môže byť komentár alebo postupnosť podtržníkov, bodiek, pomlčiek a medzier, ktoré sú jednoduchou reprezentáciou vzoru typu čiary.
15. Zadajte vzor typu čiary:

A,

Zadajte definíciu vzoru ako postupnosť čísiel oddelenú čiarkami. Na definíciu dĺžok čiar použite kladné hodnoty, na definíciu dĺžok medzier použite záporné hodnoty. Na reprezentáciu bodky použite nulu.

Písmeno "A" vo výzve k definícii vzoru určuje zarovnanie vzoru použitého na koncoch samostatných úsečiek, kružníc a oblúkov. Pomocou zarovnania typu A je zaistené, že čiary a oblúky začínajú a končia čiarkou. A je automaticky obsiahnuté v definícii. Ak pre vytvorenie typu čiary použijete textový editor, je treba zadať A na začiatku definície.

Po vytvorení typu čiary je potrebné ho načítať, čím sa tento typ sprístupní na priraďovanie rôznym objektom.

Po vytvorení súboru LIN a definovaní novej čiary v ňom príkazom Typč si môžeme tento súbor otvoriť napríklad pomocou programu Notepad, ktorý číta a modifikuje textové súbory.

Vidíme, že typ čiary je definovaný v dvoch riadkoch v súbore definície typu čiary. Prvý riadok obsahuje názov typu čiary a voliteľný popis. Druhý riadok je kód, ktorý definuje vlastný vzor typu čiary. Ako sme videli pri použití príkazu Typč druhý riadok musí začínať písmenom A (zarovnanie), nasledujúcim zoznamom popisov vzorov, ktoré definujú dĺžky vypnutého pera (medzery), dĺžka zapnutého pera (čiarky) a bodky. Ak na začiatku riadku napíšete bodkočiarku môžete do súboru LIN vložiť komentáre.

Formát definície typu čiary teda je:

**názov_typu_čiary,popis*
A,popis1,popis2,...

Napríklad typ čiary s názvom BODKOČIARKOVANÁ je definovaný ako:

** BODKOČIARKOVANÁ,Bodkočiarkovaná__ . __ . __ . __ . __ . __ . __*
A,.5,–.25,0,–.25

Označuje to opakujúci sa vzor začínajúci čiarkou dlhou 0.5 výkresových jednotiek, medzeru dlhú 0.25 výkresových jednotiek, bodkou a ďalšou medzerou dlhou 0.25

výkresových jednotiek. Tento vzor pokračuje po celej dĺžke riadku, končí čiarkou dlhou 0.5 výkresových jednotiek.

Ďalej uvádzame súhrnne informácie o každom poli v definícii typu čiary.

Názov typu čiary

Pole názvu typu čiary začína hviezdičkou (*) a musí obsahovať jedinečný názov typu čiary.

Popis

Pri úprave súboru LIN by mal popis typu čiary pomáhať predstaviť si typ čiary. Popis je taktiež zobrazený v správcovi typov čiar a v dialógu Načítať typy čiar.

Popis je voliteľný a môže obsahovať:

- Jednoduché znázornenie vzoru typu čiary pomocou ASCII textu
- Rozšírený popis typu čiary
- Komentár, napríklad "Typ čiary určený pre skryté hrany"

Ak popis vynecháte, nevkladajte za názvom typu čiary čiarku. Popis nesmie presiahnuť 47 znakov.

Pole zarovnania (A)

Pole zarovnania určuje akciu pre zarovnanie vzoru na koncoch jednotlivých čiar, kružníc a oblúkov. AutoCAD aktuálne podporuje iba zarovnanie typu A, ktoré zaručuje, že koncové body čiar a oblúkov budú začínať a končiť čiarkou.

Predpokladajme napríklad definíciu typu čiary OS, ktorá bude definovaná opakujúcim sa vzorom čiarky a bodky. AutoCAD upraví sekvenciu čiarok a bodiek, na každej čiare tak, že na koncových bodoch objektu začne a skončí čiarka. Pokiaľ to vzor dovolí, posunie ju AutoCAD tak, aby čiaru začínala a končila aspoň polovica dĺžky príslušnej čiarky. Pokiaľ je to potrebné, sú prvá a posledná čiarka predĺžené. Pokiaľ je čiara príliš krátka a nevojde sa na ňu ani jedna sekvencia čiarky a bodky, AutoCAD nakreslí medzi obidvoma koncovými bodmi súvislú čiaru. Pri oblúkoch je vzor upravený tak, aby sa čiarky kreslili v koncových bodoch. Kružnice nemajú koncové body, AutoCAD úpravou sekvencie čiarok a bodiek ponúkne rozumné zobrazenie.

Popisy vzorov

Každé pole popisov vzorov určením dĺžky segmentov vytvorí typ čiary. Údaje sú oddelené čiarkami (medzery nie sú povolené):

- Kladné desatinné číslo určuje segment dĺžky zapnutého pera (čiarky) danej dĺžky.
- Záporné desatinné číslo určuje segment vypnutého pera (medzery) danej dĺžky.
- Dĺžka 0 nakreslí bodku.

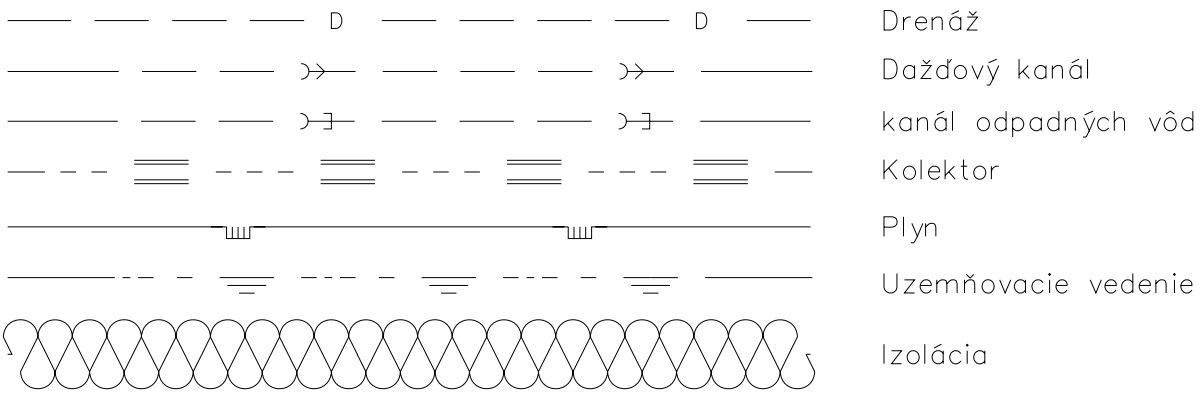
Jedna definícia môže obsahovať až 12 vzorov (za predpokladu, že dĺžka riadku neprekročí 80 znakov v súbore LIN. Vložiť je nutné iba jedno kompletné opakovanie vzoru typu čiary definované popismi vzorov. Po nakreslení typu čiary použije AutoCAD prvý popis

vzoru pre počiatočnú a koncovú čiarku. Medzi prvou a poslednou čiarkou sa vykresľuje vzor sekvenčne, začína na druhej pozícii a v prípade potreby znovu cyklicky od prvej pozície.

Zarovnanie typu A vyžaduje, aby prvý úsek bol 0, alebo kladné číslo (zapnuté pero). Pokiaľ potrebujete prázdny segment, mala by byť dĺžka druhej čiarky menšia než nula, pri plnej čiare potom väčšia ako nula. Vzor pre zarovnanie typu A musí obsahovať aspoň dve špecifikácie čiarok.

3.3 Vytváranie zložitých typov čiar

Zložitým typom čiary sa myslí taká čiara, ktorá obsahuje text alebo nejakú zvláštnu značku. Napríklad plot sa zvykne označovať s vloženým písmenom X. Ako zdroj značiek do čiar slúžia fonty AutoCADu alebo si môže užívateľ sám vytvoriť vlastnú značku zadefinovaním v súbore SHX na základe znalosti príslušných konvencií. Zložité typy čiar nie je možné vytvárať pomocou príkazu Typč. V tomto prípade je už nutné pracovať priamo s textovým súborom LIN, čo bol aj dôvod prečo sme sa s jeho štruktúrou v predchádzajúcej časti zoznámili.



Obr. 3.4 Príklady zložitých typov čiar

Syntax zložitých typov čiar je podobná syntaxe jednoduchých typov čiar v tom, že sa jedná o čiarkami oddelený zoznam vzorov. Zložité typy čiar môžu ako popisy vzorov okrem spomenutých základných prvkov ako sú čiarky, medzery a bodky obsahovať navyše aj tvary a textové objekty, iba s odlišnou syntaxou uzavretou v hranatých zátvorkách.

Znaky z existujúcich textových typov písma je možné zahrnúť do definícií čiar pomerne jednoducho. Typy čiar s vloženými znakmi môžu označovať inžinierske siete, hranice, vrstevnice a podobne. Podobne ako jednoduché typy čiar sú čiary kreslené dynamicky podľa užívateľom určených vrcholov grafických entít. Textové znaky obsiahnuté v čiarach sú vždy zobrazené kompletne, nie sú nikdy skrátené.

Vložené textové znaky sú asociované so štýlom textu vo výkrese. Ľubovoľné štýly textu asociované s typom čiary musia existovať vo výkresoch pred načítaním typu čiary.

