Textový procesor 602Text

Popis makrojazyka

Obsah

Textový procesor 602Text ... 1

Popis makrojazyka ... 1

Obsah ... 2

Makrojazyk 602Text ... 16

Vytvoření makra ... 16

Zápis makra ... 16

Nabídka funkcí makrojazyka ... 17

Překlad makra ... 17

Spuštění makra ... 17

Lokální a globální makra ... 18

Spouštěcí povely ... 18

Přejmenování makra ... 19

Vymazání makra ... 19

Změna lokálního makra na globální a naopak ... 19

Automatická makra ... 19

Programovací jazyk ... 20

Používané symboly ... 20

Klíčová slova ... 20

Identifikátory ... 21

Znakové konstanty ... 22

Řetězce znaků ... 23

Speciální symboly ... 23

Oddělovače ... 24

Konstanty a jejich deklarace ... 24

Typy a jejich deklarace ... 25

Jednoduché typy ... 25

Ordinální typy ... 27

Strukturované typy ... 28

Deklarace typů ... 29

Deklarace proměnných ... 30

Deklarace procedur a funkcí ... 30

Zápis deklarací ... 30

Implicitní parametry ... 32

Deklarace externích procedur a funkcí ... 33

Struktura programu ... 33

Přístup k proměnným ... 34
Výrazy ... 35
Precedence operátorů ... 35
Aritmetické operace ... 36
Operace s datem ... 36
Operace s časem ... 36
Operace se znakovými řetězci ... 37
Relace ... 37
Podmíněný výraz ... 37
Příkazy ... 38
Přiřazovací příkaz ... 38
Složený příkaz ... 39
Podmíněný příkaz ... 39
Cyklus WHILE ... 39
Cyklus REPEAT ... 39
Cyklus FOR ... 39
Příkaz CASE ... 41
Volání procedury ... 41
Příkaz HALT ... 42
Vstup a výstup ... 42
Hlavní odlišnosti od jazyka Pascal ... 42
Přehled procedur a funkcí ... 44
Tématický přehled procedur a funkcí ... 45
Aritmetické funkce ... 46
Čas ... 46
Databázové operace ... 46
Dokument, práce s dokumentem ... 47
Elektronická pošta a faxování ... 48
Editace ... 48
HTML ... 49
Informace o aplikaci ... 49
Kapitoly, záhlaví, zápatí ... 49
Komunikace ... 49
Kontrola pravopisu ... 50
Konverze ... 50
Kurzor – pozice a označení textu ... 51
Lišty ... 52
Makra ... 52
Mód zobrazení (stránky, osnova) ... 53
Objekty ... 53
Odstavce a sekce ... 54
Odvolání akce ... 54
Okna s dokumenty ... 54
Okraje ... 55
Ostatní procedury a funkce ... 55
Pole ... 56

Pravítko s tabelátory ... 56
Předvolby ... 56
Rejstřík, obsah ... 56
Rolování obsahu okna ... 57
Soubory ... 57
Stav menu ... 58
Stavový řádek ... 58
Stránky, formát stránek ... 58
Styly ... 59
Svislé pravítko ... 59
Synonyma ... 59
Text, vlastnosti textu ... 59
Tisk ... 61
Uživatelské ovládací prvky ... 61
Vyhledání a záměna ... 63
Záložky, pojmenované bloky ... 63
Znakové řetězce ... 63
Zobrazení ... 64
Zvětšení ... 64
Přehled standardních konstant ... 65
Encyklopedie funkcí ... 72
Abs ... 72
AddIndexEntry ... 72
AddIndexEntryDlg ... 73
Addr ... 73
AddToDbRecord ... 74
Arctan ... 75
AreObjectsOn ... 75
AreScrollBarsOn ... 75
ArrangeIcons ... 75
AtDocEnd ... 76
AtDocStart ... 76
AtObjectEnd ... 76
AtObjectStart ... 76
BackspaceDelete ... 76
BalanceColumns ... 77
Beep ... 77
BlockName ... 77
BlocksCount ... 77
Bool2str ... 78
BottomOfScreen ... 78
CanUndo ... 78
CaretCellStart ... 78
CaretEnd ... 79
CaretHome ... 79
CaretLeft ... 80
CaretObjectEnd ... 80

CaretObjectStart 81
CaretRight 81
CleanString 81
ClearMark 82
ClipCopy 82
ClipCut 82
ClipPaste 82
Close 83
ColorDlg 83
ContentsEnabled 83
ContentsExists 84
CopyBlock 84
Cos 84
CountFonts 84
CountMacros 85
CountOfEntryRef 85
CountWindows 85
CreateContents 86
CreateFooter 86
CreateHeader 86
CreateIndex 87
CurrentFont 87
CurrentMacroName 88
DatabaseEnabled 88
Date2str 88
Day 89
Day_of_week 89
Dec 90
DefaultDocDir 90
DefaultTmplDir 90
DefParaStyleDlg 90
Delete_file 91
DeleteBlock 91
DeleteChar 91
DeleteObject 91
DeleteWord 92
DialogCreate 92
DialogCreateExt 93
DialogDestroy 93
DialogFontSize 93
DialogFontResize 94
DialogRun 94
DialogSetBmp 94
DialogSetSysMenu 94
DisableAutoRun 95
DisconnectMM 95
DisplayOutline 96

DisplayPages 96
DlgBtnCancel 96
DlgBtnNo 97
DlgBtnOk 97
DlgBtnYes 97
DlgButton 98
DlgButtonDefPush 98
DlgComboBox 99
DlgEditBox 100
DlgGroupBox 100
DlgCheckBox 100
DlgInputLine 101
DlgLineX 101
DlgLineY 102
DlgListBox 102
DlgRadioBtn 103
DlgRadioBtnGroup 103
DlgRemoveCtrl 104
DlgSetCtrlBmp 104
DlgSetCtrlPos 105
DlgSetCtrlSize 105
DlgSetCtrlText 105
DlgStdBox 106
DlgStrBoxAdd 106
DlgStrBoxDelete 107
DlgStrBoxGetStr 107
DlgStrBoxGetVal 107
DlgStrBoxSetVal 108
DlgText 108
DlgTextSetAlign 109
DlgTextSetBmp 109
DocClose 109
DocMaximize 110
DocMinimize 110
DocRestore 110
DocSplit 111
DocWndMove 111
DoesBlockExist 111
DoesMarkExist 112
DoesObjectExist 112
DoesWindowExist 112
EditBlocksDlg 113
EditMarksDlg 113
Eof 113
ErasethisMacro 114
Exec 114
ExeFileName 114

Exp 114
FieldInsertDlg 114
FieldHTML 115
FieldSimple 115
FieldSuma 116
Filelenght 116
FindClose 117
FindFirstFile 117
FindNextFile 117
FixScreenPos 118
FontName 118
FormatFontDlg 118
FormatParaDlg 119
FormatSectDlg 119
FrameMarginsOn 119
GetAlignType 119
GetBottomMargin 120
GetCaretCell 120
GetCaretLine 120
GetCaretObjectId 121
GetCaretPage 121
GetCaretPos 121
GetCaretPosType 122
GetCaretRow 122
GetCaretSection 122
GetCommandLine 122
GetDocFileName 123
GetDocWndHeight 123
GetDocWndLeft 123
GetDocWndTittle 123
GetDocWndTop 123
GetDocWndWidth 124
GetFormatFont 124
GetFormatPara 125
GetFormatSect 126
GetLastCode 127
GetLeftMargin 127
GetNewWinState 128
GetNumCells 128
GetNumRows 128
GetObjectHTMLStr 128
GetOutlineLevel 129
GetPageHeight 129
GetPageType 129
GetPageWidth 129
GetPreferences 130
GetPrefFax 130

GetPrefFaxName 131
GetPrefFootnStr 131
GetRGBText 131
GetRightMargin 131
GetScale 132
GetSelEndLine 132
GetSelEndPage 132
GetSelEndPos 133
GetSelStartLine 133
GetSelStartPage 133
GetSelStartPos 134
GetSplitType 134
GetSplitVal 134
GetStyleCount 134
GetStyleName 135
GetText 135
GetTextMM 135
GetTextMMI 136
GetTmplFileName 136
GetTopMargin 136
GetTotChapters 137
GetTotPages 137
GetTotSection 137
GetUnderlineType 137
GetUserName 137
GoToChapter 138
GoToIndexEntry 138
GoToMark 139
GoToPage 139
GoToSection 139
Hours 140
HyphenDlg 140
HyphenDlgEnabled 140
ChangeCase 141
ChangeSelRanges 141
ChapterHasFooter 141
ChapterHasHeader 141
ChapterNumber 142
Char2str 142
CharLeft 143
CharRight 143
CheckBoxGetVal 144
CheckBoxSetVal 144
Chr 144
Iabs 145
Inc 145
IndexEditDlg 145

IndexEditEnabled 146
IndexEntries 146
IndexEntry 146
IndexEntryInd 147
IndexExists 147
IndexGenEnabled 147
IndexMarkEnabled 148
Info_box 148
Input_box 148
Input_box_msg 149
Input_box_msg_2 149
Input_box_msg_3 150
Input_box_msg_4 151
InputLineGetVal 151
InputLineSetVal 151
InsertDateTime 152
InsertFileObject 152
InsertFootNote 152
InsertHardHyphen 153
InsertHardSpace 153
InsertChar 153
InsertMark 154
InsertNewChapter 154
InsertNewPage 155
InsertNewPara 155
InsertNewSection 155
InsertOptHyphen 156
InsertTab 156
InsertText 156
InsertTextStr 157
Int2str 157
IsAlignCenter 158
IsAlignJustify 158
IsAlignLeft 158
IsAlignRight 158
IsAnyDocOpened 159
IsBlockSelected 159
IsBold 159
IsBrushOn 159
IsDbForSave 159
IsDocDefault 159
IsDocMaximized 160
IsDocMinimized 160
IsDocumentOpened 160
IsFieldCntsOn 160
IsFormatOn 161
IsInsertMode 161

IsItalic 161
IsMacro 161
IsModified 161
IsPageLandscape 162
IsPagePortrait 162
Isqr 162
IsReadOnly 162
IsStatusStrip 163
IsTabRulerOn 163
IsTextDefault 163
IsTextRegular 163
IsToolbar 163
IsVertRulerOn 164
IsWordOverFlow 164
LabelBlock 164
LangSetupEnabled 165
Lcase 165
LeftOfLine 165
Like 166
LineDown 166
LineUp 167
Ln 167
Ltrim 167
MacroAutoStarted 168
MacroDelete 168
MacroDescription 169
MacroIsInMenu 169
MacroIsLocal 169
MacroName 170
MacroNameToIndex 170
MacroSetDescr 170
MacroSetToMenu 171
MacroSubrListDlg 171
MacroToGlobal 171
MacroToLocal 172
MailMergeType 173
Make_date 173
Make_directory 174
Make_time 174
ManualScaleDlg 175
MarkModified 175
MarkName 175
MarksCount 175
MarkUnmodified 176
Memcpy 176
MenuCreate 176
MenuDestroy 176

MenuRun ... 177
MenuStrAdd ... 177
MenuStrDelete ... 178
MenuStrDisable ... 178
MenuStrGet ... 178
MenuStrCheck ... 179
Minutes ... 179
Money2str ... 179
Month ... 180
NewFile ... 180
NextCell ... 181
NextMMField ... 181
NextPara ... 181
NextRow ... 182
NextWindow ... 182
Now ... 182
NullCmd ... 182
ObjectsCount ... 183
ObjectId ... 183
ObjectIsSelected ... 183
ObjectListDlg ... 184
ObjectPropDlg ... 184
ObjectSelCount ... 184
ObjectSelId ... 185
ObjectSelModify ... 185
ObjectType ... 185
Odd ... 186
OpenFile ... 186
Ord ... 188
PageDown ... 188
PageFormat ... 189
PagesDisplayed ... 189
PageUp ... 190
Pred ... 190
Pref ... 190
PreferencesDlg ... 191
PrevCell ... 191
PreviousWindow ... 191
PrevPara ... 192
PrevRow ... 192
Print ... 192
PrnAdvancedDlg ... 193
PrnGetCurrent ... 193
PrnIsPortrait ... 194
PrnListCount ... 194
PrnListStr ... 194
PrnSetCurrent ... 195

PrnSetPortait 195
PrnSetupDlg 195
RadioBtnCheck 196
RadioBtnChecked 196
RadioGrpGetVal 196
RadioGrpSetVal 197
Read 197
Real2str 197
RecentFileName 198
RecentFilesCount 198
RecordCurrent 198
RecordFirst 199
RecordLast 199
RecordNext 199
RecordPrev 200
RecordsCount 200
RecordSet 200
RemoveIndexEntry 200
Replace 201
ReplaceAgain 202
ReplaceIsReady 202
Reset 202
Rewrite 203
RGB 203
RgbTrio 204
RightOfLine 204
RightOfWord 204
Round 205
Rtrim 205
SaveDbRecord 205
SaveFile 206
SaveFileAs 206
ScrollLineDown 207
ScrollLineUp 207
ScrollPageDown 207
ScrollPageUp 207
Search 208
SearchAgain 208
SearchIsReady 209
Sec1000 209
Seconds 210
Seek 210
SelectBlock 210
SelectContents 211
SelectIndex 211
SelectObject 211
SetAlign 212

SetAlignCenter 212
SetAlignJustify 212
SetAlignLeft 213
SetAlignRight 213
SetBold 213
SetBottomMargin 213
SetBrush 214
SetDbForSave 214
SetDocWndHeight 214
SetDocWndWidth 215
SetFormatFont 215
SetFormatPara 216
SetFormatSect 218
SetInitialOpen 218
SetInsertMode 219
SetIsMacro 219
SetItalic 219
SetLeftMargin 219
SetMargins 220
SetMM 220
SetNewWinState 221
SetObjectHTMLStr 221
SetOutlineLevel 221
SetOverwriteMode 222
SetPreferences 222
SetPrefFax 223
SetPrefFaxName 223
SetPrefFootnStr 223
SetRGBColor 224
SetRGBText 224
SetRightMargin 224
SetScale 225
SetScaleFullPage 225
SetStatusStrip 225
SetStatusText 225
SetTextDefault 226
SetTextRegular 226
SetToolbar 226
SetTopMargin 226
SetUnderlineType 227
SetWordOverFlow 227
Sgn 227
Sin 228
sizeof 228
SpellEnabled 228
SpellRun 229
Sqr 229

Sqrt ... 229
Str2date ... 230
Str2int ... 230
Str2money ... 230
Str2real ... 231
Str2time ... 231
Strcat ... 232
Strcopy ... 232
Strdelete ... 233
Strinsert ... 233
Strlength ... 233
StrIsRC ... 234
Strpos ... 234
Strtrim ... 234
StyleIndex ... 235
StyleIsBased ... 235
StyleUpdate ... 236
Substr ... 236
Succ ... 236
SummaryDlg ... 237
SummaryGetStr ... 237
SummarySetStr ... 237
SwitchToDoc ... 238
SwitchToWindow ... 238
ThesaurusDlg ... 238
ThesaurusEnabled ... 238
Time2str ... 239
Today ... 239
ToggleInsert ... 240
ToolbarCfgDlg ... 240
TopOfScreen ... 240
TranslateEnabled ... 241
Trunc ... 241
Undo ... 241
UnlabelBlock ... 241
UnselectBlock ... 242
UnselectObject ... 242
Upcase ... 242
UseFont ... 242
UseFontOfName ... 243
UseFontOfNameExt ... 243
UseParaStyleNum ... 244
ViewFieldCnts ... 244
ViewFormat ... 244
ViewFrameMargins ... 244
ViewObjects ... 245
ViewScrollBars ... 245

ViewTabRuler 245
ViewVertRuler 245
Wait 246
Wait_box_hide 246
Wait_box_show 246
WindowsCascade 246
WindowsTile 247
WordLeft 247
WordRight 247
Write 248
Writeln 249
Year 249
Yesno_box 249
Poznámka 1. 250
Poznámka 2. 250
Poznámka 3. 250
Poznámka 4. 250
Poznámka 5. 251
Poznámka 6. 251

Makrojazyk 602Text

Makra poskytují prostředky, které na uživatelské úrovní dovolují značné zefektivnění a zrychlení práce.

Typická makra se skládají z často opakovaných sekvencí příkazů a povelů procesoru 602Text. Jejich vyvoláním se v jediné operaci provede celá uložená sekvence, bez nutnosti zdlouhavého mačkání kláves a přemýšlení, jak se vlastně ten který příkaz jmenuje či zadává.

Základní body pro práci s makry:

- Každé makro musíte nejprve napsat ve zdrojové formě tzv. makrojazyka.
- Zdrojový text makra je dále zapotřebí přeložit do spustitelné formy.
- V této podobě je teprve možné makro spustit a vyzkoušet jeho funkci.
- Často používaným makrům je možné přiřadit spouštěcí klávesové povely.
- Hotová makra mohou figurovat jako lokální (spojená s jedním dokumentem nebo jako globální, všeobecně dostupná. Speciální kategorii tvoří tzv. automatická makra.
- S makry můžete pracovat jako s jinými součástmi prostředí 602Text: můžete je přejmenovávat i mazat; lze také měnit jejich lokální status na globální a naopak.

Vvtvoření makra

Makra se v procesoru 602Text zapisují v podobě programových instrukcí. Syntaxe i další pravidla jsou velice podobné jazyku Pascal.

K lepšímu pochopení je v podadresářích **Dokument** a **Makra** k dispozici řada příkladů funkčních maker, na kterých celou problematiku snadno pochopíte.

Obecná pravidla pro práci s makry:

- Makra se vytvářejí v podobě zdrojových textů, které se pak přeloží a zařadí do seznamu spustitelných maker.
- Texty maker se ukládají do souborů s příponou WPM. Při načítání a ukládání volte jako **Typ zobrazených souborů** položku **Makro (*.WPM)**.
- Makra dělíme na globální pro všechny dokumenty jednoho uživatele a lokální v rámci jednoho dokumentu. Přenos maker mezi uživateli realizujte přenosem zdrojových souborů a jejich následným překladem.
- Máte-li aktivovanou volbu **Nabízet šablony pro nové dokument**y, zatrhněte při založení nového makra přepínač **Pro makra**.
- Součástí maker mohou být standardní funkce makrojazyka.
- Při zápisu a editaci maker se přepněte do režimu **Jako zdroj makra** (menu **Makro)**.

Zápis makra

Při zápisu textu makra přepněte pracovní okno do režimu zápisu maker.

V menu Makra spusťte příkaz Jako zdroj makra.

Tím se zavede kontinuální číslování řádků, které je nutné k identifikaci chyb při překladu.

Vytvořte zdrojový text makra (podle příkladů v adresářích **Dokument** a **Makra**) nebo si jeden z příkladů načtěte. Deklarovaný název makra bude použit jako název do seznamu spustitelných maker.

Při načítání příkladů z adresáře **Makra** volte typ importu **ASCII Standard Windows** – tím se vám zobrazí soubory s příponou TXT**.** Kódování češtiny by mělo být vždy podle normy **WinEE**.

Nabídka funkcí makrojazyka

Příkaz Nabídka funkcí makrojazyka z menu Makra nabídne Seznam procedur a funkcí, které můžete nastavením ukazatele a potvrzením volby vkládat do vašich makrosouborů.

- Pomocí tlačítka **Vložit volání** se do makrosouboru vloží na pozici kurzoru název funkce, který je podle potřeby doprovázen prázdnými závorkami pro zápis argumentů.
- Pomocí tlačítka **Vložit deklaraci** se vloží na pozici kurzoru kompletní deklarace zvolené funkce, včetně deklarací typů argumentů. Tuto volbu použijete v praxi dosti zřídka. Jedno z možných využití se nabízí tehdy, budete-li chtít nějakou standardní proceduru/funkci ve svém zdrojovém souboru předefinovat, tj. napsat vlastní proceduru/funkci stejného jména, ale odlišného chování.

Podrobný popis funkcí makrojazyka 602Text najdete v Encyklopedii funkcí.

Překlad makra

Před použitím je třeba zdrojový text makra přeložit do spustitelné podoby.

Příkaz Přeložit makro z menu Makra nebo klávesový povel Shift+F11 přeloží obsah makrosouboru načteného v pracovním okně. Pokud není obsah v pracovním okně zobrazen v režimu Jako zdroj makra, před začátkem překladu se tento režim nastaví.

Během překladu se kontrolují jednotlivé příkazy. Pokud dojde k chybě, je ohlášena a překlad se v místě chyby ukončí. Pokud je vše v pořádku, makro se zařadí do seznamu. Během překladu lze překlad předčasně ukončit (kliknutím myší na tlačítko **Stop**, klávesovými povely **Enter** či **Esc** nebo také **Ctrl+Break**).

Chyby jsou indikovány jednak rámečkem s číslem řádku a výpisem jejich charakteru, jednak nastavením řádkového kurzoru na místo, kde k chybě došlo. Chybu opravte a spusťte překlad znovu. S chybou také samozřejmě skončí omylem spuštěný překlad jiného typu souboru.

Přeložené makro je jistým způsobem svázáno s otevřeným dokumentem, ze kterého bylo přeloženo. Toto napojení spočívá v tom, že při dalším překladu tohoto otevřeného (a event. od posledního překladu pozměněného) dokumentu je původní kód (přeložená posloupnost instrukcí) spustitelného makra nahrazen novým; při neúspěšném překladu zanikne.

Tento mechanismus se uplatňuje i v případě, kdy v dokumentu provedeme změny mající za důsledek generování odlišného jména přeloženého makra (buď změna jména v hlavičce programu; u programu bez hlavičky změna jména dokumentu funkcí **Zapsat jako**). Příslušné nahrazení proběhne včetně jména. Popsaná vazba se zruší buď zavřením příslušného dokumentu (z něj přeložené spustitelné makro pak k žádnému dokumentu nepatří; i po opětovném otevření téhož dokumentu je toto makro pro něj „cizí“) nebo přejmenováním makra.

Spuštění makra

Správně přeložená makra připravená ke spuštění jsou k dispozici ve formě seznamu.

Příkaz Spustitelná makra nebo klávesový povel Ctrl+M otevře přehled všech spustitelných maker, která jsou v dané chvíli k dispozici.

Přehled umožňuje vypouštění, přejmenovávání a spouštění maker. Ke každému makru také zobrazí doplňkové informace a lze mu přiřadit spouštěcí klávesový povel.

Makro vyberete nastavením ukazatele seznamu nebo kliknutím myší. Spustíte jej pomocí tlačítka **Spustit** (nastavením ukazatele a stiskem klávesy **Enter**, případně dvojím kliknutím myší na položku).

Může se stát, že budete chtít běh makra násilně předčasně ukončit (makro dělá to, co nemá, nebo se zacyklí v nekonečné smyčce). Použijte klávesový povel **Ctrl+Break** (zobrazí se dialog se zprávou o předčasném ukončení makra).

Lokální a globální makra

Makra rozlišujeme na lokální a globální. Lokální makra se vztahují jen k jednomu dokumentu, globální makra jsou určena pro spuštění nad libovolným dokumentem.

V dialogu **Spustitelná makra** se lokální makra přiřazují aktuálnímu dokumentu se kterým se též ukládají. Jsou v tomto dialogu označena hvězdičkou před svým jménem (která však není součástí tohoto jména). Má-li jedno lokální a jedno globální makro stejný spouštěcí klávesový povel, přednost pro spuštění má makro lokální.

Lokální makra mají sloužit hlavně pro operace nad aktuálním dokumentem; nelze z nich volat funkce, při jejichž provedení by se provedlo přepnutí do jiného dokumentu. Taková situace způsobí chybu za běhu makra a dojde k předčasnému ukončení běhu.

Jedná se o funkce:
NewFile, **OpenFile**, **DocClose**, **PreviousWindow**, **NextWindow**, **SwitchToWindow**, **SwitchToDoc**.

Lokální makra mohou být též obsažena v šablonách. Při založení nového dokumentu se do něj zkopírují lokální makra z nastavené šablony.

Tato skutečnost má význam pro běh automatických maker:

- Lokální makro **AutoNew** načtené ze šablony při založení nového dokumentu má pro spuštění přednost před globálním makrem tohoto jména.
- Lokální makro **AutoClose** má při zavírání dokumentu přednost pro spuštění před globálním makrem téhož jména.
- Lokální makra mohou mít též jména **AutoExec**, **AutoExit**; takováto lokální makra nemají však funkci svých globálních protějšků a nikdy se automaticky nespouštějí. Proto použití těchto jmen pro lokální makra nedoporučujeme.

Podrobněji o automatických makrech viz kapitola *Automatická makra*.

Spouštěcí povely

Ke každému makru můžete vytvořit popis a můžete mu také přiřadit spouštěcí klávesový povel.

Jak budete postupovat:

- Stiskněte tlačítko **Vlastnosti**.
- Do vstupního pole **Zkrácený příkaz** zapište znak, který spolu s klávesami **Ctrl** a **Alt** makro spouští klávesovým povelem. Již dříve zapsaný znak můžete zrušit stiskem kláves **Delete**, **Backspace.**
- V dialogu **Vlastnosti makra** lze všechny z nich zadávat buď stiskem **Ctrl+Alt+klávesa** nebo stisknutím klávesy s výjimkou:
  - **F1** (nápověda) – použijte spojení kláves **Ctrl+Alt+F1**
  - **Backspace** (maže předchozí hodnotu) – použijte spojení kláves **Ctrl+Alt+Backspace.**
- Do vstupního pole **Popis** zapište text, který vám v rámečku **Spustitelná makra** pomůže makro identifikovat.

Menu **Makra** umožňuje přímé spuštění již přeložených maker.

Přeimenování makra

Přeložené makro zařazené do seznamu můžete přejmenovat.

Jak budete postupovat:

- Na makro nastavte ukazatel seznamu a stiskněte tlačítko **Přejmenovat**.
- Do dialogového rámečku zapište nový název makra.

Vymazání makra

Makro zařazené do seznamu z něj můžete vymazat.

Vymazání se netýká zdrojového textu, ze kterého se makro překládá. Jak budete postupovat:

- Na makro nastavte ukazatel seznamu a stiskněte tlačítko **Smazat**.
- Potvrďte bezpečnostní dotaz.

Změna lokálního makra na globální a naopak

Lokální makro je uloženo v dokumentu (tedy i šabloně) a lze s ním pracovat pouze je-li příslušný dokument v aktivním pracovním okně.

Pomocí tlačítek **Lokální** či **Globální** můžete přesouvat makro z globální úložny do dokumentu a zpět.

Automatická makra

Program 602Text rozlišuje speciální skupinu tzv. automatických maker. Tato makra jsou určena k samočinnému provedení určitých akcí v klíčových momentech práce s programem.

Názvy automatických maker začínají slovem **Auto**.

Název makra	použití
AutoExec	Spustí se při každém startu programu s výjimkou situace, kdy je tento spuštěn jinou aplikací jako OLE server.
AutoNew	Spustí se při každém otevření nového dokumentu.
AutoOpen	Spustí se při každém otevření existujícího dokumentu.
AutoClose	Spustí se při každém zavření pracovního okna.
AutoExit	Spustí se při ukončení programu.

Při tvorbě těchto maker je nutné mít na zřeteli, že nemůže běžet více maker současně. Makra nemohou jiná makra volat ani nemohou svůj běh dočasně přerušit. Pokud proto při běhu makra `AutoExec` (nebo jiného) otvíráte nový dokument, makro `AutoNew` se v tomto případě nezavolá a požadované operace je nutné aplikovat jako následující příkazy v makru `AutoExec`. Analogicky to platí i pro otvírání existujících dokumentů, zavírání dokumentů (a to i v jiných než automatických makrech).

Programovací jazyk

Programovací jazyk 602Text je vyšší strukturovaný programovací jazyk velmi podobný Pascalu. Hlavní odlišnosti mezi oběma jazyky jsou uvedeny v následujícím výčtu.

- Přidání prostředků pro komunikaci s jádrem textového procesoru
- přidání jednoduchého uživatelského interface a prostředků pro spolupráci s Windows
- vypuštění těch rysů, které by našly ve vnitřním programovacím jazyku jen malé uplatnění.

Nové schopnosti vnitřního jazyka mají většinou podobu nových standardních procedur a funkcí. Jen v nečetných případech je rozšířena i syntaxe jazyka. Na konci této kapitoly naleznete seznam standardních podprogramů a také souhrn hlavních rozdílů mezi Pascalem a jazykem 602Text.

V dalším textu budeme předpokládat, že již znáte alespoň základy programování v některém vyšším strukturovaném jazyku, jako jsou například Pascal, C, ADA, MODULA 2, a má možnost vyjasnit si ty rysy Pascalu, na něž zde budeme odkazovat.

Referenční popis standardních procedur a funkcí najdete v kapitole *Přehled procedur a funkcí*.

Používané svmbolv

Na nejnižší syntaktické úrovni jazyka jsou tzv. symboly neboli lexikální elementy. Jsou vytvořeny ze znaků.

Symboly lze rozdělit na sedm druhů:

- klíčová slova
- identifikátory
- čísla
- znakové konstanty
- řetězce znaků
- speciální symboly
- oddělovače.

Klíčová slova

Klíčová slova jsou symboly s předem daným významem, který nemůže být v programu nijak změněn. Pokud jim pokusíte přiřadit význam jiný, dojde ke konfliktu.

Klíčovými slovy jsou:

AND	ARRAY	BEGIN
CASE	CONST	CSISTRING
CSSTRING	DIV	DO
DOWNTO	ELSE	END
FILE	FOR	FUNCTION
HALT	IF	MOD
NOT	OF	OR
PROCEDURE	PROGRAM	RECORD

REPEAT	STRING	THEN
TO	TYPE	UNTIL
VAR.	WHILE	

Velikost písmen nemá na význam vliv, proto lze klíčová slova psát jak malými, tak i velkými písmeny popřípadě oba druhy písmen kombinovat.

Identifikátory

Identifikátory se tvoří z písmen, číslic a znaku _ (podtržítko), přičemž musí začínat písmenem. Identifikátory se musí lišit od klíčových slov.

Identifikátory jsou například:

- ALFA
- Cena_zboží
- NováCena
- ODDĚLENÍ_56
- AF17

POZOR! V identifikátorech se bere v úvahu pouze prvních 16 znaků. Proto dva identifikátory, které se shodují v prvních 16 znacích, budou považovány za stejné. V identifikátorech se nerozlišuje mezi velkými a malými písmeny. Proto jsou možné různé způsoby zápisu **téhož** identifikátoru:

- NOVÁ_CENA_MELOUNU
- Nová_cena_melounů
- Nová_Cena_Melounů
- Nová_cena_melounů_v_únoru

Předdefinované identifikátory

Existuje početná množina předdefinovaných (standardních) identifikátorů, které sice můžete použít i v jiném významu, ale zastíníte tím význam původní. Pokud například definujete proměnnou *day*, nebudete moci v rozsahu její platnosti volat standardní funkci **Day**. Všechny standardní objekty naleznete v kapitole *Přehled procedur a funkcí*.

Celá čísla .....

Celá čísla se zapisují jako posloupnosti číslic, přičemž před zápornými čísly se píše znak minus ('–').

Celá čísla musí ležet v rozsahu:

`-2147483647` až `2147483647` tzn. `-MAXINT` až `MAXINT`.

Všechna celá čísla se dají přiřadit proměnné typu *integer* (viz níže, viz též konstanta `MAXINT`).

Proměnné typu s*hort* se dají přiřadit pouze celá čísla z rozsahu:

`-32767` až `32767` tzn. `-MAXSHORT` až `MAXSHORT`.

Ve specifickém kontextu v programu může být vyžadován menší rozsah hodnot celých čísel.

Reálná čísla

Reálná čísla se skládají z celé části a buď desetinné části za desetinnou tečkou (nikoli čárkou!) nebo desítkového exponentu, případně obojího. Před desítkovým exponentem se uvede písmeno E a případně i znaménko + resp. –.

Správně zapsaná reálná čísla jsou tedy například:

```
1.0
-538.045
0.02
56.871E-12
7e5
```

Reálná čísla se dají přiřadit proměnné typu *real*.

Částky peněz

Čísla vyjadřující peněžní obnosy (jinými slovy: konstanty typu *money*) musí obsahovat mezi celou a desetinnou částí znak $. Kompilátor pak explicitně pozná, že se jedná o typ *money* a nikoli typ *integer* nebo *real*.

Příklad:
```
120$
120$50
```

Zápisy peněžních částek se dají přiřazovat proměnným typu *money*. Vzhledem ke konverzím typů (viz dále) se těmto proměnným dají také přiřazovat hodnoty zadané celým nebo reálným číslem.

Kalendářní datum

Číselné konstanty vyjadřující datum, tedy hodnoty typu *date,* se zapisují v programu jako trojice čísel oddělených tečkami v pořadí: den . měsíc . rok .

Platí tedy tento formát: den.měsíc.rok

Příklad:
```
19.12.1997
```

Druhá tečka a rok se mohou vynechat, pak se míní běžný rok. Zápis roku se nesmí zkracovat, rok 97 není totéž co 1997. Celý zápis musí být na jedné řádce, uvnitř zápisu se nesmějí vyskytnout žádné mezery.

Zápis času

Čas, tedy konstanta typu *time* se zapisuje v programu v pořadí: hodiny – minuty – sekundy – tisíciny sekundy.

Platí tedy zápis: hodiny:minuty:sekundy.tisíciny_sekundy

Příklad:
```
17:48:56.908
```

Tisíciny sekundy resp. i sekundy se smějí vynechat spolu s tečkou resp. dvojtečkou před nimi. Celý zápis musí být na jedné řádce, uvnitř zápisu se nesmějí vyskytnout žádné mezery.

POZOR na záměnu proměnné typu *time* se příliš úsporně zapsaným podmíněným výrazem. Zápis jako:

```
n := n>0?n*10:0;
```

kompilátor pochopí jako násobení číselné proměnné s výrazem typu *time* a ohlásí chybu. Aby podmíněný výraz byl správně vyhodnocen, musíte za dvojtečku vložit mezeru.

Znakové konstanty

Jednotlivé znaky v textu programu se zapisují v apostrofech.

Například znak A zapíšete jako: `'A'`.

Znak ' (apostrof) zapíšete mezi ohraničujícími apostrofy zdvojeně, tzn. celkem 4 apostrofy za sebou: ''''.

Řetězce znaků

Řetězcem znaků je libovolná posloupnost znaků začínající a končící apostrofem nebo uvozovkou. Uvnitř řetězce smějí být libovolné znaky. Pokud mají být apostrof nebo uvozovka obsaženy v řetězci, napište je zdvojeně.

Řetězec znaků smí obsahovat jediný znak, což je rozšíření oproti standardnímu Pascalu. Takový řetězec však musí být omezen uvozovkami, aby se dal odlišit od znakové konstanty.

Příklad:
```
"To je řetězec"
'Reader''s Digest'
""
```

Syntaxe zápisu řetězce znaků byla dále rozšířena o možnost zapisovat znaky pomocí jejich ASCII – kódu podobným způsobem, jako v Turbo Pascalu. Potřebujete-li mezi znaky '>>' a '<<' v řetězci zapsat znaky s kódy 14 a 232 (dekadicky), napíšete řetězec takto:

```
'>>'#14#232'<<'
```

Tímto způsobem můžete vložit libovolný počet speciálních znaků na libovolné místo řetězce včetně jeho začátku, například takto:

```
#14'Tento řetězec začíná znakem,jehož kód je 14'
```

Jedno z možných využití je rozdělení řetězce v proceduře Info_box do více řádků:

```
Info_box('','první řádek'#10'druhý řádek');
```

nebo:

```
str:= date2str(today, 1)+#10""+time2str(now, 1);
Info_box('Datum a čas', str);
```

Samostatný zápis jednoho speciálního znaku ovšem je zápisem znaku, nikoli řetězce. Tam, kde potřebujete vytvořit řetězec délky 1 obsahující speciální znak, vypomůžete si spojením speciálního znaku s prázdným řetězcem, tak jako v předchozím příkladu nebo i takto:

```
Info_box('Ahoj', ""#33);
```

Pokud je řetězec delší než jedna řádka, je možno jej na konci této řádky ukončit apostrofem nebo uvozovkou a na začátku další řádky apostrofem nebo uvozovkou pokračovat. Překladač obě takové části řetězce spojí.

```
'Tento řetězec zde '
'nekončí, nýbrž pokračuje.'
```

POZOR ! Při spojování obou částí řetězců překladač mezi ně nevkládá mezeru, takže kdyby na konci prvního řetězce ve výše uvedeném příkladu mezera chyběla, výsledný text by byl:

```
Tento řetězec zdenekončí, nýbrž pokračuje.
```

Speciální symboly

Speciálními symboly jsou následující znaky a dvojice znaků.

+	(	.	<=
-	)	,	>=
/	[	..	:=
^	]	;	.=

*	\	<	.=.
:	#	>	
=	~	<>	

Dále existují dvojznaky, které lze použít jako ekvivalenty některých speciálních symbolů nebo klíčových slov. Zde uvedeme, které to jsou, a více se o nich nebudeme zmiňovat.

Tyto ekvivalenty mají za cíl usnadnit život programátorům, kteří pracují v jazyku **C**:

!=	je ekvivalentní	<>
==	je ekvivalentní	=
||	je ekvivalentní	**OR**
&&	je ekvivalentní	**AND**
!	je ekvivalentní	**NOT**
/*	je ekvivalentní	**{**
***/**	je ekvivalentní	**}**
//	má stejný význam jako v jazyku **C**	

Oddělovače

Jediným významem oddělovačů je to, že od sebe oddělují ostatní symboly. Oddělovači jsou mezery, hranice řádek a komentáře.

Komentáře začínají znakem `{` a končí znakem `}`. Uvnitř komentáře smějí být libovolné znaky kromě znaku `}`. Místo znaků `{ }` je možné použít dvojice znaků `(* *)` nebo `/* */`.

Komentář smí také začínat dvojznakem `//` a pak končí na konci řádky.

Mezi kterékoliv dva symboly v programu lze vložit libovolný počet libovolných oddělovačů.

Do symbolů se nesmějí vkládat žádné oddělovače. Nejméně jedním oddělovačem je nutno oddělit navzájem sousedící identifikátory a čísla.

Konstantv a ieiich deklarace

Standardní konstanty

Konstanty označující hodnotu NONE .....

Pro označení žádné hodnoty je ve vnitřním programovacím jazyku pro každý jednoduchý typ zvláštní konstanta. Jsou to:

`NONECHAR, NONEBOOLEAN, NONESHORT, NONEINTEGER, NONEREAL, NONETIME, NIL, NONEDATE, NONEMONEY.`

Pro typy `STRING`, `CSSTRING` a `CSISTRING` (řetězce znaků) je hodnota NONE totožná s prázdným řetězcem. Proto ji lze zapsat jako dvojici bezprostředně následujících (bez mezery) znaků ‘uvozovky’. Pro typ ukazatel slouží hodnota NIL.

Ostatní konstanty .....

- Číselný rozsah typu i*nteger* popisuje konstanta `MAXINT = 2147483647`.
- Číselný rozsah typu s*hort* popisuje konstanta `MAXSHORT = 32767`.

Řada dalších konstant je popsána v kapitole *Přehled procedur a funkcí*.

Deklarace konstant

Deklarace konstant začínají vždy klíčovým slovem `CONST`.

Deklarace mají tento obecný tvar:

identifikátor_konstanty = konstantní_hodnota;

Konstantní hodnotou smí být číslo (celé, reálné, peněžní částka, datum, čas) nebo znaková konstanta, např.:

```
CONST
      Žádost_skončit = 1001;
      Minimální_mzda = 2200$;
      Oddělovací_znak = ',';
      Konec_staré_daně = 31.12.1992;
      Pi = 3.1415926;
```

Jazyk nyní umožňuje definovat konstanty typu řetězec znaků; tyto konstanty jsou typu String. Konstanty typu CSString, CSIString nelze definovat.

Typy a jejich deklarace

Jednoduché typy

Množina jednoduchých typů jazyka

- **Short**
  Celočíselný typ s rozsahem hodnot -32767 až 32767. Hodnota zabírá 2 byty.
- **Integer**
  Celočíselný typ s rozsahem hodnot -2147483647 až 2147483647. Hodnota zabírá 4 byty.
- **Money**
  Typ má hodnoty se dvěma desetinnými místy a rozsah hodnot je od $-1.4*10^{12}$ do $1.4*10^{12}$. Hodnota zabírá 6 bytů.
- **Real**
  Typ zahrnuje reálná čísla v rozsahu absolutních hodnot přibližně od 1.7E-308 do 1.7E308, kladná i záporná. Přesnost výpočtů je 15–16 desetinných míst. Hodnota zabírá 8 bytů paměti.
- **Char**
  Hodnotami typu jsou znaky. Hodnota zabírá 1 byt.
- **Boolean**
  Typ má pouze dvě hodnoty, a to TRUE a FALSE (nepočítáme-li „nedefinovanou“ hodnotu NONEBOOLEAN). Hodnota zabírá 1 byt.
- **Date**
  Hodnotou typu je datum, které musí být od začátku našeho letopočtu. Tato hodnota zabírá 4 byty.
- **Time**
  Hodnotou typu je čas, který se měří s přesností na tisíciny sekundy. Tato hodnota zabírá 4 byty.
- **Ukazatel**
  Zvláštní typ, jehož hodnoty lze považovat za abstrakci adresy umístění proměnné v paměti. Podle typů proměnných rozlišujeme i typy ukazatelů. Každý typ *ukazatel* je vázán na jeden typ proměnných, který se nazývá doménový typ ukazatele a určuje se v popisu typu *ukazatel*. Tento popis má tvar:

```
^doménový_typ
```

Doménovým typem může být libovolný typ a v popisu typu *ukazatel* se označuje identifikátorem typu. Například, je-li v programu definován identifikátor typu OBJEKT, pak deklarací:

```
type SPOJ = ^OBJEKT;
```

se definuje typ SPOJ, jehož hodnotami jsou ukazatele identifikující proměnné typu OBJEKT.
Proměnné typu *ukazatel* se deklarují obvyklým způsobem. Deklarací:

```
var P : SPOJ;
```

je zavedena proměnná *P*, jejímž oborem hodnot jsou ukazatele identifikující proměnné typu OBJEKT.
Zvláštní hodnota, která neidentifikuje *ukazatel* na žádnou proměnnou a která může být přiřazena libovolné proměnné typu *ukazatel*, je hodnota označená identifikátorem NIL. Přiřadíme-li tuto hodnotu `proměnné P` příkazem:

```
P := nil;
```

pak hodnota proměnné *P* neukazuje na žádnou proměnnou. Jiné hodnoty, které můžeme přiřadit do proměnných typu *ukazatel*, získáme prostřednictvím standardní funkce **Addr**.
Argumentem funkce **Addr** může být proměnná nebo položka strukturované proměnné libovolného typu; typ výsledku této funkce se řídí typem argumentu a je vždy ukazatelem na tento typ.
Je-li tedy *T* libovolný typ a proměnná *P* je typu *T*, je výraz `Addr(P)` typu *^T*.
Proměnná, kterou identifikuje hodnota proměnné typu *ukazate*l, se označuje zápisem:
`proměnná_typu_ukazatel^`
Například při deklaracích:

```
type pInt = ^integer;
var I : integer;
P : pInt;
```

se následujícími dvěma příkazy přiřadí do proměnné *I* hodnota 250 :

```
P := Addr(I); P^ := 250;
```

Proměnné typu *ukazatel* může být přiřazena hodnota jiné proměnné typu *ukazatel* (téhož doménového typu). Po přiřazení:

```
P := Q;
```

kde `P` a `Q` jsou proměnné typu *ukazatel*, mají obě stejnou hodnotu, tzn. ukazují na stejnou proměnnou. Zcela jiný význam má příkaz:

```
P^ := Q^;
```

Tímto příkazem se nezmění hodnota proměnné *P*, ale hodnota proměnné, na níž *P* ukazuje; této proměnné se přiřadí hodnota té proměnné, na kterou ukazuje proměnná *Q*.
Např. při deklaracích:

```
type pInt = ^integer;
var I, J : integer;
P, Q : pInt;
```

bude po vykonání následujících příkazů proměnná *I* mít hodnotu 333:

```
P := Addr(I);
Q := Addr(J);
I := 1; J:= 333;
P^ := Q^;
```

Výrazy, které jsou téhož typu *ukazatel*, mohou být použity jako operandy relačních operátorů = a <>. Těmito výrazy mohou být jak hodnoty proměnných typu *ukazatel*, tak přímo adresy proměnných získané voláním funkce `Addr`. (Výrazy nemusí být striktně téhož typu *ukazatel*; postačí, jsou-li jejich typy kompatibilní. Dva typy *ukazatel* jsou kompatibilní, jsou-li kompatibilní

doménové typy těchto ukazatelů. To platí samozřejmě u totožných typů. Jiným jednoduchým příkladem jsou textové řetězce stejné deklarované maximální délky. Obecná pravidla typové kompatibility jsou dosti složitá a zájemce se s nimi může seznámit v doporučené literatuře. Doménové typy ukazatelů mohou ovšem být zcela libovolné, např. opět typu *ukazatel*).