Formát pre vkladanie textových znakov v popise typov čiary je nasledujúci:

["text",názovštýlutextu,merítko,otočenie,xposun,yposun]

Tento formát je pridaný ako popis k jednoduchému typu čiary. Napríklad typ čiary s názvom PRÍVOD_ZAVLAHOVEJ_VODY je definovaný takto:

**PRÍVOD_* ZAVLAHOVEJ*_VODY,---- PZV ---- PZV ---- PZV ---- PZV ----*
A,.5,-.2,["PZV",STANDARD,S=.1,R=0.0,X=-0.1,Y=-.05],-.3

Táto definícia označuje opakujúci sa vzor začínajúci čiarkou dĺžky 0,5 výkresových jednotiek, medzerou dĺžky 0,2 výkresových jednotiek, znaky PZV s určitými parametrami merítka a umiestnenia a ďalšou medzerou dĺžky 0,3 výkresových jednotiek. Textové znaky pochádzajú z textového písma s priradeným štýlom textu STANDARD s merítkom 0,1, relatívnym otočením 0 stupňov, posunom v osy S veľkosti –0,1 a posunom v osy Y veľkosti 0,55. Tento vzor sa opakuje po celej dĺžke čiary a končí čiarkou dĺžky 0,5 výkresových jednotiek. Typ čiary by sa zobrazil spôsobom zobrazeným dole.

Obr. 3.5 Zložitý typ čiary s textom

Všimnite si, že celková dĺžka ťahu je 0,5 + 0,2 + 0,3 = 1,0 a že začiatok textu je odsadený o –0,1 jednotky v smere osy X od konca prvého ťahu. Na ujasnenie týchto parametrov a ich vzájomného pomeru môžeme ten istý typ čiary napísať aj týmto odlišným spôsobom:

**PRÍVOD_* ZAVLAHOVEJ*_VODY,---- PZV ---- PZV ---- PZV ---- PZV ----*
A,.5,-.1,["PZV",STANDARD,S=.1,R=0.0,X=0,Y=-.05],-.4

Celková dĺžka ťahu je stále 0,5 + 0,1 + 0,4 = 1,0, ale začiatok textu nie je potrebné v hranatej zátvorke odsadzovať v smere osy X.

V nasledujúcej tabuľke sú zhrnuté informácie o každom poli v popise znakov.

Tabuľka 3.1 Popis parametrov textu v definícii typu čiary

Parameter	Popis parametra
Text	Textové znaky, ktoré sa majú použiť v type čiary.
Názov štýlu textu	Názov štýlu textu, ktorý sa má použiť. Ak nie je určený štýl textu, použije AutoCAD aktuálny štýl.
Merítko	S = hodnota. Merítko, ktoré sa má použiť pre štýl textu je relatívne k merítku typu čiary. Výška štýlu textu je násobená merítkom. Ak je výška textu v štýle určená ako 0, pre výšku sa použije samotná hodnota pre S = hodnota.
Otočenie	R = hodnota alebo A = hodnota. R = určuje relatívne alebo kolmé otočenie vzhľadom k čiare. A = určuje absolútne otočenie textu vzhľadom k súradnému systému, to znamená, že celý text má rovnaké otočenie bez ohľadu na jeho umiestnenie vzhľadom k čiare. K číslu môže byť pripojený symbol d pre stupne (čo je východzia možnosť), r pre radiálne alebo g pre grády. Pokiaľ je toto pole vypustené, chová sa definícia ako R = 0.
Xposun	X = hodnota. Posunutie textu v osy X typu čiary, čo znamená pozdĺž čiary. Pokiaľ je hodnota Xposun vynechaná alebo je 0, text sa vytvorí bez posunutia. Pomocou tohto poľa určíme vzdialenosť medzi textom a predchádzajúcim ťahom vypnutého alebo zapnutého pera. Táto hodnota nie je zväčšená merítkom definovaným hodnotou S = hodnota.
Yposun	Y = hodnota. Posunutie textu v ose Y čiary, čo znamená v uhle 90 stupňov od čiary. Pokiaľ je hodnota Yposun vynechaná alebo je 0, text sa vytvorí bez posunutia.

Tvary v užívateľských typoch čiar

Zložité typy čiar môžu obsahovať aj tzv. tvary, ktoré sú uložené v súboroch tvarov. Tým sa v podstate ich tvorba veľmi neodlišuje od tvorby definícií čiar s textami, nakoľko aj znaky (písmená) textu sú zadefinované v súbore fontu. Rozdiel je v praxi v tom, že značku tvaru často nemáme zadefinovanú v nejakom prebratom súbore a musíme ju zadefinovať sami. Zložité typy čiar s tvarmi (značkami) sa často využijú na označovanie inžinierskych sietí, hraníc a pod. Tvary sú zadefinované v súboroch s popismi tvarov. Tento súbor obsahuje kód popisujúci ľubovoľný tvar, ktorý chcete opakovane používať. Napríklad aj niektoré písma používané v AutoCADe sú zadefinované pomocou súborov tvarov. Tieto súbory majú príponu SHX. Výhodou tvarov je, že relatívne jednoduchú geometriu zobrazujú rýchlo, čo je výhodné pri prekresľovaní výrezu, regenerácii a pod.

Takisto ako jednoduché typy čiar sú aj zložité typy čiar kreslené dynamicky podľa definície vrcholov (alebo koncových bodov) užívateľom. Tvary a textové objekty začlenené v čiarach sú vždy zobrazené kompletne, nie sú nikdy skrátené.

Syntax definície tvarov v popise typu čiary je nasledujúca:

[názovtvaru,názovsúborushx]
alebo: *[názovtvaru,názovsúborushx,transformacia]*

Z uvedeného vyplýva, že transformácia nie je povinná a keď ju uvádzame môže sa skladať zo série nasledujúcich prvkov oddelených čiarkami:

R =## Relatívne otočenie
A =## Absolútne otočenie
S =## Merítko
X =## X posun
Y =## Y posun

V tejto syntaxe ## znamená desatinné číslo – hodnotu parametra – so znamienkom (1, –17, 0,01 a pod.), otočenie je v stupňoch a ostávajúce atribúty vo výkresových jednotkách podľa merítka typu čiary. Písmena uvádzajúce transformáciu musia byť bezprostredne nasledované znakom = a číslom, medzery nie su povolené.

Nasledujúca definícia popisuje čiaru s názvom KRUH1 skladajúcu sa z opakujúceho sa vzoru čiarka, medzera a vložený tvar KRUH zo súboru kruh.shx (aby nasledujúci príklad fungoval, musí byť súbor kruh.shx najprv vytvorený postupom, ktorý ukážeme ďalej a uložený v ceste podporných súborov).

**KRUH1,S krúžkom- - o - -*
A,15,-10,[KRUH,kruh.shx,y=.2,s=1],–10

Ako vidíme, mimo kódu v hranatých zátvorkách zodpovedá táto definícia popisu jednoduchého typu čiary.

Pre popis vloženého tvaru v hranatých zátvorkách môže byť použitých celkovo až šesť polí. Prvé dve sú povinné (názov tvaru, názov zdrojového súboru SHX) a s predpísaným umiestnením na začiatku, ďalšie štyri sú voliteľné a je možné ich radiť podľa potreby (R alebo A, a ďalej S, X, Y).

Nasledujúca syntax potom definuje tvar ako súčasť zložitého typu čiary.

[názovtvaru,názovsúborutvaru,merítko,otočenie,xposun,yposun]

Nasledujú definície polí v tejto syntaxi:

Názov tvaru

Názov kresleného tvaru. Toto pole musí byť prítomné. Pokiaľ chýba, nie je definícia platná. Pokiaľ názov tvaru neexistuje v zadanom súbore tvaru, čiara sa vykreslí bez vloženého tvaru.

Názov súboru tvaru

Názov súboru definície kompilovaného tvaru (SHX). Pokiaľ chýba, nie je definícia platná. Pokiaľ názov súboru tvaru neobsahuje úplnú cestu, bude súbor hľadaný v prehľadávanej ceste knižnice pre daný súbor. Pokiaľ názov súboru tvaru úplnú cestu obsahuje, ale v zadanom mieste nie je súbor nájdený, bude súbor hľadaný v prehľadávaných cestách systému. Pokiaľ súbor nie je nájdený, čiara sa nakreslí bez tvaru.

Merítko, Otočenie, Xposun, Yposun – podobne ako pri vkladaní textu

Vytváranie zložitých typov čiar so značkami je v mnohých ohľadoch podobné vytváraniu jednoduchých čiar alebo čiar s textom. Základná bariéra, ktorú je potrebné v tejto problematike prekonať, je ako už bolo spomenuté vytváranie vlastných súborov tvarov. Vytváranie súborov tvarov ako textových súborov v textovom editore zahŕňa písanie pomerne

zložitého kódu, ktorého vysvetlenie presahuje rámec týchto skrípt. Našťastie však Autodesk poskytuje medzi inými užitočnými nástrojmi v rozširujúcom balíku Expres Tools aj nástroj na vytváranie súboru tvarov a definovanie tvarov, ktorý je veľmi pohodlný, pretože sa textový súbor generuje na základe grafickej reprezentácie tvaru bežnými kresliacimi prostriedkami AutoCADu. Express Tools nie su nainštalované automaticky, je potrebné ich pri inštalovaní zvoliť z ponuky inštalačného programu. Na základe toho sa vytvorí v AutoCADe menu Express, v ktorého časti Tools je obsiahnutý príkaz MKSHAPE. Pred jeho použitím je potrebné najprv nakresliť vyžadovaný tvar bežnými kresliacimi príkazmi: úsečka, krivka, oblúk, kružnica a elipsa. Pri ich kreslení je potrebné používať obyčajnú Continuous čiaru; keby boli aj vytvorené iným typom čiary príkaz to nebude brať do úvahy. V našom prípade vytvorenia čiary KRUH1 uvedenej vyššie nakreslite kružnicu s polomerom 4.

Ďalej sa pokračuje takto:

16. Příkaz: **mkshape**
    Inicializace... *(ozvena AutoCADu)*
17. Vyžaduje zadať meno súboru tvarov
    Reading shape file: C:\Výkres1.shp..|Done. *(ozvena AutoCADu)*
18. Enter the name of the shape: **kruh** *(zadajte názov tvaru)*
19. Enter resolution <128>: **Enter** *(zadajte veľkosť rozlíšenia)*
20. Specify insertion base point: *(vkladací bod – v našom prípade stred kruhu)*
21. Vyberte objekty: nalezeno: 1 *(vyberieme krúžok)*
22. Vyberte objekty: **Enter**

Súbor, ktorý sa po úspešnom prebehnutí príkazu mkshape vytvorí potom môžeme používať ako zdroj pre značky v definíciách čiar už uvedeným spôsobom. Vhodnejšie je jeho umiestnenie na prehľadávanú cestu:

C:\Documents and Settings\užívatel\Data aplikací\Autodesk\AutoCAD 2006\R16.2\csy\Support

Poznámka. Ak pri výbere shp súboru vyberiete existujúci súbor, dialógový panel upozorňuje, že bude prepísaný. V skutočnosti ale k súboru bude pridaná ďalšia definícia – až na prípad, že by bol zadaný rovnaký názov tvaru aký už v súbore existuje – v tomto prípade bude predchádzajúci tvar prepísaný, ale zvyšok súboru ostane zachovaný.

Pokiaľ ide o veľkosť rozlíšenia zadávanú v priebehu príkazu, vyššie zadané hodnoty vedú k vytvoreniu presnejších tvarov (ich vernejšiu zhodu s vektorovou kresbou z ktorej boli generované) a nižšie hodnoty menej presné tvary, ale rýchlejšie prekresľované napr. pri regenerácii výkresu. Niekedy je potrebné s touto hodnotou experimentovať – napríklad ak má tvar obsahovať malú súčasť – napr. bodku, ktorú nakreslíme krátkou čiarkou, môže sa stať, že pri niektorom rozlíšení sa už bodka prakticky nezobrazí a je potrebné rozlíšenie zvýšiť.

Dobrá správa je, že Express Tools obsahujú aj graficky orientovaný nástroj na vytvorenie zložitej užívateľskej čiary, ktorým je príkaz MKLTYPE. Tento vytvára nový typ čiary na základe vyselektovaných objektov v pracovnom priestore AutoCADu.

Pred jeho použitím je potrebné najprv nakresliť vyžadovanú zostavu entít bežnými kresliacimi príkazmi: úsečka, krivka, bod, tvar a text. Pri ich kreslení je potrebné používať obyčajnú Continuous čiaru; keby boli aj vytvorené iným typom čiary príkaz to nebude brať do úvahy. Použitie je takéto:

23. Příkaz: **mkltype**
    Inicializace... *(ozvena AutoCADu)*
24. Enter linetype name: **MojaCiara** *(zadajte názov novej čiary)*

25. Enter linetype description: **PopisMojejCiary** *(zadajte popis novej čiary)*
26. Specify starting point for line definition: *(špecifikujte počiatočný bod definície)*
27. Specify ending point for line definition: *(špecifikujte koncový bod definície)*
28. Vyberte objekty: nalezeno: 4
29. Vyberte objekty: **enter**

Ak pri výbere .lin súboru vyberiete existujúci súbor, dialógový panel upozorňuje, že bude prepísaný. Ako v predchádzajúcom prípade bude iba pridaná ďalšia definícia – až na prípad, že by bol zadaný rovnaký názov čiary aký už v súbore existuje – v tomto prípade bude iba táto prepísaná, ale zvyšok súboru zostane zachovaný.

4 VYKRESĽOVANIE VÝKRESOV

Vykresľovanie v AutoCADe je pomerne komplexnou problematikou, ktorá zahŕňa viacero prvkov. Ďalej ich uvádzame v určitom poradí, nie však vždy nevyhnutne v tom istom ako budete aplikovať pri konkrétnom kreslení svojej práce v AutoCADe.

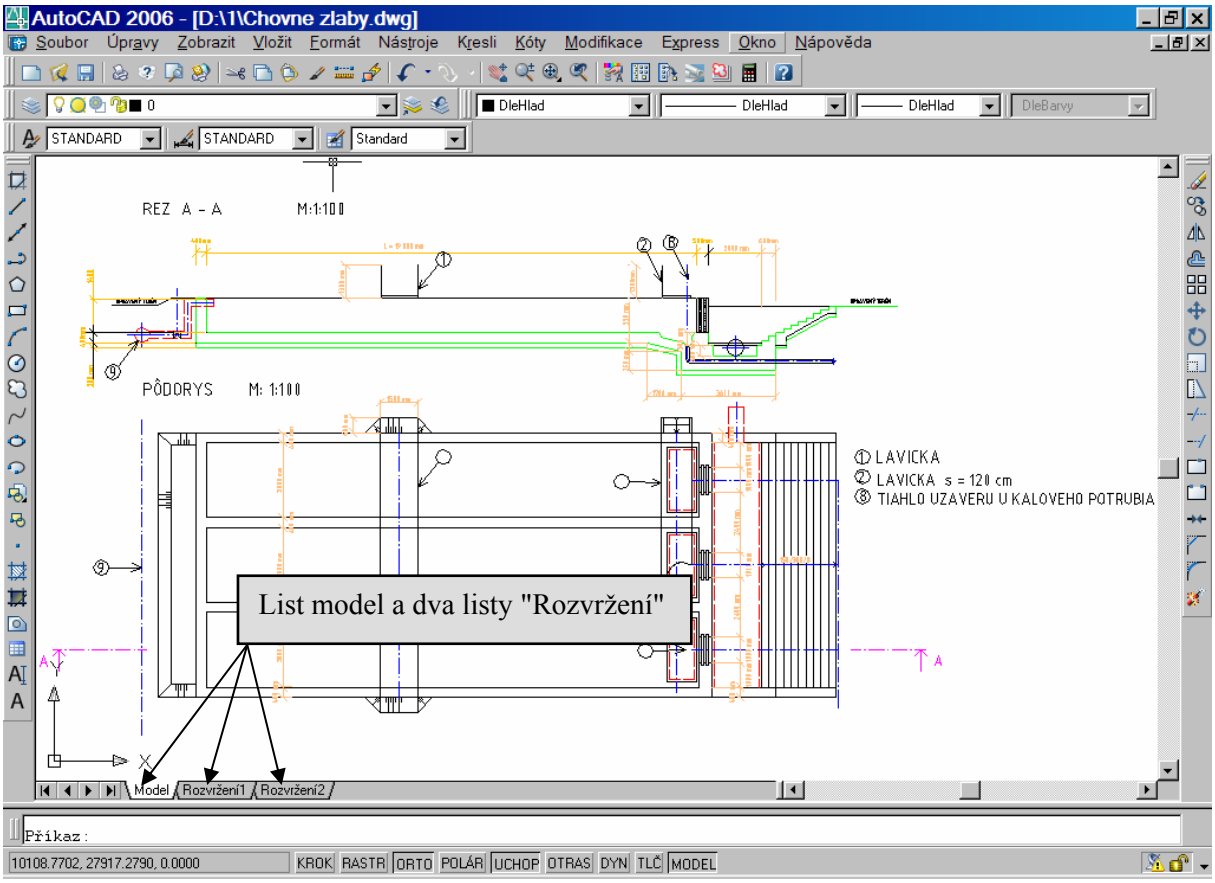
Na vykresľovanie sú potrebné vedomosti o plotri, ktorý bude slúžiť k vykresleniu a o jeho možnostiach, nastaveniach a uchovávaní týchto nastavení. Niekedy je potrebná jeho konfigurácia v správcovi plotrov. Správca plotru je dialógové okno AutoCADu, ktoré obsahuje zoznam súborov s konfiguráciou plotru (PC3) pre každú nesystémovú tlačiareň, ktorú inštalujete. Konfiguračné súbory plotrov môže byť účelné vytvoriť aj pre tlačiarne systému Windows®, pokiaľ chcete použiť východiskové vlastnosti odlišné od vlastností použitých v systéme Windows. Nastavenie konfigurácie plotru zahrňujú informácie o porte, kvalitu rastrovej a vektorové grafiky, rozmery papieru a ďalšie vlastnosti, ktoré závisia od typu plotra.

Ďalej je potrebné vedieť obsluhovať Dialógový panel na vykresľovanie, ktorý zahŕňa výber a nastavenie tlačiarne, papieru, vykresľovanej oblasti – časti výkresu, merítko a tzv. štýl vykreslenia, t. j. zjednodušene povedané priradenie toho, ktoré čiary budú mať akú farbu, hrúbku a iné vizuálne vlastnosti. Štýl vykreslenia je z nastavení v Dialógovom paneli na vykresľovanie najobsiahlejšou témou, ostatné sú pomerne jednoduché a zrozumiteľné parametre.