Hodnota výrazu:

```
P = Q
```

kde `P` a `Q` jsou proměnné (obecněji výrazy) typu *ukazatel* je TRUE tehdy, když obě proměnné ukazují na tutéž proměnnou. Jinak je hodnota FALSE. Operandem v těchto relacích může být také hodnota NIL.

Každá hodnota typu *ukazatel* zabírá 4 byty.

Ordinální typy

Typy *short, integer, char, boolean*, které jsme zde popsali nazýváme stejně jako ve standardním Pascalu typy ordinální. (Standardní Pascal ovšem typy *integer* a *short* nerozlišuje). Ordinální typy mají charakteristické použití při konstrukci strukturovaných příkazů. Jejich společnou vlastností je to, že množina hodnot každého z nich je konečná, je na ní definováno uspořádání a jsou pro ně definovány funkce **Ord**, **Succ** a **Pred**.

- Každé hodnotě *x* ordinálního typu *T* je funkcí `Ord` přiřazeno její ordinální číslo `Ord`(*x*). Funkce **Ord** přitom zachovává uspořádání, tzn., že pro každé dvě hodnoty *x* a *y* ordinálního typu *T* platí:
  x < *y* právě tehdy, když `Ord(x)` < `Ord`(*y*)
- Každé hodnotě *x* ordinálního typu *T* (s výjimkou největší hodnoty) je funkcí **Succ** přiřazen její přímý následník `Succ`(*x*), jehož ordinální číslo je o 1 větší, než ordinální číslo hodnoty *x*, tzn., že platí:
  `Ord(Succ(`*x*`)) = Ord(`*x*`) + 1`
- Každé hodnotě *x* ordinálního typu *T* (s výjimkou nejmenší hodnoty) je funkcí **Pred** přiřazen její přímý předchůdce `Pred`(*x*), jehož ordinální číslo je o 1 menší, než ordinální číslo hodnoty *x*, tzn., že platí:
  `Ord(Pred(`*x*`)) = Ord(`*x*`) - 1`

Pro hodnoty typu *boolean* platí:

`Ord(FALSE) = 0,` `Ord(TRUE) = 1,`
`Succ(FALSE) = TRUE,` `Pred(TRUE) = FALSE,`
`Pred(FALSE` a `Succ(TRUE)` není definováno.

Pro hodnoty typu *integer* platí:

`Ord`(x) = x,
`Succ`(x) = x+1 pro x < `MAXINT`
`Pred`(x) = x–1 pro x > `-MAXINT`

Pro hodnoty typu *short* platí:

`Ord`(x) = x,
`Succ`(x) = x+1 pro x < `MAXSHORT`
`Pred`(x) = x–1 pro x > `-MAXSHORT`

Pro hodnoty typu *char* vrací funkce **Ord** nezáporné číslo, které udává reprezentaci znaku v paměti počítače. Tato reprezentace závisí na příslušné zvolené normě reprezentace národních znaků. Například ve východoevropské verzi Windows se jedná o modifikovaný standard ASCII (tzv. code page 1250). Uspořádání množiny hodnot typu *char* je dáno právě hodnotami této interní reprezentace, kterou funkce `Ord` vrací.

Kromě výběru následníka a předchůdce je pro typ *char* definovaná funkce **Chr**, která je inverzní k funkci `Ord`. Argumentem této funkce musí být hodnota typu *short*, která je ordinálním číslem nějakého znaku; hodnotou funkce je příslušný znak. Pro libovolný znak *zn* tedy platí:

```
Chr(Ord(zn)) = zn
```

Funkce pro výběr následníka a předchůdce jsou definovány tak, že vracejí znak, jehož ordinální hodnota je o jedničku větší (resp. menší):

```
Succ(zn) = Chr(Ord(zn) + 1)
Pred(zn) = Chr(Ord(zn) – 1)
```

S použitím hodnot ordinálních typů při konstrukci strukturovaných příkazů se setkáme v souvislosti s příkazem větvení **case** a s příkazem cyklu **for**.

Strukturované typy

Strukturovanými typy proměnných ve 602Text jsou: řetězec znaků, typ pole, typ záznam a typ soubor.

Proměnná strukturovaného typu nesmí zabírat více paměti než 64KB. Na toto omezení je třeba pamatovat zejména při deklarování polí. Pokud jsou například složky pole typu r*eal*, pak počet složek nesmí být větší než cca 8000.

Typ řetězec znaků

Pro řetězce znaků jsou k dispozici 3 typy: **String**, **CSString** a **CSIString** . Hodnotami těchto typů jsou řetězce znaků až do určité délky. Popisy typu *řetězec znaků* vypadají takto:

- `String[délka]`
- `CSString[délka]`
- `CSIString[délka]`

Odlišnosti jednotlivých typů:

Řetězce těchto tří typů se neliší svými hodnotami, ale pouze tím, jak se mezi sebou porovnávají (viz popis relací).

Nejdelší řetězec, který se dá zapsat do proměnné takového typu, má délku stejnou jako délka uvedená v hranatých závorkách. Každý řetězec má na svém konci omezovací znak s hodnotou 0. Pro tento znak se při deklaraci řetězce rezervuje místo automaticky.

Pokud budete řetězec indexovat a pracovat s jeho jednotlivými znaky, index prvního znaku je **1**. Poslední index je o **1** větší než zadaná délka, ale do znaku s tímto indexem není dovoleno zapsat nic jiného než omezovací znak, tedy hodnotu `Chr(0)`.

Typ pole

Ze složek libovolného typu lze vytvořit typ *pole* popisem:

```
ARRAY [dolní_mez .. horní_mez] OF typ_složek;
```

kde dolní_mez a horní_mez jsou celá čísla a dolní_mez není větší než horní_mez.

Vícerozměrná pole lze deklarovat tak, že vytvoříte *pole polí*, například:

```
ARRAY [dolní_mez1 .. horní_mez1] OF
     ARRAY [dolní_mez2 .. horní_mez2] OF typ_složek;
```

tento popis typu je ekvivalentní zápisu:

```
ARRAY [dolní_mez1 .. horní_mez1, dolní_mez2 .. horní_mez2] OF
     typ_složek;
```

Složky typu char

Typ *pole*, jehož dolní mezí je 1 a jehož složky jsou typu *char*, je kompatibilní s typem *string*. Pamatujte však na to, že při deklaraci pole se na jeho konec automaticky nepřidá znak pro omezovač řetězce. Proto, chcete-li s polem 10 znaků pracovat stejně jako s řetězcem deklarovaným jako *string*`[10]` musíte jej deklarovat jako:

```
ARRAY[1..11] OF Char;
```

POZOR ! Pokud chcete s polem znaků pracovat jako s celkem, například jej celý vypsat na obrazovku nebo předložit jako parametr kterékoli standardní proceduře či funkci, pak za posledním platným znakem řetězce musí být znak s kódem 0. Tuto podmínku splňují řetězce vytvořené zápisem konstantních řetězců nebo voláním standardních funkcí s výsledkem typu s*tring* a je nutno ji zachovat operacemi, které budete nad polem resp. řetězcem znaků provádět.

Řetězce ve 602Text jsou jednou z možných implementací znakových řetězců specifikovaných ve standardním Pascalu, nejsou ovšem totožné s implementací řetězců použitou od nejstarších verzí Turbo Pascalu. Jsou spíše obdobou (až na indexaci od 1) řetězců zvaných *PChar* implementovaných v modulu *string* v Borland Pascalu. V nulté složce řetězce není uložena délka.
Nultá složka řetězce ve 602Text neexistuje.

Typ záznam

Typ *záznam* lze definovat popisem:

```
RECORD skupina ... skupina END;
```

kde skupina má tuto strukturu:

identifikátor, ... , *identifikátor* : *typ*;

Například:

```
TYPE rec = RECORD
     del : Boolean;
     datum : Date;
     obs : ARRAY[1..28] OF Boolean;
     kdo : ARRAY[1..28] OF String[16];
     co : ARRAY[1..28] OF String[30];
     END;
```

Typ soubor

Pro práci s textovými soubory uloženými pod správou operačního systému slouží typ *soubor*.

Tento typ je jediný a označuje se klíčovým slovem FILE. Jiné než textové soubory nejsou implementovány. Pro manipulaci s nimi slouží procedury a funkce:

`Reset, Rewrite, Close, Seek, Eof, Filelength, Read, Write a Writeln.`

Příklad deklarace proměnné typu s*oubor* a otevření textového souboru:

```
VAR  f : FILE;
....
Reset(f,'C:\WT602\OUTPUT\RES.TXT');
```

Deklarace typů

Deklarace typů začíná klíčovým slovem *type* a pokračuje deklaracemi ve tvaru:

identifikátor_typu = popis_typu;

kde popis typu může být popisem některého z uvedených strukturovaných typů, popisem typu *ukazatel* nebo identifikátorem již dříve definovaného typu; např.:

```
TYPE
     str = CSString[10];
     cele_cislo = integer;
```

```
rec = RECORD
     p : str;
     a, b : Integer;
     c : cele_cislo;
     r : Real
END;
rec_alias = rec;
pole = ARRAY [1 .. 100] OF rec;
Tp_cele_cislo = ^cele_cislo;
Tp_rec = ^rec;
```

Deklarace proměnnvch

Deklarace proměnných začínají klíčovým slovem **VAR** a mají tento tvar:

identifikátor, ... , identifikátor : *typ*;

kde specifikaci typu může představovat buď identifikátor standardního nebo dříve definovaného typu, nebo také popis typu (typy takto zavedené pouze svým popisem jsou tzv. nepojmenované typy).

Příklad deklarací proměnných:

```
VAR
     i, j, m : Integer;
     r : Real;
     jméno : ARRAY [1..20] OF Char;
     tab : pole;
     soubor : FILE;
```

Deklarace procedur a funkcí

Zápis deklarací

Deklarace procedury má tvar:

```
PROCEDURE identifikátor (skupina_par ... skupina_par); blok;
```

Deklarace funkce má tvar:

```
FUNCTION identifikátor(skupina_par ...
skupina_par) : typ; blok;
```

Část deklarace procedury resp. funkce uvedena před blokem se nazývá **hlavička** procedury resp. funkce.

Závorky a parametry v nich deklarované mohou být z hlavičky vypuštěny – pak jde o proceduru resp. o funkci bez parametrů.

Příklad funkce **Rand**, která vrací pseudonáhodná čísla mezi 0 a 1:

```
Function Rand : Real;
CONST
     c1 = 13849;
     c2 = 27181;
     c3 = 65536;
BEGIN
     Seed := (c1+(c2*seed)) MOD c3;
     Rand := Seed/c3;
END;
```

Konstrukce *skupina_par* deklaruje skupinu parametrů stejného typu a vypadá takto:

```
identifikátor, ... , identifikátor : typ;
```

nebo

```
VAR identifikátor, ... , identifikátor : typ;
```

V prvním případě jsou parametry předávány hodnotou a nazývají se tedy **hodnotové**. V druhém případě se předávají odkazem, předává se tedy adresa skutečného parametru. Takové parametry se nazývají **referenční**.

Jako specifikace typu může v deklaraci skupiny parametrů vystupovat (ve shodě se standardním Pascalem) pouze identifikátor typu, nikoli popis typu. Výjimkou z tohoto pravidla je typ **string**, kde je možno použít přímo popis typu *string* [délka] a není nutno pro tento typ zavádět v předchozích deklaracích zvláštní identifikátor.

Ve specifikaci typu se může před identifikátorem typu vyskytovat klíčové slovo `const`; pak se jedná o skupinu konstantních parametrů. O této vlastnosti jazyka (která je rozšířením vůči standardnímu Pascalu) pojednáváme dále.

Blok tvořící součást deklarace procedury nebo funkce má stejnou strukturu jako celý program až na dvě výjimky:

- v této implementaci jazyka nemůže v sobě obsahovat deklarace dalších procedur a funkcí
- není ukončen tečkou.

Typ, uvedený na konci hlavičky funkce je typem jejího výsledku. Nesmí to být strukturovaný typ.

V 602Text tedy deklarace podprogramů (procedur a funkcí) nelze do sebe vnořovat. Uvnitř podprogramu nemůže být deklarován jiný podprogram.

Konstantní parametry procedur a funkcí

Jak jsme již uvedli, ve specifikaci typu skupiny parametrů procedury či funkce (jak standardní tak uživatelem definované) se může vyskytovat klíčové slovo **const**. Například:

```
type  pole = array[1..10] of integer ;
procedure suma(var A, B : const pole; var result : pole) ;
var  i : integer;
begin
     for i := 1 to 10 do
          result[i] := A[i] + B[i];
end;
```

V tomto případě jsou `A, B` konstantní parametry. Překladači je tím dáno najevo, že jsou pouze vstupními parametry a není žádoucí je kdekoli v bloku procedury modifikovat. Překladač pak nahlásí chybu u takového přiřazení uvnitř procedury, ve kterém by se na levé straně vyskytoval identifikátor konstantního parametru nebo přístup k jeho složce (je-li strukturovaného typu).

Překladač nahlásí chybu i tehdy, pokud by konstantní parametr (event. jeho složka) byl skutečným parametrem při volání jiné procedury/funkce, jejíž odpovídající formální parametr by nebyl deklarován jako konstantní a byl by předáván referencí.

Poznámka A

Zabezpečení toho, aby při volání procedury nebyly některé skutečné parametry modifikovány je možno ovšem docílit také tak, že příslušné formální parametry v deklaraci procedury deklarujeme jako hodnotové a procedura si tedy při každém volání vytvoří jejich lokální kopii. To je však např. u vícerozměrných polí nevýhodné z důvodu paměťových nároků.

Poznámka B

I hodnotové parametry je možno deklarovat jako konstantní; jak ovšem plyne z předchozí poznámky není tato možnost upotřebitelná v tolika případech jako u parametrů referenčních.

Poznámka C

Je-li konstantní parametr procedury/funkce strukturovaného typu a je předáván referencí, je možno při volání použít jako skutečný parametr také konstantu příslušného strukturovaného typu. „Použitím“

konstanty se zde rozumí identifikátor dříve definované konstanty; v případě typu **řetězec znaků** (jiné strukturované konstanty zatím ani nelze definovat) je možný zápis přímo řetězce znaků. Například:

```
const
     otázka = "Vaše jméno" ?;
var
     strA, strB : string[30] ;
begin
     Input_box ( otazka, strA, 30 ) ;
     Input_box ("Vaše příjmení ? " , strB, 30) ;
     ...
```

U těchto referenčních parametrů, které v hlavičce procedury/funkce nejsou deklarovány jako konstantní nelze ovšem při jejím volání jako skutečný parametr konstantu předávat. Překladač na takovou chybu upozorní. Nepřeloží například toto:

```
Input_box ("Ahoj" , "Jak se máš" , 20) ;
```

Jedním z důvodů zavedení konstantních parametrů bylo právě eliminování podobných chyb již na úrovni překladu.

Poznámka D

Typ *string* je jediný strukturovaný typ, který může být výsledkem funkce. Proto pouze na tento typ se vztahuje další rozšíření standardního Pascalu. Je-li referenční parametr procedury\funkce strukturovaného typu a je-li deklarován jako konstantní, lze jako skutečný parametr použít návratovou hodnotu funkce kompatibilního typu. Například:

```
type
     str20 = string[20];
var
     s : str20;
function f ( d : date) : str20;
begin
     f := ´Dnes je ´ + Date2str(d, 0) + ´, jak se máte ? ´;
end;
begin
     Input_box ( f (Today ), s) ;
end.
```

Implicitní parametry

Další rozšíření syntaxe se týká tzv. implicitních (default) parametrů. Některé standardní procedury / funkce mohou mít některé (event. všechny) parametry implicitní; tzn, že v syntaxi zápisu volání takové funkce jsou tyto parametry vypustitelné a při provádění takto zavolané procedury / funkce budou mít svou implicitní hodnotu.

V deklaraci standardní procedury/funkce s implicitními parametry stejně jako v jazyku **C++** platí, že za (v pořadí zleva) prvním implicitním parametrem následují už jen implicitní parametry. Pokud při volání takové procedury/funkce je vypuštěn některý skutečný parametr odpovídající formálnímu implicitnímu parametru, je nutno vypustit i všechny následující implicitní parametry. Bez zavedení těchto konvencí (při vypouštění parametrů způsobem ponechaným pouze na fantazii programátora) by totiž překladač obecně nemohl rozpoznat, který skutečný parametr v zápisu volání přísluší ke kterému formálnímu parametru v deklaraci.

Příkladem je standardní funkce **CharLeft,** která má dva parametry; oba s implicitní hodnotou. Je možno ji zavolat těmito způsoby:

```
CharLeft(2, TRUE);      {zápis obou parametrů}
CharLeft(2);            {druhý parametr má implicitní hodnotu (FALSE )}
CharLeft;               {oba parametry mají default hodnoty (1, FALSE)}
```

Deklarace externích procedur a funkcí

Vnitřní programovací jazyk 602Text umožňuje volat i procedury a funkce implementované v externích knihovnách typu DLL.

Lze volat pouze ty procedury a funkce, které vyhovují volací konvenci `stdcall`. Všechny takovéto procedury a funkce musí být v programu před místem svého prvního použití deklarovány. Deklarace má jeden z těchto dvou tvarů:

hlavička procedury nebo funkce;
EXTERNAL '*jméno_knihovny*'

nebo

hlavička procedury nebo funkce;
EXTERNAL '*jméno_knihovny*' NAME' *jméno_funkce_v_knihovně*';

Pokud je jméno funkce v programu a v knihovně stejné, použijete první, zkrácený tvar. Jsou-li různé použijete druhý tvar. V praxi je nutno jméno funkce v knihovně uvádět pouze tehdy, pokud je delší než 16 znaků nebo pokud ve svém programu již máte stejně pojmenovaný objekt.

Jestliže jméno dynamické knihovny uvedete bez přípony, předpokládá se přípona `DLL`. Upozorňujeme, že některé knihovny mají přípony jiné než `DLL` a je nutno je uvést.

Příklad některých deklarací:

```
function GetWindowDC (hwnd : integer) : integer;
EXTERNAL "USER32.DLL" name "GetWindowDC";

function LineTo (hdc : integer; x, y : integer): boolean;
EXTERNAL "GDI32.DLL" name "LineTo";

function GetVersion : integer;
EXTERNAL "KERNEL32.DLL" name "GetVersion";
```

V deklaraci je nutno přesně dodržet popis parametrů procedury / funkce tak, jak je v externí knihovně implementován. Překladač může ověřit pouze formální správnost deklarace; případné chyby v počtu, typu nebo pořadí parametrů, event. v typu návratové hodnoty se projeví až při běhu programu jeho nekorektním chováním. Explicitně upozorňujeme na nebezpečí záměny typu *short* zabírajícího 2 byty a typu *integer*, jehož hodnoty v interním jazyku zabírají 4 byty.

Struktura programu

Program se skládá z deklarační části a z těla.

Hlavička programu

Hlavičkou programu začíná vlastní deklarační část programu. Hlavička začíná klíčovým slovem **Program**, za kterým následuje název programu:

```
Program identifikátor;
```

Hlavička programu se nesmí vyskytovat jinde než na začátku deklarační části. Její uvedení je nepovinné.

Deklarační část programu

Deklarační část začíná hlavičkou programu (nepovinnou), za níž následuje posloupnost deklarací konstant, typů, proměnných, procedur a funkcí uvedených v libovolném pořadí. Tělo je složeným příkazem (viz dále) a je ukončeno tečkou. Za touto tečkou již nesmí nic následovat.

Rozsahy platnosti identifikátorů

Identifikátory označují v programu konstanty, typy, proměnné, procedury a funkce, parametry, složky záznamů. Každý identifikátor musí být deklarován předtím, než je poprvé použit.

Deklarace identifikátorů

Identifikátory se deklarují v deklaračních částech programu, procedur a funkcí. Přitom procedury a funkce se smějí deklarovat pouze v deklarační části programu. Parametr je deklarován svým výskytem v hlavičce procedury nebo funkce. Složka záznamu je deklarována svým výskytem v deklaraci příslušného typu *záznam*.

Platnost deklarace

Je-li identifikátor deklarován v deklarační části programu, pak tato deklarace platí až do konce programu (s níže uvedenou výjimkou). Je-li identifikátor deklarován v proceduře nebo funkci, pak platí pouze uvnitř této procedury resp. funkce. V jedné deklarační části nesmí být dvakrát deklarován stejný identifikátor. Identifikátory deklarované v proceduře/funkci nesmí být stejné jako parametry této procedury/funkce. To však neplatí pro identifikátory složek záznamů, které se musí lišit pouze navzájem mezi sebou v rámci každého typu *záznam*.

Je-li v proceduře/funkci deklarován identifikátor stejný jako identifikátor deklarovaný v programu, pak tato nová deklarace překryje uvnitř této procedury/ funkce původní deklaraci.

Přístup k proměnným

Potřebujete-li v programu pracovat s proměnnou, napíšete v příslušném kontextu prostě její identifikátor. Složitější případ nastává, pokud chcete pracovat s některou složkou strukturované proměnné.

Vyznačení prvku proměnné typu pole .....

K proměnné typu *pole* nebo *řetězec znaků* lze připojit index ve tvaru:

```
[ výraz ]
```

kde výraz musí být celočíselného typu a jeho hodnota musí ležet v mezích daných deklarací pole (pro typy s*tring* v mezích 1 až délka řetězce plus 1).

Jsou-li tedy například *pole* a *řetězec znaků* deklarovány:

```
VAR
     P : ARRAY[1..10] OF Integer;
     T : String[20];
```

pak zápisy P[1], P[2], ..., P[10] označují složky pole, T[1], ..., T[20] složky řetězce – v prvním případě čísla typu *integer*, v druhém znaky. Složka T[21] smí obsahovat pouze omezovací znak řetězce.

Například do složky dvourozměrného pole s indexem 1,1 se přiřadí hodnota:

```
pole[1][1] := ...
```

nebo zkráceně:

```
pole[1, 1] := ...
```

POZOR ! Překročení mezí *pole* je jednou z nejběžnějších a nejzhoubnějších programátorských chyb. Například pokus zapisovat hodnotu do složky P[0] nebo P[11] výše deklarovaného pole vede k nezamýšlenému přepsání neznámého místa v paměti a v krajním případě i ke zhroucení aplikace.

Vyznačení složky proměnné typu záznam .....

K proměnné typu *záznam* lze připojit specifikaci složky ve tvaru:
. *identifikátor_složky*

Například v záznamu deklarovaném:

```
VAR rec : RECORD
     p : Str;
     a, b : Integer;
     r : Real
END;
```

označují zápisy `REC.p, REC.a, REC.b, REC.r` jednotlivé složky záznamu.

Přechody přes ukazatel .....

Odkaz na proměnnou, na niž odkazuje ukazatel, lze získat tak, že za přístup k tomuto ukazateli připojíme symbol ^.

Například při deklaracích:

```
TYPE uk = ^char;
VAR ch1, ch2 : char
     u1, u2 : uk;
```

lze obsah proměnných *ch1*, *ch2* přiřazovat také takto:

```
u1 := Addr(ch1);
u2 := Addr(ch2);
u1^ := u2^;
```

Podrobnější popis použití ukazatelů se nachází v popisu typu *ukazatel*.

Výrazv

Ve výrazech lze používat proměnné, konstanty, čísla, data, hodnoty času, znaky, řetězce znaků a volání funkcí.

Precedence operátorů

Operátory v pořadí od nejvyšší priority jsou:

1. **NOT**
2. ***, /, AND, DIV, MOD**
3. **+,** – (unární i binární), **OR**
4. **<, >, =, <=, >=, <>, .=, .=., ~**

Při vyhodnocování výrazu je pořadí operací dáno v první řadě strukturou výrazu – části výrazu uzavřené v závorkách se vyhodnocují předem. Pak rozhodují výše uvedené priority operátorů. Operace se stejnou prioritou se vyhodnocují zleva doprava.

Na priority operátorů je nutno myslet při konstrukci výrazů. Předpokládejme, že proměnná *I* je typu *integer*. Pak výraz:

```
I >= 10 AND I < 300
```

je chybný, neboť operátor AND má vyšší prioritu než relace >= a < . Tuto podmínku je proto nutno zapsat ve tvaru:

```
(I >= 10) AND (I < 300)
```

Při vyhodnocování logických podmínek (tj. výrazů typu *boolean*) se postupuje jen tak dlouho, dokud není zřejmé, jakou má výraz hodnotu. (Jedná se o tzv. zkrácené vyhodnocování boolovských výrazů). Například ve výrazu:

```
cond AND Fnc(I)
```

se funkce `Fnc` vůbec nezavolá, pokud proměnná *cond* má hodnotu FALSE. Toto lze s výhodou využít v podmínkách cyklu jako:

```
WHILE NOT konec AND Fnc(I) DO ...
```

Aritmetické operace

K dispozici jsou operace +, –, *, /, DIV a MOD. Operace DIV a MOD musí mít oba argumenty celočíselného typu.

Ve výrazech lze libovolně kombinovat typy *short* a *integer*. Všechny operace nad těmito typy se provádějí jako pro typ *integer* a mají výsledek typu *integer*, s výjimkou operace dělení , v níž je podíl vždy typu *real***.**

Je-li jedním argumentem typ *real*, je hodnota výrazu typu *real*.

Pro typ *money* lze používat stejné operace jako pro celočíselné typy, avšak s jednou výjimkou: nelze mezi sebou násobit ani dělit operacemi **DIV** ani **MOD** dvě hodnoty typu *money*. Typ *money* lze kombinovat ve výrazech s typy *integer, short* a *real*. Výsledek operace s typem *real* je typu *real*, v ostatních případech je typu *money*.

Nad celočíselnými typy lze provádět i bitové operace sjednocení, průniku a negace pomocí operátorů **OR**, **AND**, **NOT**.

Například:

```
10 OR 6 = 14
10 AND 6 = 2
NOT 6 = -7.
```

Operace s datem

K hodnotě typu *date* lze přičíst (resp. odečíst) celé číslo – interpretuje se to jako přičtení (resp. odečtení) dnů.

Dvě hodnoty typu *date* lze od sebe odečíst. Rozdílem je počet dnů mezi oběma daty (tedy číslo typu *integer*).

Při operacích nad daty se berou v úvahu přestupné roky (ale nepočítá se s reformou kalendáře, která proběhla v roce 1582). Operace s datem tedy probíhají správně vzhledem ke gregoriánskému kalendáři.

Operace s časem

K hodnotě typu *time* lze přičíst (resp. odečíst) celé číslo – interpretuje se to jako přičtení (resp. odečtení) tisícin sekundy. Při překročení hranice dne není výsledek operace definován.

Dvě hodnoty typu *time* lze od sebe odečíst. Rozdílem je počet tisícin sekundy mezi oběma časy (tedy číslo typu *integer*). Předpokládá se, že oba časy patří do téhož dne, takže výsledek může být záporný.

Operace se znakovými řetězci

Znakové řetězce se dají spojovat pomocí operátoru +. Výsledný řetězec může mít maximální délku 511 znaků. Pokud je součet délek spojovaných řetězců větší, bude jeho konec přesahující délku 511 znaků odříznut.

Řetězce se dají porovnávat v řadě relací uvedených níže.

POZOR ! Standardním procedurám a funkcím se parametr typu *řetězec znaků* předává zásadně odkazem. Proto nelze jako skutečný parametr použít výraz (např. obsahující dva řetězce spojené operátorem +). Místo toho je nutno oba řetězce napřed spojit do třetího řetězce a ten pak předat jako parametr.

Příkladně v záznamu REC je položka MUŽ typu *boolean*:

```
str := (REC.MUŽ ? 'narozen' : 'narozena') +
       date2str(REC.DAT_NAR,1);
Info_box('informace', str);
```

Relace

Operátory vyjadřující relace jsou <, >, =, <=, >=, <>, .=, .=., ~. Poslední tři je možno použít pouze na znakové řetězce.

Výsledkem relace je hodnota typu *boolean*. V relaci lze použít operandy stejných typů, dále lze mezi sebou libovolně kombinovat typy *short, integer, money* a *real* a také různé typy řetězců navzájem.

Typy **STRING**, **CSSTRING** a **CSISTRING** se neliší v tom, jakých hodnot mohou nabývat. Rozdíl je pouze v tom, jak se jejich hodnoty uspořádávají:

STRING	lexikografické uspořádání odvozené z interního kódu znaků.
CSSTRING	uspořádání dle pravidel českého (a slovenského) jazyka, přitom velká písmena jsou před malými.
CSISTRING	uspořádání dle pravidel českého (a slovenského) jazyka, přitom se nerozlišuje mezi velkými a malými písmeny.

Při porovnávání řetězců různého typu se postupuje tak, že způsob porovnávání určuje ten z nich, který má vyšší pořadí v tomto výčtu:

- String
- CSString
- CSIString

Speciální řetězcové relace mají tento význam:

x .= y	řetězec *y* je prefixem řetězce *x*
x .=. y	řetězec *y* je obsažen v řetězci *x*
x ~ y	řetězce *x* a *y* se navzájem liší nanejvýš mezerami, diakritikou nebo velikostí písmen.

Podmíněný výraz

Vnitřní jazyk 602Text obsahuje také tzv. podmíněný výraz převzatý z jazyka C. Jde o přibližnou obdobu podmíněného příkazu.

Podmíněný výraz má tento tvar:

výraz_1 ? *výraz_2* : *výraz_3*

kde výraz_1 je nepodmíněný výraz typu *boolean* a výraz_2 je nepodmíněný výraz stejného typu jako *výraz_3*. Přitom *výraz_3* může být i podmíněný.

Při vyhodnocování podmíněného výrazu se nejprve vyhodnotí *výraz_1*. Pokud má hodnotu TRUE, vyhodnotí se dále *výraz_2*, jinak se vyhodnotí *výraz_3*. Ten z nich, který se vyhodnotil, udává zároveň hodnotu celého podmíněného výrazu.

Příkladně absolutní hodnotu proměnné X lze s pomocí podmíněného výrazu zapsat takto:

```
X >= 0 ? X : -X
```

Příkazv

Typickým rysem strukturovaného programovacího jazyka je, že příkazy lze vkládat do sebe. Tělo hlavního programu, každé procedury i funkce je tvořeno jedním složeným příkazem. Ten obsahuje posloupnost příkazů, které mohou obsahovat další příkazy a tak dále.

K dispozici máte následující základní příkazy:

- Přiřazovací příkaz
- Složený příkaz
- Podmíněný příkaz
- Cyklus **WHILE**
- Cyklus **REPEAT**
- Cyklus **FOR**
- Příkaz **CASE**
- Volání procedury
- Příkaz **HALT**

Přiřazovací příkaz

Přiřazovací příkaz má tvar:

proměnná := *výraz*

Při provádění přiřazovacího příkazu se nejprve zjistí, jaká proměnná je na jeho levé straně, pak se spočte hodnota výrazu na pravé straně a nakonec se hodnota výrazu přiřadí proměnné.

Konverze typu při přiřazení .....

Pokud typy výrazu a proměnné v přiřazovacím příkazu nejsou shodné, překladač v některých případech sám zařídí potřebnou konverzi typu tak, aby přiřazení bylo možno provést. V takovém případě říkáme, že typy jsou kompatibilní vůči přiřazení. V tomto tvrzení je ovšem důležité pořadí obou typů; např. do proměnné typu *real* lze přiřadit výraz typu *integer*; naopak to však neplatí.

Bez omezení se provedou konverze mezi typy *integer, short* a *money*. Všechny tři tyto typy se také konvertují na typ *real*. Hodnota typu *real* se automaticky konvertuje na typ *money*. Konverzi typu *real* na typ *integer* nebo *short* musíte ale zařídit sami, například pomocí funkcí `Round` nebo `Trunc`.

Přímo přiřazovat lze také všechny typy řetězec znaků a pole znaků. Pokud však přiřazujete obsah pole znaků, je nezbytné, aby za posledním znakem v poli byl omezovací znak s hodnotou nula (`Chr`(0)).

Lze navzájem přiřazovat řetězec znaků a znak. Ze znaku vzniká přiřazením do řetězcové proměnné řetězec délky jedna, při přiřazení řetězce do znaku se přenese pouze první znak. (Tato konverze se však neprovádí ve výrazech, např. řetězec znaků a znak nelze prostým sčítáním spojovat.)
Pro jiné typy žádná automatická konverze neprobíhá. Je proto nutno buď příslušný převod naprogramovat nebo využít některou ze standardních konverzních procedur.

U typů kompatibilních vůči přiřazení se příslušné konverze provádějí také v případě volání procedury nebo funkce pro ty parametry, které jsou předávány hodnotou. Pro referenční parametry je ovšem vyžadována kompatibilita typů jako taková, která má ve standardním **Pascalu** oproti kompatibilitě vůči přiřazení pravidla podstatně přísnější. Výjimky vůči tomuto pravidlu standardního **Pascalu** jsou popsané v popisu volání procedury.

Složený příkaz

Složený příkaz sdružuje posloupnost po sobě následujících příkazů do jediného příkazu.

Složený příkaz má tento tvar:

`BEGIN` *příkaz*; ... ; *příkaz* `END`

Provedení složeného příkazu znamená postupné provedení příkazů v něm obsažených.

Podmíněný příkaz

Podmíněný příkaz má tvar:

`IF` *výraz* `THEN` *příkaz1* `ELSE` *příkaz2*

nebo jen:

`IF` *výraz* `THEN` *příkaz1*

Při jeho provádění se nejprve vyhodnotí *výraz*, jenž musí být typu *boolean*. Pokud má hodnotu TRUE, provede se *příkaz1*, pokud má hodnotu FALSE, provede se *příkaz2* (pokud existuje, jinak se neprovede nic).

Cyklus WHILE

Cyklus **WHILE** má tento tvar:

`WHILE` *výraz* `DO` *příkaz*

Výraz musí být typu *boolean*. Dokud má tento výraz hodnotu **TRUE**, opakuje se provádění příkazu uvedeného za **DO**. Protože se hodnota výrazu počítá před provedením příkazu, nemusí se tento příkaz provést ani jednou.

Cyklus REPEAT

Cyklus REPEAT má tento tvar:

`REPEAT` *příkaz*; ... ; *příkaz* `UNTIL` *výraz*

Výraz musí být typu *boolean*. Příkazy uvedené mezi **REPEAT** a **UNTIL** se opakují tak dlouho, dokud výraz nenabude hodnoty **TRUE**. Protože se hodnota výrazu počítá až po provedení příkazů, provedou se všechny příkazy nejméně jednou.

Cyklus FOR

Cyklus FOR má tvar:

(1) `FOR` *identifikátor* `:=` *mez1* `TO` *mez2* `DO` *příkaz*

nebo

```
(2)  FOR identifikátor := mez1 DOWNTO mez2 DO příkaz
```

Identifikátor musí označovat proměnnou ordinálního typu (*integer, short, char, boolean*). Tato proměnná se nazývá **řídící proměnná** příkazu **FOR**. Výrazy *mez1* a *mez2* musí být výrazy typu kompatibilního vůči přiřazení s typem řídící proměnné. Je-li příkaz **FOR** součástí procedury nebo funkce, pak řídící proměnná musí být deklarována jako lokální proměnná v této proceduře nebo funkci. (Toto omezení standardního Pascalu má význam pro zmenšení rizika vzniku chyb v programu vinou vedlejších efektů FOR cyklu).

Význam příkazu **(1)** je stejný jako význam složeného příkazu (identifikátor řídící proměnné označujeme *rp*):

```
begin
  pom1 := výraz1;
  pom2 := výraz2;
  if pom1 <= pom2 then
   begin
     rp := pom1;
     příkaz;
     while rp <> pom2 do
       begin
        rp := Succ(rp);
        příkaz
       end
    end
end
```

Proměnné *pom1* a *pom2* jsou pomocné (programátorovi nepřístupné) pracovní proměnné, které vytvoří překladač pro uložení hodnot výrazů *výraz1* a *výraz2*; hodnoty těchto výrazů se počítají pouze jednou před prvním provedením příkazu tvořícího tělo cyklu. Mezní hodnoty řídící proměnné tedy nemohou být uvnitř cyklu změněny. Uvnitř cyklu nelze explicitně změnit ani hodnotu řídící proměnné, neboť příkaz tvaru:

```
rp := výraz
```

je uvnitř těla cyklu příkazu **FOR** zakázán. Uvnitř těla cyklu příkazu **FOR** nelze také volat procedury či funkce, pokud by jedním ze skutečných parametrů byla řídící proměnná a tento parametr by byl předáván referencí. Z těchto omezení vyplývá, že počet provedení těla cyklu nelze ovlivnit instrukcemi prováděnými uvnitř cyklu a je vždy konečný. Jestliže před provedením příkazu **FOR** platí:

```
výraz1 > výraz2
```

pak se dílčí příkaz neprovede. Po skončení příkazu **FOR** není hodnota řídící proměnné definována (toto pravidlo v definici jazyka poskytuje autorům překladače určitý prostor pro výběr nejefektivnější implementace příkazu **FOR**). Je-li klíčové slovo `TO` v příkazu **FOR** nahrazeno klíčovým slovem `DOWNTO`, pak se příkaz **FOR** provádí tak, že hodnota řídící proměnné se postupně zmenšuje od hodnoty *výraz1* do hodnoty *výraz2*. Platí-li však:

```
výraz1 < výraz2
```

neprovede se nic. Význam této formy příkazu **FOR** lze tedy rozepsat :

```
begin
  pom1 := výraz1;
  pom2 := výraz2;
  if pom1 >= pom2 then
   begin
     rp := pom1;
     příkaz;
     while rp <> pom2 do
       begin
        rp := Pred(rp);
        příkaz
```

```
        end
    end
end
```

Příkaz CASE

Příkaz **CASE** slouží k rozvětvení programu do mnoha větví na základě hodnoty výrazu ordinálního typu (*short, integer, char, boolean*).

Příkaz má tento tvar:

```
CASE výraz OF
  konstanta, ... konstanta : příkaz;
  konstanta, ... konstanta : příkaz;
  ...
  ELSE : příkaz;
END
```

Větev začínající slovem **ELSE** se může vyskytnout na libovolném místě mezi ostatními větvemi, ale nejvýše jednou. Její výskyt není povinný. Každá konstanta smí být v příkazu **CASE** použita nejvýše jednou.

Při provádění příkazu **CASE** se nejprve vyhodnotí *výraz*. Jeho hodnota se porovná s uvedenými konstantami. Pokud se některá konstanta rovná hodnotě výrazu, pak se provede *příkaz* následující za touto konstantou. Pokud žádná konstanta nemá stejnou hodnotu jako *výraz*, pak má-li příkaz větev **ELSE**, provede se příkaz v této větvi, pokud větev **ELSE** schází, neprovede se nic.

Konstanty v příkazu **CASE** lze uvádět jak zápisem jejich hodnot, tak i jako identifikátory deklarované v deklarační části jako konstanty příslušného typu.

Volání procedury

Volání procedury má tento tvar:

identifikátor (*parametr* , ... , *parametr*)

Identifikátor označuje volanou proceduru. Parametry uvedené v tomto příkazu odpovídají po řadě parametrům uvedeným v deklaraci procedury. Hodnotou lze proceduře předat libovolný výraz vyhovujícího typu (automaticky se provádějí stejné konverze jako v přiřazovacím příkazu, viz výše). Odkazem lze předat pouze proměnnou stejného typu, jaký má parametr.
Výjimkou oproti standardnímu Pascalu je předávání polí. Při předávání pole odkazem postačí, když skutečný a formální parametr mají stejný typ složek a stejnou dolní mez.

Další výjimkou je předávání parametrů typu *string*. Tam, kde jedním z formálních parametrů je parametr typu *string* předávaný odkazem (referenční) a je současně deklarován jako konstantní, může být při volání skutečným parametrem kromě přístupu k proměnné tohoto typu i zápis konstanty typu *string* anebo výsledek volání funkce (standardní i uživatelem definované), jejíž návratová hodnota je typu *string*.

Při volání standardních procedur a funkcí byly vůči standardnímu Pascalu umožněny ještě další výjimky (mj. byly zavedeny tzv. implicitní parametry).
Popisem těchto odlišností se zabývá poznámka v kapitole *Zápis deklarací*.

Pokud procedura nemá žádné parametry, pak se v jejím volání neobjeví závorky ani hodnoty skutečných parametrů.
Jako proceduru lze zavolat i kteroukoli standardní funkci. V takovém případě se její návratová hodnota ignoruje.

Například standardní funkci **Read** lze volat jako proceduru:

```
Read(FIL, I);
```

Příkaz HALT

Příkaz **HALT** slouží k okamžitému ukončení běhu programu. Používá se zejména po zjištění chyb, které neumožňují pokračovat v běhu programu.

Například:

```
S:="Nazdar";
j:=Str2Int(S);
IF  j = NONEINTEGER  THEN
  BEGIN
    Info_box('Chyba', 'Chybný zápis čísla');
    HALT
  END;
```

Vstup a výstup

Veškerý vstup a výstup dat do a z programu probíhá přes uživatelské rozhraní Windows.

Vnitřní programovací jazyk není orientován na řízení pomocí vstupujících řádek textu a výstup řádek textu na obrazovku. Procedury **Read**, **Write** a **Writeln** jsou určeny pouze pro práci s diskovými soubory.

Potřebuje-li program získat nějaká data od uživatele nebo chce-li uživateli sdělit určité výsledky, pak k tomu použije buď funkci **Input_box** nebo **Input_box_msg**. Pro sdělení stručné zprávy lze využít proceduru **Info_box** nebo funkci **Wait_box_show**. Pro volbu ze dvou možností funkci **Yesno_box.**

Vytváření uživatelských dialogů

Uživatel může sám vytvořit dialogy, definovat jejich ovládací prvky:

- **DialogCreate** (`function DialogCreate`) – funkce typu **DlgBtnOk** – vytváří dialogový rámeček.
- **DialogRun** (`function DialogRun`) – funkce typu **InputLineGetVal** – spustí dialog s nastavením ovládacích prvků.
- **DialogDestroy** (`function DialogDestroy`) – ukončí zobrazení dialogového rámečku.

Hlavní odlišnosti od jazyka Pascal

Přehled hlavních konstrukcí standardního Pascalu, které v interním jazyku 602Text nejsou implementovány (standardem se rozumí norma ISO 7185 úroveň 0).

- Definice a použití návěští
- typy interval, množina, výčtové typy, variantní typy záznam
- jiné než textové soubory
- typy pole indexované jinak než celými čísly
- dynamicky alokované proměnné
- předsunuté (forward) deklarace procedur a funkcí
- vnořování procedur a funkcí

- parametry typu procedura nebo funkce
- přístup ke složkám proměnné typu záznam pomocí příkazu **WITH.**

Přehled nových syntaktických konstrukcí jazyka 602Text:

- konstantní parametry procedur a funkcí
- implicitní parametry standardních procedur a funkcí
- symboly .=, .=. a ~ vyjadřující relace mezi řetězci.

Přehled procedur a funkcí

Přehled procedur a funkcí můžete využívat několika způsoby:

- Jako tématicky řazený výčet; po skupinách podle tématického zaměření. V jednotlivých skupinách jsou funkce řazeny abecedně a doplněny stručnou charakteristikou. Při volbě jiného způsobu zobrazení vyvoláte tento výčet stiskem tlačítka se symbolem sumy.
- Jako abecedně řazený výčet ve skupinách po položkách začínajících stejným písmenem. Ve skupině jsou položky opět tříděny abecedně a doplněny charakteristikou. Stiskem tlačítka s písmenem vyberte odpovídající skupinu názvů funkcí.
- Jako abecedně řazený výčet bez rozdělení do podskupin – všechny funkce od A do Z. Při volbě jiného způsobu zobrazení vyvoláte tento výčet stiskem tlačítka se symbolem hvězdičky.
- Vedle přehledu procedur a funkcí makrojazyka máte ještě k dispozici přehled standardních konstant.

Obecné poznámky

Mezi standardní procedury a funkce vnitřního jazyka spadá jak množina funkcí definovaných normou jazyka Pascal, tak i mnoho nově přidaných. Pro jejich použití je nutno znát některá specifika, kterými se liší od procedur/funkcí definovaných uživatelem v programu.

Standardní procedury/funkce využívají některá rozšíření syntaxe vůči standardnímu Pascalu.

Parametry typu string

Pouze u standardních procedur/funkcí může v jejich deklaraci formální parametr typu *string* předávaný odkazem vystupovat bez deklarované délky. (Ve skutečnosti jsou parametry typu *string* předávány standardním procedurám a funkcím zásadně tímto způsobem). Jako skutečné parametry je pak možno při volání použít přístup k proměnným typu *string* (event. výsledky funkcí vracejících textové řetězce) různých deklarovaných délek.

Poznámky z hlediska použití

Z hlediska sémantiky (významu) je standardní procedury/funkce možno rozdělit do několika skupin:

- aritmetické
- provádějící konverze hodnot různých typů
- pro práci se soubory
- pro práci s otevřenými dokumenty, atd.

Pro některé z nich je důležitý pojem tzv. aktuálního dokumentu.

Aktuální dokument

Na začátku běhu makra může být otevřen jeden nebo více dokumentů, z nichž pak jeden je tzv. aktuální (pracovní), anebo nemusí být otevřen žádný dokument. Tento počáteční stav se za běhu makra může měnit voláním standardních funkcí, které například z aktuálního dokumentu přepnou do jiného otevřeného, případně mohou otevírat existující dokumenty, zakládat nové, zavírat okno s dokumentem apod.

Mnoho standardních funkcí (např. aritmetických) s aktuálním dokumentem nijak nepracuje a pro makra volající pouze tyto funkce není aktuální dokument důležitý. Pro funkce pracující s aktuálním dokumentem je ovšem nutné, aby tento vůbec existoval; v opačném případě (kdy není žádný dokument otevřen) dojde při jejich volání za běhu makra k chybě – běh makra bude pak předčasně ukončen. Některé z nich požadují pro stav aktuálního dokumentu ještě další omezení; například nelze volat funkci pro pojmenování označeného bloku textu, je-li aktuální dokument v objektovém režimu. Taková situace způsobí též chybu za běhu makra.

V popisu jednotlivých procedur/funkcí jsou takováto omezení vždy vyznačena.

Mnoho procedur/funkcí pracujících s aktuálním dokumentem má některé společné vlastnosti.

Pro větší přehlednost (a z důvodu šetření místem) jsou uvedeny zde. V popisu příslušných procedur/funkcí se na ně budeme pouze odvolávat (viz pozn....).

Poznámky

Poznámka 1– Boolovské funkce zjišťující některé vlastnosti písma (zda je tučné, kurzívou, apod.) vracejí, není-li označen blok, hodnotu dle textu na pozici řádkového kurzoru. Je-li označen blok, vracejí TRUE v tom případě, když zjišťovanou vlastnost má celý označený text.

Poznámka 2 – Funkce, které zjišťují některé vlastnosti písma a jejichž návratová hodnota je typu s*hort*, vracejí platnou hodnotu buď není-li označen blok (pak se tato hodnota týká textu na pozici řádkového kurzoru) anebo má-li v celém označeném bloku zmíněná vlastnost (např. velikost písma) stejnou hodnotu. V opačném případě (je označen blok a různé části označeného bloku jsou vůči této vlastnosti různé) vrací funkce hodnotu *kAmbiguous*.

Poznámka 3 – Funkce nastavující vlastnosti písma (velikost, horní/dolní index apod.) ji nastavují buď v označeném bloku anebo není-li blok označen, bude nastavená vlastnost platit pro následující vkládaný text.

Poznámka 4 – Boolovské funkce zjišťující některou vlastnost odstavce (např. zarovnávání) vracejí (není-li označen žádný blok) hodnotu podle odstavce, ve kterém je řádkový kurzor. Je-li označen blok vracejí TRUE v případě, že inkriminovanou vlastnost mají všechny odstavce označeného textu.

Poznámka 5 – Funkce, které zjišťují některé vlastnosti odstavce a jejichž návratová hodnota je typu s*hort*, vracejí platnou hodnotu buď tehdy, není-li označen blok (pak se tato hodnota týká odstavce příslušejícího pozici řádkového kurzoru) anebo má-li pro všechny odstavce v označeném bloku zmíněná vlastnost stejnou hodnotu. V opačném případě vrací funkce hodnotu *kAmbiguous*.

Poznámka 6 – Funkce nastavující vlastnost odstavce (například zarovnání doleva) ji nastavují buď pro aktuální odstavec (není-li označen blok) anebo pro všechny odstavce označeného bloku.

Tématický přehled procedur a funkcí

Upozornění – v následujícím přehledu se některé procedury a funkce vyskytují i ve více skupinách.

- Aritmetické funkce
- Databázové operace
- Editace
- HTML
- Kapitoly, záhlaví, zápatí
- Kontrola pravopisu
- Kurzor – pozice a označení textu
- Makra
- Objekty
- Odvolání akce
- Okraje
- Pole
- Předvolby
- Rolování obsahu okna
- Stav menu
- Stránky, formát stránek
- Svislé pravítko
- Text, vlastnosti textu
- Uživatelské ovládací prvky
- Záložky, pojmenované bloky
- Zobrazení
- Čas
- Dokument, práce s dokumentem
- Elektronická pošta a faxování
- Informace o aplikaci
- Komunikace
- Konverze
- Lišty
- Mód zobrazení (stránky, osnova)
- Odstavce a sekce
- Okna s dokumenty
- Ostatní procedury a funkce
- Pravítko s tabelátory
- Rejstřík, obsah
- Soubory
- Stavový řádek
- Styly
- Synonyma
- Tisk
- Vyhledání a záměna
- Znakové řetězce
- Zvětšení

Aritmetické funkce

Název	Význam – činnost
Abs	absolutní hodnota reálného čísla
Arctan	funkce arkus tangens
Cos	funkce kosinus
Exp	exponenciála (přirozená)
Iabs	absolutní hodnota celého čísla
Isqr	druhá mocnina celého čísla
Ln	přirozený logaritmus
Odd	zjištění parity čísla
Sgn	znaménko reálného čísla
Sin	funkce sinus
Sqr	druhá mocnina reálného čísla
Sqrt	druhá odmocnina

Čas

Název	Význam – činnost
Now	běžný čas
Today	dnešní datum
Wait	čekání po zadaný počet sekund

Databázové operace

Název	Význam – činnost
AddToDbRecord	přidání položky do záznamu databáze
DisconnectMM	odpojí připojenou databázi
GetTextMM	vrací obsah pole pro slučování v záznamu databáze
GetTextMMI	vrací obsah pole pro slučování v záznamu databáze
IsDbForSave	napojení databáze pro zápis
MailMergeType	vrací typ databáze nastavené pro slučování
NextMMField	označí nejbližší následující pole pro slučování za kurzorem
RecordsCount	vrací počet záznamů v nastavené databázi pro slučování
RecordFirst	nastaví první záznam jako aktuální pro slučování s databází

RecordLast	nastaví poslední záznam jako aktuální pro slučování s databází
RecordNext	nastaví následující záznam jako aktuální pro slučování s databází
RecordPrev	nastaví předcházející záznam jako aktuální pro slučování s databází
RecordCurrent	vrací číslo aktuálního záznamu
RecordSet	nastavuje zadaný záznam jako aktuální pro slučování s databází
SetDbForSave	nastaví připojenou databázi pro zápis
SetMM	otevře dialog pro nastavení databáze
SaveDbRecord	zapíše záznam do přiřazené databáze

Dokument, práce s dokumentem

Název	Význam – činnost
DocClose	zavření dokumentu
GetDocFileName	jméno diskového souboru pro dokument
GetTotSection	vrací počet sekcí v dokumentu
GetLastCode	vrací poslední typ kódování použitý při ukládání/otevírání souboru
GetTmplFileName	vrací jméno otevřené šablony
GoToChapter	jde na příslušnou kapitolu
GoToSection	jde na příslušnou sekci
IsDocDefault	zjištění, zda dokument nebyl nikdy uložen
IsMacro	zjištění, zda platí režim editace makra
IsModified	zjištění, zda byl dokument změněn od posledního uložení
IsReadOnly	zjištění, zda je aktuální dokument otevřen pouze pro čtení
MarkModified	nastavení indikátoru změny dokumentu
MarkUnmodified	vymazání indikátoru změny dokumentu
NewFile	vytvoření nového dokumentu
OpenFile	otevření existujícího dokumentu
RecentFileName	zjištění jména některého ze zapamatovaných naposledy otevřených dokumentů
RecentFilesCount	počet zapamatovaných naposledy otevřených dokumentů
SaveFile	uložení dokumentu
SaveFileAs	uložení dokumentu v požadovaném formátu
SummaryDlg	dialog pro popis dokumentu
SummaryGetStr	vrací některou z položek popisu aktuálního dokumentu
SummarySetStr	mění některou z položek popisu aktuálního dokumentu

Elektronická pošta a faxování

Název	Význam – činnost
GetPrefFax	vrací jméno položky databáze, pro výběr faxového číslo adresáta
GetPrefFaxName	vrací jméno položky databáze, pro výběr jména adresáta
SetPrefFax	nastaví jméno položky databáze, pro výběr faxového čísla
SetPrefFaxName	nastaví jméno položky databáze, pro výběr jména adresáta

Editace

Název	Význam – činnost
BackspaceDelete	mazání znaku před řádkovým kurzorem
ClipCopy	kopírování do schránky
ClipCut	vystřižení do schránky
ClipPaste	vložení obsahu schránky
DeleteBlock	mazání označeného bloku textu
DeleteChar	mazání znaku za řádkovým kurzorem
DeleteWord	mazání textu až k začátku dalšího slova
HyphenDlg	dialog pro editaci měkkých dělítek
InsertDateTime	vložení aktuálního data a času
InsertFileObject	vloží obrázek načtený ze souboru
InsertFootNote	vložení poznámky pod čarou
InsertHardHyphen	vložení tvrdého dělítka
InsertHardSpace	vložení tvrdé mezery
InsertChar	vložení znaku
InsertNewChapter	vložení konce kapitoly
InsertNewPage	vložení konce stránky
InsertNewPara	vložení konce odstavce
InsertNewSection	vloží konec sekce na pozici kurzoru
InsertOptHyphen	vložení měkkého dělítka
InsertTab	vložení tabelátoru
InsertText	vložení řetězce znaků
InsertTextStr	vkládá na aktuální řádkovou pozici řetězec text
IsInsertMode	zjištění, zda je aktivní režim vkládání
SetInsertMode	nastavení režimu vkládání
SetOverwriteMode	nastavení režimu přepisu
ToggleInsert	přepnutí režimu vkládání/přepisu

HTML

Název	Význam – činnost
FieldHTML	vkládá příslušný typ pole
GetObjectHTMLStr	vrací specielní HTML atribut daného objektu
SetObjectHTMLStr	přiřadí objektu specielní HTML atribut

Informace o aplikaci

Název	Význam – činnost
ExeFileName	vrací jméno souboru spuštěné aplikace vč. cesty
GetCommandLine	vrací příkazovou řádku
GetUserName	jméno uživatele zadané při instalaci

Kapitoly, záhlaví, zápatí

Název	Význam – činnost
CreateFooter	vytvoření zápatí
CreateHeader	vytvoření záhlaví
GetTotChapters	vrací počet kapitol
ChapterHasFooter	zjištění existence zápatí
ChapterHasHeader	zjištění existence záhlaví
ChapterNumber	nastavení číslování kapitol, stránek

Komunikace

Název	Význam – činnost
Beep	krátký zvukový signál
Info_box	informační dialogový rámeček
Input_box	dialog pro vložení textového řetězce
Input_box_msg	dialog pro vložení textového řetězce
Input_box_msg_2	dialog pro vložení dvou textových řetězců
Input_box_msg_3	dialog pro vložení tří textových řetězců
Input_box_msg_4	dialog pro vložení čtyř textových řetězců
Wait_box_hide	zavření dialogu
Wait_box_show	zobrazení dialogu bez přerušení běhu makra
Yesno_box	dialogový rámeček (ano – ne)

Kontrola pravopisu

Název	Význam – činnost
SpellEnabled	přístupnost příkazu **Překlepy**
SpellRun	spuštění kontroly pravopisu

Konverze

Název	Význam – činnost
Bool2str	konverze boolovské hodnoty na řetězec znaků
Date2str	konverze data na řetězec znaků
Day	extrakce dne s data
Day_of_week	získání dne v týdnu z data
Hours	extrakce hodin z časového údaje
Char2str	konverze znaku na řetězec znaků
Chr	konverze celých čísel na znaky
Int2str	konverze celého čísla na řetězec znaků
Make_date	vytvoření data ze dne, měsíce, roku
Make_time	vytvoření časového údaje
Minutes	extrakce minut z časového údaje
Money2str	konverze částky peněz na řetězec znaků
Month	extrakce měsíce z data
Ord	konverze znaků na celá čísla
Real2str	konverze reálného čísla na řetězec znaků
Round	zaokrouhlení reálného čísla na celé číslo
Sec1000	extrakce tisícin sekundy z časového údaje
Seconds	extrakce sekund z časového údaje
Str2date	konverze řetězce znaků na datum
Str2int	konverze řetězce znaků na celé číslo
Str2money	konverze řetězce znaků na částku peněz
Str2real	konverze řetězce znaků na reálné číslo
Str2time	konverze řetězce znaků na údaj o čase
Time2str	konverze údaje o čase na řetězec znaků
Trunc	seříznutí reálného čísla na celé číslo
Year	extrakce roku z data

Kurzor – pozice a označení textu

Název	Význam – činnost
AtDocEnd	zjištění, zda se řádkový kurzor nachází na konci aktuálního dokumentu
AtDocStart	zjištění, zda se řádkový kurzor nachází na začátku aktuálního dokumentu
AtObjectEnd	zjištění, zda se řádkový kurzor nachází na konci objektu
AtObjectStart	zjištění, zda se řádkový kurzor nachází na začátku objektu
BottomOfScreen	přesun na konec viditelného obsahu okna
CaretCellStart	umístění kurzoru v buňce (ve sloupci i řádku) textové tabulky
CaretEnd	přemístění na konec dokumentu
CaretLeft	označení textu při pohybu (šipka vlevo)
CaretHome	přemístění na začátek dokumentu
CaretObjectEnd	přesune řádkový kurzor (je-li objekt textový) na jeho konec
CaretObjectStart	přesune řádkový kurzor (je-li objekt textový) na jeho začátek
CaretRight	označení textu při pohybu (šipka vpravo)
GetCaretCell	nové číslo sloupce v textové tabulce
GetCaretLine	číslo řádky umístění řádkového kurzoru
GetCaretObjectId	vrací identifikátor objektu (je-li řádkový kurzor v objektu)
GetCaretPage	číslo stránky umístění řádkového kurzoru
GetCaretPos	celé číslo udávající pozici řádkového kurzoru
GetCaretPosType	druh umístění řádkového kurzoru
GetCaretRow	vrací číslo řádku (pod kurzorem) v textové tabulce
GetCaretSection	vrací číslo sekce, ve které se nachází řádkový kurzor
GetNumCells	vrací počet sloupců v řádku textové tabulky
GetNumRows	vrací počet řádků textové tabulky
GetSelEndLine	číslo řádky konce označeného bloku
GetSelEndPage	číslo stránky konce označeného bloku
GetSelEndPos	pozice konce označeného bloku
GetSelStartLine	číslo řádky začátku označeného bloku
GetselStartPage	číslo stránky začátku označeného bloku
GetSelStartPos	pozice začátku označeného bloku
GetTextMM	vrací obsah pole pro slučování v záznamu databáze
GetTextMMI	vrací obsah pole pro slučování v záznamu databáze
ChangeSelRanges	záměna pevného a pohyblivého konce označeného bloku
CharLeft	posun o znak doleva
CharRight	posun o znak doprava
IsBlockSelected	zjištění, zda je označen nějaký blok textu

LeftOfLine	přemístění na začátek řádky
LineDown	přemístění na následující řádek
LineUp	přemístění na předcházející řádek
NextCell	umístí kurzor do následující buňky v textové tabulce
NextMMField	označí nejbližší následující pole pro slučování za kurzorem
NextRow	umístí kurzor do následujícího řádku v textové tabulce
PrewCell	umístí kurzor do předchozí buňky v textové tabulce
PrewRow	umístí kurzor do předchozího řádku v textové tabulce
PageDown	přesun dolů o obsah okna
PageUp	přesun nahoru o obsah okna
PrevPara	přesun na začátek předchozího odstavce nebo opačně: na začátek odstavce ve kterém se nachází
RightOfLine	přemístění na konec řádky
RightOfWord	přemístění na konec slova
TopOfScreen	přesun na začátek viditelného obsahu okna
UnselectBlock	zrušení označení bloku textu
WordLeft	přemístění na předchozí slovo
WordRight	přemístění na následující slovo

Lišty

Název	Význam – činnost
IsToolbar	zjištění, zda je lišta zobrazena
SetToolbar	zobrazení/skrytí lišty

Makra

Název	Význam – činnost
CountMacros	vrací počet právě spustitelných maker
CurrentMacroName	vrací jméno právě běžícího makra
DisableAutoRun	znepřístupní automatická makra
ErasethisMacro	vypuštění makra ze seznamu spustitelných maker
IsMacro	zjištění, zda platí režim editace makra
MacroSubrListDlg	dialog pro spustitelná makra
MacroAutoStarted	zjištění, zda bylo běžící makro spuštěno automaticky
MacroDelete	vymaže makro ze seznamu spustitelných maker
MacroDescription	zjištění popisu makra v aktuálním dokumentu

MacroIsInMenu	zjištění, zda makro je/není zařazeno do menu
MacroIsLocal	zjištění, je-li běžící makro lokální v dokumentu nebo globální
MacroName	vrací jméno makra
MacroNameToIndex	zjištění indexu makra
MacroSetDescr	nastavení nového popisu makra
MacroSetToMenu	zařazení makra do menu
MacroToGlobal	zajistí změnu lokálníhomakra na globální
MacroToLocal	zajistí změnu globálníhomakra na lokální
SetIsMacro	nastavení režimu editace makra

Mód zobrazení (stránky, osnova)

Název	Význam – činnost
DisplayOutline	nastavení režimu osnovy
DisplayPages	nastavení režimu zobrazení stránek
GetOutlineLevel	zjištění úrovně osnovy
PagesDisplayed	zjištění, je-li nastaven režim zobrazení stránek
SetOutlineLevel	nastavení úrovně osnovy

Objekty

Název	Význam – činnost
DoesObjectExist	zjištění existence objektu
DeleteObject	v aktuálním dokumentu ruší (maže) objekt
DeleteObjGroup	zruší všechny objekty v seskupení
ObjectId	pro objekt zadaného indexu vrací jeho identifikátor
ObjectIsSelected	zjištění, zda je objekt vybrán (označen)
ObjectListDlg	dialog pro seznam objektů
ObjectPropDlg	určí pořadí karty při inicializaci dialogu pro aktivní objekt
ObjectsCount	vrací počet všech objektů v aktuálním dokumentu
ObjectSelCount	vrací počet vybraných objektů
ObjectSelId	vrací číslo index-tého vybraného objektu
ObjectSelModify	přidává/ubírá objekt do vícenásobné selekce
ObjectType	zjištění typu objektu zadaného identifikátorem
SelectObject	výběr objektu zadaného identifikátorem
UnselectObject	zruší výběr (označení) objektu

Odstavce a sekce

Název	Význam — činnost
FormatSectDlg	vyvolá dialog pro nastavení sekcí
GetFormatPara	zjišťuje vlastnosti vybraných odstavců textu
GetFormatSect	zjišťuje vlastnosti vybraných sekcí textu
SetFormatPara	nastaví vlastnosti pro vybrané odstavce textu
SetFormatSect	nastaví vlastnosti pro vybrané sekce textu

Odvolání akce

Název	Význam – činnost
CanUndo	zjištění, zda je přístupný příkaz **Odvolat**
Undo	příkaz **Odvolat**

Okna s dokumenty

Název	Význam – činnost
ArrangeIcons	uspořádání ikon minimalizovaných dokumentů
CountWindows	počet oken
DocClose	zavření dokumentu
DocMaximized	maximalizování okna dokumentu
DocMinimize	minimalizování okna dokumentu
DocRestore	obnova původní velikosti okna
DocSplit	nastavení rozdělení okna
DocWndMove	přemístění okna
DoesWindowExist	zjištění, zda existuje okno se zadaným řetězcem v záhlaví
GetDocWndHeight	vertikální rozměr okna
GetDocWndLeft	horizontální souřadnice levého horního rohu okna
GetDocWndTittle	textový řetězec v záhlaví okna
GetDocWndTop	vertikální souřadnice levého horního rohu okna
GetDocWndWidth	horizontální rozměr okna
GetSplitType	zjištění druhu rozdělení okna
GetSplitVal	zjištění poměru rozdělení okna
IsAnyDocOpened	zjištění, zda je vůbec nějaké okno otevřeno
IsDocMaximized	zjištění, zda je okno maximalizováno
IsDocMinimized	zjištění, zda je okno minimalizováno

IsDocumentOpened	zjištění, zda existuje okno zadaného dokumentu
NextWindow	přepnutí do dalšího okna
PreviousWindow	přepnutí do předchozího okna
SetDocWndHeight	nastavení vertikálního rozměru okna
SetDocWndWidth	nastavení horizontálního rozměru okna
SwitchTo Window	přepnutí do okna zadaného řetězce v záhlaví
SwitchToDoc	přepnutí do okna zadaného dokumentu
WindowsCascade	uspořádání oken do kaskády
WindowsTile	uspořádání oken do mozaiky

Okraje

Název	Význam – činnost
GetBottomMargin	velikost spodního okraje
GetLeftMargin	velikost levého okraje
GetRightMargin	velikost pravého okraje
GetTopMargin	velikost horního okraje
SetBottomMargin	nastavení spodního okraje
SetLeftMargin	nastavení levého okraje
SetMargins	nastavení okrajů
SetRightMargin	nastavení pravého okraje
SetTopMargin	nastavení horního okraje

Ostatní procedury a funkce

Název	Význam – činnost
Addr	vrací adresu argumentu
Dec	zmenší hodnotu proměnné
Exec	spuštění jiné aplikace
Inc	zvětší hodnotu proměnné
Memcpy	kopírování části paměti
NullCmd	žádný příkaz
Pred	předchůdce zadané ordinální hodnoty
SetInitialOpen	nastavení interního indikátoru pro vytváření prázdného dokumentu při startu
sizeof	pro proměnnou libovolného typu vrací její velikost v bajtech
Succ	následník zadané ordinální hodnoty

StrIsRC	zjištění, je-li řetězec rodné číslo

Pole

Název	Význam – činnost
FieldInsertDlg	volá dialog **Vložit pole**
FieldSimple	vkládá do aktuálního dokumentu jednoduché pole
FieldSuma	vkládá do pole **Součet** (ve sloupci i řádce) textové tabulky
ViewFieldCnts	nastaví/potlačí zobrazení obsahu polí

Pravítko s tabelátory

Název	Význam – činnost
IsTabRulerOn	zjištění, zda je pravítko s tabelátory zobrazeno
ViewTabRuler	zobrazení/skrytí pravítka s tabelátory

Předvolby

Název	Význam – činnost
DefaultDocDir	implicitní adresář pro otvírání dokumentů
DefaultTmplDir	implicitní adresář pro otvírání šablon
GetNewWinState	vrací stav tlačítka pro nové okno
GetPreferences	zjištění specifikované položky předvoleb
GetPrefFootnStr	zjištění implicitní značky poznámky pod čarou
PreferencesDlg	dialog pro předvolby
SetNewWinState	nastavuje stav tlačítka pro nové okno
SetPreferences	nastavení specifikované položky předvoleb
SetPrefFootnStr	nastavení implicitní značky poznámky pod čarou

Rejstřík, obsah

Název	Význam – činnost
AddIndexEntry	přidání položky rejstříku
AddIndexEntryDlg	dialog pro přidání položky rejstříku
ContentsEnabled	přístupnost příkazu **Obsah**
ContentsExist	zjištění, zda byl již vytvořen obsah
CountOfEntryRef	počet odkazů na položku rejstříku

CreateContents	vytvoření obsahu
CreateIndex	vytvoření rejstříku
GoToIndexEntry	skok na položku rejstříku
IndexEditDlg	dialog pro editaci položek rejstříku
IndexEditEnabled	přístupnost příkazu **Opravit rejstřík**
IndexEntries	počet položek rejstříku
IndexEntry	vrací položku rejstříku daného indexu
IndexEntryInd	pro položku rejstříku zjišťuje index
IndexExists	zjištění, zda byl již vytvořen rejstřík
IndexGenEnabled	přístupnost příkazu **Vytvořit rejstřík**
IndexMarkEnabled	přístupnost příkazu **Označit položku rejstříku**
RemoveIndexEntry	zrušení položky rejstříku
SelectContents	označení vytvořeného obsahu
SelectIndex	označení vytvořeného rejstříku jako bloku

Rolování obsahu okna

Název	Význam – činnost
FixScreenPos	odrolování k pozici řádkového kurzoru
ScrollLineDown	odrolování o jednu nebo více řádek dolů
ScrollLineUp	odrolování o jednu nebo více řádek nahoru
ScrollPageDown	odrolování o obsah okna dolů
ScrollPageUp	odrolování o obsah okna nahoru

Soubory

Název	Význam – činnost
Close	uzavření souboru
Delete_file	smazání souboru z disku
Eof	zjištění konce souboru
Filelength	zjištění délky souboru
FindClose	ukončí vyhledávání spuštěné funkcí **FindFirstFile**
FindFirstFile	najde první soubor
FindNextFile	najde další soubor
Make_directory	vytvoření adresáře na disku
Read	čtení z textového souboru
Reset	otevření existujícího souboru

Rewrite	vytvoření a otevření nového souboru
Seek	nastavení pozice v souboru
Write	zápis do textového souboru
Writeln	zápis do textového souboru a odřádkování

Stav menu

Název	Význam – činnost
ContentsEnabled	přístupnost příkazu **Obsah**
DatabaseEnabled	přístupnost příkazu **Nastavení databáze**
HyphenDlgEnabled	přístupnost příkazu **Měkká dělítka**
IndexEditEnabled	přístupnost příkazu **Opravit rejstřík**
IndexGenEnabled	přístupnost příkazu **Vytvořit rejstřík**
IndexMarkEnabled	přístupnost příkazu **Označit položku rejstříku**
IsWordOverFlow	zjištění, zda je nastaven přepínač **Přetékání zvolených slov**
LangSetupEnabled	přístupnost příkazu **Volba jazyka**
SetWordOverFlow	nastavení přepínače **Přetékání zvolených slov**
SpellEnabled	přístupnost příkazu **Překlepy**
ThesaurusEnabled	přístupnost příkazu **Synonyma**
TranslateEnabled	přístupnost příkazu **Překlad slova**

Stavový řádek

Název	Význam – činnost
IsStatusStrip	zjištění, zda je stavový řádek zobrazen
SetStatusStrip	zobrazení/skrytí stavového řádku
SetStatusText	vypsání textu na stavový řádek

Stránky, formát stránek

Název	Význam – činnost
GetPageHeight	výška stránky
GetPageType	formát stránky
GetPageWidth	šířka stránky
GetTotPages	celkový počet stran
GoToPage	přesun na zadanou stranu
IsPageLandscape	zjištění, zda je orientace stránky na šířku

IsPagePortrait	zjištění, zda je orientace stránky na výšku
PageFormat	nastavení formátu stránky

Styly

Název	Význam – činnost
DefParaStyleDlg	dialog pro definici stylu odstavce
GetStyleCount	počet stylů
GetStyleName	jméno daného stylu
StyleIndex	návratová hodnota je číslo stylu, který je označen jménem
StyleIsBased	návratová hodnota je číslo stylu, z něhož je styl daný parametrem odvozen
StyleUpdate	udržuje a mění jednotný styl dokumentů
UseParaStyleNum	použití stylu odstavce

Svislé pravítko

Název	Význam – činnost
IsVertRulerOn	zjištění, zda je svislé pravítko zobrazeno
ViewVertRuler	zobrazení/skrytí svislého pravítka

Synonyma

Název	Význam – činnost
ThesaurusDlg	dialog pro synonyma
ThesaurusEnabled	přístupnost příkazu **Synonyma**

Text, vlastnosti textu

Název	Význam – činnost
BalanceColumns	vyrovnat délky sloupců
ColorDlg	dialog pro výběr barvy
CountFonts	vrací počet aktuálně dostupných druhů písma
CurrentFont	vrací název fontu, kterým je zapsán text na pozici řádkového kurzoru
FontName	vrací název jednoho z aktuálně dostupných fontů
FormatFontDlg	dialog pro nastavení vlastností písma
FormatParaDlg	dialog pro nastavení vlastností odstavce

GetAlignType	zjištění druhu zarovnání
GetFormatFont	zjištění zadané vlastnosti textu
GetFormatPara	zjišťuje vlastnost odstavce, resp. vybraných odstavců textu
GetRGBText	zjištění barvy textu/objektu na pozici řádkového kurzoru
GetText	vrací zadanou část textu jako řetězec znaků
GetUnderlineType	zjištění druhu podtržení
IsAlignCenter	zjištění, zda je zarovnání centrované
IsAlignJustify	zjištění, zda je zarovnání oboustranné
IsAlignLeft	zjištění, zda je zarovnání nalevo
IsAlignRight	zjištění, zda je zarovnání napravo
IsBold	zjištění, zda je text tučný
IsBrushOn	zjištění, je-li aktivován štěteček
IsItalic	zjištění, zda je text kurzívou
IsTextDefault	zjištění, zda je text daný stylem odstavce
IsTextRegular	zjištění, zda je text obyčejný
RGB	vytváří podle intenzity zadaných barevných složek jako návratovou hodnotu jejich barevnou kombinaci
RgbTrio	rozkládá zadanou hodnotu barevné kombinace na jednotlivé barevné složky
SetAlign	nastavení zadaného druhu zarovnání
SetAlignCenter	nastavení centrovaného zarovnání
SetAlignJustify	nastavení oboustranného zarovnání
SetAlignLeft	nastavení zarovnání nalevo
SetAlignRight	nastavení zarovnání napravo
SetBold	nastavení tučného písma
SetBrush	aktivuje/deaktivuje štěteček
SetFormatFont	nastavení zadané vlastnosti textu
SetFormatPara	zajišťuje nastavení vlastností odstavce, resp. vybraných odstavců textu
SetItalic	nastavení kurzívy
SetRGBColor	nastavení barvy
SetRGBText	nastavení barvy písma
SetTextDefault	nastavení textu podle stylu odstavce
SetTextRegular	nastavení obyčejného textu
SetUnderlineType	nastavení druhu podtržení
UseFont	nastaví typ písma pro označený blok textu/následující vkládaný text
UseFontOfName	nastaví písmo identifikované svým jménem

Tisk

Název	Význam – činnost
Print	tisk dokumentu
PrnAdvancedDlg	zobrazí dialog pro specifická nastavení tiskárny
PrnGetCurrent	řetězec aktuálně nastavený pro zvolený list
PrnIsPortrait	zjištění připravenosti tiskárny k tisku
PrnListCount	řetězec aktuálně nastavený pro zvolený seznam
PrnListStr	řetězec odpovídající indexu ve zvoleném seznamu vlastností
PrnSetCurrent	nastavuje parametry tiskárny
PrnSetPortrait	nastavuje tiskárnu pro tisk
PrnSetupDlg	zobrazí dialog pro základní nastavení tiskárny

Uživatelské ovládací prvky

Název	Význam – činnost
DialogCreate	vytvoří prázdný dialog
DialogCreateExt	vytvoří prázdný dialog s možností volby písma
DialogDestroy	zruší dialog
DialogFontSize	vrací velikost písma v dialogu
DialogFontReSize	umožňuje změnu velikost písma v dialogu
DialogRun	spustí dialog
DialogSetBmp	nastaví bitmapu coby pozadí uživatelského dialogu
DialogSetSysMenu	zobrazení/potlačení systémového tlačítka
DlgBtnCancel	vytvoří v dialogu tlačítko Zrušit
DlgBtnNo	vytvoří v dialogu tlačítko NO
DlgBtnOk	vytvoří v dialogu tlačítko OK
DlgBtnYes	vytvoří v dialogu tlačítko Ano
DlgButton	vytvoří nadepsané tlačítko v dialogu
DlgButtonDefPush	nastaví tlačítko, které bude reagovat na klávesu Enter
DlgComboBox	vloží do uživatelského dialogu seznam, do něhož je možno vpisovat
DlgEditBox	vloží do uživatelského dalogu pole pro víceřádkový vstup
DlgGroupBox	vloží rámeček sekce do uživatelského dialogu
DlgCheckBox	vloží zaškrtávací přepínač do uživatelského dialogu
DlgInputLine	vytvoří vstupní řádek v uživatelském dialogu
DlgLineX	vloží vodorovnou čáru do uživatelského dialogu
DlgLineY	vloží svislou čáru do uživatelského dialogu

DlgListBox	vloží do uživatelského dialogu seznam, který slouží pouze pro výběr
DlgRadioBtn	vytvoří nový vícepolohový přepínač se zadaným textem, polohou a rozměry a přidává jej do skupiny
DlgRadioBtnGroup	vytváří v uživatelském dialogu "skrytý" řídící prvek, představující skupinu přepínačů
DlgRemoveCtrl	odstraní zadaný ovládací prvek z dialogu
DlgSetCtrlBmp	umožní zobrazit ovládací prvek bitmap
DlgSetCtrlPos	nastaví u daného řídícího prvku v uživatelském dialogu novou polohu vzhledem k levému hornímu rohu dialogu
DlgSetCtrlSize	nastaví u daného řídícího prvku v uživatelském dialogu jeho nové rozměry
DlgSetCtrlText	nastaví nový text pro daný ovládací prvek
DlgStdBox	do uživatelského dialogu vloží standardní seznam
DlgStrBoxAdd	přidá do seznamu znakových řetězců v listboxu/comboboxu nový řetězec
DlgStrBoxDelete	vypouští ze seznamu znakových řetězců v listboxu/comboboxu) znakový řetězec
DlgStrBoxGetStr	vrací hodnotu znakového řetězce v daném listboxu/comboboxu
DlgStrBoxGetVal	vrací identifikátor znakového řetězce, který byl při opuštění dialogu v příslušném listboxu/comboboxu selektován
DlgStrBoxSetVal	nastaví v seznamu znakových řetězců v listboxu/comboboxu interní indikátor
DlgText	vloží doprovodný text do uživatelského dialogu
DlgTextSetAlign	nastaví zarovnání popisného textu
DlgTextSetBmp	zobrazí bitmapu místo doprovodného textu
CheckBoxGetVal	zjišťuje stav přepínače po ukončení dialogu
ChechBoxSetVal	nastaví stav přepínače před vlastním spuštěním dialogu
InputLineGetVal	zjištění hodnoty řetězce znaků vstupní řádky v uživatelském dialogu
InputLineSetVal	zadá nové hodnoty řetězce znaků na vstupní řádce v uživatelském dialogu
MenuCreate	vytvoří uživatelské menu
MenuDestroy	uvolňuje z paměti interní šablonu uživatelského menu
MenuRun	zobrazí uživatelské menu, jehož šablona byla již dříve vytvořena
MenuStrAdd	do uživatelského menu přidá novou položku, jejímž obsahem je textový řetězec
MenuStrDelete	z uživatelského menu vypouští položku
MenuStrDisable	nastavuje přístupnost/položky uživatelského menu
MenuStrGet	návratovou hodnotou funkce je obsah (textový řetězec) položky uživatelského menu
MenuStrCheck	nastaví zaškrtnutí položky v uživatelském menu
RadioBtnCheck	nastaví/zruší u přepínače stav "zatrženo"
RadioBtnChecked	zjištění, je-li přepínač ve stavu "zatrženo"

RadioGrpGetVal	pro danou skupinu přepínačů vrací identifikátor toho přepínače, který je zatržen
RadioGrpSetVal	nastaví/zruší v zadané skupině přepínačů v zadaném dialogu u přepínače stav "zatrženo"

Vyhledání a záměna

Název	Význam – činnost
Replace	záměna textového řetězce
ReplaceAgain	opakování poslední záměny textového řetězce
ReplaceIsReady	zjištění možnosti záměny textového řetězce
Search	vyhledání textového řetězce
SearchAgain	opakování posledního vyhledání textového řetězce
SearchIsReady	zjištění možnosti vyhledání textového řetězce

Záložky, pojmenované bloky

Název	Význam – činnost
BlockName	vrací jméno index-tého pojmenovaného bloku
BlocksCount	vrací počet pojmenovaných bloků v aktuálním dokumentu
ClearMark	zrušení záložky
CopyBlock	zkopíruje příslušný pojmenovaný blok do schránky
DoesBlockExist	existuje pojmenovaný blok daného jména?
DoesMarkExist	zjištění, zda existuje záložka daného jména
EditBlocksDlg	dialog pro pojmenované bloky
EditMarksDlg	dialog pro editaci záložek
GoToMark	přemístění na záložku
InsertMark	vložení záložky
LabelBlock	pojmenování bloku
MarkName	vrací jméno index-té záložky
MarksCount	vrací počet záložek v aktuálním dokumentu
SelectBlock	označení pojmenovaného bloku
UnlabelBlock	zrušení pojmenování bloku

Znakové řetězce

Název	Význam – činnost
CleanString	odstranění netisknutelných znaků z řetězce

ChangeCase	konverze velkých písmen v řetězci za malá a naopak
Lcase	konverze řetězce na malá písmena
Like	podobnost řetězců
Ltrim	odstranění mezer ze začátku řetězce
Pref	relace prefix mezi řetězci
Rtrim	odstranění mezer z konce řetězce
Strcat	spojení dvou řetězců (zřetězení), ekvivalent +
Strcopy	vybrání podřetězce z řetězce
Strdelete	zrušení podřetězce v řetězci
Strinsert	vložení řetězce dovnitř řetězce
Strlength	zjištění délky řetězce
StrIsRC	zjištění, je-li řetězec rodné číslo
Strpos	hledání podřetězce v řetězci
Strtrim	odstranění mezer ze začátku a konce řetězce
Substr	relace obsažen mezi řetězci
Upcase	konverze řetězce na velká písmena

Zobrazení

Název	Význam – činnost
AreObjectsOn	zjištění, zda jsou zobrazeny objekty
AreScrollBarsOn	zjištění, zda jsou zobrazeny posuvníky
FrameMarginsOn	zjištění, zda jsou zobrazeny okraje textových rámců
IsFieldCntsOn	zjištění, zda je zobrazen obsah polí
IsFormatOn	zjištění, zda jsou zobrazeny skryté znaky
ViewFieldCnts	nastavení zobrazení/potlačení obsahu polí
ViewFormat	nastavení zobrazení/potlačení skrytých znaků
ViewFrameMargins	nastavení zobrazení/potlačení okrajů textových rámců
ViewObjects	nastavení zobrazení/potlačení objektů
ViewScrollBars	nastavení zobrazení/potlačení posuvníků

Zvětšení

Název	Význam – činnost
GetScale	zjištění nastaveného zvětšení v procentech
ManualScaleDlg	dialog pro nastavení zvětšení
SetScale	nastaví zvětšení v procentech

SetScaleFullPage	nastavení zvětšení “pro celou stránku”

Přehled standardních konstant

Konstanty TRUE a FALSE

Konstanty TRUE a FALSE jsou typu ***boolean*.**

Konstanty označující hodnoty NONE

Tyto konstanty označují nedefinované (NONE) hodnoty různých typů.

Konstanta	Typ
NONECHAR	char
NONEBOOLEAN	boolean
NONESHORT	short
NONEINTEGER	integer
NONEREAL	real
NONETIME	time
NONEDATE	date
NONEMONEY	money
NIL	ukazatel

Konstanty označující hodnoty ID...

Tyto konstanty zpravidla označují hodnoty vracené standardními funkcemi, které vyvolávají vždy nebo při jistém nastavení parametrů *dialog*. Návratová hodnota pak charakterizuje způsob opuštění tohoto dialogu; obecněji provedení či neprovedení požadované akce.

Všechny jsou typu s*hort*.

Konstanta	Význam – činnost
IDOK	potvrzení výběru v dialogu **OK**
IDCANCEL	zrušení výběru v dialogu
IDGOTO	provedena akce “jdi na...”
IDINSERT	provedena akce “vložit...”
IDDELETE	provedeno vypuštění
IDERROR	došlo k chybě (nepřípustné hodnoty parametrů, nedostatek paměti apod.)

Konstanty kOn, kOff, kAmbiguous

Tyto konstanty zpravidla označují hodnoty vracené standardními funkcemi, které zjišťují aktuální vlastnosti písma (zda je tučné, kurzívou apod.). Návratová hodnota *kOn* znamená, že zjišťovaná vlastnost je nastavena; hodnota *kOff* znamená, že nastavena není. Hodnota *kAmbiguous* je vrácena, pokud nelze rozhodnout (je označen blok, přičemž jeho část má zjišťovanou vlastnost a část ne).

Všechny jsou typu s*hort*.

Konstanta	Význam
kOn	ano
kOff	ne
kAmbiguous	nelze rozhodnout

Konstanty kCHP...

Tyto konstanty vystupují jako hodnoty toho z parametrů standardních funkcí **GetFormatFont**, **SetFormatFont**, který udává, jaká vlastnost písma se bude zjišťovat, resp. nastavovat.
Všechny jsou typu s*hort*.