Dôležitým rozhodnutím a zvláštnou kapitolou pri voľbe spôsobu vykresľovania je to, či budete svoj výkres kresliť z modelového alebo výkresového priestoru, t. j. či budete tvoriť tzv. *layout* čiže v českom AutoCADe *rozvržení*, alebo nie. Postup vykresľovania pomocou *layouts* volíme hlavne pri zložitejších výkresoch, napríklad keď chceme mať vo výkrese zobrazený ten istý objekt z rôznych pohľadov, v rôznych mierkach, s rôznymi zapnutými/vypnutými hladinami prípadne s textami alebo rozpiskami, ktoré nie su súčasťou kresby v modelovom priestore alebo sú dopredu pripravené v tzv. šablóne výkresu. Keďže zo skúsenosti s vedomosťami bežného užívateľa – študenta stavebnej fakulty považujeme mnohé problémy okolo vykresľovania za pomerne známe, sústredíme sa v kapitole o vykresľovaní najmä na problematiku výkresového priestoru a štýlov vykresľovania pri nastavení dialógového panelu *Vykresli*.

4.1 Výkresový priestor

Existujú dve rôzne pracovné prostredia, alebo "priestory", v ktorých môžete vytvárať objekty vo výkrese a zároveň z ktorých môžete vytvorené predmety vykresľovať. V AutoCADe vlastne tvoríte určitý model reality zložený z geometrických objektov – tento je väčšinou vytvorený v trojrozmernom súradnom priestore známom ako *modelový priestor*. Konečné rozvrhnutie pohľadu na model, ktoré chceme dokumentovať v grafickej časti projektu, prípadne popis rohové razítko a pod. daného modelu je možné pripraviť v dvojrozmernom priestore nazývanom *výkresový priestor*. Tieto priestory sú prístupné na dvoch alebo viac listoch, ktorých "ušká" vidíte dole v grafickej oblasti: list *Model* a jeden alebo viac listov rozvrhnutia (*Rozvržení* v českom a *Layout* v anglickom AutoCADe).



Obr. 4.1 Výkresový a modelový priestor

Pri využívaní oboch priestorov môžete postupovať tak, že v prvej fáze pri práci na karte *Model* kreslíte iba model predmetu (geometriu) zvyčajne v mierke 1 : 1. Je to dobrý zvyk, ktorý má svoju výhodu pri kótovaní, prípadne zisťovaní rôznych informácií z výkresu ako sú dĺžky, plochy a podobne. Tak isto je výhodné zadávať pri kreslení objektov priamo ich rozmery a neprepočítavať ich podľa mierky. Po vytvorení tohto modelu (v 2D alebo v 3D) pokračujeme v práci v "*rozvržení – layout*" tak, že do tohto výkresového priestoru umiestňujeme jeden alebo viac *výrezov* (rôzne modifikovaných pohľadov na modelový priestor a v ňom existujúcich objektov) a tieto pohľady doplníme ďalšími objektami – kótami, poznámkami, rohovými pečiatkami a pod. Čiže v modelovom priestore sa zaoberáte objektom a jeho geometriou a v "rozvržení" pripravujete list výkresu tak ako ho chcete vykresliť. To samozrejme neznamená, žeby kóty, poznámky a rohové pečiatky nebolo možné robiť v modelovom priestore – po získaní informácií o práci s kartami *Rozvržení* sa však viete kvalifikovanejšie rozhodnúť pre optimálny postup vhodný pre konkrétny výkres. Menšie výkresy pravdepodobne zostanú kompletne vyvárané aj vykresľované cez modelový priestor, pri väčších môže byť vhodnejší popisovaný kombinovaný prístup s modelom a "rozvržením".

Hlavným pojmom pri vytváraní obsahu *Rozvržení* je *výrez*, ktorý slúži ako okno, alebo pohľad do priestoru modelu. Každý takýto výrez obsahuje pohľad, ktorý zobrazuje model v určenej mierke a smere. Tým je povedané, že môžete v rámci jedného listu vykresľovať pohľady na model v rôznom merítku, t. j. napr. celkový pohľad a jeden alebo viac detailov modelu. Podobne môžete určiť, ktoré hladiny budú v každom výreze viditeľné – t. j. vášmu pohľadu – výrezu nasadíte isté "okuliare" – celok vidím, detaily nevidím a podobne.

Súhrnne – po vytvorení modelu objektu v liste *Model* použite pri príprave *"rozvržení – layout"* nasledujúce kroky:

1. Klepnite na záložku *"rozvržení – layout"*. Všetko ďalšie robíte v rámci tejto záložky.
2. Definujte nastavenie vzhľadu stránky (v správcovi nastavenia stránky), ako je vykresľovacie zariadenie, veľkosť výkresu, oblasť vykreslenia, mierka vykreslenia a orientácia výkresu.
3. Ak chcete vložte rohovú pečiatku (pokiaľ nevytvárate výkres zo šablóny výkresu, ktorá už pre "rozvržení" rohovú pečiatku obsahuje).
4. Vytvorte novú hladinu, ktorá sa použije pre výrezy.
5. Vytvorte výrezy v tejto hladine a umiestnite ich.
6. Nastavte orientáciu, mierku a viditeľnosť hladín v každom výreze.
7. Podľa potreby pridajte do *"rozvržení – layout"* kóty, poznámky a ďalšie objekty.
8. Keď skončíte s usporiadaním *"rozvržení – layout"*, vypnite hladinu, ktorá obsahuje objekty hraníc výrezov. Obsah pohľadov zostane viditeľný a *"rozvržení – layout"* môžeme vykresliť aj bez zobrazenia rámčekov výrezov.
9. Vytlačte.

Implicitne sa nový výkres otvára s dvoma kartami *Rozvržení* pomenovanými *Rozvržení1* a *Rozvržení2*. Ak použijete výkresovou šablónu alebo otvoríte existujúci výkres, listy *"rozvržení – layout"* môžu mať rôzne názvy. Nový list *rozvržení* možno vytvoriť pomocou jednej z nasledujúcich metód:

- Pridajte nový list *"rozvržení – layout"* bez nastavenia a potom určite nastavenie v Správcovi nastavení stránky (z menu Súbor alebo klepnutím pravým tlačidlom myši na list *rozvržení*, čím sa zobrazí miestne menu s voľbami).

- Pomocou sprievodcu *"Vytvoriť rozvržení"* vytvorte nový list a určite nastavenia (Nástroje – Průvodci – Vytvořit rozvržení).

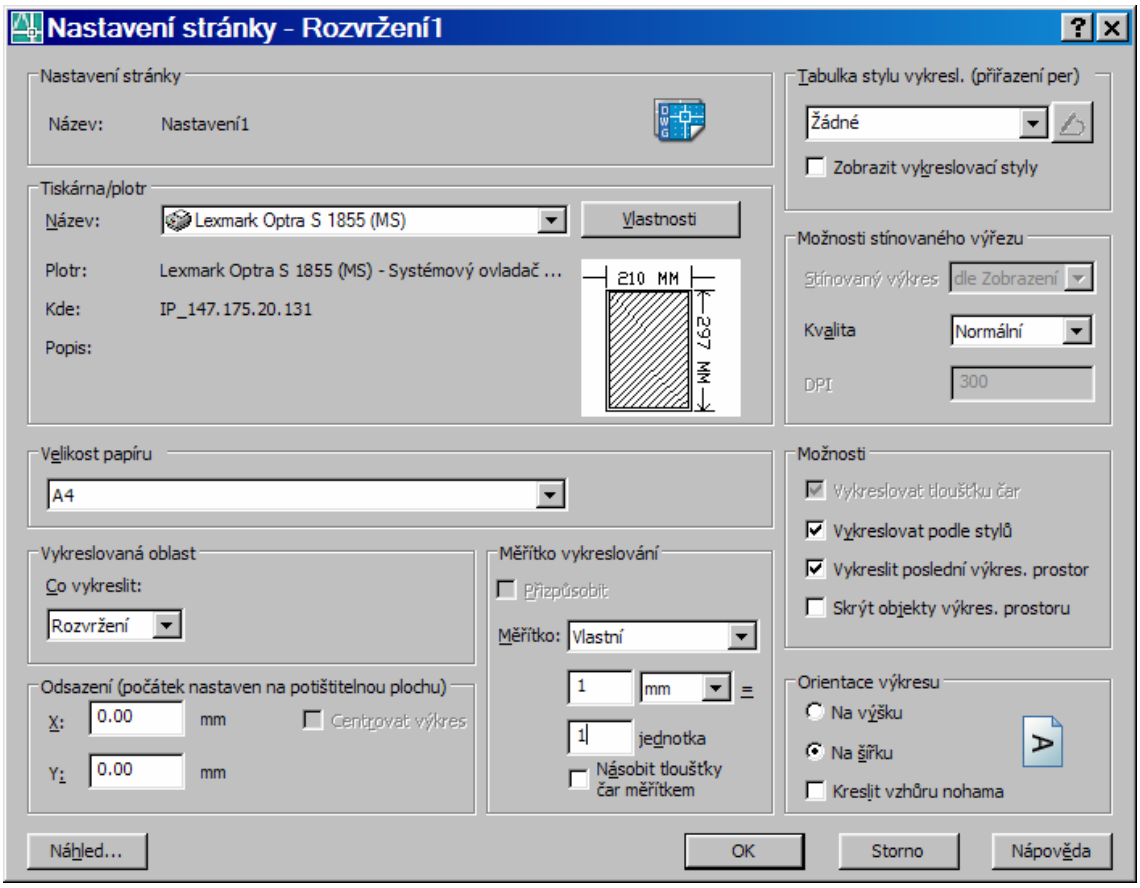
- Skopírujte list *rozvržení* a jeho nastavenia z iného výkresového súboru (použite DesignCenter z menu Nástroje.