Konstanta	Čeho se týká
kCHPsize	velikost písma
kCHPbold	zda je písmo tučné
kCHPitalic	zda je písmo kurzívou
kCHPunderline	druh podtržení písma
kCHPsupersub	normální písmo /horní index /dolní index
kCHPcaps	vlastnost “všechna písmena velká”

Konstanty kPP...

Tyto konstanty vystupují jako hodnoty toho z parametrů standardních funkcí **GetFormatPara**, **SetFormatPara**, který udává, jaká vlastnost odstavce se bude zjišťovat, resp. nastavovat.
Všechny jsou typu *short*.

Konstanta	Význam
kPPAlign	nastaví zarovnání odstavce
kPPBorder	nastaví typ orámování odstavce
kPPBorderOffset	nastaví oddělení (vzdálenost) písma a čar kolem odstavce (v aktuálních jednotkách)
kPPFirstIndent	nastaví odsazení první řádky odstavce
kPPHyphenation	nastaví druh dělení slov
kPPLeading	nastaví vzdálenost řádek v procentech
kPPLeftIndent	nastaví odsazení odstavce nalevo
kPPLevel	nastaví úroveň osnovy
kPPLine	nastaví typ čar kolem odstavce
kPPLowerSpace	nastaví místo za odstavcem
kPPNumber	nastaví druh číslování osnovy
kPPRightIndent	nastaví odsazení odstavce napravo
kPPShade	nastaví stínování odstavce
kPPSkipBullets	nastaví příznak pro použití odrážek
kPPUpperSpace	nastaví místo před odstavcem

Konstanty kSP...

Tyto konstanty vystupují jako hodnoty toho z parametrů standardních funkcí **GetFormatSect**, **SetFormatSect**, který udává, jaká vlastnost odstavce se bude zjišťovat, resp. nastavovat.
Všechny jsou typu *short*.

Konstanta	Význam
kSPColInc	nastaví pøírùstek sloupce
kSPColSpace	nastaví velikost mezery mezi sloupci
kSPMaxCol	nastaví poèet sloupcù
kSPColLine	nastaví typ èáry mezi sloupci
kSPColHeigh	nastaví výšku sloupce

Konstanty tSNormal, tSuper, tSub

Tyto konstanty vystupují jako návratové hodnoty, resp. jako hodnoty jednoho z parametrů standardních funkcí **GetFormatFont**,**SetFormatFont .** Všechny jsou typu s*hort*.

Konstanta	Význam

tSNormal	normální text
tSuper	horní index
tSub	dolní index

Konstanty al...

Tyto konstanty specifikují druh zarovnání; vystupují jako návratové hodnoty, resp. jako hodnoty parametrů standardních funkcí zjišťujících nebo nastavujících zarovnání.

Všechny jsou typu *short*.

Konstanta	Význam
alLeft	zarovnání doleva
alCenter	zarovnání na střed
alRight	zarovnání doprava
alJustify	oboustranné zarovnání

Konstanty cpt...

Tyto konstanty mohou být návratovými hodnotami standardní funkce **GetCaretPosType**, která zjišťuje druh umístění řádkového kurzoru (caretu) v dokumentu.

Všechny jsou typu *short*.

Konstanta	Význam
cptTextFrame	textový rámec
cptFootnote	poznámka pod čarou
cptTable	tabulka
cptNormal	normální text

Konstanty st...

Tyto konstanty vystupují ve standardních funkcích týkajících se druhu rozdělení okna dokumentu.
Všechny jsou typu *short*.

Konstanta	Význam
stNoSplit	nijak nerozděleno
stHorSplit	rozděleno vodorovně
stVertSplit	rozděleno svisle

Konstanty kPrint...

Tyto konstanty slouží pro zadání hodnoty jednoho z parametrů standardní funkce **Print**. Všechny jsou typu *short*.

Konstanta	Význam
PrintAllPages	tisknout všechny stránky
PrintEvenPages	tisknout sudé stránky
PrintOddPages	tisknout liché stránky

Konstanty kPref...

Tyto konstanty vystupují ve standardních funkcích týkajících se nastavení předvoleb.
Všechny jsou typu *short*.

Konstanta	Čeho se týká

kPREF_DEV_FONTS	nabízet písma tiskáren
kPREF_B_HLP	zobrazovat bublinovou nápovědu
kPREF_SUPP_SUMM	nabízet popis při ukládání
kPREF_S_WINDOWS	ukládat rozložení oken
kPREF_TEMPL_NEW	nabízet šablony pro nové dokumenty
kPREF_AUTOSAVE	automatické ukládání
kPREF_AUTO_INT	interval automatického ukládání
kPREF_GROUP_DEL	při odvolání seskupovat mazání textu
kPREF_GROUP_INS	při odvolání seskupovat vkládání textu
kPREF_GROUP_MOVE	při odvolání seskupovat pohyby kurzoru
kPREF_IMG_DITHER	výpočet odstínů šedi pro tisk obrázků
kPREF_GLOB_UNITS	používané délkové jednotky
kPREF_HORIZONTAL	preference vodorovného uspořádání mozaiky

Konstanty kCheck...

Tyto konstanty slouží pro zadání způsobu kontroly dokumentu standardní funkcí **SpellRun**. Všechny jsou typu s*hort*.

Konstanta	Význam
kCheckFromCaret	kontrolovat od pozice řádkového kurzoru
kCheckEntireDoc	kontrolovat celý dokument
kCheckSelection	kontrolovat označený blok textu

Konstanty kSave...

Tyto konstanty použijte při zadání hodnoty parametru specifikujícího činnost zavírání dokumentu standardní funkcí **DocClose**. Všechny jsou typu s*hort*.

Konstanta	Význam
kSaveAlways	vždy ukládat
kSaveNever	nikdy neukládat
kSaveNormal	dotaz na uložení změněného dokumentu

Konstanty ct...

Tyto konstanty specifikují jednotlivé druhy kódování národních znaků. Všechny jsou typu s*hort*.

Konstanta	Význam
ctKEYBCS2	kódování podle bratří Kamenických
ctLATIN2	kódování podle normy Latin 2
ctKOI8CS	kódování podle KOI8–čs (JSEP, SMEP)
ctCP852	oficiální kódová stránka pro východoevropské prostředí (MS DOS od verze 5.0)
ctWINANSI	kódování standardní verze Windows
ctWINEE	kódování východoevropské verze Windows
ctMAC	kódování počítačů Macintosh
ctMAZOVIA	polské kódování Mazovia

Konstanty ft...

Tyto konstanty specifikují jednotlivé typy souborů co do formátu. Všechny jsou typu *integer*.

Interní formáty.....

Identifikátor	Význam
ftT602	Text602

ftWT_10	WinText 1.0
ftWT_20	WinText 2.0
ftWT_21	WinText 2.1
ftWT_30	WinText 3.0
ftWT_30_macro	WinText 3.0
ftWT_30_template	WinText 3.0 – šablona
ftWt_50	WinText 5.0 (a novější: 602Text)
ftWT_50_macro	WinText 5.0 (a novější: 602Text)
ftWT_50_template	WinText 5.0 (a novější: 602Text) - šablona
ftHTML	HTML

Externí formáty

ASCII STANDARD

Identifikátor	Význam
ftASCII_PC	standard pro PC (DOS)
ftASCII_MAC	standard pro Macintosh
ftASCII_Win	standard pro PC (Windows)
ftASCII_UNIX	standard pro Unix

ASCII PLAIN

Identifikátor	Význam
ftASCII_PC_pl	PC (DOS) plain
ftASCII_MAC_pl	Macintosh plain
ftASCII_Win_pl	PC (Windows) plain
ftASCII_UNIX_pl	Unix plain

ASCII STRIPPED

Identifikátor	Význam
ftASCII_PC_st	PC (DOS) stripped
ftASCII_MAC_st	Macintosh stripped
ftASCII_Win_st	PC (Windows) stripped
ftASCII_UNIX_st	Unix stripped

ASCII SMART

Identifikátor	Význam
ftASCII_PC_sm	PC (DOS) smart
ftASCII_MAC_sm	Macintosh smart
ftASCII_Win_sm	PC (Windows) smart
ftASCII_UNIX_sm	Unixsmart

COMMUNICATIONS FORMÁT

Identifikátor	Význam
ftCommunications	Communications format

WORDSTAR

Identifikátor	Význam
ftWordStar_Win	WordStar for Windows
ftWordStar_70	WordStar 7.0
ftWordStar_60	WordStar 6.0
ftWordStar_55	WordStar 5.5
ftWordStar_50	WordStar 5.0

ftWordStar_40	WordStar 4.0
ftWordStar_345	WordStar 3.45
ftWordStar_33x	WordStar 3.30, 3.31

MSWORD

Identifikátor	Význam
ftMSWord_60	MSWord 6.0
ftMSWord_50_55	MSWord 5.0 , 5.5
ftMSWord_40	MSWord 4.0
ftMSWord_30_31	MSWord 3.0 , 3.1

WORDPERFECT

Identifikátor	Význam
ftWordPerfect_42	WordPerfect 4.2
ftWordPerfect_41	WordPerfect 4.1
ftWordPerfect_51	WordPerfect 5.1
ftWordPerfect_50	WordPerfect 5.0

WORDPERFECT FOR WINDOWS

Identifikátor	Význam
ftWinWP5x	WordPerfect for Windows 5.1 – 5.2

PROFESSIONAL WRITE

Identifikátor	Význam
ftProfWrite_2x	Professional Write 2.0 – 2.2
ftProfWrite_10	Professional Write 1.0

PROFESSIONAL WRITE PLUS

Identifikátor	Význam
ftProfWritePlus	Professional Write Plus

PFS:FIRST CHOISE

Identifikátor	Význam
ftPFS30	PFS:First Choise 3.0
ftPFS20	PFS:First Choise 2.0
ftPFS10	PFS:First Choise 1.0
ftPFS_WRITE_VerC	PFS:Write Ver C

MS RTF

Identifikátor	Význam
ftMS_RTF	Microsoft RTF
ftMS_RTF_ANSI	Microsoft RTF Ansi char set
ftMS_RTF_CP850	Microsoft RTF Code page 850
ftMS_RTF_PC	Microsoft RTF PC char set
ftMS_RTF_MAC	Microsoft RTF Mac char set

AMI PRO

Identifikátor	Význam

ftAmiPro_1x_30	Ami Professional 1.x , 2.0, 3.0

WORDSTAR FOR WINDOWS

Identifikátor	Význam
ftWordStar_Win	WordStar for Windows

LEGACY

Identifikátor	Význam
ftLegacy_1x_20	Legacy 1.0 – 2.0

ONGO

Identifikátor	Význam
ftOnGO	on GO

MS WRITE FOR WINDOWS

Identifikátor	Význam
ftMSWinWrite_3x	MicrosoftWindows Write 3.x

MS WORD FOR WINDOWS

Identifikátor	Význam
ftMSWinWord_60	Microsoft Word for Windows 6.0
ftMSWinWord_20	Microsoft Word for Windows 2.0
ftMSWinWord_1x	Microsoft Word for Windows 1.x

MACWORD

Identifikátor	Význam
ftMacWord_50_51	MacWord 5.0 – 5.1
ftMacWord_40	MacWord 4.0
ftMacWord_30	MacWord 3.0

Encyklopedie funkcí

Tvar popisu funkcí a procedur

Popis každé funkce nebo procedury obsahuje níže popsanou strukturu. Některé části popisu mohou chybět; například u procedur se neuvádí hodnota.

Jméno_funkce

```
function jméno(param_1 : typ; param_n : typ) : typ;
```

Parametry: *param_1* popis prvního parametru
param_n popis dalšího parametru

Implicitní hodnota: hodnoty implicitních parametrů

Popis: činnost funkce resp. procedury v souvislosti s jejími parametry

Hodnota: návratová hodnota funkce a co lze z této hodnoty vyčíst

Poznámka: doplňková informace nehodící se do jiné položky tohoto výčtu

Omezení: předpoklady korektního chování procedury (funkce)

Příklad: příklad volání funkce

Viz: jména jiných procedur a funkcí z téže oblasti.

Abs

```
function Abs(
  r : real) : real;
```

Parametry: *r* reálné číslo

Funkce vrací absolutní hodnotu čísla zadaného jako parametr. Na rozdíl od podobné funkce v Pascalu je typ výsledku vždy *real*.

Popis: Funkce spočte absolutní hodnotu reálného čísla.

Viz: **Iabs**

AddIndexEntry

```
function AddIndexEntry(
  dlg : boolean;
  var entry : const string) : short;
```

Parametry: *dlg* vyvolávat dialog
entry nová položka rejstříku

Popis: Funkce slouží pro přidávání položek do rejstříku.
Pro ***dlg*** **= TRUE** - podobně jako u funkce **AddIndexEntryDlg** se vyvolá dialog **Položka rejstříku** (příkaz **Označit položku rejstříku**), s tím rozdílem, že obsah vstupního pole tohoto dialogu je dán parametrem *entry*. Proto se také neoznačuje žádný "vhodný kandidát" za položku rejstříku před vyvoláním dialogu. Odkaz na položku vloženou do rejstříku bude dán okamžitou pozicí řádkového kurzoru (eventuelně jím bude ten označený textový blok), odpovídající stavu před zavoláním funkce.
Podle způsobu opuštění dialogu funkce vrátí:

IDOK - přidat do rejstříku
IDCANCEL - zrušit.

Pro ***dlg*** **= FALSE** - provede se analogická činnost s tím rozdílem, že příslušné dialogové okno se nevyvolá, ale položka *entry* se rovnou přidá do rejstříku.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **IndexExists**
IndexEntry
IndexEntryIn
CountOfEntryRefIndex
EditDlgAddIndexEntryDlg
GoToIndexEntry
SelectIndex
RemoveIndexEntry

AddIndexEntryDlg

```
function AddIndexEntryDlg : short;
```

Popis: Funkce odpovídá položce menu **Pomůcky** (příkaz **Rejstřík/Označit položku rejstříku**). Pokud není v textu označen blok, stejně jako při vyvolání příslušného dialogu z menu se označí v textu "vhodný kandidát" za položku rejstříku a po vyvolání dialogu je označený text nabízen ve vstupním poli.
Podle způsobu opuštění tohoto dialogu funkce vrátí:
IDOK – přidat do rejstříku
IDCANCEL – zrušit

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **IndexExists**
IndexEntries
IndexEntry
IndexEntryInd
SelectIndex
IndexEditDlg
CountOfEntryRef
GoToIndexEntry
AddIndexEntry
RemoveIndexEntry

Addr

```
function Addr(
  var s : anytype) : ^anytype;
```

Parametry: *s* proměnná libovolného typu

Popis: Je-li T libovolný typ a S proměnná typu T, funkce vrací adresu proměnné S**,** tedy výraz `Addr` (S) je typu ^T.

Při deklaracích:

```
Type T = ... {popis typu};
TpT = ^T;
var    S : T;
pS : TpT;
```

lze dosazovat:

```
pS : = Addr(S)
```

Výsledky funkce **Addr** mohou být také použity jako operandy relačních operátorů = a <>. Dosazování a porovnávání nelze ovšem provádět libovolně, ale se zřetelem k typové kompatibilitě.

Při deklaraci:

```
Type
 TA = ... {popis typu};
 TB = ... {popis typu};
 TpTA = ^TA;
 TpTB = ^TB;
var
 a1, a2 : TA;
 pa1, pa2 : TpTA
 b1, b2 : TB;
 pb : TpTB;
```

lze provádět např. tato přiřazení a porovnání:

```
pa1 := Addr(a1);
pa2 := Addr(a2);
pa1 := pa2;
if pa2 = Addr(a1) then pb := Addr(b1);
```

ovšem každý z následujících příkazů:

```
pa1 := Addr(b1);
pb := pa1;
pb := Addr(a2);
if pb = Addr(a1) then pa2 := Addr(b2);
```

je syntakticky správný a přeloží se pouze tehdy, jsou-li typy TA,TB kompatibilní.

Hodnota: Funkce vrací adresu svého parametru.

Poznámka: Podrobnější informace o použití ukazatelů a funkci **Addr** se nacházejí v popisu vnitřního jazyka u popisu typu ukazatel.

AddToDbRecord

```
function AddToDbRecord [
  var field_name : const string;
  field_type : integer;
  var field_value : const string]:boolean;
```

Parametry:

field_name	jméno položky v databázi	
field_value	hodnota položky	
field_type	typ položky:	
	BOOLEAN	1
	CHAR	2
	INT16	3
	INT32	4
	MONEY	5
	FLOAT	6
	STRING	7
	CSSTRING	8
	CSISTRING	9
	DATE	10
	TIME	11

Popis: Přidání položky do záznamu (record) databáze. Vlastní zápis kompletního záznamu se provede funkcí **SaveDbRecord**.

Hodnota: TRUE = byl-li zápis úspěšný

Viz: **SetMM**
DisconnectMM
SaveDbRecord
MailMergeType
RecordsCount
RecordFirst
RecordLast

RecordNext
RecordPrev
RecordCurrent
RecordSet
NextMMField
GetTextMM
GetTextMMI

Arctan

```
function Arctan(
  r : real) : real;
```

Parametry: *r* reálné číslo

Popis: Funkce počítá arkus tangens vyjádřený v radiánech.

Hodnota: Funkce vrátí hodnotu arkus tangenty.

AreObjectsOn

```
function AreObjectsOn : boolean;
```

Popis: Funkce zjišťuje, zda-li jsou zobrazeny objekty.

Poznámka: Nemá-li žádný aktuální dokument nebo je-li aktuální dokument v zobrazení osnovy, vrací FALSE.

Příklad:
```
if not AreObjectsOn then ViewObjects;
```

Viz: **ViewObjects**

AreScrollBarsOn

```
function AreScrollBarsOn : boolean;
```

Popis: Funkce zjišťuje, zda-li jsou zobrazeny posuvníky.

Poznámka: Nemá-li žádný aktuální dokument, vrací FALSE.

Příklad:
```
if not AreScrollBarsOn then ViewScrollbars;
```

Viz: **ViewScrollBars**

ArrangeIcons

```
procedure ArrangeIcons;
```

Popis Procedura spustí příkaz **Uspořádat ikony** z menu **Okno**.

Viz: **WindowsTile**
WindowsCascade

AtDocEnd

```
function AtDocEnd : boolean;
```

Popis: Funkce zjišťuje, zda se řádkový kurzor nachází na konci aktuálního dokumentu.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **AtDocStart**
CaretEnd
CaretHome

AtDocStart

```
function AtDocStart : boolean;
```

Popis: Funkce zjišťuje, zda se řádkový kurzor nachází na začátku aktuálního dokumentu.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **AtDocEnd**
CaretEnd
CaretHome

AtObjectEnd

```
function AtObjectEnd : boolean;
```

Popis: Funkce zjišťuje, zda se řádkový kurzor nachází na konci objektu (textového rámce anebo textové tabulky).

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **AtObjectStart**
CaretObjectEnd
CaretObjectStart

AtObjectStart

```
function AtObjectStart : boolean;
```

Popis: Funkce zjišťuje, zda se řádkový kurzor nachází na začátku objektu (textového rámce anebo textové tabulky).

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **AtObjectEnd**
CaretObjectEnd
CaretObjectStart

BackspaceDelete

```
function BackspaceDelete : boolean;
```

Popis: Funkce vymaže znak před řádkovým kurzorem (existuje-li, tj. není-li řádkový kurzor na začátku dokumentu). Pokud se znak vypustil, vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **DeleteChar**
DeleteWord

BalanceColumns

```
procedure BalanceColumns;
```

Popis: Procedura odpovídá položce menu **Formát** příkaz **Vyrovnat délky sloupců**.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Beep

```
procedure Beep(
  beep_type : short);
```

Parametry: *beep_type* druh "pípnutí"

Implicitní hodnota: *beep_type* = 1

Popis: Procedura slouží pro krátké zvukové znamení.

Poznámka: Parametr *beep_type* má smysl zadávat od 1 do 6, pro jiné hodnoty procedura provede to samé co pro *beep_type* = 1. Pro konkrétní hodnoty závisí druh zvukového znamení na systémovém nastavení Windows (**Control panel/Sounds**); u většiny PC to bude standardní píp.

Příklad:

```
for j:= 4 downto 2 do
begin
beep; beep(j);
end;
```

BlockName

```
function BlockName(
  index : integer) : string;
```

Parametry: *index* pořadové číslo pojmenovaného bloku (1<= index <= BlocksCount)

Popis: Funkce vrací jméno index-tého pojmenovaného bloku.

Omezení: Požaduje neminimalizovaný aktuální dokument.

Viz: **DoesBlockExist**
SelectBlock
EditBlockDlg
LabelBlock
UnlabelBlock
BlocksCount
CopyBlock

BlocksCount

```
function BlocksCount : integer;
```

Popis: Funkce vrací počet pojmenovaných bloků v aktuálním dokumentu.

Omezení: Požaduje neminimalizovaný aktuální dokument.

Viz: **DoesBlockExist**
SelectBlock
EditBlockDlg
LabelBlock

UnlabelBlock
BlockName
CopyBlock

Bool2str

```
function Bool2str(
  b : boolean) : string;
```

Parametry: *b* boolovská hodnota, která bude konvertována

Popis: Funkce převede hodnotu *b* na řetězec znaků.

Hodnota: Funkce vrátí řetězec znaků TRUE nebo FALSE.

BottomOfScreen

```
function BottomOfScreen(
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Implicitní hodnota: *shift* = FALSE

Popis: Funkce přesune kurzor na spodní řádek aktuálního okna a zanechá sloupec beze změny (je-li to možné). Pokud se řádkový kurzor pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharLeft**
CharRight
LineDown
LineUp
CaretEnd
CaretHome
LeftOfLine
RightOfLine
WordLeft
WordRight
RightOfWord
PageDown
PageUp
TopOfScreen

CanUndo

```
function CanUndo : boolean;
```

Popis: Funkce testuje přístupnost příkazů **Odvolat**.

Viz: **Undo**

CaretCellStart

```
function CaretCellStart(
  cell_index, row_index : integer) : boolean;
```

Parametry: *cell_index* sloupec buněk v textové tabulce
row_index řádek buněk v textové tabulce

Popis: Umístění kurzoru v buňce textové tabulky ve sloupci *cell_index* a řádku *row_index*. Při chybě vrací FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru umístěném v textové tabulce.

Viz: **GetCaretCell**
GetCaretRow
GetNumCells
GetNumRows
PrevCell
NextCell
PrevRow
NextRow

CaretEnd

```
function CaretEnd(
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Implicitní hodnota: *shift* = FALSE

Popis: Funkce přesune kurzor na konec dokumentu. Pokud se řádkový kurzor pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharLeft**
CharRight
LineDown
LineUp
CaretHome
LeftOfLine
RightOfLine
WordLeft
WordRight
RightOfWord
PageDown
PageUp
BottomOfScreen
TopOfScreen

CaretHome

```
function CaretHome(
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Implicitní hodnota: *shift* = FALSE

Popis: Funkce přesune kurzor na začátek dokumentu. Pokud se řádkový kurzor pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharLeft**
CharRight
LineDown
LineUp
CaretEnd
LeftOfLine
RightOfLine
WordLeft
WordRight
RightOfWord
PageDown
PageUp
BottomOfScreen
TopOfScreen

CaretLeft

```
function CaretLeft(
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Implicitní hodnota: *shift* = FALSE

Popis: Funkce simuluje stisk klávesy ← (šipka vlevo). Na rozdíl od fukce **CharLeft** nepřeskakuje tabelační znaky. Pokud se řádkový kurzor pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharRight**
CharLeft
LineDown
LineUp
CaretEnd
CaretHome
LeftOfLine
RightOfLine
RightOfWord
PageDown
PageUp
BootomOfScreen
TopOfScreen
CaretRight

CaretObjectEnd

```
function CaretObjectEnd(
  id : integer) : boolean;
```

Parametry: *id* identifikátor objektu

Popis: Pokud je parametr *id* identifikátorem existujícího objektu a tento objekt je textový (textový rámec nebo textová tabulka), funkce přesune řádkový kurzor na jeho konec a vrátí TRUE. Jinak se neprovede nic a návratová hodnota bude FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **AtObjectEnd**
AtObjectStart
CaretObjecStart

GetCaretObjectId

CaretObjectStart

```
function CaretObjectStart(
  id : integer) : boolean;
```

Parametry: *id* identifikátor objektu

Popis: Pokud je parametr *id* identifikátorem existujícího objektu a je-li tento objekt textový (textový rámec nebo textová tabulka), funkce přesune řádkový kurzor na jeho začátek a vrátí TRUE. Jinak se neprovede nic a návratová hodnota bude FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **AtObjectEnd**
AtObjectStart
CaretObjectEnd
GetCaretObjectId

CaretRight

```
function CaretRight(
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Implicitní hodnota: shift = FALSE

Popis: Funkce simuluje stisk klávesy ➔ (šipka vpravo). Na rozdíl od funkce **CharRight** nepřeskakuje tabelační znaky. Pokud se řádkový kurzor pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharRight**
CharLeft
LineDown
LineUp
CaretEnd
CaretHome
LeftOfLine
RightOfLine
RightOfWord
PageDown
PageUp
BottomOfScreen
TopOfScreen
CaretLeft

CleanString

```
procedure CleanString(
  var text : string);
```

Parametry: *text* upravovaný řetězec znaků

Popis: Procedura "čistí" řetězec od znaků, které nelze vložit do textu (a nahrazuje je mezerami).

Poznámka: Funkce je automaticky volána na parametr typu *string* ve funkci **InsertText**.

Viz: **InsertText**
InsertTextStr

ClearMark

```
function ClearMark(
  var mark_name : const string) : boolean;
```

Parametry: *mark_name* jméno záložky

Popis: Funkce vymaže záložku *mark_name* je-li nastavena. Pokud ano, vymaže ji a vrací TRUE. V opačném případě vrací FALSE.

Poznámka: Funkce zohledňuje rozdíl mezi malými a velkými písmeny.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **EditMarksDlg**
InsertMark
GoToMark
DoesMarkExist
MarkName
MarksCount

ClipCopy

```
function ClipCopy : boolean;
```

Popis: Funkce kopíruje označený text do schránky. Nebyl-li žádný text označen, vrací FALSE.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **ClipCut**
ClipPaste

ClipCut

```
function ClipCut : boolean;
```

Popis: Funkce vystřihne označený text do schránky. Nebyl-li žádný text označen, vrací FALSE.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **ClipCopy**
ClipPaste

ClipPaste

```
function ClipPaste : boolean;
```

Popis: Funkce vlepí obsah schránky na pozici řádkového kurzoru. Nebyl-li text vlepen, vrací FALSE.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **ClipCopy**
ClipCut

Close

```
procedure Close(
  f : file);
```

Parametry: *f* proměnná typu soubor (file)

Popis: Procedura uzavírá dříve otevřený soubor. Každý soubor, který byl otevřen funkcí **Reset** nebo **Rewrite**, by měl být uzavřen procedurou **Close**. Pokud není explicitně uzavřen, uzavře jej 602Text při ukončení běhu programu. Soubor musí být bezpodmínečně uzavřen dříve, než je podruhé otevřen funkcí **Reset** nebo **Rewrite**. Dojde-li před uzavřením souboru ke zhroucení operačního systému, nebude zaznamenána změna velikosti souboru vyplývají z operací zápisu provedených nad tímto souborem.

Příklad:
```
VAR
f : FILE;
BEGIN
Rewrite(f,'C:\Report.TXT');
...
Close(f);
END.
```

Viz: **Read**
Write
Writeln
Reset
Rewrite
Seek
Eof
Filelenght

ColorDlg

```
function ColorDlg : short;
```

Popis: Funkce zajišťuje vyvolání dialogu **Barvy**. Podle způsobu jeho opuštění vrací IDOK, IDCANCEL.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **SetRgbColor**
FormatFontDlg

ContentsEnabled

```
function ContentsEnabled : boolean;
```

Popis: Funkce vrací přístupnost položky menu **Pomůcky** příkaz **Obsah**.

Příklad:
```
if ContentsEnabled and not ContentsExists
then CreateContents;
```

Viz: **CreateContents**
ContentsExists
HyphenDlgEnabled
DatabaseEnabled

IndexMarkEnabled
IndexEditEnabled
IndexGenEnabled
SpellEnabled
LangSetupEnabled
TranslateEnabled
ThesaurusEnabled

ContentsExists

```
function ContentsExists : boolean;
```

Popis: Funkce zjišťuje, má-li aktuální dokument vytvořen obsah.

Omezení: Potřebuje aktuální dokument.

Viz: **SelectContents**
IndexExists

CopyBlock

```
function CopyBlock(
  var name : const string) : boolean;
```

Parametry: *name* jméno pojmenovaného bloku

Popis: Funkce zkopíruje příslušný pojmenovaný blok do schránky.

Omezení: Požaduje neminimalizovaný aktuální dokument.

Viz: **DoesBlockExist**
SelectBlock
EditBlocksDlg
LabelBlock
UnlabelBlock
BlocksCount
BlockName

Cos

```
function Cos(
  r : real) : real;
```

Parametry: *r* reálné číslo

Popis: Funkce počítá kosinus úhlu zadaného v radiánech.

Hodnota: Funkce vrátí hodnotu kosinu zadaného úhlu.

CountFonts

```
function CountFonts : short;
```

Popis: Funkce vrací počet aktuálně dostupných druhů písma (fontů).

Viz: **FontName**

CurrentFontUseFont
UseFontOfName

CountMacros

```
function CountMacros(
   global : boolean) : integer;
```

Parametry: *global* globální makra

Implicitní hodnota: *global* = TRUE

Popis: Funkce vrací počet právě spustitelných maker, resp. lokálních maker v aktuálním dokumentu.

Poznámka: Pokud při zavolání funkce není otevřen žádný aktuální dokument **CountMacros** vrací (FALSE) nulu.

Viz: **MacroName**
MacroIsInMenu
MacroSetToMenu
MacroDescription
MacroSetDesc
MacroDelete

CountOfEntryRef

```
function CountOfEntryRef(
   var entry : const string) : short;
```

Parametry: *entry* položka rejstříku (textový řetězec)

Popis: Pro danou položku rejstříku aktuálního dokumentu *entry* vrací počet referencí (odkazů) na ni v tomto dokumentu. Pokud se daná položka v rejstříku nenachází, funkce vrátí 0.

Omezení: Potřebuje aktuální dokument.

Viz: **IndexExists**
IndexEntries
IndexEntry
IndexEntryInd
IndexEditDlg
GoToIndexEntry
AddIndexEntryDlg
AddIndexEntry
RemoveIndexEntry
SelectIndex

CountWindows

```
function CountWindows : short;
```

Popis: Funkce vrací počet oken s dokumenty.

Viz: **IsAnyDocOpened**

CreateContents

```
function CreateContents(
  dlg : boolean;
  firstlevel, lastlevel : short;
  ignore_old : boolean) : short;
```

Parametry:

dlg	zobrazovat dialog
firstlevel	první úroveň
lastlevel	poslední úroveň
ignore_old	mazat dříve vytvořený obsah

Implicitní hodnota: *ignore_old* = FALSE

Popis: Funkce vytváří obsah dokumentu.
Pro ***dlg* = TRUE** - zavolá dialog **Obsah**, jeho vstupní pole inicializuje parametry *firstlevel*, *lastlevel;* přičemž *firstlevel* = 0 a zároveň *lastlevel* = 0 interpretuje jako "všechny". Mají-li jiné hodnoty a neplatí-li přitom (1 <= *firstlevel* <= *lastlevel* <= 9), inicializuje se dialog jako pro hodnoty *firstlevel* = 1, *lastlevel* = 9. Pokud již existuje obsah a parametr *ignore_old* = TRUE, smaže se tento při vytvoření nového bez dalšího dotazu (jinak se objeví další dialog). Funkce vrátí IDOK či IDCANCEL podle reakce na první; event. ještě druhý dialog.
Pro ***dlg* = FALSE** - rovnou se vytvoří obsah. Ostatní parametry mají stejný význam s tímto jediným rozdílem: není-li (*firstlevel; lastlevel*) = (0; 0) a neplatí-li (1 <= *firstlevel) && (firstlevel* <=*lastlevel) &&* (*lastlevel* <= 9), neprovede se nic a funkce vrátí IDERROR.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy). Řádkový kurzor se nesmí nacházet v textovém rámci ani v poznámce pod čarou.

Viz: **ContentsEnabled**

CreateFooter

```
function CreateFooter : short;
```

Popis: Odpovídá položce menu **Formát** (příkaz **Kapitola**/**Zápatí)**. Pokud aktuální kapitola ještě žádné nemá, vloží se nové a funkce vrátí IDOK. V opačném případě vyvolá příslušný dialog a dle volby v něm se vrátí:
Esc – IDCANCEL
Nové – IDOK
Jdi na – IDGOTO.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CreateHeader**
ChapterHasHeader

CreateHeader

```
function CreateHeader : short;
```

Popis: Odpovídá položce menu **Formát (**příkaz **Kapitola**/**Záhlaví)**. Pokud aktuální kapitola ještě žádné nemá, vloží se nové a vrátí IDOK. V opačném případě vyvolá příslušný dialog a dle volby v něm se vrátí:
Esc – IDCANCEL
Nové – IDOK
Jdi na – INGOTO

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CreateFooter**
ChapterHasHeader

CreateIndex

```
function CreateIndex(
  dlg : boolean;
  separator : short;
  check_exists : boolean) : short;
```

Parametry:
dlg zobrazovat dialog
separator druh oddělování položek rejstříku
check_exists konrolovat existenci dříve vytvořeného rejstříku

Implicitní hodnota: *check_exists* = TRUE

Popis: Odpovídá činnosti vytvořit rejstřík:
Pro ***dlg* = TRUE** - pokud ještě není označena žádná položka rejstříku, vrátí IDERROR. Jinak se vyvolá dialog **Rejstřík** (menu **Pomůcky**, submenu **Vytvořit rejstřík**) a jeho vstupní pole se parametrem *separator_style* inicializuje:
0 – žadné dělení položek
1 – prázdná řádka
2 – písmeno
Pro jiné hodnoty *separator_style* se uplatní aktuální stav rejstříku v tomto dokumentu (či 0, pokud žádný ještě nebyl vytvořen). Při opuštění dialogu:
a) při stisku klávesy **Esc** – vrátí IDCANCEL
b) při stisku klávesy **Enter**:
`check_CANCEL`, jinak `IDOK`;
`check_exists = FALSE`
vrátí `rovexists = TRUE`
existuje-li rejstřík, zeptá se, zda přepsat novým; při odpovědi **Ne** vrátí IDnou IDOK.
Pro ***dlg* = FALSE** - přeskočí první dialog, rovnou se použijí hodnoty *separator_style*, *check_exists* a vrátí IDOK/IDCANCEL (s tím rozdílem, že chybné hodnoty *separator_style* nenahrazuje nějakou jinou, ale vrátí IDERROR).

Poznámka: Funkce může vrátit IDERROR také proto, že nebude dost paměti pro vytvoření rejstříku.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **IndexExists**

CurrentFont

```
function CurrentFont : string;
```

Popis: Funkce vrací název fontu, kterým je zapsán text na pozici řádkového kurzoru (stejný textový řetězec se nachází na pevném liště, je-li ovšem tato zobrazena).

Poznámka: Je-li označen blok textu, ve kterém je text zapsaný dvěma nebo více fonty, funkce vrátí prázdný řetězec.

Omezení: Vyžaduje aktuální neminimalizovaný dokument při aktivním řádkovém kurzoru.

Viz: **CountFonts**
FontName
UseFont
UseFontOfName

CurrentMacroName

```
function CurrentMacroName : string;
```

Popis: Funkce vrací jméno právě běžícího makra.

Viz: **MacroName**
MacroAutoStarted
MacroIsInMenu
MacroIsLocal
CountMacros

DatabaseEnabled

```
function DatabaseEnabled : boolean;
```

Popis: Funkce vrací přístupnost položky menu **Pomůcky** příkaz **Nastavení databáse**.

Viz: **HyphenDlgEnabled**
ContentsEnabled
IndexMarkEnabled
IndexEditEnabled
IndexGenEnabled
LangSetupEnabled
TranslateEnabled
ThesaurusEnabled
SpellEnabled

Date2str

```
function Date2str(
  dt : date;
  prez : short) : string;
```

Parametry: *dt* hodnota, která se má konvertovat
prez způsob konverze

Popis: Funkce převede hodnotu zadanou prvním parametrem na řetězec znaků. Při konverzi se použije hodnota parametru *prez* takto:

- je-li *prez* nulové zápis data bude obsahovat pouze den a měsíc
- je-li *prez* nenulové zápis data bude obsahovat den, měsíc a rok.

Hodnota: Funkce vrací textový zápis data *dt*.

Viz: **Int2str**
Str2int
Real2str
Str2real
Money2str
Str2money
Str2date
Time2str
Str2time

Day

```
function Day(
  dt : date) : short;
```

Parametry: *dt* datum (ve vnitřním formátu 602Text)

Popis: Funkce extrahuje den z data.

Hodnota: Funkce vrací číslo označující den v měsíci, toto číslo leží v intervalu 1 až 31.

Viz: **Make_date**
Month
Year
Today
Make_time
Hours
Minutes
Seconds
Sec1000
Now
Day_of_week

Day_of_week

```
function Day_of_week(
  dt : date) : short;
```

Parametry: *dt* datum

Popis: Funkce zjišťuje den v týdnu odpovídající zadanému datu.

Hodnota: Funkce vrací číslo dne v týdnu. Neděle je označena číslem 0, pondělí 1, atd. Pokud má datum nepřípustný formát, funkce vrací hodnotu 7.

Příklad:

```
var
s, sv : string[30];
begin
case Day_of_week(Today) of
0 : s := 'neděle';
      1 : s := 'pondělí' :
      2 : s := 'úterý' :
      3 : s := 'středa' :
      4 : s := 'čtvrtek':
      5 : s := 'pátek' :
      6 : s := 'sobota';
end;
sv := 'Dnes je ' + s;
Info_box('', sv);
end.
```

Viz: **Make_date**
Day
Month
Year
Today

Dec

```
procedure Dec(
   var v : ordinaltype;
   int : integer);
```

Parametry: *v* proměnná libovolného ordinálního typu
int výraz typu integer

Implicitní hodnota: *int* = 1

Popis: Má-li při volání procedury `Dec` parametr *int* svou implicitní hodnotu 1, dosadí příkaz **Dec** (*v*) do proměnné *v* hodnotu jejího předchůdce, tj. má stejný efekt jako přiřazení `v := Pred (v)`.

Je-li proměnná *v* celočíselného typu, zmenší se tedy její hodnota o jedničku. Obecně má příkaz **Dec**(v, int)stejný význam jako příkaz:

```
if int > 0 then
for j:= 1 to int do v := Pred(v)
else
for j:= -1 downto int do v := Succ(v);
```

(kde proměnná **j** je pomocná celočíselná proměnná).

Je-li proměnná v číselného typu, zmenší se tedy její hodnota o hodnotu výrazu *int*.

Poznámka: Další podrobnosti naleznete v popisu vnitřního jazyka v části týkající se ordinálních typů.

Viz: **Inc**
Succ
Pred

DefaultDocDir

```
function DefaultDocDir : string;
```

Popis: Funkce vrací název implicitního adresáře pro otevírání dokumentů.

Viz: **DefaultTmplDir**

DefaultTmplDir

```
function DefaultTmplDir : string;
```

Popis: Funkce vrací název implicitního adresáře pro otevření šablon.

Viz: **DefaultDocDir**

DefParaStyleDlg

```
function DefParaStyleDlg : short;
```

Popis: Funkce vyvolá dialog **Definice stylu odstavce –** vrací IDOK nebo IDCANCEL.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetStyleCount**
GetStyleName
UseParaStyleNum

Delete_file

```
function Delete_file(
  var filename : const string) : boolean;
```

Parametry: *filename* plné jméno souboru podle konvencí DOSu

Popis: Funkce smaže soubor označený parametrem *filename*.

Hodnota: Funkce vrací TRUE, pokud se podařilo soubor úspěšně smazat. V opačném případě vrátí FALSE.

Příklad: `Delete_file("C:\WT602\OUTPUT\out.txt");`

Viz: **Make_directory**

DeleteBlock

```
function DeleteBlock : boolean;
```

Popis: Pokud byl označen blok textu, funkce jej vymaže. Nebyl-li žádný text označen, funkce vrací FALSE.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

DeleteChar

```
function DeleteChar : boolean;
```

Popis: Funkce vymaže znak na aktuální pozici řádkového kurzoru (existuje-li takový). Pokud se znak vypustil, vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **BackspaceDelete**
DeleteWord

DeleteObject

```
function DeleteObject(
  id : integer): boolean;
```

Parametry: *id* identifikátor objektu

Popis: Funkce ruší (maže) v aktuálním dokumentu objekt daného identifikátoru *id*. Není-li žádný dokument otevřen nebo příslušný objekt neexistuje, běh makra je předčasně ukončen. Pokud daný objekt existuje, ale představuje skupinu jiných objektů (viz funkce **ObjectType**) funkce neprovede nic a vrátí FALSE.

Poznámka: Funkce neprovede nic (a vrátí FALSE) také tehdy, pokud sice daný objekt existuje, ale je součástí seskupení více objektů. Ke zrušení takového objektu je třeba použít funkci **DeleteObjGroup**, kde parametrem bude identifikátor příslušného seskupení.

Omezení: Funkce vyžaduje otevřený dokument.

Viz: **DeleteObjGroup**
ObjectId

ObjectIsSelected
ObjectsCount
ObjectSelCount
ObjectSelId
ObjectSelModify
ObjectType
ObjectPropDlg

DeleteObjGroup

```
function DeleteObjGroup(
  id : integer;
  all : boolean): boolean;
```

Parametry: *id* identifikátor objektu
all zrušit všechny objekty v seskupení

Implicitní hodnota: *all* = FALSE

Popis: Funkce ruší (maže) v aktuálním dokumentu pseudoobjekt daného identifikátoru *id*, jenž představuje seskupení více objektů. Není-li žádný dokument otevřen nebo tento objekt neexistuje, běh makra je předčasně ukončen.
Pokud daný objekt existuje, ale je jiného typu než seskupení objektů (viz funkce **ObjectType**) funkce neprovede nic a vrátí FALSE. Jinak je toto seskupení zrušeno a funkce vrátí TRUE; pokud má navíc parametr *all* hodnotu TRUE jsou zrušeny (smazány) všechny objekty původně v seskupení obsažené.

Omezení: Funkce vyžaduje otevřený dokument.

Viz: **DeleteObject**
ObjectId
ObjectIsSelected
ObjectsCount
ObjectSelCount
ObjectSelId
ObjectSelModify
ObjectType
ObjectPropDlg

DeleteWord

```
function DeleteWord : boolean;
```

Popis: Funkce vymaže text od pozice řádkového kurzoru až na začátek dalšího slova. Pokud se něco vypustilo, vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **BackspaceDelete**
DeleteChar

DialogCreate

```
function DialogCreate(
  x, y, cx, cy : short;
  var title : const string) : short;
```

Parametry: *x* x-ová souřadnice levého horního rohu dialogu

	y	y-ová souřadnice levého horního rohu dialogu
	cx	vodorovný rozměr dialogu
	cy	svislý rozměr dialogu
	title	nadpis dialogu

Popis: Vytváří prázdný dialog; lze mu přidávat ovládací prvky.

Poznámka: Vrací identifikátor uživatelského dialogu.

Viz: **DialogDestroy**
DialogRun
DialogCreateExt
DialogSetSysMenu

DialogCreateExt

```
function DialogCreateExt(
  x, y, cx, cy : short;
  var fontsize : short;
  var fontname : short;
  var title : const string) : short;
```

Parametry:	*x*	x-ová souřadnice levého horního rohu dialogu
	y	y-ová souřadnice levého horního rohu dialogu
	cx	vodorovný rozměr dialogu
	cy	svislý rozměr dialogu
	fontsize	velikost písma použitého v dialogu
	fontname	název písma použitého v dialogu
	title	nadpis dialogu

Popis: Vytváří prázdný dialog, do kterého lze přidávat ovládací prvky. Na rozdíl od funkce **DialogCreate** lze zadat i písmo, jímž budou vypsány všechny texty v dialogu.

Poznámka: Vrací identifikátor uživatelského dialogu.

Viz: **DialogCreate**
DialogDestroy
DialogRun
DialogSetSysMenu

DialogDestroy

```
procedure DialogDestroy(
  dlg_id : short);
```

Parametry:	*dlg_id*	identifikátor uživatelského dialogu

Popis: Zruší dialog; stav jeho ovládacích prvků bude nepřístupný.

Viz: **DialogCreate**
DialogRun

DialogFontSize

```
function DialogFontSize(
  dlg_id : short) : short;
```

Parametry:	*dlg_id*	identifikátor uživatelského dialogu

Popis: Funkce vrací velikost písma použitého v uživatelském dialogu.

Viz: **DialogCreateExt**
DialogFontResize
DialogSetSysMenu

DialogFontResize

```
procedure DialogFontResize(
  dlg_id : short) : short;
  newsize : const string);
```

Parametry: *dlg_id* identifikátor uživatelského dialogu
newsize nová velikost písma.

Popis: Procedura umožňuje změnit velikost písma použitého v uživatelském dialogu**.**

Viz: **DialogCreateExt**
DialogFontSize
DialogSetSysMenu

DialogRun

```
function DialogRun(
  dlg_id : short) : short;
```

Parametry: *dlg_id* identifikátor uživatelského dialogu

Popis: Spustí dialog vytvořený pomocí funkce **DialogCreate.**

Poznámka: Vrací identifikátor tlačítka, kterým byl ukončen.

Viz: **DialogCreate**
DialogDestroy
DialogSetSysMenu

DialogSetBmp

```
function DialogSetBmp(
  dlg_id : short;
  var filemane : const string) : boolean;
```

Parametry: *dlg_id* identifikátor uživatelského dialogu
filename jméno bitmapy

Popis: Nastavuje bitmapu coby pozadí uživatelského dialogu. Pokud se to nepodaří vrací FALSE.

Viz: **DlgSetCtrlBmp**
DlgTextSetBmp
DialogSetSysMenu

DialogSetSysMenu

```
procedure DialogSetSysMenu(
```

```
  dlg_id : short;
  on : boolean);
```

Parametry: *dlg_id* identifikátor uživatelského dialogu
on zobrazení/potlačení systémového tlačítka

Implicitní hodnota: *on* = TRUE

Popis: Funkce umožňuje pro dříve vytvořený uživatelský dialog nastavit hodnotu interního identifikátoru, který ovlivňuje, zda při zobrazení tohoto dialogu funkcí **DialogRun** bude též zobrazeno systémové tlačítko. Toto tlačítko umožňuje dialog ukončit myší stejně jako kdybychom na klávesnici stiskli **Esc** nebo **Alt+F4** anebo v tomto dialogu stiskli tlačítko, jehož identifikátor je IDCANCEL.

Poznámka: Aby bylo možné uživatelský dialog některým z těchto způsobů ukončit je nutné, aby obsahoval tlačítko, jehož identifikátor je IDCANCEL. Pokud dialog takové systémové tlačítko neobsahuje bude jeho případné zobrazení zašedlé a nefunkční.

Viz: **DialogCreate**
DialogCreateExt
DialogDestroy
DialogFontSize
DialogRun
DialogSetBmp
DialogFontResize

DisableAutoRun

```
procedure DisableAutoRun(
  disable : boolean);
```

Parametry: *disable* potlačí automatické spouštění automatických maker

Implicitní hodnota: *disable* = TRUE

Popis: Po zavolání této procedury s hodnotou parametru *disable* = TRUE je znepřístupněno automatické spouštění maker: **AutoOpen**, **AutoNew**, **AutoClose** a **AutoExit**. Tato makra se pak "chovají" (tzn. jsou spustitelná) pouze jako všechna ostatní makra. Tento stav platí až do doby, kdy je zrušen zavoláním **DisableAutoRun** (FALSE) anebo do ukončení a nového spuštění programu 602Text.

Poznámka: Funkci nelze použít pro zákaz běhu bezprostředně toho makra, ve kterém se její volání nachází (funkce je volána, když už makro běží). Nelze ji tedy použít pro zákaz automatického spuštění makra **AutoExec**, neboť po spuštění programu 602Text se **AutoExec** automaticky spouští jako první. Chcete-li potlačit automatické spuštění makra **AutoExec**, držte během spuštění 602Text klávesu **Shift**.

Viz: **ErasethisMacro**

DisconnectMM

```
function DisconnectMM : boolean;
```

Popis: Odpojí připojenou databázi.

Hodnota: TRUE = bylo-li odpojování databáze úspěšné

Viz: **SetMM**
AddToDbRecord
SaveDbRecord

MailMergeType
RecordsCount
RecordFirst
RecordLast
RecordNext
RecordPrev
RecordCurrent
RecordSet
NextMMField
GetTextMM
GetTextMMI

DisplayOutline

```
procedure DisplayOutline;
```

Popis: Procedura přepíná do zobrazení osnovy.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **DisplayPages**
PagesDisplayed

DisplayPages

```
procedure DisplayPages;
```

Popis: Procedura přepíná do režimu zobrazení stránek.

Omezení: Vyžaduje neminimalizovaný aktuální dokument.

Viz: **PagesDisplayed**
DisplayOutline

DlgBtnCancel

```
function DlgBtnCancel(
  dlg_id, x, y, cx, cy : short) : boolean;
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
x	x-ová souřadnice levého horního rohu tlačítka vzhledem k rámečku dialogu
y	y-ová souřadnice levého horního rohu tlačítka vzhledem k rámečku dialogu
cx	vodorovný rozměr tlačítka
cy	svislý rozměr tlačítka

Implicitní hodnoty: *cx* = 42
cy = 14

Popis: V dialogu vytvoří tlačítko **Zrušit**. Bude-li dialog ukončen tímto tlačítkem, funkce vrací IDCANCEL.

Viz: **DlgButton**
DlgBtnNo
DlgBtnOk
DlgBtnYes

DlgBtnNo

```
function DlgBtnNo(
  dlg_id, x, y, cx, cy : short) : boolean;
```

Parametry:
- *dlg_id* identifikátor uživatelského dialogu
- ***x*** x-ová souřadnice levého horního rohu tlačítka vzhledem k rámečku dialogu
- y y-ová souřadnice levého horního rohu tlačítka vzhledem k rámečku dialogu
- *cx* vodorovný rozměr tlačítka
- *cy* svislý rozměr tlačítka

Implicitní hodnoty:
- *cx* = 28
- *cy* = 14

Popis: V dialogu vytvoří tlačítko **Ne**. Bude-li dialog ukončen tímto tlačítkem, funkce vrací IDNO.

Viz: **DlgButton**, **DlgBtnCancel**, **DlgBtnOk**, **DlgBtnYes**, **DialogRun**

DlgBtnOk

```
function DlgBtnOk(
  dlg_id, x, y, cx, cy : short) : boolean;
```

Parametry:
- *dlg_id* identifikátor uživatelského dialogu
- *x* x-ová souřadnice levého horního rohu tlačítka vzhledem k rámečku dialogu
- *y* y-ová souřadnice levého horního rohu tlačítka vzhledem k rámečku dialogu
- *cx* vodorovný rozměr tlačítka
- *cy* svislý rozměr tlačítka

Implicitní hodnoty:
- *cx* = 42
- *cy* = 14

Popis: V dialogu vytvoří tlačítko **OK**. Bude-li dialog ukončen tímto tlačítkem funkce vrací IDOK.

Viz: **DlgButton**, **DlgBtnCancel**, **DlgBtnNo**, **DlgBtnYes**, **DialogRun**

DlgBtnYes

```
function DlgBtnYes(
  dlg_id, x, y, cx, cy : short) : boolean;
```

Parametry:
- *dlg_id* identifikátor uživatelského dialogu
- *x* x-ová souřadnice levého horního rohu tlačítka vzhledem k rámečku dialogu
- *y* y-ová souřadnice levého horního rohu tlačítka vzhledem k rámečku dialogu
- *cx* vodorovný rozměr tlačítka
- *cy* svislý rozměr tlačítka

Implicitní hodnoty:
- *cx* = 28

cy = 14

Popis: V dialogu vytvoří tlačítko **Ano**. Bude-li dialog ukončen tímto tlačítkem funkce vrací IDYES.

Viz: **DialogRun**
DlgButton
DlgBtnCancel
DlgBtnNo
DlgBtnOk

DlgButton

```
function DlgButton(
  dlg_id : short;
  var text : const string;
  id, x, y, cx, cy : short) : boolean;
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
text	text na tlačítku
id	identifikátor tlačítka
x	x-ová souřadnice levého horního rohu tlačítka vzhledem k dialogu
y	y-ová souřadnice levého horního rohu tlačítka vzhledem k dialogu
cx	vodorovný rozměr tlačítka
cy	svislý rozměr tlačítka

Implicitní hodnoty: *cx* = 42
cy = 14

Popis: Vytvoří nadepsané tlačítko v dialogu.

Poznámka: Bude-li dialog ukončen tímto tlačítkem funkce **DialogRun** vrátí ID.

Viz: **DlgBtnCancel**
DlgBtnNo
DlgBtnOk
DlgBtnYes
DlgButtonDefPush
DlgRemoveCtrl
DlgText
DlgCheckBox
DlgGroupBox
DlgInputLine
DlgListBox
DlgComboBox
DlgSetCtrlBmp
DlgLineY
DlgLineX

DlgButtonDefPush

```
procedure DlgButtonDefPush(
  dlg_id, id : short;
  on : boolean);
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
i*d*	identifikátor tlačítka
on	nastavuje/ruší danou vlastnost

Popis: Funkce pro tlačítko, které v uživatelském dialogu nastavuje vlastnost „reaguj na klávesu **Enter**“ („buď implicitní pro stisknutí“) nebo tuto vlastnost ruší. V případě

nastavení této vlastnosti pak takové tlačítko reaguje na stisk klávesy **Enter** v situaci, kterou popíšeme dále. Protože v uživatelském dialogu může být implicitní nejvýše jedno tlačítko, nastavení této vlastnosti automaticky ruší eventuální předchozí nastavení jiného tlačítka. Uživatelský dialog reaguje na stisk klávesy **Enter** tímto způsobem:
- Pokud právě je některé tlačítko aktivní je po stisknutí klávesy **Enter** dialog ukončen stejně, jako po stisknutí tohoto tlačítka myší.

- Pokud není žádné tlačítko aktivní (aktivní ovládací prvek je jiného typu), mohou nastat tyto situace:

- Některé tlačítko je nastaveno jako implicitní pro stisknutí; uživatelský dialog je pak ukončen stejně jako při stisknutí tlačítka myší.
- Žádné tlačítko není nastaveno jako implicitní, ale dialog obsahuje tlačítko s identifikátorem IDOK: uživatelský dialog je pak ukončen stejně jako při stisknutí tohoto tlačítka;
- Žádné tlačítko není implicitní a dialog neobsahuje tlačítko s dentifikátorem IDOK: uživatelský dialog není ukončen.

Viz: **DlgButton**
DlgBtnCancel
DlgBtnNo
DlgBtnOk
DlgBtnYes

DlgComboBox

```
function DlgComboBox(
  dlg_id : short;
  drop_down : boolean;
  id, x, y, cx, cy : short) : boolean;
```

Parametry:	*dlg_id*	identifikátor uživatelského dialogu
	drop_down	nastavuje, zda bude seznam rozbalovací
	id	identifikátor seznamu
	x	x-ová souřadnice levého horního rohu seznamu vzhledem k rámečku dialogu
	y	y-ová souřadnice levého horního rohu seznamu vzhledem k rámečku dialogu
	cx	vodorovný rozměr seznamu
	cy	svislý rozměr seznamu

Implicitní hodnoty: *cx* = 60
cy = 80

Popis: Umožňuje do uživatelského dialogu vložit seznam do něhož je možno vpisovat.

Viz: **DlgListBox**
DlgRemoveCtrl
DlgText
DlgButton
DlgCheckBox
DlgGroupBox
DlgInputLine
DlgLineX
DlgLineY
DlgStdBox

DlgEditBox

```
function DlgEditBox (
  dlg_id : short ;
  var text : const string;
  id, x, y, cx, cy : short) : boolean
```

Popis: Umožňuje vytvořit v uživatelském dialogu víceřádkový text.

Viz: **DlgInputLine**

DlgGroupBox

```
function DlgGroupBox(
  dlg_id : short;
   var text : const string;
  id, x, y, cx, cy : short) : boolean;
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
text	text názvu sekce ovládacích prvků
id	identifikátor čárového obdélníku – rámečku sekce – okolo ovládacích prvků dialogu
x	x-ová souřadnice levého horního rohu obdélníku vzhledem k rámečku dialogu
y	y-ová souřadnice levého horního rohu obdélníku vzhledem k rámečku dialogu
cx	vodorovný rozměr rámečku sekce
cy	svislý rozměr rámečku sekce

Popis: Umožňuje vložit rámeček sekce do uživatelského dialogu.

Viz: **DlgRemoveCtrl**
DlgText
DlgButton
DlgCheckBox
DlgInputLine
DlgListBox
DlgComboBox
DlgLineX
DlgLineY

DlgCheckBox

```
function DlgCheckBox(
  dlg_id : short;
  var text : const string;
  id, x, y, cx, cy : short) : boolean;
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
text	popis (zaškrtávacího) přepínače ☑
id	identifikátor přepínače
x	x-ová souřadnice levého horního rohu přepínače vzhledem k rámečku dialogu
y	y-ová souřadnice levého horního rohu přepínače vzhledem k rámečku dialogu
cx	vodorovný rozměr přepínače
cy	svislý rozměr přepínače

Implicitní hodnoty: cx = 80
cy = 10

Popis: Umožňuje vložit přepínač ☑ do uživatelského dialogu.

Viz: **DlgRemoveCtrl**
DlgCheckBox
CheckBoxGetVal
CheckBoxSetVal
DlgText
DlgButton
DlgGroupBox
DlgInputLine
DlgListBox
DlgComboBox
DlgLineX
DlgLineY

DlgInputLine

```
function DlgInputLine(
  dlg_id : short;
  var text : const string;
  id, x, y, cx, cy : short) : boolean;
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
text	text pro vstupní řádek
id	identifikátor vstupního řádku
x	x-ová souřadnice levého horního rohu vstupního řádku vzhledem k rámečku dialogu
y	y-ová souřadnice levého horního rohu vstupního řádku vzhledem k rámečku dialogu
cx	vodorovný rozměr vstupního řádku
cy	svislý rozměr vstupního řádku

Implicitní hodnoty: *cx* = 120
cy = 12

Popis: Umožňuje vytvořit vstupní řádek v uživatelském dialogu.

Viz: **DlgRemoveCtrl**
InputLineGetVal
DlgGroupBox
InputLineSetVal
DlgCheckBox
DlgButton
DlgText
DlgListBox
DlgComboBox
DlgLineY
DlgLineX

DlgLineX

```
function DlgLineX(
  dlg_id, id, x, y, cx : short) : boolean;
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
id	identifikátor ovládacího prvku
x	horizontální poloha v dialogu
y	vertikální poloha v dialogu
cx	horizontální rozměr

Popis: Umožňuje vložit vodorovnou čáru do uživatelského dialogu.

Viz: **DlgRemoveCtrl**
DlgButton
DlgCheckBox
DlgGroupBox
DlgInputLine
DlgListBox
DlgComboBox
DlgLineY

DlgLineY

```
function DlgLineY(
  dlg_id, id, x, y, cy : short) : boolean;
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
id	identifikátor ovládacího prvku
x	horizontální poloha v dialogu
y	vertikální poloha v dialogu
cy	vertikální rozměr

Popis: Umožňuje vložit svislou čáru do uživatelského dialogu.

Viz: **DlgRemoveCtrl**
DlgButton
DlgCheckBox
DlgGroupBox
DlgInputLine
DlgListBox
DlgComboBox
DlgLineX

DlgListBox

```
function DlgListBox(
  dlg_id : short;
  drop_down : boolean;
  id, x, y, cx, cy : short) : boolean;
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
drop_down	nastavuje, zda bude seznam rozbalovací
id	identifikátor seznamu
x	x-ová souřadnice levého horního rohu seznamu vzhledem k rámečku dialogu
y	y-ová souřadnice levého horního rohu seznamu vzhledem k rámečku dialogu
cx	vodorovný rozměr seznamu
cy	svislý rozměr seznamu

Implicitní hodnoty: *cx* = 60

cy = 80

Popis: Umožňuje do uživatelského dialogu vložit seznam, který slouží pouze pro výběr.

Viz: **DlgComboBox**
DlgRemoveCtrl
DlgText
DlgButton
DlgCheckBox
DlgGroupBox
DlgInputLine
DlgLineY
DlgLineX

DlgRadioBtn

```
function DlgRadioBtn(
  dlg_id, grp_id : short;
  var text : const string;
  id, x, y, cx, cy : short) : boolean;
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
grp_id	identifikátor skupiny (vícepolohových) přepínačů ⊙
text	popis přepínače ⊙
id	identifikátor přepínače
x	vodorovná poloha přepínače vzhledem k rámečku dialogu
y	vertikální poloha přepínače vzhledem k rámečku dialogu
cx	horizontální rozměr přepínače
cy	vertikální rozměr přepínače

Implicitní hodnoty: *cx* = 80
cy = 10

Popis: Funkce vytváří nový přepínač ⊙ se zadaným textem, polohou a rozměry a přidává jej do skupiny vícepolohových přepínačů identifikované parametrem *grp_id*.

Hodnota: Funkce vrací TRUE při úspěšném vytvoření nového přepínače ⊙, při neúspěchu vrací FALSE.

Viz: **DlgRadioBtnGroup**
RadioBtnCheck
RadioBtnChecked
RadioGrpGetVal
RadioGrpSetVal

DlgRadioBtnGroup

```
function DlgRadioBtnGroup(
  dlg_id, id : short) : boolean;
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
id	identifikátor skupiny přepínačů

Popis: Funkce vytváří v daném uživatelském dialogu "skrytý" řídící prvek, představující skupinu přepínačů. I pro tento "skrytý" řídící prvek platí, že jeho identifikátor je jedinečný (nesmí patřit žádnému řídícímu prvku dosud v dialogu umístěnému; při dalším vkládání řídících prvků jej už nelze použít). Po svém vytvoření je tato skupina prázdná; nové přepínače do ní přidáme funkcí **DlgRadioBtn**. Příslušnost

přidaných přepínačů k této skupině spočívá z uživatelského hlediska v tom, že právě jeden přepínač ze skupiny bude označen; příslušnou hodnotu nastavíme před zobrazením dialogu funkcí **RadioGrpSetVal** nebo **RadioBtnCheck**. Po ukončení dialogu ji zjistíme funkcí **RadioGrpGetVal** nebo **RadioBtnChecked**.

Hodnota: Funkce vrací TRUE při úspěšném vytvoření nového řídícího prvku, při neúspěchu (např. z důvodu nedostatku paměti) vrací FALSE.

Poznámka: Jednotlivé přepínače můžeme z dialogu odstranit funkcí **DlgRemoveCtrl**; celou skupinu odstraníme zavoláním této funkce, kde identifikátorem řídícího prvku bude identifikátor skupiny.

Viz: **DlgRadioBtn**
RadioBtnCheck
RadioBtnChecked
RadioGrpGetVal
RadioGrpSetVal

DlgRemoveCtrl

```
procedure DlgRemoveCtrl(
  dlg_id, id : short);
```

Parametry: *dlg_id* identifikátor uživatelského dialogu
id identifikátor ovládacího prvku dialogu

Popis: Odstraní zadaný ovládací prvek z dialogu. Lze použít např. k modifikaci dialogu mezi opakovaným voláním funkce **DialogRun**.

Viz: **DlgText**
DlgButton
DlgCheckBox
DlgGroupBox
DlgInputLine
DlgListBox
DlgComboBox
DlgLineY
DlgLineX

DlgSetCtrlBmp

```
function DlgSetCtrlBmp(
  dlg_id, id, index : short;
  var filename : const string) : boolean;
```

Parametry: *dlg_id* identifikátor uživatelského dialogu
id identifikátor ovládacího prvku
index stav ovládacího prvku
filename plné jméno bitmapy

Hodnoty: výčet stavů:
0 nestlačený a nezaškrtnutý
1 stlačený a nezaškrtnutý
2 nestlačený a zaškrtnutý
3 stlačený a zaškrtnutý

Popis: Umožňuje zobrazit ovládací prvek bitmap.

Poznámka: Nedefinované stavy jsou automaticky generovány ze základního, tzn., že můžete zadat pouze stav 0.

Omezení: Lze použít pouze pro tyto ovládací prvky: tlačítko, zaškrtávací přepínač, vícepolohový přepínač.

Viz: **DlgButton**
DlgBtnOK
DlgBtnCancel
DlgBtnYes
DlgBtnNo
DlgCheckBox
DlgRadioBtn
DlgTextSetBmp
DialogSetBmp

DlgSetCtrlPos

```
procedure DlgSetCtrlPos(
  dlg_id, id, x, y : short);
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
id	identifikátor ovládacího prvku dialogu
x	horizontální poloha ovládacího prvku vzhledem k rámečku dialogu
y	vertikální poloha ovládacího prvku vzhledem k rámečku dialogu.

Popis: Funkce u daného řídícího prvku v uživatelském dialogu nastavuje novou polohu vzhledem k levému hornímu rohu dialogu.

Omezení: Funkce má smysl pouze pokud již byl ovládací prvek s identifikátorem *id* vytvořen.

Viz: **DlgSetCtrlSize**
DlgSetCtrlText

DlgSetCtrlSize

```
procedure DlgSetCtrlSize(
  dlg_id, id, cx, cy : short);
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
id	identifikátor řídícího prvku
cx	horizontální rozměr řídícího prvku vzhledem k rámečku dialogu
cy	vertikální rozměr řídícího prvku vzhledem k rámečku dialogu

Popis: Funkce u daného řídícího prvku v uživatelském dialogu nastavuje jeho nové rozměry.

Omezení: Funkce má smysl pouze pokud byl již řídící prvek s identifikátorem *id* vytvořen.

Viz: **DlgSetCtrlPos**
DlgSetCtrlText

DlgSetCtrlText

```
function DlgSetCtrlText(
  dlg_id, id : short; var text : const string) : boolean;
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
id	identifikátor ovládacího prvku
text	text na ovládacím prvku

Popis: Funkce nastavuje nový text pro daný ovládací prvek.

Omezení: Funkce má smysl pouze pokud již byl ovládací prvek s identifikátorem *id* vytvořen.

Viz: **DlgSetCtrlPos**
DlgSetCtrlSize

DlgStdBox

```
function DlgStdBox(
  dlg_id, type : short;
  drop_down : boolean;
  id, x, y, cx,cy : short) ; boolean;
```

Parametry:

dlg_id	identifikátor uživatelského dialogu	
type	možné hodnoty:	1 Seznam barev
		2 Seznam stínování
		3 Seznam čar
drop_down	nastavuje, zda bude seznam rozbalovací	
id	identifikátor seznamu	
x	x-ová souřadnice levého horního rohu seznamu vzhledem k rámečku dialogu	
y	y-ová souřadnice levého horního rohu seznamu vzhledem k rámečku dialogu	
cx	vodorovný rozměr seznamu	
cy	svislý rozměr seznamu	

Popis: Umožňuje do uživatelského dialogu vložit standardní seznam z 602Text. Tento seznam je již plný a nelze do něj vkládat.

Viz: **DlgListBox**
DlgLineY
DlgRemoveCtrl
DlgComboBox
DlgText
DlgButton
DlgCheckBox
DlgGroupBox
DlgInputLine
DlgLineX

DlgStrBoxAdd

```
function DlgStrBoxAdd(
  dlg_id, id, str_id : short;
  var str : const string) : boolean;
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
id	identifikátor seznamu typu „listbox“ nebo seznamu typu „combobox“
str_id	identifikátor řetězce znaků
str	řetězec znaků

Popis: Funkce přidává do seznamu znakových řetězců v listboxu nebo comboboxu identifikovaném identifikátorem *id* nový řetězec s identifikátorem *str_id*. Pokud identifikátor *str_id* již předtím příslušel jinému řetězci v seznamu, je tento řetězec modifikován na novou hodnotu *str*. V případě comboboxu může mít parametr *str_id* i "žádnou" hodnotu NONESHORT. V tom případě hodnotou *str* nastavujeme inicializační obsah vstupní řádky comboboxu, který neodpovídá selekci žádného z přidaných řetězců. Aby byl tento stav po inicializaci dialogu navozen, je nutno také

nastavit tuto "žádnou" selekci funkcí **DlgStrBoxSetVal**, kde jako identifikátor řetězce použijeme hodnotu NONESHORT.

Viz: **DlgStrBoxDelete**
DlgStrBoxGetVal
DlgStrBoxSetVal
DlgStrBoxGetStr
DlgListBox
DlgComboBox

DlgStrBoxDelete

```
procedure DlgStrBoxDelete(
  dlg_id, id, str_id : short);
```

Parametry:
dlg_id identifikátor uživatelského dialogu
id identifikátor seznamu typu „listbox“ nebo typu „combobox“
str_id identifikátor znakového řetězce

Popis: Procedura vypouští ze seznamu znakových řetězců v listboxu (comboboxu) znakový řetězec specifikovaný identifikátorem *str_id*.

Poznámka: Pokud v daném listboxu (comboboxu) neexistuje řetězec s identifikátorem *str_id*, neprovede se nic.

Viz: **DlgStrBoxAdd**
DlgStrBoxGetVal
DlgStrBoxSetVal
DlgStrBoxGetStr
DlgListBox
DlgComboBox

DlgStrBoxGetStr

```
function DlgStrBoxGetStr(
  dlg_id, id, str_id : short) : string;
```

Parametry:
dlg_id identifikátor uživatelského dialogu
id identifikátor seznamu typu „listbox“ nebo typu „combobox“
str_id identifikátor znakového řetězce

Popis: Funkce vrací hodnotu znakového řetězce nacházejícího se v daném listboxu nebo comboboxu.

Poznámka: Pokud se v příslušném listboxu (comboboxu) nenachází řetězec, jehož identifikátor je *str_id*, funkce vrátí řetězec nulové délky.

Viz: **DlgStrBoxGetVal**
DlgStrBoxSetVal
DlgStrBoxAdd
DlgStrBoxDelete
DlgListBox
DlgComboBox

DlgStrBoxGetVal

```
function DlgStrBoxGetVal(
```

```
  dlg_id, id : short) : short;
```

Parametry: *dlg_id* identifikátor uživatelského dialogu
id identifikátor seznamu typu „listbox“ nebo typu „combobox“

Popis: Funkce vrací identifikátor toho znakového řetězce, který byl při opuštění dialogu v příslušném listboxu (comboboxu) selektován (u comboboxu byl tedy též obsahem vstupní řádky). Toto platí ovšem za předpokladu, že od opuštění dialogu nebyla funkcí **DlgStrBoxSetVal** nastavena hodnota jiná.

V případě comboboxu ovšem může být při opuštění dialogu obsahem vstupní řádky znakový řetězec, který neodpovídá selekci žádného z původně vložených řetězců, neboť je zapsán uživatelem. Tomuto stavu "žádné selekce" odpovídá návratová hodnota NONESHORT. Hodnotu příslušného znakového řetězce získáte voláním funkce **DlgStrBoxGetStr** (*dlg_id, id*, NONESHORT).

Poznámka: Návratová hodnota NONESHORT se uplatní pochopitelně též v případě, kdy příslušný listbox (combobox) byl prázdný.

Viz: **DlgStrBoxSetVal**
DlgStrBoxGetStr
DlgStrBoxAdd
DlgStrBoxDelete
DlgListBox
DlgComboBox

DlgStrBoxSetVal

```
procedure DlgStrBoxSetVal(
  dlg_id, id, str_id : short);
```

Parametry: *dlg_id* identifikátor uživatelského dialogu
id identifikátor seznamu typu „listbox“ nebo typu „combobox“
str_id identifikátor řetězce znaků

Popis: Funkce nastavuje v seznamu znakových řetězců v listboxu nebo comboboxu interní indikátor, označující, který znakový řetězec bude při vyvolání dialogu po jeho inicializaci selektován. V případě comboboxu může mít parametr *str_id* i "žádnou" hodnotu NONESHORT. V tom případě inicializační obsah vstupní řádky comboboxu nebude neodpovídat selekci žádného z přidaných řetězců. (Příslušnou hodnotu zadáte zavoláním funkce **DlgStrBoxAdd** s hodnotou identifikátoru stringu NONESHORT).

Viz: **DlgStrBoxGetVal**
DlgStrBoxGetStr
DlgStrBoxAdd
DlgStrBoxDelete
DlgListBox
DlgComboBox

DlgText

```
function DlgText(
  dlg_id : short;
  var text : const string;
  id, x, y, cx, cy : short) : boolean;
```

Parametry: *dlg_id* identifikátor uživatelského dialogu
text doprovodný text
id identifikátor ovládacího prvku
x horizontální poloha textu vzhledem k rámečku dialogu

	y	vertikální poloha textu vzhledem k rámečku dialogu
	cx	horizontální rozměr textu
	cy	vertikální rozměr textu
Implicitní hodnoty:	*cx* = 80 *cy* = 10	

Popis: Umožňuje vložit doprovodný text do uživatelského dialogu.

Viz: **DlgRemoveCtrl**
DlgTextSetAlign
DlgButton
DlgCheckBox
DlgGroupBox
DlgInputLine
DlgListBox
DlgComboBox

DlgTextSetAlign

```
procedure DlgTextSetAlign(
  dlg_id, id, value : short);
```

Parametry:	*dlg_id*	identifikátor uživatelského dialogu
	id	identifikátor textu
	value	nastavení zarovnání

Popis: Funkce nastaví zarovnání popisného textu:
0 – doleva
1 – uprostřed
2 – doprava

Viz: **DlgText**

DlgTextSetBmp

```
function DlgTextSetBmp(
  dlg_id, id : short;
  var filename : const string) : boolean;
```

Parametry:	*dlg_id*	identifikátor uživatelského dialogu
	id	identifikátor textu
	filename	jméno bitmapy

Popis: Umožňuje zobrazit bitmapu místo doprovodného textu.

Viz: **DlgText**
DlgSetCtrlBmp
DialogSetBmp

DocClose

```
function DocClose(
  save_type : short) : boolean;
```

Parametry:	*save_type*	specifikuje činnost při zavírání okna

Implicitní hodnota: *save_type* = kSaveNormal

Popis: Funkce zavírá okno s dokumentem; hodnota parametru *save_type* specifikuje činnost:

- *save_type* = **kSaveNormal -** upozorní uživatele na nutnost uložení dokumentu, který byl od posledního uložení změněn
- *save_type* = **kSaveAlways -** před uzavřením okna uloží dokument bez upozornění
- *save_type* = **kSaveNever -** dokument neuloží.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

DocMaximize

```
procedure DocMaximize;
```

Popis: Procedura maximalizuje okno s aktuálním dokumentem.

Omezení: Potřebuje aktuální dokument.

Viz: **IsDocMaximized**
IsDocMinimized
DocMinimize
DocRestore

DocMinimize

```
procedure DocMinimize(
  stay_active : boolean);
```

Parametry: *stay_active* zda má aktuální dokument zůstat aktivní

Implicitní hodnota: *stay_active* = FALSE

Popis: Procedura minimalizuje okno s aktuálním dokumentem:

- *stay_active* = **TRUE** - zůstane aktivní okno tohoto dokumentu (tj. ikona)
- *stay_active* = **FALSE -** aktivním se stane další okno v pořadí (tj. to, na které by "přišla řada" při normálním minimalizování myší anebo při přepínání na další okno kombinací kláves **Ctrl+F6**).

Omezení: Potřebuje aktuální dokument.

Viz: **IsDocMaximized**
IsDocMinimized
DocMaximize
DocRestore

DocRestore

```
procedure DocRestore;
```

Popis: Procedura obnoví okno s aktuálním dokumentem do původní velikosti.

Omezení: Potřebuje aktuální dokument.

Viz: **IsDocMaximized**
IsDocMinimized

DocMaximize
DocMinimize
DocRestore

DocSplit

```
procedure DocSplit)
  splittype, percent : short);
```

Parametry: *splittype* druh rozdělení okna
percent procentuální poměr rozdělení

Popis: Procedura nastavuje typ rozdělení okna a kvantitativní velikosti. K chybě dojde, pokud typ rozdělení nemá jednu z hodnot *stNoSplit, stHorSplit, stVertSplit.* Stejně tak, pokud neplatí 0 <= *percent* <= 100. Jinak se nastaví odpovídající rozdělení.

Viz: **GetSplitType**
GetSplitVal

DocWndMove

```
function DocWndMove(
  left, top : short) : boolean;
```

Parametry: *left* horizontální souřadnice
top vertikální souřadnice

Popis: Funkce nastaví nové souřadnice levého horního rohu okna s dokumentem vůči rodičovskému oknu. Souřadnice jsou relativní vůči levému rohu rodičovského okna, tj. okna 602Text a jsou zde vždy v bodech (nikoli v aktuálních jednotkách).

Omezení: Vyžaduje aktuální dokument.

Viz: **GetDocWndLeft**
GetDocWndTop

DoesBlockExist

```
function DoesBlockExis(
  var block_name : const string) : boolean;
```

Parametry: *block_name* jméno bloku

Popis: Funkce zjišťuje, existuje-li v aktuálním dokumentu uvedeného jména.

Poznámka: Funkce zohledňuje rozdíl mezi malými a velkými písmeny.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **EditBlocksDlg**
SelectBlock
LabelBlock
UnlabelBlock
BlocksCount
BlockName
CopyBlock

DoesMarkExist

```
function DoesMarkExist(
  var mark_name : const string) : boolean;
```

Parametry: *mark_name* jméno záložky

Popis: Funkce testuje, zda-li je v dokumentu nastavena záložka *mark_name*.

Poznámka: Funkce zohledňuje rozdíl mezi malými a velkými písmeny.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **EditMarksDlg**
GoToMark
InsertMark
ClearMark
MarkName
MarksCount

DoesObjectExist

```
function DoesObjectExist(
  id : integer) : boolean
```

Parametry: *id* identifikátor objektu

Popis: Funkce zjišťuje, zda existuje objekt daného *id.*

Viz: **ObjectListDlg**
ObjectId
ObjectsCount
ObjectType
SelectObject
UnselectObject
ObjectSelCount
ObjectSelId
ObjectSelModify
ObjectIsSelected

DoesWindowExist

```
function DoesWindowExist(
  var title : const string;
  case_senzitive : boolean) : boolean;
```

Parametry: *tittle* titulní řetězec
case_senzitive rozlišovat malá/velká písmena

Implicitní hodnota: *case_senzitive* = FALSE

Popis: Funkce hledá okno dokumentu, které má v titulku řetězec *title*. Druhý parametr *case_senzitive* udává, zda se má při prohlížení rozlišovat mezi malými a velkými písmeny.

Viz: **SwitchToWindow**
NextWindow
PreviousWindow
CountWindows
GetDocWndTittle
GetDocFileName

EditBlocksDlg

```
function EditBlocksDlg : short;
```

Popis: Funkce volá dialog **Pojmenované bloky** a vrací IDGOTO, IDINSERT, IDCANCEL (IDINSERT zde má význam **Pojmenovat**).

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **SelectBlock**
LabelBlock
UnlabelBlock
DoesBlockExist
BlocksCount
BlockName
CopyBlock

EditMarksDlg

```
function EditMarksDlg : short;
```

Popis: Funkce vyvolá dialog **Záložky**; vrací IDGOTO, IDINSERT, IDCANCEL **.**

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GoToMark**
InsertMark
DoesMarkExist
ClearMark
MarkName
MarksCount

Eof

```
function Eof(
  f : file) : boolean;
```

Parametry: *f* proměnná typu soubor (file)

Popis: Funkce zjišťuje, zda soubor *f* byl dočten do konce. Tento soubor musí být otevřen. Vedlejším efektem funkce je přeskočení mezer a oddělovačů řádek před testováním konce souboru.

Hodnota: Funkce vrátí TRUE, pokud je soubor dočten do konce nebo pokud nepřečtený zbytek souboru sestává v okamžiku volání **Eof** pouze z mezer a oddělovačů řádek. Jinak vrátí FALSE.

Viz: **Read**
Write
Writeln
Reset
Rewrite
Seek
Close
Filelenght

ErasethisMacro

```
procedure ErasethisMacro;
```

Popis: Speciální procedura způsobující, že po ukončení makra se toto vypustí ze seznamu spustitelných maker. K tomuto vypuštění dojde i po předčasném ukončení běhu makra (**Ctrl+Break**), byla-li funkce zavolána ještě před tímto ukončením. Možné využití je např. pro makra, u kterých je žádoucí, aby proběhla právě jen jednou.

Viz: **SetInitialOpen**

Exec

```
function Exec(
  var pathname, params : const string) : short;
```

Parametry: *pathname* cesta a jméno souboru obsahujícího program
params parametry, které se programu předají na příkazové řádce

Popis: Funkce spustí program uložený v souboru *pathname* a předá mu parametry *params*. Tento program bude prováděn souběžně s 602Text. Funkce je implementována tak, že volá systémovou funkci **WinExec**.

Hodnota: Funkce vrátí totéž, co vrátí systémová funkce **WinExec**. Bližší údaje viz systémové manuály Windows.

ExeFileName

```
function ExeFileName : string;
```

Popis: Vrací jméno souboru spuštěné aplikace (`602TEXT.EXE`) včetně úplné cesty, tj. například:
`C:\602TEXT\EXEC\602TEXT.EXE`.

Exp

```
function Exp(
  r : real) : real;
```

Parametry: *r* reálné číslo

Popis: Funkce počítá přirozenou exponenciálu, tedy hodnotu *r*-té mocniny základu přirozených logaritmů.

Hodnota: Funkce vrátí hodnotu exponenciály v bodě *r*.

FieldInsertDlg

```
function FieldInsertDlg : short;
```

Popis: Volá dialog **Vložit pole**. Pokud bulo nějaké pole pomocí tohoto dialogu vybráno a vloženo vrací IDOK; jinak vrací IDCANCEL.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **FieldSimple**
InsertFootNote
FieldHTML
FieldSuma

FieldHTML

```
function FieldHTML(
  dlg : boolean;
  type : short;
  var str1, str2, str3 : const string) : short;
```

Parametry:

	boolean	volat dialog
	type	typ pole: **– 1** kHTMLMark **– 2** kHTMLSymbol **– 3** kHTMLAddr
	str1, *str2*, *str3*	řetězce inicializující příslušný typ pole

Implicitní hodnoty:

str1= ""	(prázdný řetězec)
str2 = ""	(prázdný řetězec)
str3 = ""	(prázdný řetězec)

Popis: Vkládá příslušný typ pole. Pro `kHTMLMark` a `kHTMLSymbol` má smysl pouze první parametr (řetězec) – tento text je obsahem HTML značky i symbolu. Pro `kHTMLAddr` mají parametry následující význam:

- text, který se zobrazuje jako obsah pole
- URL odkaz
- další atributy pole.

Implicitně jsou další parametry (řetězce) prázdné.

Při vložení pole vrací IDOK, jinak IDCANCEL.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **FielInsertDlg**
FieldSimple
FieldSuma
InsertFootNote
GetObjectHTMLStr

FieldSimple

```
function FieldSimple(
  field_type : short) : boolean;
```

Parametry: *field_type* typ vkládaného pole

Popis: Vkládá do aktuálního dokumentu některý z tzv. jednoduchých fieldů tj. takový, který nevyžaduje zadat dodatečná data. Jedná se o tyto druhy polí:

hodnota *field_type*	činnost v aktuálním dokumentu
1	vloží číslo stránky
2	vloží číslo kapitoly

3	vloží čas
4	vloží datum
5	vloží aktuální časový údaj zahájení tisku
6	vloží aktuální datum zahájení tisku
7	vloží nadpis
8	vloží obsah
9	vloží jméno autora
10	vloží klíčové slovo
11	vloží poznámku
12	vloží šablonu
13	vloží jméno aplikace
14	vloží název souboru

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **FieldInsertDlg**
InsertFootNote
FieldHTML
FieldSuma

FieldSuma

```
function FieldSuma(
  dlg, horizontal : boolean;
  from : short) : short;
```

Parametry:

dlg	volat dialog
horizontal	součet v řádce
from	od jakého sloupce (event. řádky)

Implicitní hodnoty: *from* = 1

Popis: Vkládá do pole **Součet** (ve sloupci i řádce) textové tabulky. Při vložení pole vrací IDOK; jinak vrací IDCANCEL.

Viz: **FieldInsertDlg**
FieldSimple
FieldHTML
InsertFootNote

Filelenght

```
function Filelenght(
  f : file) : integer;
```

Parametry: *f* proměnná typu soubor (file)

Popis: Funkce zjišťuje délku souboru *f*. Tento soubor musí být v okamžiku volání funkce otevřen.

Hodnota: Funkce vrátí délku souboru *f* v bajtech. Dojde-li k chybě (např. když soubor není otevřen), funkce vrátí hodnotu -1.

Viz: **Read**
Write
Writeln
Reset
Rewrite
Close
Eof
Seek

FindClose

```
procedure FindClose(handle : integer);
```

Parametry: *handle* identifikační číslo (vráceno z funkce **FindFirstFile**)

Popis: Ukončí vyhledávání spuštěné funkcí **FindFirstFile**.

Příklad:

```
program ff;
var
 handle:integer;
 szFile:string[200];
begin
 handle := FindFirstFile("c:\*.*", szFile);
 info_box('handle',int2str(handle));
 info_box('První nalezený soubor', szFile);
 while(FindNextFile(handle,szFile)) do begin
  info_box('Další nalezený soubor',szFile);
 end;
 FindClose(handle);
   end.
```

Viz: **FindFirstFile**
FindNextFile

FindFirstFile

```
function FindFirstFile(
  var path_name : const string;
  var file_name : string) : integer;
```

Parametry: *path_name* hledaný soubor, lze použít znaky '*' a '?'
file_name první nalezený soubor

Popis: Najde první soubor zadaný v *path_name*, výsledek vloží do *file_name*.

Hodnota: Identifikační číslo, které se použije při volání funkce **FindNextFile** a **FindClose .**

Příklad:

```
program ff;
var
 handle:integer;
 szFile:string[200];
begin
 handle := FindFirstFile("c:\*.*", szFile);
 info_box('handle',int2str(handle));
 info_box('První nalezený soubor', szFile);
 while(FindNextFile(handle,szFile)) do begin
  info_box('Další nalezený soubor',szFile);
 end;
 FindClose(handle);
end.
```

Viz: **FindNextFile**
FindClose

FindNextFile

```
function FindNextFile(
  handle : integer;
```

```
  var file_name : string) : boolean;
```

Parametry: *handle* identifikační číslo (vráceno z funkce **FindFirstFile**)
file_name nalezený soubor

Popis: Najde soubor zadaný v *path_name* při volání funkce **FindFirstFile**.