- Importujte z existujúceho súboru výkresové šablóny (DWT) alebo výkresového súboru (DWG) pomocou Správcu nastavenia stránky.

Nové *rozvržení* môžete vytvoriť aj pomocou sprievodcu *Vytvořit rozvržení*. Sprievodca vyžaduje najmä tieto informácie:

- Názov nového *rozvržení*
- Tlačiareň pripojenú k *rozvržení*
- Veľkosť papiera
- Orientácia výkresu na papieri
- Nastavenia a umiestnenia výrezu(ov)

Informácie vložené pomocou sprievodcu môžete neskôr upravovať.



Obr. 4.2 Zadávanie parametrov pre layout – rozvržení v správcovi nastavenia stránky

Pri vytváraní nového súboru používame súbor šablóny, ktorá obsahuje počiatočné nastavenia výkresu. Tieto nastavenia sa týkajú ako modelového tak aj výkresového priestoru, a preto sa zmienime aj o súvislostiach medzi šablónami a *"Rozvrženími – Layouty"* a ich prvotnými nastaveniami, keďže sa zo šablón preberajú.

Súbor šablóny výkresu obsahuje zadefinované nastavenia. Môžeme vybrať niektorú z existujúcich šablón alebo si vytvoriť šablónové súbory sami. Šablóny výkresu majú príponu .dwt. Keď vytvoríte nový výkres pomocou existujúcej šablóny a vo výkrese vykonáte zmeny, neovplyvnia tieto zmeny pôvodnú šablónu.

Dobrou pomôckou je to, že môžete nastaviť šablónu s ktorou sa Vám bude AutoCAD otvárať pomocou zadania prepínača /t takto:

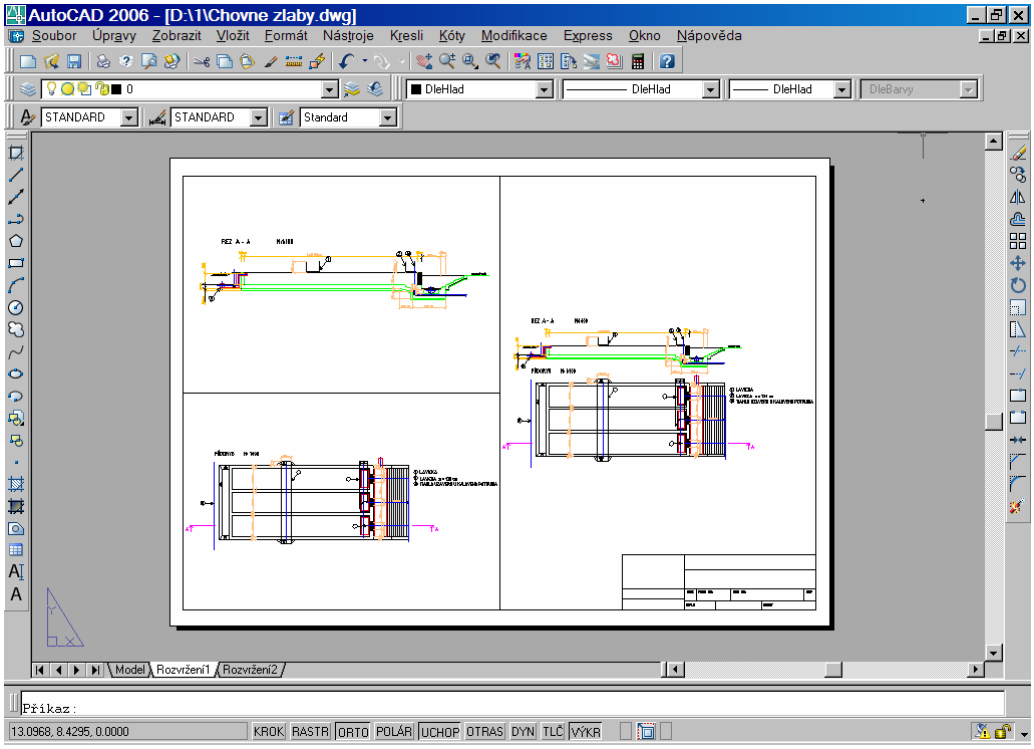
"C:\Program Files\AutoCAD 2006\acad.exe" /t MojaSablona.dwt

Tento údaj zadávate ako Target v Properties (Vlastnostiach), ku ktorým sa dopracujete kliknutím pravým tlačidlom myši na ikonu AutoCADu na ploche v záložke Shotcut.

Keď vytvárate "*Rozvržení – Layout"*, môžete si zvoliť, či má byť použitá informácia z existujúcej šablóny. Šablónu možno v tomto prípade importovať zo súboru DWG, DXF alebo DWT. V novom *"Rozvržení – Layout"* sú použité objekty výkresového priestoru a nastavenia stránky z existujúcej šablóny. Všetky objekty (rozpisky, výrezy...) sú teda zobrazené vo výkresovom priestore.

Každý výkres môže byť uložený ako šablóna výkresu (súbor DWT) so všetkými nakreslenými objektmi a nastaveniami (bloky, hladiny, kódovacie štýly) modelového aj výkresového priestoru. Možno použiť aj voľbu *Uložit jako* príkazu *ROZVRŽENÍ*. Súbor

šablóny je uložený v priečinku súborov šablón výkresu definovanom na karte *Soubory* dialógu *Možnosti*.
Pomocou okna *Návrhové centrum™* môžete pretiahnuť *"Rozvržení – Layout"* aj s objektami z ľubovoľného výkresu do aktuálneho výkresu.



Obr. 4.3Vkladanie výrezov

Po týchto základných nastaveniach rôznymi postupmi už nasleduje samotné zostavenie obsahu daného výkresu. Ako bolo zmienené, základnou akciou je vkladanie výrezov. Je možné vytvoriť jeden výrez, ktorý zaberie celý výkres, alebo v *"rozvržení – layout"* vytvoriť viac výrezov. Príkaz *MPOHLED* (tiež z menu *Zobraziť – Výrezy*) ponúka niekoľko volieb pre tvorbu jedného nebo viacerých výrezov (v rôznom usporiadaní). Keď vytvoríte výrezy, môžete meniť ich veľkosť napríklad ťahaním za uzly, zväčšovať ich alebo nimi hýbať podľa potreby.

Príkaz *MPOHLED* ponúka aj dve voľby, ktoré pomáhajú pri definovaní nepravidelne tvarovaného výrezu. Pomocou voľby *Objekt* (tiež je prístupný z menu *Zobraziť – Výrezy*) môžete vybrať uzavretý objekt, napríklad kružnicu nebo uzatvorenú krivku vytvorenú vo výkresovom priestore, a previesť ju na výrez. Objekt, ktorý definuje rámček výrezu, je združený s vytvoreným výrezom. Voľbu *Polygón* je možné použiť na vytvorenie nepravouhlého výrezu výberom bodov. Postupnosť výziev je rovnaká ako postupnosť výziev pri tvorbe krivky.

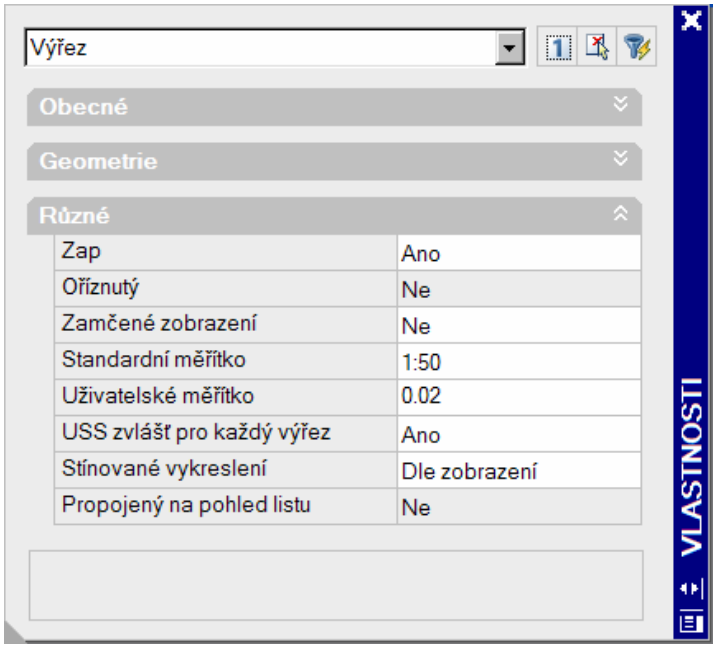
Rámček existujúceho výrezu môžete predefinovať pomocou príkazu *VOŘEŽ*. Ak chcete orezať výrez, môžete pomocou ukazovacieho zariadenia buď vybrať existujúci objekt, ktorý má slúžiť ako nová hranica, alebo vybrať body nových hraníc.

Niekedy je vhodné časť pohľadu na modelový priestor vo výreze zneviditeľniť, napríklad na vloženie popisu do priestoru objektu. Na to slúži príkaz *PŘEKRÝT*. Prekrývací

objekt je mnohouholníková oblasť, ktorá maskuje pod ňou ležiace objekty pomocou aktuálnej farby pozadia. Tato oblasť je ohraničená rámčekom, ktorý môžete zapnúť pri úpravách a vypnúť pri vykresľovaní. Prekrývací objekt možno vytvoriť zadaním mnohouholníkovej oblasti niekoľkými bodmi alebo je možné previesť na prekrývací objekt uzatvorenú krivku. Obsahuje voľbu pre nezobrazovanie rámčeku prekrývacej oblasti.