Hodnota: TRUE = bylo-li hledání úspěšné

Příklad:

```
program ff;
var
 handle:integer;
 szFile:string[200];
begin
 handle := FindFirstFile("c:\*.*", szFile);
 info_box('handle',int2str(handle));
 info_box('První nalezený soubor', szFile);
 while(FindNextFile(handle,szFile)) do begin
  info_box('Další nalezený soubor',szFile);
 end;
 FindClose(handle);
end.
```

Viz: **FindFirstFile**
FindClose

FixScreenPos

```
function FixScreenPos : boolean;
```

Popis: Funkce nastaví zobrazení okna tak, aby byl řádkový kurzor viditelný. Vrací TRUE, pokud se obsah okna pohnul.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

FontName

```
function FontName(
  index : short) : string;
```

Parametry: *index* index fontu

Popis: Funkce vrací název jednoho z aktuálně dostupných fontů. Fonty jsou očíslovány od jedné do **CountFonts**. Při použití indexu mimo tento interval funkce vrátí prázdný řetězec.

Viz: **CountFonts**
CurrentFont
UseFont
UseFontOfName

FormatFontDlg

```
function FormatFontDlg : short;
```

Popis: Funkce zajišťuje vyvolání dialogu **Písmo**. Vrací IDOK**,** IDCANCEL.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **SetFormatFont**
GetFormatFont
ColorDlg

FormatParaDlg

```
function FormatParaDlg : short;
```

Popis: Funkce odpovídá položce menu **Formát** příkaz **Odstavec**. Funkce po zavolání dialogu vrací IDOK (**OK**), IDCANCEL (**Esc**), IDERROR (dialog je opuštěn jako **OK**, ale nebylo dost paměti na příslušné změny v textu).

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

FormatSectDlg

```
function FormatSectDlg : short;
```

Popis: Funkce odpovídá položce menu **Formát** příkaz **Sekce a sloupce**. Po zavolání dialogu vrací IDOK (**OK**) nebo IDCANCEL (**Esc**).

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **FormatFontDlg**
FormatParaDlg

FrameMarginsOn

```
function FrameMarginsOn : boolean;
```

Popis: Funkce zjišťuje, zda-li jsou zobrazeny okraje textových rámců.

Poznámka: Nemá-li žádný dokument nebo je-li aktuální dokument v zobrazení osnovy, vrací FALSE.

Viz: **ViewFrameMargins**

GetAlignType

```
function GetAlignType : short;
```

Popis: Funkce zjišťuje aktuální hodnotu zarovnání písma (doleva, na střed ...).

Hodnota: Funkce vrací podle druhu zarovnání hodnotu některé ze standardních konstant *alLeft, alCenter, alRight, alJustify* nebo *kAmbiguous*.

Poznámka: Viz Poznámka 4.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **IsAlignLeft**
IsAlignCenter
IsAlignRight
IsAlignJustify
SetAlign

GetBottomMargin

```
function GetBottomMargin : real;
```

Popis: Funkce vrací okamžité nastavení spodního okraje.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **SetMargins**
GetLeftMargin
SetLeftMargin
GetRightMargin
SetRightMargin
GetTopMargin
SetTopMargin
SetBottomMargin

GetCaretCell

```
function GetCaretCell : integer;
```

Popis: Vrací číslo sloupce v textové tabulce, ve které je umístěn kurzor.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru umístěném v textové tabulce.

Viz: **CaretCellStart**
GetCaretRow
PrevCell
NextCell
PrevRow
NextRow
GetNumCells
GetNumRows

GetCaretLine

```
function GetCaretLine(
 continually : boolean) : short;
```

Parametry: *continually* uvažovat kontinuální číslování řádek

Implicitní hodnota: *continually* = TRUE

Popis: Funkce vrací číslo řádky, na které stojí řádkový kurzor.
Je-li parametr ***continually*** = **TRUE** vrácená hodnota odpovídá kontinuálnímu číslování řádek, tj. řádky jsou číslovány od prvního řádku první strany (čísla řádek odpovídající tomuto číslování jsou na stavovém řádku zobrazována v režimu editace makra).

Je-li parametr ***continually*** = **FALSE** číslování řádek začíná vždy od počátku každé stránky.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Příklad:
```
if IsMacro then j : = GetCaretLine
          else j : = GetCaretLine(FALSE) ;
```

Viz: **GetSelStartLine**
GetSelEndLine

GetCaretObjectId

```
function GetCaretObjectId : integer;
```

Popis: Pokud se řádkový kurzor právě nachází v objektu (textovém rámci nebo textové tabulce), funkce vrací identifikátor tohoto objektu. Jinak je návratová hodnota NONEINTEGER.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **ObjectsCount**
ObjectId
ObjectType

GetCaretPage

```
function GetCaretPage(
  global : boolean) : short;
```

Parametry: *global* uvažovat číslování stránek nezávisle na nastavení
Implicitní hodnota: *global* = TRUE

Popis: Funkce vrací číslo stránky, na které stojí řádkový kurzor.
Je-li parametr ***global*** = ***implicitní hodnota*** **TRUE** – nastavení číslování stránek nemá vliv na vrácenou hodnotu; tj. pro první stranu od začátku dokumentu je vždy návratová hodnota 1, pro druhou 2, atd.
Je-li parametr ***global*** **= FALSE** – návratová hodnota odpovídá aktuálně nastavenému číslování stránek.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetSelEndPage**
GetSelStartPage

GetCaretPos

```
function GetCaretPos : integer;
```

Popis: Funkce udává pořadí znaku, před kterým stojí řádkový kurzor (znaky jsou číslovány od 0), neboli také počet znaků, které jsou před řádkovým kurzorem. Tato konvence v číslování platí i pro funkce **GetSelEndPos, GetSelStartPos**. Je nutno mít na zřeteli, že do celkového počtu znaků v textu se započítávají i znaky v poznámkách pod čarou.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetSelEndPos**
GetSelStartPos

GetCaretPosType

```
function GetCaretPosType : short;
```

Popis: Funkce dle typu pozice řádkového kurzoru vrací některou ze standardních konstant:

NONESHORT	není otevřen žádný dokument
cptTextFrame	textový rámec
cptFootnote	poznámka pod čarou
cptTable	tabulka
cptNormal	normální text

Viz: **GetCaretPos**
GetSelStartPos
GetSelEndPos

GetCaretRow

```
function GetCaretRow : integer;
```

Popis: Vrací číslo řádku v textové tabulce, ve které je umístěn kurzor.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru umístěném v textové tabulce.

Viz: **CaretCellStart**
GetCaretCell
PrevCell
NextCell
PrevRow
NextRow
GetNumCells
GetNumRows

GetCaretSection

```
function GetCaretSection : short;
```

Popis: Funkce vrací číslo sekce, ve které se nachází řádkový kurzor.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetCaretLine**
GetCaretPage

GetCommandLine

```
function GetCommandLine : string;
```

Popis: Funkce vrací hodnotu příkazové řádky, z níž byl 602Text spuštěn.

GetDocFileName

```
function GetDocFileName : string;
```

Popis: Funkce vrací řetězec – název souboru (včetně přístupové cesty) odpovídající aktivnímu dokumentu. Pokud byl dokument vytvořen pomocí příkazu **Nový** a dosud nebyl uložen (ani automaticky), vrací název souboru, do kterého bude automaticky uložen (dle nastavení v předvolbách).

Omezení: Potřebuje aktuální dokument.

Viz: **IsDocDefaul**
GetDocWndTittle

GetDocWndHeight

```
function GetDocWndHeight : short;
```

Popis: Funkce vrací výšku okna s dokumentem. Souřadnice jsou zde vždy v bodech (nikoli v aktuálních jednotkách).

Omezení: Vyžaduje aktuální dokument.

Viz: **SetDocWndHeight**
GetDocWndTop
GetDocWndLeft
GetDocWndWidth

GetDocWndLeft

```
function GetDocWndLeft : short;
```

Popis: Vrací souřadnici – horizontální vzdálenost levého horního rohu okna s dokumentem vůči levému hornímu rohu jeho rodičovského okna (tj. okna aplikace 602Text). Výsledné souřadnice jsou relativní k rodičovskému oknu – zde jsou vždy v bodech (nikoli v aktuálních jednotkách).

Omezení: Vyžaduje aktuální dokument.

Viz: **GetDocWndTop**
GetDocWndWidth
GetDocWndHeight

GetDocWndTittle

```
function GetDocWndTittle : string;
```

Popis: Funkce vrací řetězec vypsaný v záhlaví aktivního okna.

Omezení: Potřebuje aktuální dokument.

Viz: **GetDocFileName**

GetDocWndTop

```
function GetDocWndTop : short;
```

Popis: Vrací souřadnici – vertikální vzdálenost levého horního rohu okna s dokumentem vůči levému hornímu rohu jeho rodičovského okna (tj. okna aplikace 602Text).
Výsledné souřadnice jsou relativní k rodičovskému oknu; zde jsou vždy v bodech (nikoli v aktuálních jednotkách).

Omezení: Vyžaduje aktuální dokument.

Viz: **GetDocWndLeft**
GetDocWndWidth
GetDocWndHeight

GetDocWndWidth

function `GetDocWndWidth` : short;

Popis: Funkce vrací šířku okna s dokumentem – zde vždy v bodech (nikoli v aktuálních jednotkách).

Omezení: Vyžaduje aktuální dokument.

Viz: **SetDocWndWidth**
GetDocWndTop
GetDocWndLeft
GetDocWndHeight

GetFormatFont

```
function GetFormatFont(
  kind : short) : short;
```

Parametry: *kind* druh zjišťované vlastnosti písma

Popis: Funkce vrací vlastnosti písma. Hodnota parametru *kind* může být jednou z těchto standardních konstant:

***kind* = kCHPsize** - vrátí velikost písma (v bodech) nebo hodnotu *kAmbiguous* (označeno písmo různých velikostí).

kind = kCHPbold - vrátí kOff, kOn, kAmbiguous.

kind = kCHPitalic - vrátí kOff, kOn, kAmbiguous.

***kind* = kCHPunderline** - podle druhu podtržení vrací :

0	bez podtržení
1	vše podtrženo
2	slova
kAmbiguous	označeno písmo různého typu podtržení

***kind* = kCHPsupersub** - podle druhu písma (normální / superscript / subscript) vrací hodnoty těchto standardních konstant:

0 = tSNormal	normální text
1= tSuper	superscript (horní index)
2= tSub	subscript (dolní index)
kAmbiguous	nelze rozhodnout

***kind* = kCHPcaps** - podle toho, zda má písmo vlastnost "všechna písmena velká", vrací hodnoty *kOn, kOff, kAmbiguous.*

Pro jiné hodnoty parametru *kind* vznikne běhová chyba.

Poznámka: Viz Poznámka 2.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **SetFormatFont**

GetFormatPara

```
function GetFormatPara(
  kind : short) : real;
```

Parametry: *kind* druh zjišťované vlastnosti odstavce

Popis: Funkce zjišťuje vlastnost odstavce, resp. vybraných odstavců textu. Zjišťovaná vlastnost je dána parametrem *kind* (viz následující výčet). Pokud je vybráno více odstavců, z nichž ne všechny mají pro zjišťovanou vlastnost tutéž hodnotu (např. pro *kind = kPPAlign* nemají všechny stejný typ zarovnání), funkce vrátí "žádnou" hodnotu NONEREAL.

kind = kPPAlign - funkce vrací typ zarovnání odstavce. Návratové hodnoty jsou celočíselné konstanty *alLeft*, *alCenter*, *alRight*, *alJustify* anebo NONEREAL.

kind = kPPLeftIndent - funkce vrací hodnotu odsazení odstavce nalevo v aktuálních jednotkách (centimetry, palce, atd.); za podmínek popsaných výše vrací NONEREAL.

kind = kPPRightIndent - funkce vrací hodnotu odsazení odstavce napravo v aktuálních jednotkách (centimetry, palce atd.), případně NONEREAL.

kind = kPPFirstIndent - funkce vrací hodnotu odsazení první řádky odstavce v aktuálních jednotkách, případně NONEREAL.

kind = kPPColSpace - návratovou hodnotou je velikost mezery mezi sloupci v aktuálních jednotkách, případně NONEREAL.

kind = kPPMaxCol - návratovou hodnotou je maximální počet sloupců odstavce, případně NONEREAL.

kind = kPPUpperSpace - funkce vrací místo před odstavcem v aktuálních jednotkách, resp. NONEREAL.

kind = kPPLowerSpace - funkce vrací místo za odstavcem v aktuálních jednotkách, resp. NONEREAL.

kind = kPPShade - funkce vrací celočíselnou hodnotu od 0 do 19 včetně, charakterizující typ stínování. Jednotlivé hodnoty mají tentýž význam jako ve funkci **SetFormatPara**. Jinou možnou návratovou hodnotou je NONEREAL (viz výše).

kind = kPPLine - funkce vrací celočíselnou hodnotu od 0 do 11 včetně, charakterizující typ čar kolem odstavce. Jednotlivé hodnoty mají tentýž význam jako ve funkci **SetFormatPara**. Jinou možnou návratovou hodnotou je NONEREAL (viz výše).

kind = kPPBorder - funkce vrací nezáporné celé číslo mezi 0 a 15 včetně, jehož hodnota charakterizuje typ orámování odstavce, tj. přítomnosti/nepřítomnosti jednotlivých čar kolem odstavce. Přítomnost/nepřítomnost jednotlivých čar charakterizuje nastavení/nenastavení jednotlivého bitu v dvojkovém vyjádření návratové hodnoty s tímto přiřazením:

nejnižší bit $(0001)_2$ čára nalevo
druhý nejnižší bit $(0010)_2$ čára napravo
třetí nejnižší bit $(0100)_2$ čára nahoře
čtvrtý nejnižší bit $(1000)_2$ čára dole

Například:
$11 = (1011)_2$ čára je dole, napravo a vlevo

Jinou možnou návratovou hodnotou je NONEREAL (viz výše).

kind = kPPLevel - slouží pro zjištění úrovně osnovy. Návratové hodnoty jsou buďto NONEREAL anebo nezáporná celá čísla mezi 0 a 9 včetně; přitom hodnota 0 má význam "všechny".

kind = kPPNumber - návratová hodnota charakterizuje druh číslování osnovy; je to buďto NONEREAL anebo některé z těchto nezáporných celých čísel:

0	žádné
1	legální (1.1 apod.)
2	osnova (1,A apod.)

kind = kPPLeading - vrací vzdálenost řádek v procentech (celé číslo mezi 100 a 240 včetně) anebo NONEREAL.

kind = kPPHyphenation - vrací druh dělení slov. Návratové hodnoty jsou buďto tato nezáporná celá čísla:

0	žádné
1	všechny řádky
2	ob jeden řádek
3	ob dva řádky
4	anebo NONEREAL (viz výše).

kind = kPPBorderOffset - vrací oddělení (vzdálenost) písma a čar kolem odstavce (v aktuálních jednotkách) anebo NONEREAL.

kind = kPPSubPageHeight - vrací výšku sloupců (v aktuálních jednotkách) anebo NONEREAL.

kind = kPPColInc - vrací přírůstek sloupce (v aktuálních jednotkách) anebo NONEREAL.

kind = kPPBullets - jedná se o vlastnost odstavec - odrážky. Pro návratovou (resp. nastavovanou) hodnotu platí tyto možnosti:

0	žádné odrážky
1 – 6	odrážky dle pořadí v dialogu (rozepsat)
100 – 104	číslování dle pořadí v dialogu (rozepsat)

kind = kPPSkipBullets - vrací buďto NONEREAL (viz výše) anebo příznak indikující, mají-li se nebo nemají použít odrážky. Jeho možnými hodnotami jsou tato celá čísla:

kOn = 1	použít
kOff = 0	nepoužít

kind = kPPRulLine - vrací buďto NONEREAL anebo celočíselnou hodnotu mezi 0 až 11 včetně, charakterizující typ čáry mezi sloupci. Jednotlivé hodnoty mají tentýž význam jako pro *kind = kPPLine* (typ čar kolem odstavce).

Pro jiné hodnoty parametru *kind* nastane běhová chyba.

Poznámka: Viz Poznámka 3.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **FormatParaDlg**
SetFormatPara
SetFormatFont
GetFormatFont
SetFormatSect
GetFormatSect

GetFormatSect

```
function GetFormatSect(
  kind : short) : real;
```

Parametry: *kind* druh zjišťované vlastnosti sekce

Popis: Funkce zjišťuje vlastnost sekce, resp. vybraných sekcí textu. Zjišťovaná vlastnost je dána parametrem *kind* (viz následující výčet). Pokud je vybráno více sekcí, z nichž ne všechny mají pro zjišťovanou vlastnost tutéž hodnotu (např. pro *kind = kSPColHeight* nemají všechny stejnou výšku sloupců), funkce vrátí "žádnou" hodnotu NONEREAL.

kind = kSPColSpace - návratovou hodnotou je velikost mezery mezi sloupci v aktuálních jednotkách, případně NONEREAL.

kind = kSPMaxCol - návratovou hodnotou je maximální počet sloupců odstavce, případně NONEREAL.

kind = kSPColHeight - vrací výšku sloupců (v aktuálních jednotkách), případně NONEREAL.

kind = kSPColInc - vrací přírůstek sloupce (v aktuálních jednotkách), případně NONEREAL.

kind = kSPColLine - vrací buďto NONEREAL anebo celočíselnou hodnotu mezi 0 až 11 včetně, charakterizující typ čáry mezi sloupci. Jednotlivé hodnoty mají tentýž význam jako pro *kind = kPPLine* ve funkci **GetFormatPara** (typ čar kolem odstavce).

Pro jiné hodnoty parametru *kind* nastane běhová chyba.

Poznámka: Viz Poznámka 3.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetFormatPara**
SetFormatPara
GetFormatFont
SetFormatFont
SetFormatSect

GetLastCode

```
function GetLastCode : boolean;
```

Popis: Vrátí poslední typ kódování použitý při ukládání nebo otevírání souboru.

Viz: **OpenFile**
SaveFile
SaveFileAs

GetLeftMargin

```
function GetLeftMargin : real;
```

Popis: Funkce vrací okamžitou hodnotu levého okraje.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **SetMargins**
SetLeftMargin
GetRightMargin
SetRightMargin
GetTopMargin
SetTopMargin
GetBottomMargin
SetBottomMargin

GetNewWinState

```
function GetNewWinState : boolean;
```

Popis: Funkce vrací stav tlačítka **Nové okno**.

Viz: **SetNewWinState**

GetNumCells

```
function GetNumCells : integer;
```

Popis: Vrací počet řádků textové tabulky, ve kterém je umístěn kurzor.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru umístěném v textové tabulce.

Viz: **CaretCellStart**
GetCaretCell
GetCaretRow
PrevCell
NextCell
PrevRow
NextRow
GetNumRows

GetNumRows

```
function GetNumRows : integer;
```

Popis: Vrací počet řádků textové tabulky.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru umístěném v textové tabulce.

Viz: **CaretCellStart**
GetCaretCell
GetCaretRow
PrevCell
NextCell
PrevRow
NextRow
GetNumCells

GetObjectHTMLStr

```
function GetObjectHTMLStr(
  id : integer) : string;
```

Parametry: *id* identifikátor objektu

Popis: Funkce vrací specielní HTML atribut daného objektu.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **SetObjectHTMLStr**
FieldHTML

GetOutlineLevel

```
function GetOutlineLevel : short;
```

Popis: Funkce vrací pro aktuální odstavec úroveň osnovy 0 – 9; 0 = všechny.

Omezení: Potřebuje neminimalizovaný aktuální dokument v zobrazení osnovy.

Viz: **SetOutlineLevel**

GetPageHeight

```
function GetPageHeight : real;
```

Popis: Funkce vrací výšku stránky v jednotkách, jaké jsou nastaveny v předvolbách.

Omezení: Potřebuje aktuální dokument.

Viz: **GetPageWidth**
GetPageType
IsPageLandscape
IsPagePortrait
PageFormat

GetPageType

```
function GetPageType : short;
```

Popis: Funkce vrací nastavený typ stránky dokumentu, přičemž vrácené hodnoty mají význam:

0 A4
1 Dopis
2 Legal
3 Nastavená uživatelem

Omezení: Potřebuje aktuální dokument.

Viz: **GetPageWidth**
GetPageHeight
IsPageLandscape
IsPagePortrait
PageFormat

GetPageWidth

```
function GetPageWidth : real;
```

Popis: Funkce vrací šířku stránky v jednotkách, jaké jsou nastaveny v předvolbách.

Omezení: Potřebuje aktuální dokument.

Viz: **GetPageHeight**
GetPageType
IsPageLandscape

IsPagePortrait
PageFormat

GetPreferences

```
function GetPreferences(
  kind : short) : short;
```

Parametry: *kind* druh zjišťované hodnoty

Popis: Funkce vrací hodnoty nastavené v dialogu **Předvolba** (menu **Pomůcky**):

pro *kind* = kPREF_DEV_FONTS – **Nabízet písma tiskáren**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE

pro *kind* = kPREF_B_HLP – **Zobrazovat "bublinovou" nápovědu**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE

pro *kind* = kPREF_SUPP_SUMM – **Nabízet popis při ukládání**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE

pro *kind* = kPREF_S_WINDOWS – **Ukládat rozložení oken**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE

pro *kind* = kPREF_TEMPL_NEW – **Nabízet šablony pro nové dokumenty**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE

pro *kind* = kPREF_AUTOSAVE – **Automatické ukládání**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE

pro *kind* = kPREF_AUTO_INT – **Automatické ukládání (interval)**: pro hodnoty value od 0 do 99 tuto hodnotu použije pro nastavení intervalu ukládání a vrací TRUE; pro jiné hodnoty se neprovede nic a funkce vrací FALSE

pro *kind* = kPREF_IMG_DITHER **– Výpočet odstínů šedi pro tisk obrázků**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE

pro *kind* = kPREF_GLOB_UNITS – **Jednotky**: pro hodnoty (1<= value) && (value <=5): 1– palce, 2 – cm, 3 – body, 4 – pica, 5 – dekadické palce funkce vrací TRUE; pro jiné hodnoty parametru value je bez efektu a vrací FALSE

pro *kind* = kPREF_HORIZONTAL – **Preferovat vodorovné uspořádání mozaiky**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE.

Pro jiné než vyjmenované hodnoty parametru *kind* vrací 0.

Viz: **PreferencesDlg**
SetPreferences
GetPrefFootnStr
SetPrefFootnStr

GetPrefFax

```
function GetPrefFax : string;
```

Popis: Vrací jméno položky databáze, kde se bude při faxování se slučováním hledat faxové číslo adresáta.

Viz: **SetPrefFax**
GetPrefFaxName
SetPrefFaxName
GetPrefFootnStr
SetPrefFootnStr

GetPrefFaxName

```
function GetPrefFaxName : string;
```

Popis: Vrací jméno položky databáze, kde se bude při faxování se slučováním hledat jméno adresáta.

Viz: **GetPrefFax**
SetPrefFax
SetPrefFaxName
GetPrefFootnStr
SetPrefFootnStr

GetPrefFootnStr

```
function GetPrefFootnStr : string;
```

Popis: Návratová hodnota funkce je řetězec, jenž je implicitní značkou poznámky pod čarou (tato položka předvoleb se volí v menu **Pomůcky** příkazem **Předvolby**).

Viz: **SetPrefFootnStr**
GetPreferences
SetPreferences
PreferencesDlg
GetPrefFax
SetPrefFax
GetPrefFaxName
SetPrefFaxName

GetRGBText

```
function GetRGBText(
  var colorref : integer) : boolean;
```

Parametry: *colorref* výsledná barevná kombinace

Popis: Funkce slouží ke zjištění barvy textu na aktuální pozici řádkového kurzoru, event.označeného bloku textu. Pokud barva nemůže být zjištěna (je označen vícebarevný blok textu), funkce vrací FALSE a hodnotu výstupního parametru nezmění. Jinak vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při zobrazeném textovém kurzoru.

Viz: **SetRGBText**
SetRGBColor
RGB
RgbTrio

GetRightMargin

```
function GetRightMargin : real;
```

Popis: Funkce vrací okamžitou hodnotu pravého okraje.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **SetMargins**
GetLeftMargin
SetLeftMargin
SetRightMargin
GetTopMargin
SetTopMargin
GetBottomMargin
SetBottomMargin

GetScale

```
function GetScale : short;
```

Popis: Funkce vrací v procentech nastavené zvětšení.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **SetScale**
SetScaleFullPage
ManualScaleDlg

GetSelEndLine

```
function GetSelEndLine(
  continually : boolean) : short;
```

Parametry: *continually* uvažovat kontinuální číslování řádek
Implicitní hodnota: *continually* = TRUE

Popis: Funkce vrací číslo řádky konce označeného bloku, resp. řádkového kurzoru, není-li žádný blok v textu označen.
Je-li parametr ***continually*** = **TRUE** vrácená hodnota odpovídá kontinuálnímu číslování řádek; tj. řádky jsou číslovány od prvního řádku první strany (čísla řádek odpovídající tomuto číslování jsou na stavovém řádku zobrazována v režimu editace makra).
Je-li parametr ***continually*** = **FALSE** číslování řádek začíná vždy od počátku každé stránky.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetCaretLine**
GetSelStartLine

GetSelEndPage

```
function GetSelEndPage(
  global : boolean) : short;
```

Parametry: *global* uvažovat číslování stránek nezávisle na nastavení

Implicitní hodnota: *global* = TRUE

Popis: Funkce vrací číslo stránky konce označeného bloku, resp. řádkového kurzoru, není-li žádný blok v textu označen.

Je-li parametr ***global*** = ***implicitní hodnota*** **TRUE** nastavení číslování stránek nemá na vrácenou hodnotu vliv; tj. pro první stranu od začátku dokumentu je vždy návratová hodnota 1, pro druhou 2, atd.
Je-li parametr ***global*** = **FALSE** návratová hodnota odpovídá aktuálně nastavenému číslování stránek.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetCaretPage**
GetSelStartPage

GetSelEndPos

```
function GetSelEndPos : integer;
```

Popis: Funkce vrací pozici konce označeného bloku, resp. řádkového kurzoru, není-li žádný blok v textu označen.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetCaretPos**
GetSelStartPos

GetSelStartLine

```
function GetSelStartLine(
  continually : boolean) : short;
```

Parametry: *continually* uvažovat kontinuální číslování řádek

Implicitní hodnota: *continually* = TRUE

Popis: Funkce vrací číslo řádky počátku označeného bloku, resp. řádkového kurzoru, není-li žádný blok v textu označen.
Je-li parametr ***continually*** = **TRUE** vrácená hodnota odpovídá kontinuálnímu číslování řádek; tj. řádky jsou číslovány od prvního řádku první strany (čísla řádek odpovídající tomuto číslování jsou na pevné liště zobrazována v režimu editace makra).
Je-li parametr ***continually*** = **FALSE** číslování řádek začíná vždy od počátku každé stránky.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetCaretLine**
GetSelEndLine

GetSelStartPage

```
function GetSelStartPage(
  global : boolean) : short;
```

Parametry: *global* uvažovat kontinuální číslování stránek

Implicitní hodnota: *global* = TRUE

Popis: Funkce vrací číslo stránky začátku označeného bloku, resp. řádkového kurzoru, není-li žádný blok v textu označen.

Je-li parametr ***global*** = ***implicitní hodnota*** **TRUE** nastavení číslování stránek nemá na vrácenou hodnotu vliv; tj. pro první stranu od začátku dokumentu je vždy návratová hodnota 1, pro druhou 2, atd.

Je-li parametr ***global*** = **FALSE** návratová hodnota odpovídá aktuálně nastavenému číslování stránek.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetCaretPage**
GetSelEndPage

GetSelStartPos

```
function GetSelStartPos : integer;
```

Popis: Funkce vrací pozici počátku označeného bloku, resp. řádkového kurzoru, není-li žádný blok v textu označen.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetCaretPos**
GetSelEndPos

GetSplitType

```
function GetSplitType : short;
```

Popis: Funkce vrátí typ rozdělení okna aktuálního dokumentu: *stNoSplit, stHorSplit, stVertSplit.*

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **GetSplitVal**
DocSplit

GetSplitVal

```
function GetSplitVal : short;
```

Popis: Funkce vrátí procentuální poměr velikostí u obou částí rozdělených oken.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **GetSplitType**
DocSplit

GetStyleCount

```
function GetStyleCount : short;
```

Popis: Funkce vrací počet definovaných stylů aktuálního dokumentu nabízených v dialogu **Styl odstavce**.

Omezení: Potřebuje aktuální dokument.

Viz: **GetStyleName**
UseParaStyleNum

DefParaStyleDlg

GetStyleName

```
function GetStyleName(
  style_index : short) : string;
```

Parametry: *style_index* číslo stylu

Popis: Funkce vrací název stylu číslo index (v rozmezí od 1 do GetStyleCount).

Poznámka: Styly od 1 do GetStyleCount mají pořadí dle interního setřídění dle abecedy, které se většinou shoduje s pořadím v dialogu **Styl odstavce**.

Omezení: Potřebuje aktuální dokument.

Viz: **GetStyleCount**
UseParaStyleNum
DefParaStyleDlg

GetText

```
function GetText(
  sel_start, sel_end : integer) : string;
```

Parametry: *sel_start* celočíselná souřadnice začátku
sel_end celočíselná souřadnice konce

Popis: Funkce vrací část textu aktuálního dokumentu – textový řetězec mezi pozicemi *sel_start, sel_end.* Pro (*sel_start* >= *sel_end*) nebo (*sel_start* < *0*) nebo (*sel_end* < *0*) vrátí prázdný. Pokud *sel_start* < *sel_end* a obě jsou existující pozice v dokumentu, vrátí řetězec délky minimum ze (*sel_end* – *sel_start*, 255), tj. ne delší než 255 znaků. Funkce nevrací text uvnitř polí (ten není ve výpočtu pozice uvažován).

Omezení: Potřebuje libovolný otevřený aktuální dokument.

Viz: **GetCaretPos**
GetSelStartPos
GetSelEndtPos

GetTextMM

```
function GetTextMM(
  var field_name : const string) : string;
```

Parametry: *fild_name* jméno pole pro slučování

Popis: Vrací obsah pole pro slučování v aktuálním záznamu databáze ("{JMENO}"="Novakova").

Omezení: Potřebuje aktuální dokument napojený na databázi.

Viz: **MailMergeType**
RecordsCount
RecordFirst
RecordLast
RecordNext
RecordPrev

RecordCurrent
RecordSet
GetTextMMI
SetMM
NextMMField

GetTextMMI

```
function GetTextMMI(
  field_index : integer) : string;
```

Parametry: *field_index* index pole daný pořadím položky v záznamu

Popis: Vrací obsah pole pro slučování v aktuálním záznamu databáze.

Omezení: Potřebuje aktuální dokument napojený na databázi.

Viz: **MailMergeType**
RecordsCount
RecordFirst
RecordLast
RecordNext
RecordPrev
RecordCurrent
RecordSet
GetTextMM
SetMM
NextMMField

GetTmplFileName

```
function GetTmplFileName : string;
```

Popis: Vrátí jméno otevřené šablony.

Viz: **GetDocFileName**

GetTopMargin

```
function GetTopMargin : real;
```

Popis: Funkce vrací okamžité nastavení horního okraje.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **SetMargins**
GetLeftMargin
SetLeftMargin
GetRightMargin
SetRightMargin
SetTopMargin
GetBottomMargin
SetBottomMargin

GetTotChapters

```
function GetTotChapters : short;
```

Popis: Vrací počet kapitol v aktuálním dokumentu.

Viz: **GoToChapter**
GoToSection
GoToPage

GetTotPages

```
function GetTotPages : short;
```

Popis: Funkce vrací celkový počet stránek dokumentu.

Omezení: Vyžaduje aktuální dokument.

Viz: **GoToPage**

GetTotSection

```
function GetTotSection : short;
```

Popis: Funkce vrací počet sekcí v dokumentu.

Omezení: Vyžaduje aktuální dokument.

Viz: **GoToSection**

GetUnderlineType

```
function GetUnderlineType : short;
```

Popis: Funkce zjišťuje vlastnost podtrženého textu:

0	bez podtržení
1	vše podtrženo
2	podtržení slov
- 1 = **kAmbiguous**	nelze určit

Poznámka: Viz Poznámka 2.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **SetUnderlineType**
GetFormatFont

GetUserName

```
function GetUserName : string;
```

Popis: Vrací jméno uživatele zadané při instalaci produktu.

GoToChapter

```
function GoToChapter(
  dlg : boolean;
  chapter : short) : short;
```

Parametry: *dlg* zobrazení dialogu
chapter číslo kapitoly

Popis: Fukce vyvolá z menu **Úpravy** příkaz **Jdi na.**
Pro ***dlg* = TRUE** - funkce volá příslušný dialog a inicializuje jej parametrem *chapter*. Dle způsobu ukončení dialogu vrací IDOK (a jde na příslušnou kapitolu) či IDCANCEL.
Pro ***dlg* = FALSE** - je-li (1<= *chapter*) && (*chapter* <= počet kapitol), vrací IDOK (a jde na příslušnou kapitolu). Jinak vrací IDERROR.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GoToSection**
GoToPage
GetTotChapters

GoToIndexEntry

```
function GoToIndexEntry(
  var entry : const string;
  reference : integer) : boolean;
```

Parametry: *entry* položka rejstříku
reference číslo odkazu na tuto položku

Implicitní hodnota: *reference* = 1

Popis: Pro danou položku rejstříku aktuálního dokumentu *entry* funkce umístí řádkový kurzor v dokumentu k odkazu na tuto položku (a event. označí tento odkaz jako blok). Parametr *reference* pro ty položky, na které je více odkazů, specifikuje, o který z odkazů se jedná. (Pro položky rejstříku, které mají v textu pouze jeden odkaz, můžeme využít toho, že implicitní hodnota parametru *reference* je 1). Nachází-li se položka *entry* v rejstříku a pro parametr *reference* platí (1 <= *reference*) && (*reference* <= **CountOfEntryRef** (*entry*)), funkce vrátí TRUE. V opačném případě se neprovede nic a funkce vrátí FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **IndexExists**
IndexEntries
IndexEntry
IndexEntryInd
IndexEditDlg
CountOfEntryRef
AddIndexEntryDlg
AddIndexEntry
RemoveIndexEntry
SelectIndex

GoToMark

```
function GoToMark(
  dlg : boolean;
  var mark_name : const string) : short;
```

Parametry: *dlg* zobrazení dialogu
mark_name jméno záložky

Popis: Pro ***dlg* = TRUE** vyvolá dialog **Záložky**. Jeho vstupní pole se inicializuje druhým parametrem a vrátí IDGOTO, IDINSERT, IDCANCEL dle volby.
Pro ***dlg* = FALSE** vrátí buď IDGOTO, pokud záložka existuje a jde na ni nebo IDERROR, pokud žádná taková není.

Poznámka: Funkce zohledňuje vůči druhému parametru rozdíl mezi malými a velkými písmeny.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **EditMarksDlg**
InsertMark
DoesMarkExis
ClearMark
MarkName
MarksCount

GoToPage

```
function GoToPage(
  dlg : boolean;
  page : short) : short;
```

Parametry: *dlg* zobrazovat dialog
page číslo stránky

Popis: Fukce vyvolá z menu **Úpravy** příkaz **Jdi na.**
Pro ***dlg* = TRUE** funkce vyvolá příslušný dialog. Je-li (1<= *page*) && (*page* <= celkový počet stran), inicializuje je parametrem *page* (jinak inicializuje počtem stran). Dle způsobu ukončení dialogu vrací IDOK (a jde na příslušnou stranu) či IDCANCEL.
Pro ***dlg* = FALSE** je-li (1<= *page*) && (*page* <= **GetTotPages**, vrací IDOK (a jde na příslušnou stranu). Jinak vrací IDERROR.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetTotPages**
GoToChapter
GoToSection

GoToSection

```
function GoToSection(
  dlg : boolean;
  section : short) : short;
```

Parametry: *dlg* zobrazení dialogu
section číslo sekce

Popis: Fukce vyvolá z menu **Úpravy** příkaz **Jdi na.**
Pro ***dlg* = TRUE** funkce vyvolá příslušný dialog. Je-li (1<= *section*) && (*section* <= celkový počet sekcí), inicializuje je parametrem *section* (jinak inicializuje počtem

sekcí). Dle způsobu ukončení dialogu vrací IDOK (a jde na příslušnou sekci) či IDCANCEL.
Pro ***dlg* = FALSE** je-li (1<= *section*) && (*section*<= **GetTotSection**, vrací IDOK (a jde na příslušnou sekci). Jinak vrací IDERROR.

Poznámka: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetTotSection**
GoToPage
GoToChapter

Hours

```
function Hours(
  tm : time) : short;
```

Parametry: *tm* údaj o čase ve vnitřním tvaru 602Text

Popis: Funkce extrahuje počet hodin z časového údaje.

Hodnota: Funkce vrací počet hodin z intervalu 0 až 23.

Viz: **Make_date**
Day
Month
Year
Today
Make_time
Minutes
Seconds
Sec1000
Now

HyphenDlg

```
function HyphenDlg : short;
```

Popis: Pokud není v menu **Pomůcky** přístupna položka příkaz **Dělení slova**, funkce vrací IDCANCEL. Jinak vyvolá dialog **Měkká dělítka** a podle návratu z něj vrací IDOK nebo IDCANCEL.

Viz: **HyphenDlgEnabled**

HyphenDlgEnabled

```
function HyphenDlgEnabled : boolean;
```

Popis: Funkce vrací přístupnost položky menu **Pomůcky** příkaz **Dělení slova**.

Viz: **HyphenDlg**
DatabaseEnabled
ContentsEnabled
IndexMarkEnabled
IndexEditEnabled
IndexGenEnabled
LangSetupEnabled
SpellEnabled

TranslateEnabled
ThesaurusEnabled

ChangeCase

```
function ChangeCase(
  var str : string) : string;
```

Parametry: *str* řetězec znaků

Popis: Funkce konvertuje všechna malá písmena v řetězci *str* na velká, velká písmena na malá. Znaky jako '8', ') ', atd. ponechává v řetězci beze změny.

Hodnota: Funkce vrací řetězec *str*.

Viz: **Upcase**
Lcase

ChangeSelRanges

```
procedure ChangeSelRanges;
```

Popis: Je-li v aktuálním dokumentu označen blok textu, funkce prohodí jeho pevný a pohyblivý konec. Není-li žádný blok označen, neprovede se nic.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru**.**

Příklad:
```
if IsAnyDocOpened and not IsDocMinimized then
if IsTextMode and IsBlockSelected then
if GetSelStartPos = GetCaretPos then
ChangeSelRanges;
```

Viz: **IsBlockSelected**

ChapterHasFooter

```
function ChapterHasFooter : boolean;
```

Popis: Funkce zjišťuje, zda aktuální kapitola má již nějaké zápatí.

Omezení: Potřebuje neminimalizovaný aktuální dokument .

Viz: **ChapterHasHeader**
CreateFooter

ChapterHasHeader

```
function ChapterHasHeader : boolean;
```

Popis: Funkce zjišťuje, zda aktuální kapitola má již nějaké záhlaví.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **ChapterHasFooter**
CreateHeader

ChapterNumber

```
function ChapterNumber(
  dlg, increase : boolean;
  number : short;
  restart_pages : boolean;
  start_page_number : short) : short;
```

Parametry:

dlg	zobrazovat dialog
increase	platnost vzestupného číslování kapitol
number	nastavené číslo kapitoly
restart_pages	nezávislé číslování stran v kapitole
start_page_number	jakým číslem strany začít

Popis: Funkce pro aktuální dokument nastavuje hodnoty, jež se zadávají v dialogu pro číslování kapitol.
Pro ***dlg*** **= TRUE** - vyvolá se zmíněný dialog a jeho vstupní pole se inicializují ostatními parametry funkce:
– increase
Hodnota TRUE znamená vzestupné číslování kapitol (parametr *number* se pak ignoruje).

Hodnota FALSE znamená nastavené číslování kapitol (parametr *number* znamená příslušné nastavené číslo kapitoly).
– restart_pages

Hodnota TRUE znamená, že v každé kapitole se číslování stran začne znovu od první strany této kapitoly (nezávisle na číslování předchozích částí dokumentu); hodnota *start_page_number* je číslo první strany.

Hodnota FALSE znamená vzestupné číslování stran v celém dokumentu; hodnota *start_page_number* udává číslo první strany dokumentu.

Funkce vrací IDOK či IDCANCEL dle způsobu opuštění dialogu.

Možné chybné hodnoty parametrů:
increase = TRUE a neplatí (1<= number) && (number <= 999) –pak se použije hodnota number = 1

neplatí (1 <= start_page_number) && (start_page_number <= 999) – pak se parametry restart_pages, start_page_number ignorují; pro inicializaci dialogu se použije aktuální nastavení.

Pro ***dlg*** **= FALSE** - dialog se přeskočí, zadané hodnoty se rovnou použijí k nastavení s tím rozdílem, že pro chybné hodnoty parametrů se neprovede nic. Pro chybné hodnoty parametrů funkce vrátí IDERROR, jinak vrací IDOK.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Char2str

```
function Char2str(
  ch : char) : string;
```

Parametry:

ch	konvertovaný znak

Popis: Funkce převádí znak *ch* na *string* délky 1. Využití najde v situacích, kde se neprovádí potřebná konverze znaku na *string* (např. ve výrazech typu *string* : = *string* + *znak*).

Hodnota: Funkce vrací řetězec délky 1 obsahující znak *ch*.

Příklad:

```
var ch : char;
       S : string[30];
begin
```

```
        S := '';
for ch := 'A' to 'Z' do
        S:= S + Char2str(ch);
end.
```

CharLeft

```
function CharLeft(
  count : integer;
  shift : boolean) : boolean;
```

Parametry: *count* počet znaků
shift označení textu při pohybu

Implicitní hodnoty: *count* = 1
shift = FALSE

Popis: Funkce přesune kurzor o daný počet znaků vlevo. Pokud se řádkový kurzor pohnul (ne nutně o celý daný počet) vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Příklad:
```
if CharLeft then
CharRight(1, True);
```

Viz: **CharRight**
LineDown
LineUp
CaretEnd
CaretHome
BottomOfScreen
LeftOfLine
RightOfLine
WordLeft
WordRight
RightOfWord
PageUp
PageDown
TopOfScreen

CharRight

```
function CharRight(
  count : integer;
  shift : boolean) : boolean;
```

Parametry: *count* počet znaků
shift označení textu při pohybu

Implicitní hodnoty: *count* = 1
shift = FALSE

Popis: Funkce přesune kurzor o specifikovaný počet znaků vpravo. Přeskakuje přes tabelační znaky. Pokud se řádkový kurzor pohnul (ne nutně o celý daný počet) vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharLeft**
LineDown
LineUp

CaretEnd
CaretHome
LeftOfLine
RightOfLine
WordLeft
WordRight
RightOfWord
PageDown
PageUp
BottomOfScreen
TopOfScreen

CheckBoxGetVal

```
function CheckBoxGetVal(
  dlg_id, id : short) : boolean;
```

Parametry: *dlg_id* identifikátor uživatelského dialogu *id*

Popis: Zjišťuje stav přepínače po ukončení dialogu.

Viz: **DlgCheckBox**
CheckBoxSetVal

CheckBoxSetVal

```
procedure CheckBoxSetVal(
  dlg_id, id : short;
  checked : boolean);
```

Parametry: *dlg_id* identifikátor uživatelského dialogu
id identifikátor přepínače ☑
checked zaškrtnutí přepínače

Popis: Nastavuje stav přepínače před vlastním spuštěním dialogu.