Ak chcete zadať mierku pohľadu vo vykreslenom výkrese, môžete ho zmeniť pomocou:

- palety Vlastnosti
- Voľba XP príkazu ZOOM
- Panelu nástrojov Výřezy



Obr. 4.4 Nastavenie merítka z panelu vlastností

Keď pracujete vo výkresovom priestore, faktor mierky pohľadu vo výreze predstavuje pomer medzi skutočnou veľkosťou modelu zobrazeného vo výrezoch a jeho veľkosťou vo výkrese. Pomer je určený podielom jednotiek výkresového priestoru a modelového priestoru. Napríklad pre výkres v štvrtinovej mierke by bol pomer zväčšenia jedna jednotka výkresového priestoru ku štyrom jednotkám modelového priestoru, čiže 1 : 4.

Zmena veľkosti alebo pretiahnutie okraja výrezu nemení mierku pohľadu vo vnútri výrezu.

Keď nastavíte mierku výrezu, nemožno zoomovať vo výreze bez zmeny mierky výrezu. Pomocou uzamknutia mierky výrezu (z panelu *Vlastnosti*) uzamknete mierku, ktorú ste nastavili pre vybraný výrez. Keď je mierka zamknutá, môžete pokračovať v úpravách objektov zobrazených vo výreze, bez ovplyvnenia mierky výrezu. Uzamknutie je dostupné i pre nepravouhlé výrezy. Ak chcete uzamknúť nepravouhlé okno, je nutné v palete *Vlastnosti* vybrať objekt výrezu a nie hranicu výrezu.

Keď máte vo výreze niekoľko výrezov v rôznych mierkach, bude pravdepodobne vhodné, aby prípadné čiarkované a iné čiary sa zobrazovali vo všetkých výrezoch rovnako,

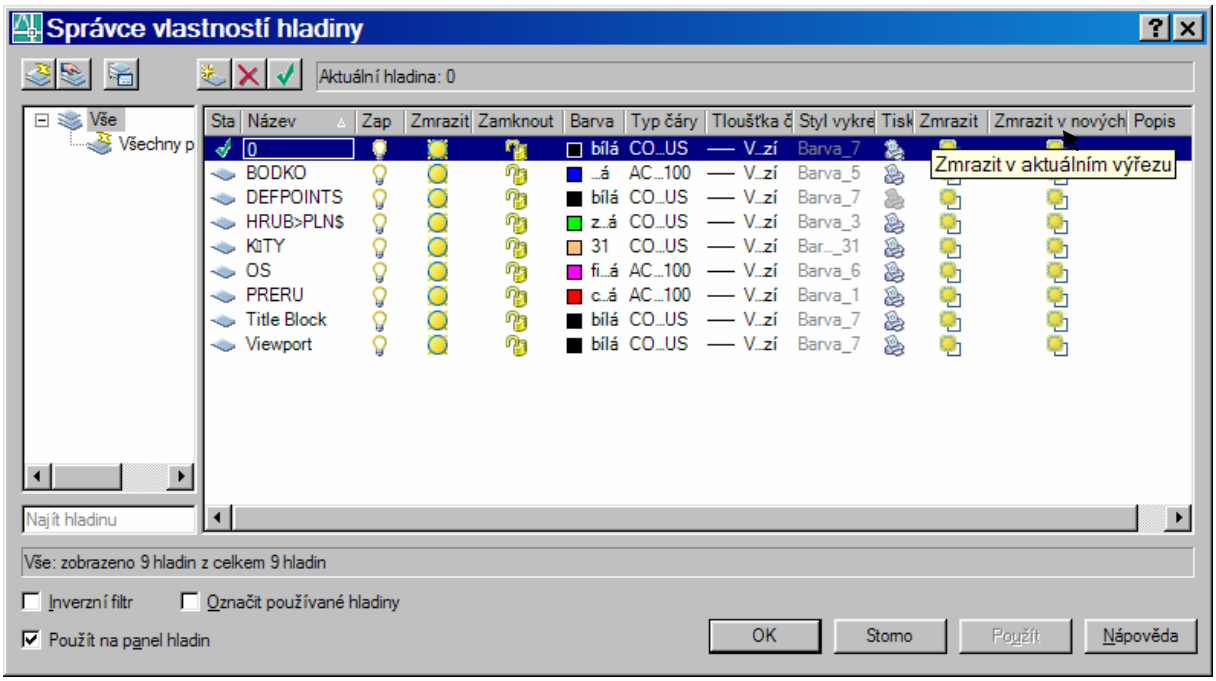
t. j. v rovnakej mierke čiary. Toto riadi systémová premenná PSLTSCALE, ktorá pri hodnote 1 zabezpečuje rovnakú mierku čiary aj pri rôznych mierkach výrezov a pri hodnote 0 sa tieto mierky riadia merítkom výrezu. Okrem toho je vo výrezoch stále možné používať panel vlastností na nastavenie individuálnej mierky pre jednu čiaru alebo meniť globálnu mierku čiary z menu *Formát – Typ čáry...*

Viditeľnosť objektov zobrazených vo výrezoch môžete ovládať niekoľkými spôsobmi. Tieto metódy sú užitočné pre zvýrazňovanie alebo skrývanie rôznych prvkov výkresu. Môžete použiť:

- Zapnutie/vypnutie výrezu (z panelu Vlastnosti)
- Zapínanie a vypínanie hladín
- Príkaz *Překrýt* (Wipeout)
- Riadkovanie nastavené na 0 (parameter štýlu vykresľovania)

Dôležitou výhodou použitia výrezov vo výkresovom priestore je, že môžete selektívne zmraziť hladiny (v každom výreze inak). Zmrazenie a rozmrazenie hladiny prevádzate bez vplyvu na ostatné výrezy. Zmrazené hladiny sú neviditeľné. Nie sú regenerované ani vykresľované. Výsledkom je, že v každom výrezu uvidíte rôzne objekty.

Najjednoduchší postup, ako zmraziť nebo rozmraziť hladiny v aktívnom výreze, je pomocou Správcu vlastností hladiny.



Obr. 4.5 Zmrazenie (zneviditeľnenie) hladiny cez správcu hladín

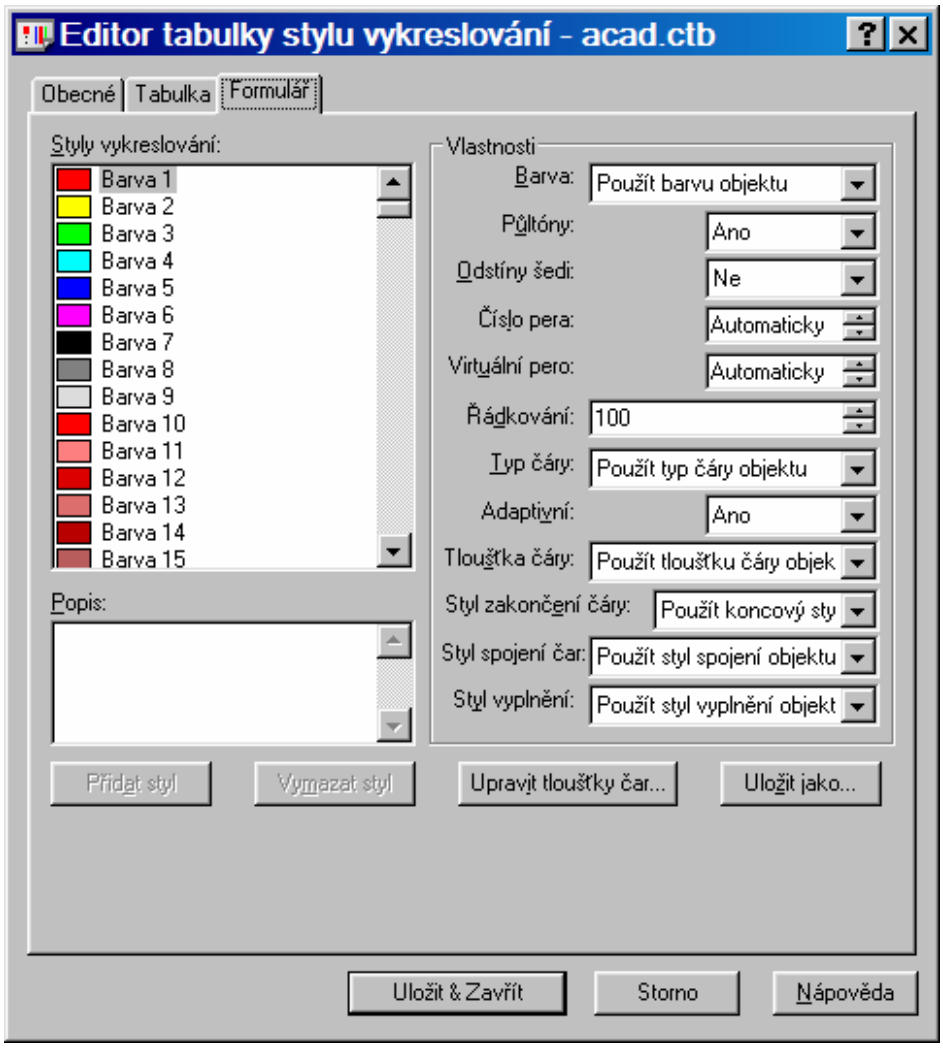
V Správcovi vlastností hladín na pravej strane pomocou stĺpca označeného *Zmraziť v aktuálnom výreze* zmrazte jednu alebo viac hladín v aktuálnom výreze. Ak chcete zobraziť stĺpec Zmraziť v aktuálnom výreze, musíte sa nachádzať v liste *"rozvržení – layout"*. Určite aktuálny výrez poklepaním v priestore vymedzenom rámčekom.

Ďalším spôsobom ako ovládať viditeľnosť objektov vo výrezoch je použitie už uvedeného príkazu *PŘEKRÝT*. Okrem toho možno využiť parameter vykresľovacieho štýlu *Riadkovanie*. Riadkovanie znamená, že je pri vykreslení použité na objekt menej atramentu.

Objekty sa budú z toho dôvodu javiť bledšie. Riadkovanie možno použiť na odlíšenie objektov vo výkresu, bez zmeny vlastnosti farby objektu.

Ak chcete prideliť hodnotu riadkovania objektu, je mu nutné prideliť štýl vykresľovania a potom v tomto štýle definovať hodnotu riadkovania podobne ako sa priradí napríklad hrúbka pera.

Objektu môže byť pridelená hodnota riadkovania v rozmedzí od 0 do 100. Začiatočné nastavenie hodnota 100, znamená, že nie je použité žiadne riadkovanie, a objekt je zobrazený s normálnou intenzitou atramentu. Hodnota riadkovania 0 znamená, že objekt neobsahuje žiaden atrament a je teda v tomto výreze neviditeľný.



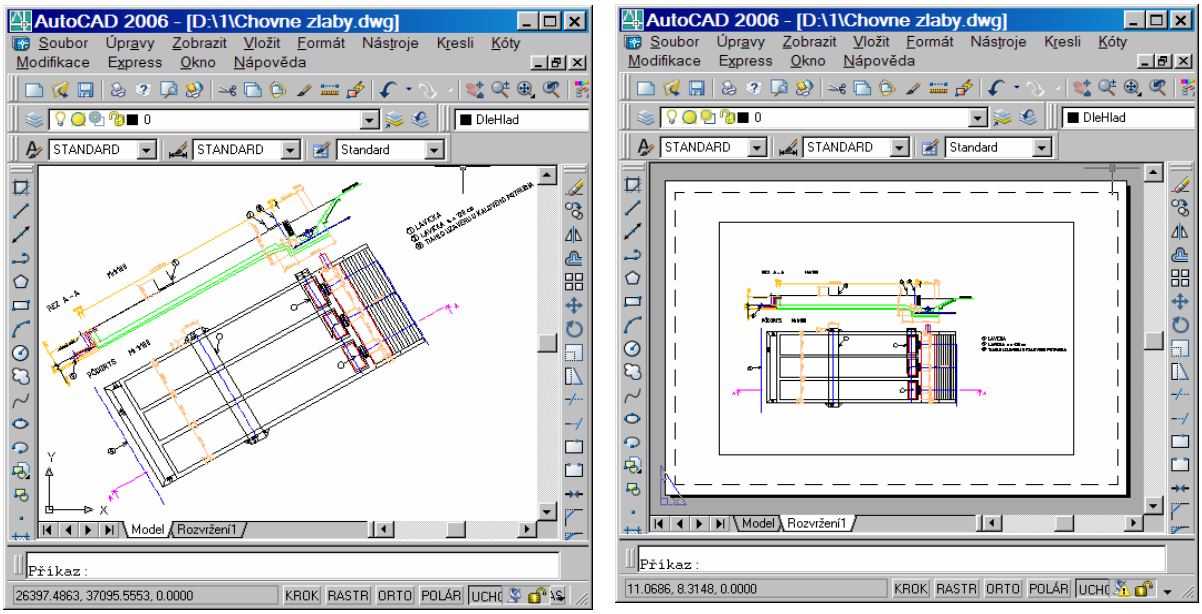
Obr. 4.6 Parameter riadkovania v editore štýlov vykresľovania

Často je potrebné, aby model zostal vo výkrese na pôvodných súradniciach (napríklad geodetických), avšak na vykreslenie je vhodné ho pootočiť. Celý pohľad môžete otočiť vo výreze pomocou zmeny USS a príkazu PŮDORYS. Pomocou príkazu USS môžete totiž otáčať rovinu XY v ľubovoľnom uhle okolo osy Z. Keď zadáte príkaz PŮDORYS, pohľad sa otočí do orientácie roviny XY. Postupujete podľa nasledujúcich krokov:

1. Prepnite sa do *"Rozvržení – Layout"* a do daného výrezu.

2. Vyberte z menu *Nástroje* položku *Nový USS* a potom položku *Otočiť okolo osi Z.*
3. Zadajte hodnotu otočenia, alebo ukážte odkliknutím dvoch bodov čiary, ktorá má byť rovnobežná s osou X (čo bude pravdepodobne spodná hrana papiera).
4. Vyberte položku *Zobrazit 3D pohledy Půdorys*. Vyberte položku *Aktuální USS.*
5. Celý pohľad sa otočí v rámci výrezu.

V prípade, že máte nainštalované Express Tools, môžete použiť aj príkaz *Express Tools → Layout Tools → Align Space.* V tomto prípade sa postupuje tak, že v danom výreze, ktorý chcete otočiť sa vyberú dva body (v modelovom priestore cez výrez), príkaz sa potom prepne do papierového priestoru a vyberiete dva body v papierovom priestore s ktorými chcete zdrojové body z modelového priestoru zarovnať.

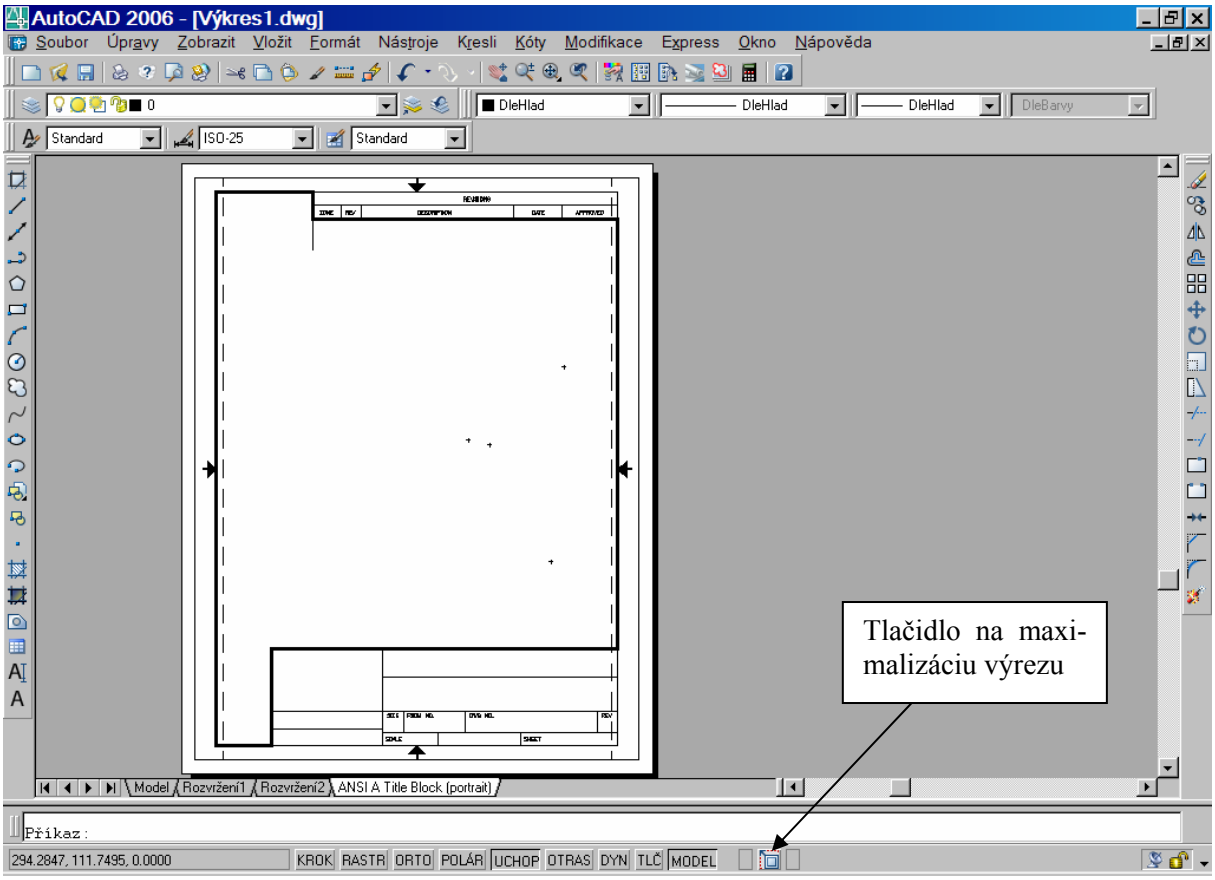


Obr. 4.7 Otočenie objektov vo výreze

Okrem ovládania viditeľnosti a pohľadu na objekty z modelového priestoru môžeme, ako už z predchádzajúceho vyplýva, pridávať vo výkresovom priestore ľubovoľné objekty, ktoré budú iba súčasťou výkresového priestoru a v modeli sa vôbec neobjavia. Pri práci na tejto karte máme k dispozícii takmer všetky príkazy AutoCADu (treba brať do úvahy, že výkresový priestor je dvojrozmerný). Pri kreslení vo výkresovom priestore však treba brať do úvahy, že v rámci práce v karte *"Rozvržení – Layout"* možno cez výrezy, ktoré sú akýmisi oknami do modelového priestoru, do modelového priestoru aj zasiahnuť a modifikovať jeho objekty, prípadne vytvárať nové. Je preto potrebné orientovať sa keď pracujeme na karte *"Rozvržení – Layout"*, či naša činnosť ovplyvňuje výkresový alebo modelový priestor.