Viz: **DlgCheckBox**
CheckBoxGetVal

Chr

```
function Chr(
  i : short) : char;
```

Parametry: *i* číslo, které bude převedeno na znak

Popis: Funkce převádí celé číslo na znak, jehož kód má stejnou hodnotu. Při této konverzi se používá standard ASCII a východoevropská norma reprezentace národních znaků ve Windows (tzv. code page 1250). Hodnota funkce je definována pro hodnoty parametru z intervalu 0 až 255. Funkce **Chr** je inverzní k funkci **Ord** (v případě, kdy argumentem funkce **Ord** je výraz typu *char*).

Hodnota: Hodnotou funkce je znak, jehož kód je parametrem funkce.

Viz: **Ord**

Iabs

```
function Iabs(
  i : integer) : integer;
```

Parametry: *i* celé číslo
Funkce vrací absolutní hodnotu čísla zadaného jako parametr.

Popis: Funkce spočte absolutní hodnotu celého čísla.

Inc

```
procedure Inc(
  var v : ordinaltype;
  int : integer);
```

Parametry: *v* proměnná libovolného ordinálního typu
int výraz typu integer

Implicitní hodnota: *int* = 1

Popis: Má-li při volání procedury **Inc** parametr *int* svou implicitní hodnotu 1, dosadí příkaz Inc (*v*) do proměnné *v* hodnotu jejího následníka, tj. má stejný efekt jako přiřazení: `v : = Succ (v).`

Je-li proměnná *v* celočíselného typu, zvětší se tedy její hodnota o jedničku.

Obecně má příkaz **Inc**(v, int)stejný význam jako příkaz:

```
if int > 0 then
for j:= 1 to int do v := Succ(v)
else
for j:= -1 downto int do v := Pred(v);
```

(kde proměnná **j** je pomocná celočíselná proměnná).

Je-li proměnná ***v*** číselného typu, zvětší se tedy její hodnota o hodnotu výrazu *int*.

Poznámka: Další podrobnosti naleznete v popisu vnitřního jazyka v části týkající se ordinálních typů.

Viz: **Dec**
Succ
Pred

IndexEditDlg

```
function IndexEditDlg : short;
```

Popis: Funkce odpovídá položce menu **Pomůcky** příkaz **Opravit rejstřík**. Pokud v rejstříku není žádná položka (**IndexEntries** = 0), neprovede se nic a funkce vrátí IDCANCEL. Jinak se vyvolá zmíněný dialog a podle způsobu jeho opuštění funkce vrátí:

IDCANCEL	zrušit
IDGOTO	skok na položku rejstříku
IDDELETE	odstranění položky rejstříku

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **IndexExists**

IndexEntries
IndexEntry
IndexEntryInd
CountOfEntryRef
GoToIndexEntry
AddIndexEntryDlg
AddIndexEntry
RemoveIndexEntry
SelectIndex

IndexEditEnabled

```
function IndexEditEnabled : boolean;
```

Popis: Funkce vrací přístupnost položky menu **Pomůcky** příkaz **Opravit rejstřík.**

Viz: **HyphenDlgEnabled**
DatabaseEnabled
ContentsEnabled
SpellEnabled
IndexMarkEnabled
IndexGenEnabled
LangSetupEnabled
TranslateEnabled
ThesaurusEnabled

IndexEntries

```
function IndexEntries : integer;
```

Popis: Vrací počet dosud zařazených položek v rejstříku.

Omezení: Potřebuje aktuální dokument.

Viz: **IndexExist**
IndexEntry
IndexEntryInd
IndexEditDlg
SelectIndex
CountOfEntryRef
GoToIndexEntry
AddIndexEntryDlg
AddIndexEntry
RemoveIndexEntry

IndexEntry

```
function IndexEntry(
  index : integer) : string;
```

Parametry: *index* číslo položky rejstříku

Popis: Vrací textový řetězec – položku rejstříku, která má číslo *index*.

Poznámka: Číslování položek rejstříku začíná od 1 do IndexEntries, pro (index < 1) nebo (index > IndexEntries) vrátí funkce prázdný řetězec.

Omezení: Potřebuje aktuální dokument.

Viz: **IndexExists**
IndexEntries
IndexEntryInd
IndexEditDlg
CountOfEntryRef
GoToIndexEntry
AddIndexEntryDlg
AddIndexEntry
RemoveIndexEntry
SelectIndex

IndexEntryInd

```
function IndexEntryInd(
  var entry : const string) : integer;
```

Parametry: *entry* položka rejstříku (textový řetězec)

Popis: Pro danou položku rejstříku vrací její index.

Poznámka: Pokud se daná položka v rejstříku nenachází, funkce vrátí NONEINTEGER.

Omezení: Potřebuje aktuální dokument.

Viz: **IndexExists**
IndexEntries
IndexEntry
IndexEditDlg
SelectIndex
CountOfEntryRef
GoToIndexEntry
AddIndexEntryDlg
AddIndexEntry
RemoveIndexEntry

IndexExists

```
function IndexExists : boolean;
```

Popis: Funkce zjišťuje, zda má aktuální dokument vytvořený rejstřík.

Omezení: Potřebuje aktuální dokument.

Viz: **CreateIndex**

IndexGenEnabled

```
function IndexGenEnabled : boolean;
```

Popis: Funkce vrací přístupnost položky menu **Pomůcky** příkaz **Vytvořit rejstřík**.

Viz: **HyphenDlgEnabled**
DatabaseEnabled
ContentsEnabled
SpellEnabled

IndexMarkEnabled
IndexEditEnabled
LangSetupEnabled
TranslateEnabled
ThesaurusEnabled

IndexMarkEnabled

```
function IndexMarkEnabled : boolean;
```

Popis: Funkce vrací přístupnost položky menu **Pomůcky** příkaz **Označit položku rejstříku.**

Viz: **HyphenDlgEnabled**
DatabaseEnabled
ContentsEnabled
IndexEditEnabled
IndexGenEnabled
LangSetupEnabled
TranslateEnabled
ThesaurusEnabled
SpellEnabled

Info_box

```
procedure Info_box(
  var caption, text : const string);
```

Parametry: *caption* nadpis informačního okna
text text v informačním okně

Popis: Procedura otevře na modální obrazovce modální okno obsahující nadpis (*caption)*, text (*text)*, informační ikonu a tlačítko **OK**. Procedura skončí poté, co uživatel stiskne toto tlačítko.
Stiskne-li uživatel místo toho klávesový povel **Ctrl+Break**, přeruší běh programu ve vnitřním jazyku. Takto lze ukončit věčný cyklus, v němž se neustále otevírá informační okno.

Příklad:
```
case Day_of_week(Today) of 0,6 :
    Info_box('Rada', 'O víkendu nepracujte !')
end;
```

Input_box

```
function Input_box(
  var caption : const string;
  var text : string;
  maxlen : short) : boolean;
```

Parametry: *caption* nadpis dialogu
text řetězec editovaný v dialogu
maxlen maximální délka řetězce text

Popis: Funkce otevře na obrazovce dialogový rámeček a umožní uživateli zadat jeden údaj ve formě řetězce znaků. Tento řetězec vrátí v proměnné *text*.

Obsah proměnné *text* musí být před voláním této funkce inicializován řetězcem, které se má v dialogovém rámečku objevit po jeho otevření.

Ukončit zadávání řetězce můžete buď stiskem tlačítka **OK** nebo **Zrušit akci**. V druhém případě obsah proměnné *text* nebude změněn.

Není možné zadat více znaků, než je hodnota parametru *maxlen*.

Hodnota: Pokud uživatel uzavře dialog stiskem tlačítka **OK,** pak funkce vrátí TRUE. Pokud použije tlačítko **Zrušit akci**, funkce vrátí FALSE.

Poznámka: Použije-li uživatel namísto popsaných způsobů k opuštění tohoto dialogu klávesový povel **Ctrl+Break**, přeruší běh programu ve vnitřním jazyku. Podobně jako u ostatních dialogů lze takto ukončit věčný cyklus, v němž se neustále otevírá dialogový rámeček.

Příklad:

```
var
  a : boolean;
  ss : string[10];
  dat : date;
  begin
    repeat
    ss := date2str(Today,1);
    a := Input_box('Zadejte datum',ss,10);
    dat := Str2date(ss);
    until (dat <> nonedate) or (a = False);
    .......
  end.
```

Viz: **Input_box_msg**

Input_box_msg

```
function Input_box_msg(
   var caption, msg : const string;
   var text : string;
  maxlen : short) : boolean;
```

Parametry:

caption	nadpis dialogu
msg	informační text v dialogu
text	řetězec editovaný v dialogu
maxlen	maximální délka řetězce text

Popis: Funkce provádí podobnou činnost jako **Input_box**, ovšem dodatečný parametr – *msg* – dovolí v tomto dialogu zobrazit nad vstupním polem další informaci.

Hodnota: Pokud uživatel uzavře dialogový rámeček stiskem tlačítka **OK,** pak funkce vrátí TRUE. Pokud použije tlačítko **Zrušit akci**, funkce vrátí FALSE.

Poznámka: Použije-li uživatel namísto popsaných způsobů k opuštění tohoto dialogu klávesový povel **Ctrl+Break**, přeruší běh programu ve vnitřním jazyku. Podobně jako u ostatních dialogů lze takto ukončit věčný cyklus, v němž se neustále otevírá dialogový rámeček.

Viz: **Input_box**

Input_box_msg_2

```
function Input_box_msg_2(
  var caption, msg1 : const string;
  var text1 : string;
```

```
  maxlen1 : short;
  var msg2 : const string;
  var text2 : string;
  maxlen2 : short) : boolean;
```

Parametry:

caption	nadpis dialogu
msg1, msg2	informační texty v dialogu
text1, text2	řetězce editované v dialogu
maxlen1, maxlen2	maximální délky řetězců text1, text2

Popis: Funkce je analogická funkci **Input_box_msg**, slouží ovšem pro vstup dvou znakových řetězců. Jako vstupně/výstupní parametry slouží řetězce *text1, text2*. Vstupní parametry *msg1, msg2* slouží pro zobrazení dodatečných informací nad vstupními poli.

Hodnota: Pokud uživatel uzavře dialogový rámeček stiskem tlačítka **OK**, pak funkce vrátí TRUE. Pokud použije tlačítko **Zrušit akci**, funkce vrátí FALSE.

Poznámka: Použije-li uživatel namísto popsaných způsobů k opuštění tohoto dialogu klávesový povel **Ctrl+Break**, přeruší běh programu ve vnitřním jazyku. Podobně jako u ostatních dialogů lze takto ukončit věčný cyklus v němž se neustále otevírá dialogový rámeček.

Viz: **Input_box_msg**
Input_box_msg_3
Input_box_msg_4

Input_box_msg_3

```
function Input_box_msg_3(
  var caption, msg1 : const string;
  var text1 : string;
  maxlen1 : short;
  var msg2 : const string;
  var text2 : string;
  maxlen2 : short;
  var msg3 : const string;
  var text3 : string
  maxlen3 : short) : boolean;
```

Parametry:

caption	nadpis dialogu
msg1, msg2, msg3	informační texty dialogu
text1, text2, text3	řetězce editované v dialogu
maxlen1, maxlen2, maxlen3	maxim. délky řetězců text1, text2, text3

Popis: Funkce je analogická funkci **Input_box_msg**, slouží ovšem pro vstup tří znakových řetězců. Jako vstupně/výstupní parametry slouží řetězce *text1, text2, text3*; vstupní parametry *msg1, msg2, msg3* slouží pro zobrazení dodatečných informací nad vstupními poli.

Hodnota: Pokud uživatel uzavře dialogový rámeček stiskem tlačítka **OK**, pak funkce vrátí TRUE. Pokud použije tlačítko **Zrušit akci**, funkce vrátí FALSE.

Poznámka: Použije-li uživatel namísto popsaných způsobů k opuštění tohoto dialogu klávesový povel **Ctrl+Break**, přeruší běh programu ve vnitřním jazyce. Podobně jako u ostatních dialogů lze takto ukončit věčný cyklus, v němž se neustále otevírá dialogový rámeček.

Viz: **Input_box_msg**
Input_box_msg_2
Input_box_msg_4

Input_box_msg_4

```
function Input_box_msg_4(
  var caption, msg1 : const string;
  var text1 : string;
  maxlen1 : short;
  var msg2 : const string;
  var text2 : string;
  maxlen2 : short;
  var mgs3 : const string;
  var text3 : string;
  maxlen3 : short;
  var msg4 : const string;
  var text4 : string;
  maxlen4 : short) : boolean;
```

Parametry:

caption	nadpis dialogu
msg1, msg2, msg3, msg4	informační texty v dialogu
text1, text2, text3, text4	řetězce editované v dialogu
maxlen1, maxlen2, maxlen3, maxlen4	max.délky řetězců: text1, text2, text3, text4

Popis: Funkce je analogická funkci **Input_box_msg**, slouží ovšem pro vstup čtyř znakových řetězců. Jako vstupně/výstupní parametry slouží řetězce *text1, text2, text3, text4*; vstupní parametry *msg1, msg2, msg3, msg4* slouží pro zobrazení dodatečných informací nad vstupními poli.

Hodnota: Pokud uživatel uzavře dialogový rámeček stiskem tlačítka **OK,** pak funkce vrátí TRUE. Pokud použije tlačítko **Zrušit akci**, funkce vrátí FALSE.

Poznámka: Použije-li uživatel namísto popsaných způsobů k opuštění tohoto dialogu klávesovým povelem **Ctrl+Break**, přeruší tím běh programu ve vnitřním jazyku. Podobně jako u ostatních dialogů lze takto ukončit věčný cyklus, v němž se neustále otevírá dialogový rámeček.

Viz: **Input_box_msg**
Input_box_msg_2
Input_box_msg_3

InputLineGetVal

```
function InputLineGetVal(
  dlg_id, id : short) : string;
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
id	identifikátor vstupní řádky

Popis: Funkce slouží pro zjištění hodnoty řetězce znaků vstupní řádky v uživatelském dialogu.

Viz: **DlgInputLine**
InputLineSetVal

InputLineSetVal

```
procedure InputLineSetVal(
  dlg_id, id : short;
  var text : const string;
```

```
  maxlen : short);
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
id	identifikátor vstupní řádky
text	řetězec znaků – text vstupní řádky
maxlen	maximální počet znaků vstupní řádky

Implicitní hodnota: *maxlen* = 256

Popis: Procedura slouží pro zadání nové hodnoty řetězce znaků na vstupní řádce v dialogu, event. ke změně hodnoty maximální délky vstupní řádky.

Poznámka: Hodnotou parametru *maxlen* musí být kladné celé číslo mezi 1 a 256 včetně. Menší hodnota způsobí předčasné ukončení běhu makra s chybovou hláškou. Hodnota větší než 256 je "zaokrouhlena" na 256.

Viz: **InputLineGetVal**
DlgInputLine

InsertDateTime

```
function InsertDateTime : boolean;
```

Popis: Funkce vloží aktuální datum a čas na pozici kurzoru.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **InsertHardHyphen**
InsertOptHyphen
InsertHardSpace
InsertNewPage
InsertNewChapter
InsertNewPara
InsertTab
InsertChar
InsertText
InsertTextStr

InsertFileObject

```
function InsertFileObject(
  var path_name:const string;
  full_size:boolean):boolean;
```

Popis: Vloží do dokumentu obrázek načtený z grafického souboru:

- návratová hodnota **TRUE** - není-li v dokumentu označen žádný rámeček (objekt) bude vložen obrázek v původní velikosti.
- návratová hodnota **FALSE** - je-li v dokumentu označen jakýkoli rámeček (objekt) bude obrázek vložen do tohoto rámečku (objektu) a přizpůsobí mu svou velikost.

InsertFootNote

```
function InsertFootNote : boolean;
```

Popis: Funkce vyvolá z menu **Vložit** příkaz **Poznámku pod čarou**.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

InsertHardHyphen

```
function InsertHardHyphen : boolean;
```

Popis: Funkce vyvolá z menu **Vložit** příkaz **Tvrdé dělítko**.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **InsertDateTime**
InsertOptHyphen
InsertHardSpace
InsertNewPage
InsertNewChapter
InsertNewPara
InsertTab
InsertTextStr
InsertChar
InsertText

InsertHardSpace

```
function InsertHardSpace : boolean;
```

Popis: Funkce z menu **Vložit** příkaz **Tvrdou mezeru**.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **InsertDateTime**
InsertHardHyphen
InsertOptHyphen
InsertNewPage
InsertNewChapter
InsertNewPara
InsertTab
InsertChar
InsertText
InsertTextStr

InsertChar

```
function InsertChar(
  ch : char) : boolean;
```

Parametry: *ch* vkládaný znak

Popis: Vloží zadaný znak na aktuální pozice řádkového kurzoru. Funkce však není určena pro vkládání tabelátorů, znaků pro přechod na nový řádek, atd. Tyto netisknutelné znaky převádí na mezery s výjimkou tabelátoru (chr (9)), pro který se zavolá funkce **InsertTab**.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **InsertDateTime**
InsertHardHyphen
InsertOptHyphen
InsertHardSpace

InsertNewPage
InsertNewChapter
InsertNewPara
InsertTab
InsertText
InsertTextStr

InsertMark

```
function InsertMark(
  dlg : boolean;
  var name : const string;
  check_duplicity : boolean) : short;
```

Parametry:

dlg	zobrazovat dialog
name	jméno záložky
check_duplicity	kontrolovat existenci záložky stejného jména

Popis: Funkce slouží pro vložení záložky do textu.
Pro ***dlg* = TRUE** vyvolá dialog **Záložky**. Vstupní pole se inicializuje druhým parametrem; vrátí IDGOTO, IDINSERT, IDCANCEL dle volby, tedy stejná činnost jako v **GoToMark**. Parametr *check_duplicity* je ignorován.
Pro ***dlg* = FALSE** vrátí buď IDINSERT (vloží záložku) nebo IDERROR v případě, že *check_duplicity* = TRUE a záložka existuje (a tedy ji nevloží, starou ponechá).

Poznámka: Funkce zohledňuje vůči druhému parametru rozdíl mezi malými a velkými písmeny.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **EditMarksDlg**
GoToMark
DoesMarkExist
ClearMark
MarkName
MarksCount

InsertNewChapter

```
function InsertNewChapter : boolean;
```

Popis: Fukce vyvolá z menu **Vložit** příkaz **Konec kapitoly**.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **InsertDateTime**
InsertHardHyphen
InsertOptHyphen
InsertNewPage
InsertHardSpace
InsertNewPara
InsertTab
InsertChar
InsertText
InsertNewSection
InsertTextStr

InsertNewPage

```
function InsertNewPage : boolean;
```

Popis: Funkce vyvolá z menu **Vložit** příkaz **Konec stránky**.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **InsertDateTime**
InsertHardHyphen
InsertOptHyphen
InsertChar
InsertHardSpace
InsertNewChapter
InsertNewPara
InsertText
InsertTab
InsertNewSection
InsertTextStr

InsertNewPara

```
function InsertNewPara : boolean;
```

Popis: Funkce vloží konec odstavce na pozici kurzoru.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **InsertDateTime**
InsertHardHyphen
InsertOptHyphen
InsertChar
InsertHardSpace
InsertNewPage
InsertNewChapter
InsertText
InsertTab
InsertNewSection
InsertTextStr

InsertNewSection

```
function InsertNewSection : boolean;
```

Popis: Funkce vloží konec sekce na pozici kurzoru.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **InsertDateTime**
InsertHardHyphen
InsertOptHyphen
InsertChar
InsertHardSpace
InsertNewPage
InsertNewPara
InsertNewChapter
InsertText
InsertTab
InsertTextStr

InsertOptHyphen

```
function InsertOptHyphen : boolean;
```

Popis: Funkce vyvolá z menu **Vložit** příkaz **Měkké dělítko**.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **InsertDateTime**
InsertHardHyphen
InsertHardSpaceInsertNewPage
InsertNewChapter
InsertNewPara
InsertTab
InsertChar
InsertText
InsertTextStr

InsertTab

```
function InsertTab : boolean;
```

Popis: Funkce vloží tabelátor na aktuální pozici řádkového kurzoru.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **InsertDateTime**
InsertHardHyphen
InsertOptHyphen
InsertChar
InsertHardSpace
InsertNewPage
InsertNewChapter
InsertNewPara
InsertText

InsertText

```
function InsertText(
  var text : string) : boolean;
```

Parametry: *text* vkládaný řetězec

Popis: Vloží řetězec *text* na aktuální řádkovou pozici kurzoru. Na argument *text* je přitom automaticky zavolána funkce **CleanString,** která všechny netisknutelné znaky v řetězci *text* (tzn. ty, které jsou v ordinálním uspořádání menší než mezera) převádí na mezeru. Pokud se provedlo vložení vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CleanString**
InsertDateTime
InsertHardHyphen
InsertOptHyphen
InsertHardSpace
InsertNewPage

InsertNewChapter
InsertNewPara
InsertChar

InsertTextStr

```
function InsertTextStr(
  var text : const string) : boolean;
```

Parametry: *text* vkládaný řetězec

Popis: Funkce (podobně jako funkce **InsertText** vkládá do aktuálního dokumentu na aktuální řádkovou pozici řetězec *text*. Na rozdíl od této funkce se však na parametr *text* neaplikuje implicitně funkce **CleanString**; parametr *text* zde tedy není vstupně / výstupním, ale pouze vstupním parametrem a může tedy být i řetězovou konstantou (ať už vyjádřenou přímo zápisem hodnoty nebo identifikátorem dříve definované konstanty tohoto typu).

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Poznámka: I tato funkce však kontroluje, zda vkládaný řetězec neobsahuje netisknutelné znaky. Jsou-li zjištěny — běh makra je předčasně ukončen.

Viz: **CleanString**
InsertDateTime
InsertHardHyphen
InsertOptHyphen
InsertHardSpace
InsertNewPage
InsertNewChapter
InsertNewPara
InsertNewSection
InsertChar

Int2str

```
function Int2str(
  int : integer) : string;
```

Parametry: *int* hodnota, která se má konvertovat

Popis: Funkce převede hodnotu zadanou parametrem na řetězec znaků a vrátí jej.

Hodnota: Funkce vrací dekadický zápis čísla *int*.

Viz: **Str2int**
Real2str
Str2real
Money2str
Str2money
Date2str
Str2date
Time2str
Str2time

IsAlignCenter

```
function IsAlignCenter : boolean;
```

Popis: Funkce zjišťuje, zda je zarovnávání centrované.

Poznámka: Viz Poznámka 4.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **SetAlignCenter**
SetAlign
GetAlignType

IsAlignJustify

```
function IsAlignJustify : boolean;
```

Popis: Funkce zjišťuje, zda je zarovnávání oboustranné.

Poznámka: Viz Poznámka 4.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **SetAlignJustify**
SetAlign
GetAlignType

IsAlignLeft

```
function IsAlignLeft : boolean;
```

Popis: Funkce zjišťuje, zda je zarovnávání nalevo.

Poznámka: Viz Poznámka 4.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **SetAlignLeft**
SetAlign
GetAlignType

IsAlignRight

```
function IsAlignRight : boolean;
```

Popis: Funkce zjišťuje, zda je zarovnávání napravo.

Poznámka: Viz Poznámka 4.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **SetAlignRight**
SetAlign
GetAlignType

IsAnyDocOpened

```
function IsAnyDocOpened : boolean;
```

Popis: Funkce zjišťuje, je-li vůbec nějaký dokument otevřen.

Viz: **CountWindows**

IsBlockSelected

```
function IsBlockSelected : boolean;
```

Popis: Funkce testuje, zda je v textu označen nějaký blok.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **UnselectBlock**

IsBold

```
function IsBold : boolean;
```

Popis: Funkce zjišťuje, zda je aktuální text (nebo označený blok) psaný tučným písmem.

Poznámka: Viz Poznámka 1.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **SetBold**
GetFormatFont

IsBrushOn

```
function IsBrushOn : boolean;
```

Popis: Funkce zjišťuje, je-li aktivován štěteček. Není-li otevřen žádný dokument, vrací FALSE.

Viz: **SetBrush**

IsDbForSave

```
function IsDbForSave : boolean;
```

Popis: Vrací TRUE, když je napojena databáze pro zápis (viz příkazy **Inteligentní šablona** a **Vytvoření databáze**).

IsDocDefault

```
function IsDocDefault : boolean;
```

Popis: Pokud byl aktuální dokument vytvořen pomocí příkazu **Nový** a dosud nebyl uložen (ani přes autosave) funkce vrátí TRUE. Jinak hlásí FALSE.

Omezení: Potřebuje aktuální dokument.

Viz: **GetDocFileName**

IsDocMaximized

```
function IsDocMaximized : boolean;
```

Popis: Funkce zjišťuje, je-li aktuální okno s dokumentem v maximalizovaném tvaru.

Omezení: Potřebuje aktuální dokument.

Viz: **IsDocMinimized**
DocMaximize
DocMinimize
DocRestore

IsDocMinimized

```
function IsDocMinimized : boolean;
```

Popis: Funkce zjišťuje, je-li aktuální okno s dokumentem v minimalizovaném tvaru.

Omezení: Potřebuje aktuální dokument.

Viz: **IsDocMaximized**
DocMaximize
DocMinimize
DocRestore

IsDocumentOpened

```
function IsDocumentOpened(
  var path_name : const string) : boolean;
```

Parametry: *path_name* název souboru včetně cesty

Popis: Funkce hledá okno s dokumentem dle specifikace odpovídajícího souboru (nutno zadat kompletně diskovou jednotku i přístupovou cestu, např. `"c:\tmp\readme.wpd").`
Tato funkce nečiní rozdílu mezi velkými a malými písmeny, tj. `"C:\TMP\README.WPD"` má stejný efekt.

Viz: **SwitchToDoc**
GetDocWndTittle
GetDocFileName

IsFieldCntsOn

```
function IsFieldCntsOn : boolean;
```

Popis: Funkce zjišťuje, zda-li je zobrazen obsah polí.

Poznámka: Nemá-li žádný aktuální dokument, vrací FALSE.

Viz: **ViewFieldCnts**

IsFormatOn

```
function IsFormatOn : boolean;
```

Popis: Funkce zjišťuje, zda jsou zobrazeny skryté znaky.

Poznámka: Nemá-li žádný aktuální dokument, vrací FALSE.

Viz: **ViewFormat**

IsInsertMode

```
function IsInsertMode : boolean;
```

Popis: Funkce testuje, zda je aktivní režim vkládání.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **SetInsertMode**
SetOverWriteMode
ToggleInsert

IsItalic

```
function IsItalic : boolean;
```

Popis: Funkce zjišťuje, zda je aktuální text (nebo označený blok) psaný kurzívou.

Poznámka: Viz Poznámka 1.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **SetItalic**
GetFormatFont

IsMacro

```
function IsMacro : boolean;
```

Popis: Funkce zjišťuje, zda je aktuální dokument nastaven v režimu editace makra.

Omezení: Potřebuje aktuální dokument.

Viz: **SetIsMacro**

IsModified

```
function IsModified : boolean;
```

Popis: Funkce testuje, zda se obsah dokumentu změnil od posledního uložení do odpovídajícího souboru.

Omezení: Potřebuje aktuální dokument.

Viz: **MarkModified**
MarkUnmodified

IsPageLandscape

```
function IsPageLandscape : boolean;
```

Popis: Funkce zjišťuje, zda nastavení stránky (orientace) dokumentu je na šířku (Landscape).

Omezení: Potřebuje aktuální dokument.

Viz: **GetPageWidth**
GetPageHeight
GetPageType
IsPagePortrait
PageFormat

IsPagePortrait

```
function IsPagePortrait : boolean;
```

Popis: Funkce zjišťuje, zda nastavení stránky (orientace) dokumentu je na výšku (Portrait).

Omezení: Potřebuje aktuální dokument.

Viz: **GetPageWidth**
GetPageHeight
GetPageType
IsPageLandscape
PageFormat

Isqr

```
function Isqr(
  i : integer) : integer;
```

Parametry: *i* celé číslo

Popis: Funkce spočte druhou mocninu celého čísla.

Hodnota: Funkce vrací druhou mocninu čísla zadaného jako parametr. Pokud druhá mocnina argumentu překračuje rozsah hodnot typu *integer*, není definováno, co funkce vrátí.

IsReadOnly

```
function IsReadOnly : boolean;
```

Popis: Funkce zjišťuje, zda je aktuální dokument otevřen pouze pro čtení.

Omezení: Vyžaduje aktuální dokument.

Viz: **IsMacro**

IsStatusStrip

```
function IsStatusStrip : boolean;
```

Popis: Funkce zjišťuje, zda je zobrazen stavový řádek.

Viz: **SetStatusStrip**
SetStatusText

IsTabRulerOn

```
function IsTabRulerOn : boolean;
```

Popis: Funkce zjišťuje, zda-li je zobrazeno pravítko s tabelátory.

Poznámka: Není-li žádný dokument aktivní nebo je-li aktuální dokument v zobrazení osnovy, vrací FALSE.

Viz: **ViewTabRuler**

IsTextDefault

```
function IsTextDefault : boolean;
```

Popis: Funkce zjišťuje, je-li text daný stylem odstavce.

Poznámka: Viz Poznámka 1.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **SetTextDefault**

IsTextRegular

```
function IsTextRegular : boolean;
```

Popis Funkce zjišťuje, zda-li je text obyčejný.

Poznámka: Viz Poznámka 1.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **SetTextRegular**

IsToolbar

```
function IsToolbar(
  index : short) : boolean;
```

Parametry: *index* index lišty

Implicitní hodnota: index = 2

Popis: Funkce zjišťuje, zda je příslušná nástrojová lišta zapnuta:

1	Osnova
2	Formát
3	Objekt
4	Tabulka
5	Standardní
6	Pomůcky
7	Pole a databáze
8	Náhled
9	HTML
10	Zobrazení
11	Formulářové objekty
12	Štítky
13	WEB

Viz: **SetToolbar**

IsVertRulerOn

```
function IsVertRulerOn : boolean;
```

Popis: Funkce zjišťuje, zda-li je zobrazeno svislé pravítko.

Poznámka Nemá-li žádný dokument nebo je-li aktuální dokument v zobrazení osnovy, vrací FALSE.

Viz: **ViewVertRuler**

IsWordOverFlow

```
function IsWordOverFlow : boolean;
```

Popis: Funkce zjišťuje, zda je nastaveno přetékání zvolených slov.

Viz: **SetWordOverFlow**

LabelBlock

```
function LabelBlock(
  dlg : boolean;
  var block_name : const string;
  check_duplicity : boolean) : short;
```

Parametry:

dlg	zobrazovat dialog
block_name	jméno bloku
check_duplicity	kontrolovat existenci bloku stejného jména

Popis: Pro ***dlg* = TRUE** - vyvolá dialog **Pojmenované bloky**. Vstupní pole se inicializuje druhým parametrem. Vrátí IDGOTO, IDINSERT, IDCANCEL dle volby. (Tedy stejná činnost jako v **SelectBlock**. Parametr *check_duplicity* je ignorován).

Pro ***dlg* = FALSE** - není-li označen žádný blok, vrátí IDERROR. Jinak vrátí buď IDINSERT (pojmenuje blok) nebo IDERROR v tom případě, že check_duplicity = TRUE a jiný blok toho jména již existuje (a tedy ho nepojmenuje, starý ponechá).

Poznámka: Funkce zohledňuje vůči parametru *block_name* rozdíl mezi malými a velkými písmeny.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **EditBlocksDlg**
SelectBlock
UnlabelBlock
DoesBlockExist
BlocksCount
BlockName
CopyBlock

LangSetupEnabled

```
function LangSetupEnabled : boolean;
```

Popis: Funkce vrací přístupnost položky menu **Pomůcky** příkaz **Volba jazyka.**

Viz: **HyphenDlgEnabled**
DatabaseEnabled
ContentsEnabled
IndexMarkEnabled
IndexEditEnabled
IndexGenEnabled
TranslateEnabled
ThesaurusEnabled
SpellEnabled

Lcase

```
function Lcase(
  var str : string) : string;
```

Parametry: *str* řetězec znaků

Popis: Funkce konvertuje všechna velká písmena v řetězci *str* na malá. Umí konvertovat i národní znaky.

Hodnota: Funkce vrací řetězec *str*.

Viz: **Upcase**
ChangeCase

LeftOfLine

```
function LeftOfLine(
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Implicitní hodnota: *shift* = FALSE

Popis: Funkce přesune kurzor na začátek aktuální řádky. Pokud se řádkový kurzor pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharLeft**
CharRight
LineDown
LineUp
TopOfScreen
RightOfLine
BottomOfScreen
CaretHome
CaretEnd
WordLeft
WordRight
PageUp
RightOfWord
PageDown

Like

```
function Like(
  var s1, s2 : const string) : boolean;
```

Parametry: *s1* řetězec znaků
s2 řetězec znaků

Popis: Funkce zjišťuje, zda řetězce *s1* a *s2* si jsou podobné; zda se liší pouze mezerami, velikostí písmen a diakritikou.

Hodnota: Funkce vrátí TRUE, pokud mezi *s1* a *s2* jsou pouze výše uvedené rozdíly (nebo ani ty). Jinak funkce vrátí FALSE.

LineDown

```
function LineDown(
  count : integer;
  shift : boolean) : boolean;
```

Parametry: *count* počet řádek
shift označení textu při pohybu

Implicitní hodnoty: *count* = 1
shift = FALSE

Popis: Funkce přesune kurzor o zadaný počet řádků dolů. Vrací TRUE, pokud se řádkový kurzor pohnul (ne nutně o celý daný počet).

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharLeft**
CharRight
LineUp
CaretEnd
CaretHome
LeftOfLine
RightOfLine
WordLeft
WordRight
RightOfWord
PageDown
PageUp

BottomOfScreen
TopOfScreen

LineUp

```
function LineUp(
  count : integer;
  shift : boolean) : boolean;
```

Parametry: *count* počet řádek
shift označení textu při pohybu

Implicitní hodnoty: *count* = 1
shift = FALSE

Popis: Funkce přesune řádkový kurzor o zadaný počet řádků nahoru. Vrací TRUE, pokud se řádkový kurzor pohnul (ne nutně o celý daný počet).

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharLeft**
CharRight
LineDown
CaretEnd
CaretHome
LeftOfLine
RightOfLine
WordLeft
WordRight
RightOfWord
PageDown
PageUp
BottomOfScreen
TopOfScreen

Ln

```
function Ln(
  r : real) : real;
```

Parametry: *r* reálné číslo

Popis: Funkce počítá přirozený logaritmus. Parametr funkce musí být kladný, jinak dojde k chybě při výpočtu.

Hodnota: Funkce vrátí hodnotu přirozeného logaritmu svého parametru.

Ltrim

```
function Ltrim(
  var str : string) : string;
```

Parametry: *str* řetězec znaků

Popis: Funkce odstraní z řetězce *str* mezery na začátku. Mezery vyskytující se mezi jinými znaky v řetězci ponechá beze změny. Ponechá beze změny též mezery vyskytující se

na konci (pokud se ovšem řetězec neskládá ze samých mezer). Řetězec *str*, předaný odkazem, je touto funkcí změněn.

Hodnota: Funkce vrací řetězec *str*.

Viz: **Rtrim**
Strtrim

MacroAutoStarted

```
function MacroAutoStarted : boolean;
```

Popis: Funkce slouží ke zjištění, zda právě běžící makro bylo spuštěno automaticky, tj. jedním z automatických maker (**AutoNew**, **AutoOpen**,...) a při práci s programem nastala situace, kdy se spouští automaticky bez zásahu uživatele.

Poznámka: Automatická makra mohou být ovšem spuštěna uživatelem jako všechna ostatní makra (klávesovou zkratkou,...) – v tom případě tato funkce vrací FALSE.

Viz: **MacroIsInMenu**
MacroIsLocal
CurrentMacroName

MacroDelete

```
function MacroDelete(
  var name : const string;
  global : boolean) : boolean) : boolean;
```

Parametry: *name* jméno makra
global globální makro

Implicitní hodnota: *global* = TRUE

Popis: Funkce vypouští makro jménem *name* ze seznamu spustitelných maker:
Je-li ***global* =TRUE** hledá je v seznamu globálních maker.
Je-li ***global* = FALSE** hledá je v seznamu lokálních maker v aktuálním dokumentu (a to v dokumentu aktuálním v okamžiku zavolání funkce, který nemusí být totožný s dokumentem aktuálním v okamžiku spuštění makra).
Při hledání se rozlišují velká a malá písmena. Pokud bylo makro nalezeno a vypuštěno, funkce vrátí TRUE; jinak FALSE.

Poznámka: Voláním této funkce nelze smazat právě běžící makro, tzn., že volání **MacroDelete** (**CurrentMacroName**, **NotMacroLocal**) vždy vrátí FALSE. Pro tento účel použijte proceduru **ErasethisMacro**.

Viz: **MacroName**
MacroNameToIndex
MacroIsInMenu
MacroSetToMenu
MacroDescription
MacroSetDescr
MacroIsLocal
ErasethisMacro

MacroDescription

```
function MacroDescription(
  var name : const string;
  global : boolean) : string;
```

Parametry: *name* jméno makra
global globální makro

Implicitní hodnota: *global* = TRUE

Popis: Funkce prochází seznam spustitelných maker (podle parametru *global* buďto globálních nebo lokálních) v aktuálním dokumentu a nalezne-li makro jménem *name* vrací jeho popis. Jinak je návratovou hodnotou prázdný řetězec. Při hledání se velká a malá písmena nerozlišují.

Viz: **MacroSetDescr**
MacroName
MacroNameToIndex

MacroIsInMenu

```
function MacroIsInMenu(
  var name : const string;
  global : boolean) : boolean;
```

Parametry: *name* jméno makra
global globální makro

Implicitní hodnota: *global* = TRUE

Popis: Funkce prochází seznam spustitelných maker (podle parametru *global* buď globálních nebo lokálních) v aktuálním dokumentu a nalezne-li makro jménem *name* vrací TRUE/FALSE podle toho, zda makro je nebo není zařazeno do menu. Jinak vrací FALSE. Při hledání se velká a malá písmena nerozlišují.

Viz: **MacroSetToMenu**
MacroAutoStarted
MacroIsLocal
CurrentMacroName

MacroIsLocal

```
function MacroIsLocal : boolean;
```

Popis: Funkce slouží ke zjištění, zda právě běžící makro je v nějakém dokumentu lokální (pak je návratová hodnota TRUE) nebo je-li globální (pak je návratová hodnota FALSE).

Viz: **MacroAutoStarted**
MacroIsInMenu
CurrentMacroName

MacroName

```
function MacroName(
  index : integer;
  global : boolean) : string;
```

Parametry: *index* index makra
global globální makro

Implicitní hodnota: *global* = TRUE

Popis: Funkce vrací jméno makra ze seznamu spustitelných maker, buď globálních nebo lokálních v aktuálním dokumentu. Pro tento účel jsou globální a lokální makra číslována zvlášť a to od 1 do počtu maker (vrací funkce **CountMacros**).

Poznámka: Pokud je hodnota parametru *index* chybná (neodpovídá mu žádné makro) funkce vrací prázdný řetězec.

Viz: **CountMacros**
MacroNameToIndex
MacroIsInMenu
MacroSetToMenu
MacroDescription
MacroSetDesc
MacroDelete

MacroNameToIndex

```
function MacroNameToIndex(
  var name : const string;
  global : boolean) : integer;
```

Parametry: *name* jméno makra
global globální makro

Implicitní hodnota: *global* = TRUE

Popis: Funkce prochází seznam spustitelných maker (podle parametru *global* buď globálních nebo lokálních) v aktuálním dokumentu. Nalezne-li makro *name* vrací jeho index. Globální a lokální makra jsou číslována zvlášť a to od jedné jejich počtu. Při hledání se velká a malá písmena nerozlišují.

Poznámka: Pokud makro daného jména není nalezeno funkce vrací NONEINTEGER.

Viz: **MacroName**
CountMacros
MacroIsInMenu
MacroSetToMenu
MacroDescription
MacroSetDescr
MacroDelete

MacroSetDescr

```
function MacroSetDescr(
  var name, description : const string;
  global : boolean) : boolean;
```

Parametry: *name* jméno makra
description nový popis makra
global globální makro

Implicitní hodnota: *global* = TRUE

Popis: Funkce prochází seznam spustitelných maker (podle parametru *global* buď globálních nebo lokálních) v aktuálním dokumentu a nalezne-li makro jménem *name* nastaví jeho nový popis daný parametrem *description* a vrátí TRUE. Jinak je návratová hodnota FALSE. Při hlednání se velká a malá písmena nerozlišují.

Viz: **MacroDescription**
MacroName
MacroNameToIndex
MacroIsInMenu
MacroSetToMenu
MacroDelete

MacroSetToMenu

```
function MacroSetToMenu(
  var name : const string;
  on, global : boolean) : boolean;
```

Parametry: *name* jméno makra
on zařadit makro do menu
global globální makro

Implicitní hodnoty: *on* = TRUE
global = TRUE

Popis: Funkce prochází seznam spustitelných maker (podle parametru *global* buď globálních nebo lokálních) v aktuálním dokumentu. Pokud nalezne makro jménem *name* bude návratová hodnota funkce TRUE a podle parametru *on* se nastaví nebo zruší vlastnost makra „zařadit do menu“. Jinak je návratová hodnota FALSE. Při hledání se velká a malá písmena nerozlišují.

Viz: **MacroIsInMenu**
MacroName
MacroDescription
MacroSetDecsr
MacroDelete

MacroSubrListDlg

```
function MacroSubrListDlg : short;
```

Popis: Funkce vyvolá dialog **Seznam procedur a funkcí**. Dle způsobu jeho opuštění vrací IDINSERT nebo IDCANCEL.

MacroToGlobal

```
function MacroToGlobal(
  var name : const string;
  copy : boolean;
```

```
  prez : short) : boolean;
```

Parametry:
name jméno makra
copy kopírovat anebo přesunout
prez činnost při existenci makra téhož jména

Implicitní hodnoty:
copy = FALSE
prez = 0

Popis: Funkce slouží pro přesun anebo kopírování makra za úložny lokálních maker aktuálního dokumentu do úložny globálních maker.
Při hodnotě parametru:

- *copy* = TRUE se provádí kopírování
- *copy* = FALSE se makro přesune a ze seznamu lokálních maker je tedy vypuštěno.

Makro je identifikováno svým jménem *name* (nerozlišují se velká a malá písmena).

- Pokud v seznamu lokálních maker není makro odpovídajícího jména, funkce neprovede nic a vrátí FALSE.
- Pokud odpovídající makro existuje a v seznamu globálních maker se makro tohoto jména nevyskytuje, funkce provede zkopírování (resp. přesun) a vrátí TRUE.
- Pokud makro téhož jména v seznamu globálních maker již existuje, podle parametru prez se funkce zachová takto:

při hodnotě *prez* = 0
objeví se dialog informující, že makro tohoto jména již existuje; podle reakce uživatele se nahradí:
— pouze kód makra (návratová hodnota TRUE) nebo
— nahradí se vše (návratová hodnota TRUE) nebo
— neprovede se nic (návratová hodnota FALSE).

při hodnotě *prez* = 1
nahradi se pouze kód makra; návratová hodnota bude TRUE.

při hodnotě *prez* = 2
Nahradí se vše (i popis, kláv. zkratka); návratová hodnota bude TRUE.

při hodnotě *prez* = 3
neprovede se nic; návratová hodnota bude FALSE.

Poznámka: Funkce vrátí FALSE (a neprovede nic) také při nedostatku paměti pro kopírování, resp. přesun.

Omezení: Potřebuje aktuální dokument.

Viz: **MacroToLocal**

MacroToLocal