K modelovému priestoru môžete pristupovať z výrezu z karty *"Rozvržení – Layout"* na úpravu objektov, zmrazenie a rozmrazenie hladín a úpravy pohľadu.

Ak plánujete vytvoriť alebo upraviť objekty, použite tlačidlo v stavovom riadku pre maximalizáciu výrezu. Rozbalením maximalizovaného výrezu se vyplní celá grafická oblasť. Nastavenia viditeľnosti hladiny sa zachovávajú a ovplyvňujú ich zobrazenie spolu s novými/modifikovanými objektmi v rámci celého výkresu, t. j. napr. aj v iných výrezoch.



Obr. 4.8 Prístup do modelového priestoru z karty *Rozvržení*

Pri práci v modelovom priestore možno ľubovoľne zoomovať a pri návrate do výkresového priestoru obnovením výrezu je obnovené pôvodné umiestnenie a mierka objektov vo výreze.

Ak chcete iba meniť pohľad, prípadne zmeniť viditeľnosť hladín, vhodnejší je trochu odlišný postup a to poklepaním do vyžadovaného výrezu, čím získate opäť prístup k modelovému priestoru. Rámček výrezu sa zvýrazní a kurzor nitkového kríža bude viditeľný iba v aktuálnom výreze. Všetky aktívne výrezy v *"Rozvržení – Layout"* zostanú pri práci viditeľné. Hladiny v aktuálnom výreze môžte zmrazovať a rozmrazovať v Správcovi vlastností hladiny. Ak sa chcete vrátiť do výkresového priestoru, poklepte na prázdnu oblasť mimo výrezu.

V tomto prípade sa však po návrate do výkresového mierku v prípade zoomovania vo výreze zmení mierka. Pokiaľ nastavíte mierku vo výreze pred získaním prístupu do modelového priestoru, môžete uzamknutím mierky zabrániť zmenám (cez panel *Vlastnosti* výrezu). Ak je mierka (*měřítko – scale*) uzamknutá, nie je možné pri práci v modelovom priestore použiť príkaz ZOOM.

4.2 Nastavenie vykresľovania

Bez ohľadu na to, či budete používať vykresľovanie z modelového alebo výkresového priestoru je potrebné nastaviť parametre vykresľovania z dialógového panely *Vykresli*. Toto je v skutočnosti prvé s čím sa užívateľ AutoCADu zoznámi, pretože zvyčajne bude svoje prvé

výkresy tlačiť z modelového priestoru a s nezmeneným – farebne závislým vykresľovacím štýlom, pravdepodobne nastaveným implicitne na to aby sa kreslilo farbou, hrúbkami atď. tak ako sú zvolené pre jednotlivé objekty výkresu.

Najprv je potrebné vybrať vykresľovacie zariadenie. Môžete zmeniť jeho konfiguráciu pomocou tlačidla Vlastnosti.

Vybraná tlačiareň alebo ploter v dialógu *Nastavení stránky* určuje medze tlače, ktoré sú vo výkresovom priestore označené prerušovanou čiarou. Ak zmeníte veľkosť papiera alebo vykresľovacie zariadenie, môže sa zmeniť aj plocha výkresovej stránky, na ktorú je možné tlačiť.

Keď nastavujete vykreslenie musíte definovať oblasť a tak určiť, čo bude vykreslené. V prípade vykresľovania z výkresového priestoru, východiskovou voľbou vykresľovanej oblasti je *Rozvržení*. Táto voľba vykreslí všetky objekty v oblasti tlače vybraného rozmeru papiera. V prípade špeciálnych požiadaviek na oblasť vykreslenia sú k dispozícii aj ďalšie voľby, ktoré už zrejme poznáte z vykresľovania z modelového priestoru. Voľba *Displej* vykreslí všetky na monitore zobrazené objekty výkresu. Voľba *Maximálně* vykreslí všetky viditeľné objekty vo výkrese. Voľba *Pohled* vykreslí uložené pohľady. Pomocou voľby *Okno* môžete definovať oblasť, ktorá bude vykreslená dvoma rohmi ohraničujúceho okna. Pokiaľ zvolíte vykreslenie inej oblasti než celého rozvrhnutia, môžete umiestniť kresbu na stred listu papiera.

Odsadenie vykresľovania určuje odsadenie vzhľadom k dolnému ľavému rohu oblasti určenej na tlač alebo k hrane papiera podľa nastavenia v dialógu *Možnosti*, na karte *Vykresľovania a publikování*. Odsadiť je možné zadaním kladných alebo záporných hodnôt, následkom však môže byť orezanie vykresľovanej oblasti.

Pri vykresľovaní z výkresového priestoru zvyčajne vykresľujete v mierke 1 : 1. V dialógu *Nastavenie stránky* alebo *Vykresľovania* môžete vybrať mierku zo zoznamu alebo ho zadať.

Orientáciu výkresu možno nastaviť pomocou voľby *Na výšku* a *Na šírku*. Voľbou *Na šírku* sa nastaví výkres na papieri tak, že dlhý okraj papiera bude horizontálny, voľbou *Na výšku* zorientujete papier tak, že horizontálny bude krátky okraj. Výberom voľby *Kreslit vzhůru nohama* sa nastaví, či je vykreslená najprv horná alebo dolná časť výkresu.

Ďalšia množina údajov, ktoré sa zadávajú na vykresľovanie je súčasťou Tabuľky štýlov vykresľovania. Táto sa otvára tlačidlom v pravom hornom rohu dialógu na nastavenie stránky. Prakticky ide o priradenie pier, ich farby, hrúbok atď. Tabuľka štýlu vykresľovania je súbor štýlov vykresľovania priradených rozvrhnutiu alebo aj listu Model.

Je možné použiť existujúcu tabuľku, vytvoriť novú tabuľku štýlov vykresľovania, alebo upraviť existujúcu tabuľku štýlov vykresľovania.

Existujú dva typy tabuľky štýlu vykresľovania: tabuľky farebne závislých štýlov vykresľovania a tabuľky pomenovaných štýlov vykresľovania.

Tabuľky farebne závislých štýlov vykresľovania určujú charakteristiky, napríklad hrúbku čiary, pomocou farby objektu. Každý objekt s červenou farbou je vykreslený rovnako (napr. hrúbka a farba pera). Štýly vykresľovania v tabuľke farebne závislého štýlu vykresľovania, je možné editovať, nie je však možné štýly vykresľovania pridávať ani odstrániť. V tabuľke farebne závislého štýlu vykresľovania je 256 štýlov vykresľovania, jeden na každú farbu.

Tabuľky pomenovaného štýlu vykresľovania (STB) obsahujú užívateľsky definované štýly vykresľovania. Keď používate tabuľku pomenovaného štýlu vykresľovania, objekty s rovnakou farbou môžu byť vykreslené odlišne podľa štýlu vykresľovania priradeného objektu. Tabuľka pomenovaného štýlu vykresľovania môže obsahovať toľko štýlov, koľko

potrebujete. Pomenované štýly môžu byť priradené objektom alebo hladinám rovnako ako ostatné vlastnosti (hladina, farba, typ čiary...).

Priradením rôznych tabuliek štýlu vykresľovania každému "rozvržení" vo výkrese môžete riadiť, akým spôsobom sú objekty vykreslené. Z toho istého modelového priestoru potom môžete generovať výkresy farbné aj čiernobiele alebo vykresľované rôznymi zariadeniami.

Ak chcete vidieť efekt Tabuľky štýlu vykresľovania v rozvrhnutí, zaškrtnite vľavo v dialógu *Nastavení stránky* v poli *Tabuľka štýlu vykresľovania* voľbu *Zobrazit vykreslovací styly*.

Na vytváranie a modifikovanie tabuliek štýlov vykresľovania používame Editor tabuliek štýlov vykresľovania. Vypisuje všetky štýly vykresľovania z tabuľky štýlu vykresľovania a ich nastavenia. Na úpravu nastavenia štýlov vykresľovania môžete použiť kartu *Tabuľka* alebo kartu *Formulár*.

V prípade pomenovaných štýlov vykresľovania budú sa ďalej uvedené vlastnosti priraďovať pre *Název*, inak sa budú priraďovať farbe, ktorou je objekt nakreslený. Popis informuje o štýle vykresľovania. V časti *Vlastnosti* (napravo v záložke formulár) sa určuje nastavenie nového štýlu vykresľovania, ktorý pridávate do tabuľky štýlu vykresľovania.

Určuje sa tu viacero vlastností – hrúbku čiary, farbu, ktorou sa budú objekty vykresľovať (môžete aplikovať "*Použít barvu objektu*" alebo ľubovoľnú inú farbu). Ak váš ploter používa k pre priblíženie farieb štruktúru bodiek, čím vykresľuje viac farieb, než je ich k dispozícii v AutoCADe možno nastaviť *áno* pre tzv. poltóny; ďalej sa tu nastavuje prevedenie do odtieňov šedej, štýly koncov a spojení čiar a ďalšie vlastnosti. Zmysel tlačidiel pridať štýl, odstrániť štýl a uložiť štýl je zrejmý z ich názvov.

OBSAH