```
function MacroToLocal(
  var name : const string;
  copy : boolean;
  prez : short) : boolean;
```

Parametry:
name jméno makra
copy kopírovat anebo přesunout
prez činnost při existenci makra téhož jména

Implicitní hodnoty:
copy = FALSE
prez = 0

Popis: Funkce slouží pro přesun anebo kopírování makra za úložny globálních maker do úložny lokálních maker aktuálního dokumentu.

Při hodnotě parametru:
copy = TRUE se provádí kopírování
copy = FALSE se makro přesune a ze seznamu globálních maker je tedy vypuštěno.

Makro je identifikováno svým jménem *name* (nerozlišují se velká a malá písmena).

- Pokud v seznamu globálních maker není makro odpovídajícího jména, funkce neprovede nic a vrátí FALSE.
- Pokud odpovídající makro existuje a v seznamu lokálních maker se makro tohoto jména nevyskytuje, funkce provede zkopírování (resp. přesun) a vrátí TRUE.
- Pokud makro téhož jména v seznamu lokálních maker již existuje, podle parametru prez se funkce zachová takto:

při hodnotě *prez* = 0
objeví se dialog informující, že makro tohoto jména již existuje; podle reakce uživatele se nahradí:
- pouze kód makra (návratová hodnota TRUE); nebo
- nahradí se vše (návratová hodnota TRUE); nebo
- neprovede se nic (návratová hodnota FALSE).

při hodnotě *prez* = 1
nahradí se pouze kód makra; návratová hodnota bude TRUE.

při hodnotě *prez* = 2
nahradí se vše (i popis, kláv. zkratka), návratová hodnota bude TRUE.

při hodnotě *prez* = 3
neprovede se nic; návratová hodnota bude FALSE.

Poznámka: Funkce vrátí FALSE (a neprovede nic) také při nedostatku paměti pro kopírování, resp. přesun.

Omezení: Potřebuje aktuální dokument.

Viz: **MacroToGlobal**

MailMergeType

```
function MailMergeType : short;
```

Popis: Funkce vrací typ databáze nastavené pro slučování.

Hodnota:

0	není nastavena žádná databáze
1	DBF soubor
3	CSV soubor
4	DTA soubor
5	WinBase.

Omezení: Vyžaduje aktuální dokument.

Viz: **RecordCurrent**
RecordFirst
RecordLast
RecordNext
RecordSet
RecordPrev
RecordsCount

Make_date

```
function Make_date(
```

```
  day, month, year : short) : date;
```

Parametry: *Day* den v měsíci (1 až 31)
Month měsíc (1 až 12)
Year rok (nula a více)

Popis: Funkce vytváří datum ze 3 složek: dne, měsíce a roku. Datum musí patřit do našeho letopočtu.

Hodnota: Funkce vrací vytvořené datum ve vnitřním formátu 602Text. Je-li datum zadáno chybně, funkce vrací NONEDATE.

Viz: **Hours**
Day
Month
Year
Today
Make_time
Minutes
Seconds
Sec1000
Now
Day_of_week

Make_directory

```
function Make_directory(
  var dirname : const string) : boolean;
```

Parametry: *dirname* jméno adresáře i s cestou podle konvencí DOSu

Popis: Funkce vytvoří adresář podle parametru *dirname*.

Hodnota: Funkce vrací TRUE, pokud se adresář podařilo vytvořit. V opačném případě vrátí FALSE.

Viz: **Delete_file**

Make_time

```
function Make_time(
  hours, minutes, seconds, sec1000 : short) : time;
```

Parametry: *hours* počet hodin
minutes počet minut
seconds počet sekund
sec1000 počet setin sekundy

Popis: Funkce vytváří údaj o čase ze 4 složek: hodin, minut, sekund a tisícin sekundy. Povolený rozsah pro hodiny je 0 až 23, pro minuty a sekundy 0 až 59, pro tisíciny 0 až 999.

Hodnota: Funkce vrací vytvořený údaj o čase ve vnitřním formátu 602Text.

Viz: **Make_date**
Hours
Day
Month
Year
Today

Minutes
Seconds
Now
Sec1000

ManualScaleDlg

```
function ManualScaleDlg : short;
```

Popis: Funkce vyvolá dialog **Zvětšení** (menu **Zobrazit**). Dle způsobu jeho opuštění vrací IDOK nebo IDCANCEL.

Omezení: Vyžaduje neminimalizovaný aktuální dokument.

Viz: **GetScale**
SetScale
SetScaleFullPage

MarkModified

```
procedure MarkModified;
```

Popis: Procedura nastaví interní indikátor testující, zda se obsah aktuálního dokumentu změnil od posledního uložení do odpovídajícího souboru.

Omezení: Potřebuje aktuální dokument.

Viz: **IsModified**
MarkUnmodified

MarkName

```
function MarkName(
  index : integer) : string;
```

Parametry: *index* pořadové číslo záložky (1<= index <= MarksCount)

Popis: Funkce vrací jméno index-té záložky.

Omezení: Požaduje neminimalizovaný aktuální dokument.

Viz: **MarksCount**
DoesMarkExist
GoToMark
EditMarksDlg
InsertMark
ClearMark

MarksCount

```
function MarksCount : integer;
```

Popis: Funkce vrací počet záložek v aktuálním dokumentu.

Omezení: Požaduje neminimalizovaný aktuální dokument.

Viz: **MarkName**
DoesMarkExist
GoToMark
EditMarksDlg
InsertMark
ClearMark

MarkUnmodified

```
procedure MarkUnmodified;
```

Popis: Procedura vymaže interní indikátor změny dokumentu od posledního uložení do souboru.

Omezení: Potřebuje aktuální dokument.

Viz: **IsModified**
MarkModified

Memcpy

```
procedure Memcpy(
  var destination, source;
  size : short);
```

Parametry:

destination	proměnná, do níž se kopíruje
source	proměnná, z níž se kopíruje
size	počet přenášených bajtů

Popis: Procedura slouží pro přenos (kopírování) dat bez ohledu na jejich typy. Přenáší se počet bajtů zadaný parametrem *size* z proměnné *source* do proměnné *destination*. Je-li tento počet větší, než je velikost *destination*, může snadno dojít ke zhroucení systému. Je-li tento počet větší, než je velikost *source*, není definováno, co se přenese. Obě proměnné mohou být libovolného typu.
Tato procedura dovoluje obcházet typové kontroly jazyka. Je to její hlavní účel.

Viz: **sizeof**

MenuCreate

```
function MenuCreate : short ;
```

Popis: Funkce vytváří interní šablonu uživatelského menu, existující po dobu běhu makra. Na tuto šablonu se odvoláme identifikátorem, který je návratovou hodnotou této funkce.

Viz: **MenuDestroy**
MenuStrDelete
MenuStrGet
MenuStrCheck
MenuStrDisable
MenuRun

MenuDestroy

```
procedure MenuDestroy(
```

```
  menu_id : short);
```

Parametry: *menu_id* identifikátor uživatelského menu

Popis: Procedura destruuje (uvolňuje z paměti) interní šablonu uživatelského menu, dříve tvořenou funkcí **MenuCreate**.

Viz: **MenuCreate**
MenuRun
MenuStrAdd
MenuStrDelete
MenuStrGet
MenuStrCheck
MenuStrDisable

MenuRun

```
function MenuRun(
  menu_id, x, y : short) : short;
```

Parametry:

menu_id	identifikátor uživatelského menu
x	horizontální souřadnice levého horního rohu
y	vertikální souřadnice levého horního rohu

Popis: Funkce zobrazí uživatelské menu, jehož šablona byla dříve vytvořena funkcí **MenuCreate**. Souřadnice *x, y* jsou v pixlech; v případě nevhodně zadaných souřadnic (např. je-li jedna z nich záporná) jsou tyto parametry ignorovány a menu je umístěno uprostřed obrazovky. Návratovou hodnotou je identifikátor položky menu vybrané z menu nebo z klávesnice.

Viz: **MenuCreate**
MenuDestroy
MenuStrAdd
MenuStrDelete
MenuStrGet
MenuStrCheck
MenuStrDisable

MenuStrAdd

```
function MenuStrAdd(
  menu_id, str_id : short;
  var str : const string) : boolean;
```

Parametry:

menu_id	identifikátor uživatelského menu
str_id	identifikátor přidávané položky menu
str	text přidávané položky menu

Popis: Funkce do uživatelského menu zadaného identifikátorem *menu_id* přidává novou položku, jejímž obsahem je textový řetězec *str*. Pokud již v tomto menu existuje položka identifikovaná identifikátorem *id* je starý obsah této položky nahrazen novým.

Poznámka: Je-li řetězec *str* délky 1 a platí `str [1] = chr (9)`, bude v menu namísto tohoto řetězce tzv. separátor.

Viz: **MenuCreate**
MenuDestroy
MenuRun
MenuStrDelete
MenuStrGet

MenuStrCheck
MenuStrDisable

MenuStrDelete

```
procedure MenuStrDelete(
  menu_id, str_id : short);
```

Parametry: *menu_id* identifikátor uživatelského menu
str_id identifikátor položky menu

Popis: Funkce z uživatelského menu zadaného identifikátorem *menu_id* vypouští položku identifikovanou identifikátorem *id*.

Viz: **MenuCreate**
MenuDestroy
MenuRun
MenuStrAdd
MenuStrGet
MenuStrDisable
MenuStrCheck

MenuStrDisable

```
procedure MenuStrDisable(
  menu_id, str_id, state : short);
```

Parametry: *menu_id* identifikátor uživatelského menu
str_id identifikátor položky menu
state specifikace nastavení:
0 – povoleno
1 – zašednutí a potlačení
2 – zakázáno

Popis: Funkce nastavuje přístupnost (nebo zašednutí) položky uživatelského menu.

Viz: **MenuCreate**
MenuDestroy
MenuRun
MenuStrDelete
MenuStrGet
MenuStrCheck
MenuStrAdd

MenuStrGet

```
function MenuStrGet(
  menu_id, str_id : short) : string;
```

Parametry: *menu_id* identifikátor uživatelského menu
str_id identifikátor položky menu

Popis: Návratovou hodnotou funkce je obsah (textový řetězec) položky uživatelského menu.

Viz: **MenuCreate**
MenuDestroy
MenuRun

MenuStrAdd
MenuStrDelete
MenuStrCheck
MenuStrDisable

MenuStrCheck

```
procedure MenuStrCheck(
  menu_id, str_id : short
  on : boolean);
```

Parametry:
menu_id identifikátor uživatelského menu
str_id identifikátor položky menu
on určuje, zda má být položka zaškrtnutá

Popis: Funkce nastavuje zaškrtnutí položky s identifikátorem *str_id* v uživatelském menu *id*.

Viz:
MenuCreate
MenuDestroy
MenuRun
MenuStrDelete
MenuStrGet
MenuStrDisable

Minutes

```
function Minutes(
  tm : time) : short;
```

Parametry: *tm* údaj o čase ve vnitřním formátu 602Text

Popis: Funkce extrahuje počet minut z časového údaje.

Hodnota: Funkce vrací počet minut z intervalu 0 až 59.

Viz:
Hours
Day
Month
Year
Today
Make_time
Make_date
Seconds
Now
Sec1000

Money2str

```
function Money2str(
  m : money;
  prez : short) : string;
```

Parametry:
m hodnota, která se má konvertovat
prez způsob konverze

Popis: Funkce převede hodnotu zadanou prvním parametrem na řetězec znaků a vrátí jej. Při konverzi se použije hodnota parametru *prez* takto:

- Je-li *prez* **nulové** desetinná část bude oddělena čárkou a je-li nulová, bude nahrazena pomlčkou (např. 89,30 nebo 123,-).
- Je-li *prez* **nenulové** desetinná část bude obsahovat dvě číslice oddělené od celé části tečkou (např. 89.30 nebo 123.00).

Hodnota: Funkce vrací desítkový zápis čísla *m*.

Viz: **Int2str**
Str2int
Real2str
Str2real
Str2money
Date2str
Str2time
Str2date
Time2str

Month

```
function Month(
  dt : date) : short;
```

Parametry: *dt* datum ve vnitřním formátu 602Text

Popis: Funkce extrahuje měsíc z data.

Hodnota: Funkce vrací číslo označující měsíc, toto číslo leží v intervalu 1 až 12.

Viz: **Hours**
Day
Year
Today
Make_time
Make_date
Minutes
Seconds
Sec1000
Now
Day_of_week

NewFile

```
function NewFile(
  macro : boolean) : boolean;
```

Parametry: *macro* platnost režimu editace maker

Implicitní hodnota: *macro* = FALSE

Popis: Funkce otevře nové okno s prázdným dokumentem. Vrátí TRUE, pokud byl nový dokument skutečně založen nebo FALSE, pokud okno nelze otevřít (při překročení maximálního počtu oken nebo stisku tlačítka **Cancel** při dotazu na uložení předchozího neuloženého dokumentu apod.). Podle parametru *macro* se nastaví nebo nenastaví režim "zobrazit jako zdroj makra".

Viz: **DocClose**
OpenFile
SaveFile

NextCell

```
function NextCell : boolean;
```

Popis: Umístění kurzoru do následující buňky v textové tabulce. Při chybě vrací FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru umístěném v textové tabulce.

Viz: **CaretCellStart**
GetCaretCell
GetCaretRow
PrevCell
PrevRow
NextRow
GetNumCells
GetNumRows

NextMMField

```
function NextMMField : integer;
```

Popis: V dokumentu označí nejbližší následující pole pro slučování umístěné za aktuální pozicí kurzoru. Vrací jeho index - lze použít jako parametr ve funkci **GetTextMMI**.

Omezení: Potřebuje neminimalizovaný aktuální dokument napojený na databázi.

Viz: **MailMergeType**
RecordsCount
RecordFirst
RecordLast
RecordNext
RecordPrev
RecordCurrent
RecordSet
GetTextMMI
GetTextMM
SetMM

NextPara

```
function NextPara(
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Popis: Funkce přesune kurzor na začátek dalšího odstavce (pokud takový odstavec existuje). Pokud se řádkový kurzor pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **PrevPara**

NextRow

```
function NextRow : boolean;
```

Popis: Umístění kurzoru do předchozího řádku v textové tabulce. Při chybě vrací FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru umístěném v textové tabulce.

Viz: **CaretCellStart**
GetcaretCell
GetCaretRow
PrevCell
NextCell
PrevRow
GetNumCells
GetNumRows

NextWindow

```
function NextWindow : boolean;
```

Popis: Pokud je počet oken větší než jedna (viz **CountWindows**), funkce přepne z aktivního okna na další okno (jako **Ctrl+F6**) a vrátí TRUE. Jinak vrátí FALSE.

Viz: **PreviousWindow**
CountWindows
DoesWindowExist
SwitchToWindow
GetDocWndTittle
GetDocFileName

Now

```
function Now : time;
```

Popis: Funkce je bez parametrů a udává okamžitý čas ve vnitřním formátu 602Text**.**

Hodnota: Funkce vrátí čas, který získá dotazem u operačního systému počítače. Tento čas je s přesností na sekundy.

Viz: **Hours**
Day
Month
Year
Today
Make_time
Make_date
Minutes
Seconds
Sec1000

NullCmd

```
procedure NullCmd;
```

Popis: Žádná operace. Může být využita např. chceme-li, aby stisk některé kombinace kláves neměl žádnou odezvu.

ObjectsCount

```
function ObjectsCount : integer;
```

Popis: Funkce vrací počet všech objektů nacházejících se v aktuálním dokumentu.

POZOR – každé seskupení objektů je do tohoto počtu též započítáváno jako samostatný objekt.

Omezení: Vyžaduje aktuální dokument.

Viz: **DoesObjectExist**
ObjectId
DeleteObject
DeleteObjGroup
ObjectPropDlg

ObjectId

```
function ObjectId(
  index : integer) : integer;
```

Parametry: *index* index objektu

Popis: Funkce pro objekt zadaného indexu (od 1 do **ObjectsCount** včetně) vrací jeho identifikátor, tj. s ním spojené číslo, které je zobrazováno v dialogu **Seznam objektů** (menu **Úpravy**).

Poznámka: Identifikátor objektu má (na rozdíl od přístupu k objektům přes indexy) tu výhodu, že při vkládání nebo mazání objektů se identifikátory zbylých objektů nemění. Proto v procedurách (funkcích) "operujících" s jednotlivým objektem je tento reprezentován zásadně svým identifikátorem.

Omezení: Vyžaduje aktuální dokument.

Viz: **ObjectsCount**
DoesObjectExist
DeleteObject
DeleteObjGroup
ObjectPropDlg

ObjectIsSelected

```
function ObjectIsSelected(
  id : integer) : boolean;
```

Parametry: *id* identifikátor objektu

Popis: Funkce zjišťuje, zda je objekt vybrán (označen).

Omezení: Vyžaduje aktuální dokument.

Viz: **ObjectSelId**
ObjectSelModify
ObjectSelCount

DeleteObject
DeleteObjGroup
ObjectPropDlg

ObjectListDlg

```
procedure ObjectListDlg;
```

Popis: Procedura vyvolá dialog **Seznam objektů** (menu **Úpravy**).

Omezení: Potřebuje neminimalizovaný aktuální dokument.

ObjectPropDlg

```
function ObjectPropDlg(
  id, init : integer) : short;
```

Parametry: *id* identifikátor objektu
init číslo karty pro inicializaci

Implicitní hodnota: *init* = 0

Popis: Pokud v aktuálním dokumentu existuje objekt zadaného identifikátoru *id* funkce vyvolá dialog **Vlastnosti *xxxx***, přičemž parametr *init* ovlivňuje, která karta bude při inicializaci dialogu zobrazena jako první.
0 – Umístění
1 – Zarovnání
2 – Obtékán
3 – Čáry a stínování
4 – Různé

Podle způsobu opuštění dialogu funkce pak vrací IDOK/IDCANCEL. Pokud objekt daný identifikátorem *id* neexistuje, funkce neprovede nic a vrátí IDERROR.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **ObjectId**
ObjectIsSelected
ObjectsCount
ObjectSelCount
ObjectSelId
ObjectSelModify
ObjectType
DeleteObject
DeleteObjGroup

ObjectSelCount

```
function ObjectSelCount : integer;
```

Popis: Funkce vrací počet vybraných objektů.

Omezení: Vyžaduje aktuální dokument.

Viz: **ObjectIsSelected**
ObjectSelId

ObjectSelModify
DeleteObject
DeleteObjGroup
ObjectPropDlg

ObjectSelId

```
function ObjectselId(
  index : integer) : integer;
```

Parametry: *index* číslo od 1 do ObjectSelCount ( )

Popis: Funkce vrací číslo index-tého vybraného objektu.

Omezení: Vyžaduje aktuální dokument.

Viz: **ObjectIsSelected**
ObjectSelModify
ObjectSelCount
DeleteObject
DeleteObjGroup
ObjectPropDlg

ObjectSelModify

```
procedure ObjectSelModify(
  id : integer;
  add : boolean);
```

Parametry: *id* identifikátor objektu
add určuje, zda objekt přidat k dalším vybraným objektům (TRUE) nebo naopak ubrat

Popis: Funkce přidává nebo ubírá objekt do vícenásobné selekce; selekce ostatních objektů se nemění.

Omezení: Vyžaduje aktuální dokument.

Viz: **ObjectIsSelected**
ObjectSelId
ObjectSelCount
DeleteObject
DeleteObjGroup
ObjectPropDlg

ObjectType

```
function ObjectType(
  id : integer) : integer;
```

Parametry: *id* identifikátor objektu

Popis: Funkce zjišťuje typ objektu zadaného identifikátorem *id*.
Pro různé typy objektů vrací funkce tyto hodnoty:

1 vodorovná čára
2 svislá čára
3 čtverec nebo obdélník

4	kružnice nebo elipsa
5	čtverec nebo obdélník se zaoblenými rohy
6	textový rámec
7	textová tabulka
8	prázdný rámec
9	obrázek
10	OLE objekt
11	seskupení objektů
12	vypínač
13	přepínač
14	tlačítko pro odeslání obrázku
15	tlačítko Reset
16	pole pro jednoduchý text
17	pole pro víceřádkový text
18	nabídka
19	pole pro heslo
20	pole pro skrytý text

Pokud v aktuálním dokumentu objekt se zadaným identifikátorem neexistuje, funkce vrátí NONEINTEGER.

Omezení: Vyžaduje aktuální dokument.

Viz: **ObjectsCount**
ObjectId
GetCaretObjectId
DeleteObject
DeleteObjGroup
ObjectPropDlg

Odd

```
function Odd(
  i : integer) : boolean;
```

Parametry: *i* celé číslo

Popis: Funkce zjišťuje, zda číslo je sudé nebo liché.

Hodnota: Funkce má hodnotu TRUE, pokud je její parametr lichý. Pokud je parametr sudý má hodnotu FALSE.

OpenFile

```
function OpenFile(
  dlg : boolean;
  var path_name : const string;
  format : integer;
  code : short) : short;
```

Parametry:

dlg	zobrazovat dialog
path_name	název souboru
format	typ souboru
code	kódování souboru

Implicitní hodnoty:
path_name = "*.wpd"
format = ftWT_30
code = ctWINEE

Popis: Funkce otevírá existující dokument buď prostřednictvím dialogu vyvolávaného příkazem **Otevřít** anebo přímo.
Význam jednotlivých parametrů:

1. dlg =
TRUE - vyvolá zmíněný dialog a další činnost se řídí reakcí uživatele
FALSE - přímo se otevře dokument specifikovaný zbylými parametry

2. *path_name*:
Specifikuje název souboru, event. i s diskem a cestou, např.:
`c:\602text\makra\macro02.wpm`

Nejsou-li disk nebo cesta udány, doplní se implicitními hodnotami.

Je možno užít i globální znaky **'?'** a **'* '** např. :
`"c:\602text\makra\*.wpm"`
`"c:\texty\pozn?.txt"`
což má ovšem smysl pouze pro *dlg* = TRUE.

Jako implicitní hodnota se užije "*.wpd" (při `OpenFile(TRUE)`) .

3. *format*:
Specifikuje typ souboru. Je nutno použít některou z předdefinovaných konstant.

INTERNÍ FORMÁT

Identifikátor	Význam
ftT602	Text602
ftWT_10	WinText 1.0
ftWT_20	WinText 2.0
ftWT_21	WinText 2.1
ftWT_30	WinText 3.0
ftWT_30_macro	WinText 3.0
ftWT_30_template	WinText 3.0 – šablona
tWt_50	WinText 5.0 (a novější: 602Text)
ftWT_50_macro	WinText 5.0 (a novější: 602Text)
ftWT_50_template	WinText 5.0 (a novější: 602Text) - šablona
ftHTML	HTML

EXTERNÍ FORMÁTY

ASCII STANDARD

Identifikátor	Význam
ftASCII_PC	standard pro PC (DOS)
ftASCII_MAC	standard pro Macintosh
ftASCII_Win	standard pro PC (Windows)
ftASCII_UNIX	standard pro Unix

Pro *dlg* = TRUE:- parametr format je ignorován.

Pro dlg = FALSE - pro interní formáty je činnost prakticky stejná (není-li dokument konkrétního zadaného formátu, zkusí se otevřít v ostatních formátech). Pouze pro `ftWT_30_macro` se navíc po otevření nastaví režim "makro".

Pro externí formáty vrátí funkce IDERROR, není-li přítomna importní .dll knihovna pro zadaný formát.

Jako implicitní hodnota se použije formát `= ftWT_30`
`(OpenFile(false, "c:\602text\makra\macro02.wpm")).`

4. *code*:
Specifikuje kódování souboru. Je nutno použít některou z předdefinovaných konstant:

ctKEYBCS2	ctWINANSI
ctLATIN2	ctWINEE
ctKOI8CS	ctMAC
ctCP852	ctMAZOWIA

Jako implicitní hodnota se použije `code = ctWINEE.`

5. ***návratová hodnota* :**
IDOK - je-li soubor správně otevřen.
IDCANCEL v jedné ze situací :
bylo-li *dlg* = TRUE a dialog opuštěn **Esc**
- soubor už je otevřen, jsou v něm změny a na dotaz: **Načíst znovu?** odpoví uživatel **Ne.**
- "nové okno" je na FALSE a při čtení do okna souboru, ve kterém jsou změny, na dotaz: **Uložit soubor?** odpoví uživatel **Cancel.**
- příliš mnoho oken = IDERROR, nepodařilo-li se soubor otevřít
(chybné hodnoty parametru *path_name, format, code* atd.).

Viz: **RecentFilesCount**
RecentFileName
SaveFile

Ord

```
function Ord(
  t : ordinaltype) : integer;
```

Parametry: *t* výraz libovolného ordinálního typu

Popis: Funkce vrací ordinální hodnotu příslušného výrazu. Je-li tedy např. výraz *t* typu *Char*, funkce konvertuje znaky na celá čísla dle standardu ASCII a východoevropské normy národních znaků ve Windows (tzv. code page 1250).

Poznámka: Další podrobnosti naleznete v popisu vnitřního jazyka (v části týkající se ordinálních typů).

Hodnota: Hodnotou funkce je ordinální hodnota výrazu zadaného parametrem.

Viz: **Succ**
Pred
Chr

PageDown

```
function PageDown[
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Implicitní hodnota: *shift* = FALSE

Popis: Funkce přesune kurzor dolů o celkový počet řádků v okně. Pokud se řádkový kurzor pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharLeft**
CharRight
LineDown
LineUp
CaretEnd
CaretHome
LeftOfLine
RightOfLine
WordLeft

WordRight
RightOfWord
PageUp
BottomOfScreen
TopOfScreen

PageFormat

```
function PageFormat(
  dlg : boolean;
  sizetype : short;
  portrait : boolean;
  width, height : real) : short;
```

Parametry:

dlg	zobrazovat dialog
sizetype	formát stránky
portrait	orientace stránky
width	šířka stránky
height	výška stránky

Popis: Funkce odpovídá položce menu **Soubor** příkaz **Formát stránky:**

Pro ***dlg* = TRUE** se vyvolá dialog inicializovaný zadanými parametry:

sizetype = 0	A4
1	Dopis
2	Legal
3	Nastavená uživatelem

Při jiných (chybných) hodnotách se použije aktuální hodnota *sizetype*.

Portrait = TRUE : orientace na výšku (Portrait)
Portrait = FALSE : orientace na šířku (Landscape)

Parametry *width, height* mají význam jen při sizetype = 3 (nastavené uživatelem), jinak se ignorují. Výška i šířka se zadávají v aktuálních jednotkách, jaké jsou zvoleny v dialogu pro nastavení. Ovšem ne všechny číselné hodnoty jsou přípustné (záporné či příliš velké; závisí právě na aktuálních jednotkách). Při zadání takových hodnot se dialog jimi neinicializuje, ale použije se aktuální *sizetype, width, height*.
Návratová hodnota: IDOK či IDCANCEL.

Pro ***dlg* = FALSE** se přeskočí dialog a parametry se rovnou použijí pro nastavení s tím rozdílem, že pro chybné *sizetype, width, height* se neprovede nic a návratová hodnota bude IDERROR.
Návratová hodnota : IDERROR či IDOK.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **GetPageWidth**
GetPageHeight
GetPageType
IsPageLandscape
IsPagePortrait

PagesDisplayed

```
function PagesDisplayed : boolean;
```

Popis: Funkce zjišťuje, zda-li jsou zobrazeny stránky (TRUE) či osnova.

Omezení: Vyžaduje neminimalizovaný aktuální dokument.

Viz: **DisplayPages**
DisplayOutline

PageUp

```
function PageUp(
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Implicitní hodnota: *shift* = FALSE

Popis: Funkce přesune kurzor nahoru o celkový počet řádků v okně. Vrací TRUE, pokud se řádkový kurzor pohnul.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharLeft**
CharRight
LineDown
LineUp
CaretEnd
CaretHome
LeftOfLine
RightOfLine
WordLeft
WordRight
RightOfWord
PageDown
BottomOfScreen
TopOfScreen

Pred

```
function Před(
  t : ordinaltype) : ordinaltype;
```

Parametry: *t* výraz libovolného ordinálního typu

Popis: Funkce vrací pro hodnotu *t* libovolného ordinálního typu hodnotu jejího předchůdce (je-li ovšem tento definován; v opačném případě hodnota výsledku není definována).

Poznámka: Další podrobnosti naleznete v popisu vnitřního jazyka (v části týkající se ordinálních typů).

Hodnota: Hodnotou funkce je následník výrazu zadaného parametrem; je téhož typu jako tento parametr.

Viz: **Ord**
Succ
Chr

Pref

```
function Pref(
  var pre, str : count string) : boolean;
```

Parametry: *pre* řetězec znaků, který má být prefixem *str*
str řetězec znaků, který má mít prefix p*re*

Popis: Funkce zjišťuje, zda řetězec *pre* je prefixem řetězce *str*, tedy zda řetězec *str* začíná řetězcem *pre* a pak případně (jakkoli) pokračuje.

Hodnota: Pokud je *pre* prefixem *str* funkce vrátí TRUE - jinak vrátí FALSE.

PreferencesDlg

```
function PreferencesDlg : short;
```

Popis: Funkce vyvolá dialog **Předvolba**. Dle způsobu jeho opuštění vrací IDOK nebo IDCANCEL.

Viz: **GetPreferences**
SetPreferences

PrevCell

```
function PrevCell : boolean;
```

Popis: Umístění kurzoru do předchozí buňky v textové tabulce. Při chybě vrací FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru umístěném v textové tabulce.

Viz: **CaretCellStart**
GetCaretCell
GetCaretRow
PrevCell
PrevRow
NextRow
GetNumCells
GetNumRows

PreviousWindow

```
function PreviousWindow : boolean;
```

Popis: Je-li otevřeno více oken s dokumenty (CountWindows > 1), funkce z aktivního okna přepne na předchozí okno (jako **Ctrl+Shift+F6**) a vrátí TRUE. Jinak vrací FALSE.

Viz: **NextWindow**
CountWindows
DoesWindowExist
SwitchToWindow
GetDocWndTittle
GetDocFileName

PrevPara

```
function PrevPara(
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Popis: Pokud se řádkový kurzor nachází na začátku odstavce je přesunut na začátek předchozího odstavce. V opačném případě je kurzor přesunut na začátek odstavce ve kterém se nachází.

Pokud se řádkový kurzor pohnul je návratová hodnota TRUE .

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **NextPara**

PrevRow

```
function PrevRow : boolean;
```

Popis: Umístění kurzoru do předchozího řádku v textové tabulce. Při chybě vrací FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řídkovém kurzoru umístěném v textové tabulce.

Viz: **CaretCellStart**
GetCaretCell
GetCaretRow
PrevCell
NextCell
NextRow
GetNumCells
GetNumRows

Print

```
function Print(
  dlg : boolean;
  kopii, stranky, firstpage, lastpage : short) : short;
```

Parametry: *dlg* zobrazovat dialog
kopii počet tištěných kopií
stranky všechny, sudé nebo liché
firstpage první strana
lastpage poslední strana

Popis: Funkce odpovídá položce menu **Soubor** příkaz **Tisk.**

Pro ***dlg* = TRUE** se zavolá tento dialog, přičemž se inicializuje zadanými parametry :

Kopii – mezi 1 a 99; při jiných (chybných) hodnotách se použije hodnota 1.
Stranky – kPrintAllPages (0) – všechny
– kPrintEvenPages (1) – sudé
– kPrintOddPages (2) – liché
Při jiných (chybných) hodnotách se použije 0 – všechny.

Firstpage, lastpage - hodnoty first*p*age = *lastpage* = 0 se interpretují jako rozsah tisku *Celý dokument*.
Pro jiné přípustné hodnoty (od – do) musí platit:

(1 <= firstpage) && (firstpage <= lastpage) && (lastpage <= GetTotPages).
Pokud toto neplatí, použijí se hodnoty:
firstpage = 1
lastpage = GetTotPages.

Návratová hodnota:
IDOK – obsah tisku byl předán správně Print Manageru.
IDCANCEL – dialog je opuštěn klávesou **Esc** nebo během předávání tisku bylo zvoleno **Zrušit tisk**.
IDERROR – při tisku došlo k chybě.

Pro ***dlg* = FALSE** se přeskočí dialog. Parametry se rovnou použijí pro tisk s tím rozdílem, že pro chybné hodnoty parametru (*kopii, stranky, firstpage, lastpage*) se neprovede nic a návratová hodnota bude IDERROR.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **GetTotPages**

PrnAdvancedDlg

```
function PrnAdvancedDlg : short;
```

Popis: Funkce zobrazí dialog pro specifická nastavení tiskárny.

Viz: **PrnGetCurrent**
PrnIsPortrait
PrnListCount
PrnListStr
PrnSetCurrent
PrnSetPortrait
PrnSetupDlg

PrnGetCurrent

```
function PrnGetCurrent(
  list : short) : string;
```

Parametry: *list* typ požadované informace:
0 – tiskárna
1 – formát papíru
2 – typ podavače

Hodnota: Řetězec, který je aktuálně nastaven pro zvolený list.

Popis: Funkce vrací aktuální nastavení tiskárny, tj. jméno tiskárny, použitý papír a podavač.

Viz: **PrnAdvancedDlg**
PrnIsPortrait
PrnListCount
PrnListStr
PrnSetCurrent
PrnSetPortrait
PrnSetupDlg

PrnIsPortrait

```
function PrnIsPortrait : boolean;
```

Hodnota: TRUE = na výšku

Popis: Funkce zjišťuje, zda je tiskárna připravena tisknout **Na výšku** nebo **Na šířku**.

Viz: **PrnGetCurrent**
PrnAdvancedDlg
PrnListCount
PrnListStr
PrnSetCurrent
PrnSetPortrait
PrnSetupDlg

PrnListCount

```
function PrnListCount(
  list : short) : short;
```

Parametry: *list* typ požadované informace:
0 – seznam instalovaných tiskáren
1 – seznam formátů papíru
2 – seznam typů podavačů

Hodnota: Řetězec, který je aktuálně nastaven pro zvolený seznam.

Popis: Funkce vrací počet možných nastavení zvoleného parametru.

Viz: **PrnGetCurrent**
PrnAdvancedDlg
PrnIsPortrait
PrnListStr
PrnSetCurrent
PrnSetPortrait
PrnSetupDlg

PrnListStr

```
function PrnListStr(
  list, index : short) : string;
```

Parametry: *list* typ požadované informace:
0 – seznam instalovaných tiskáren
1 – seznam formátů papíru
2 – seznam typů podavačů

index pořadové číslo od 1 do PrnListCount

Hodnota: Řetězec odpovídající indexu ve zvoleném seznamu vlastností.

Popis: Dle pořadového čísla a specifikace vlastnosti zjišťuje funkce možné nastavení parametrů tiskárny (použitý papír, podavač), resp. různá nastavení tiskárny.

Viz: **PrnGetCurrent**
PrnAdvancedDlg
PrnIsPortrait
PrnListCount
PrnSetCurrent

PrnSetPortrait
PrnSetupDlg

PrnSetCurrent

```
function PrnSetCurrent(
  list : short;
  var item : const string) : boolean;
```

Parametry: *list* typ požadované informace:
0 – seznam instalovaných tiskáren
1 – seznam formátů papíru
2 – seznam typů podavačů
item řetězec definující požadované nastavení

Hodnota: TRUE = podaří-li se nastavit tiskárnu požadovaným způsobem.

Popis: Funkce nastavuje parametry tiskárny (použitý papír, podavač), resp. samu tiskárnu.

Omezení: Řetězec pro nastavení vlastností lze získat v dialogu pro nastavení tiskárny případně pomocí funkce **PrnListStr**.

Viz: **PrnGetCurrent**
PrnAdvancedDlg
PrnIsPortrait
PrnListCount
PrnListStr
PrnSetPortrait
PrnSetupDlg

PrnSetPortait

```
procedure PrnSetPortrait(
  on : boolean);
```

Parametry: *on* způsob nastavení tiskárny (TRUE = na výšku)

Popis: Funkce nastavuje tiskárnu pro tisk **Na výšku** resp. **Na šířku**.

Viz: **PrnGetCurrent**
PrnAdvancedDlg
PrnIsPortrait
PrnListCount
PrnListStr
PrnSetCurrent
PrnSetPortrait
PrnSetupDlg

PrnSetupDlg

```
function PrnSetupDlg : short;
```

Popis: Funkce zobrazí dialog pro základní nastavení tiskárny.

Viz: **PrnGetCurrent**
PrnAdvancedDlg
PrnIsPortrait

PrnListCount
PrnListStr
PrnSetCurrent
PrnSetPortrait

RadioBtnCheck

```
procedure RadioBtnCheck(
  dlg_id, id : short);
```

Parametry: *dlg_id* identifikátor uživatelského dialogu
id identifikátor přepínače

Popis: Procedura nastavuje u přepínače identifikovaném identifikátorem *id* stav "zatrženo". U toho přepínače (ze skupiny přepínačů), který měl již stav "zatrženo" nastaven před zavoláním procedury, je tento stav zrušen.

Viz: **RadioBtnChecked**
RadioGrpGetVal
RadioGrpSetVal
DlgRadioBtn
DlgRadioBtnGroup

RadioBtnChecked

```
function RadioBtnChecked(
  dlg_id, id : short) : boolean;
```

Parametry: *dlg_id* identifikátor uživatelského dialogu
id identifikátor přepínače

Popis: Funkce slouží ke zjištění, zda přepínač identifikovaný identifikátorem *id* je ve stavu "zatrženo". Pokud ano, vrací TRUE, jinak je návratová hodnota FALSE.

Viz: **RadioBtnCheck**
RadioGrpGetVal
RadioGrpSetVal
DlgRadioBtn
DlgRadioBtnGroup

RadioGrpGetVal

```
function RadioGrpGetVal(
  dlg_id, grp_id : short) : short;
```

Parametry: *dlg_id* identifikátor uživatelského dialogu
grp_id identifikátor skupiny přepínačů

Popis: Funkce pro danou skupinu přepínačů vrací identifikátor toho přepínače, který je zatržen.

Poznámka: Pokud je dotyčná skupina přepínačů prázdná, návratová hodnota funkce je NONESHORT.

Viz: **RadioBtnCheck**
RadioBtnChecked
RadioGrpSetVal

DlgRadioBtn
DlgRadioBtnGroup

RadioGrpSetVal

```
procedure RadioGrpSetVal(
  dlg_id, grp_id, id : short);
```

Parametry:

dlg_id	identifikátor uživatelského dialogu
grp_id	identifikátor skupiny přepínačů
id	identifikátor přepínače

Popis: Procedura nastavuje v zadané skupině přepínačů v zadaném dialogu u přepínače identifikovaném identifikátorem *id* stav "zatrženo". U toho přepínače ze skupiny, který měl stav "zatrženo" nastaven před zavoláním procedury, je tento stav zrušen.

Viz:
RadioBtnCheck
RadioBtnChecked
RadioGrpGetVal
DlgRadioBtn
DlgRadioBtnGroup

Read

```
function Read(
  var f : file;
  var prom) : boolean;
```

Parametry:

f	proměnná typu soubor (file)
prom	proměnná některého z typů Boolean, Char, Short, Integer, Real, String, CSString nebo CSIString

Popis: Funkce čte z textového souboru zadaného prvním parametrem hodnotu do proměnné uvedené jako druhý parametr. Přitom převádí textový zápis hodnoty na hodnotu příslušného typu. Soubor musí být předem otevřen.

Hodnota: Hodnota funkce je TRUE, pokud se povedlo přečíst a zapsat do parametru přečtený údaj. Není-li v souboru hodnota, která by se dala převést na příslušný typ je hodnota funkce FALSE.

Viz:
Write
Writeln
Close
Reset
Rewrite
Seek
Eof
Filelenght

Real2str

```
function Real2str(
  r : real;
  prez : short) : string;
```

Parametry:

r	hodnota, která se má konvertovat
prez	způsob konverze

Popis: Funkce převede hodnotu zadanou prvním parametrem na řetězec znaků a vrátí jej. Při konverzi se použije hodnota parametru *prez* takto:

prez = **absolutní hodnota –** udává počet desetinných míst
prez = **kladné –** výsledek bude v semilogaritmickém tvaru
prez = **záporné** – výsledek bude v desetinném tvaru

Hodnota: Funkce vrací dekadický zápis čísla *r*.

Viz: **Int2str**
Str2int
Str2real
Money2str
Str2money
Date2str
Str2date
Time2str
Str2time

RecentFileName

```
function RecentFileName(
  file_number : short) : string;
```

Parametry: *file_number* číslo souboru z menu

Popis: Funkce vrací z nabízených naposledy otevřených souborů z menu **Soubor** soubor číslo *file_number*.

Poznámka: Číslování těchto souborů začíná od jedničky.

Viz: **RecentFilesCount**

RecentFilesCount

```
function RecentFilesCount : short;
```

Popis: Funkce vrací počet nabízených naposledy otevřených souborů z menu **Soubor.**

Viz: **RecentFileName**

RecordCurrent

```
function RecordCurrent : integer;
```

Popis: Funkce vrací číslo aktuálního záznamu. Pokud není nastavena žádná databáze pro slučování, resp. není nastaven příkaz **Obsah polí** (menu **Zobrazit**) pak je návratová hodnota nula.

Omezení: Vyžaduje aktuální dokument.

Viz: **MailMergeType**
RecordFirst
RecordLast
RecordNext
RecordPrev
RecordsCount

RecordSet

RecordFirst

```
function RecordFirst : boolean;
```

Popis: Funkce nastaví první záznam jako aktuální pro slučování s databází. Není-li nastaven **Obsah polí** (menu **Zobrazit**) dojde zároveň k nastavení tohoto přepínače.

Omezení: Vyžaduje aktuální dokument.

Viz: **MailMergeType**
RecordCurrent
RecordLast
RecordNext
RecordPrev
RecordsCount
RecordSet

RecordLast

```
function RecordLast : boolean;
```

Popis: Funkce nastaví poslední záznam jako aktuální pro slučování s databází. Není-li nastaven **Obsah polí** (menu **Zobrazit**), dojde k nastavení tohoto přepínače.

Omezení: Vyžaduje aktuální dokument.

Viz: **MailMergeType**
RecordCurrent
RecordFirst
RecordNext
RecordPrev
RecordsCount
RecordSet

RecordNext

```
function RecordNext : boolean;
```

Popis: Funkce nastaví následující záznam jako aktuální pro slučování s databází. Není-li nastaveno **Obsah polí** (menu **Zobrazit**) vrací funkce FALSE.

Omezení: Vyžaduje aktuální dokument.

Viz: MailMergeType

RecordCurrent
RecordFirst
RecordLast
RecordPrev
RecordsCount
RecordSet

RecordPrev

```
function RecordPrev : boolean;
```

Popis: Funkce nastaví předcházející záznam jako aktuální pro slučování s databází. Není-li nastaveno **Obsah polí** (menu **Zobrazit**) vrací funkce FALSE.

Omezení: Vyžaduje aktuální dokument.

Viz: **MailMergeTyp**
RecordCurren
RecordFirst
RecordLast
RecordNext
RecordsCount
RecordSet

RecordsCount

```
function RecordsCount : integer;
```

Popis: Funkce vrací počet záznamů v nastavené databázi pro slučování.

Omezení: Vyžaduje aktuální dokument.

Viz: **MailMergeType**
RecordCurrent
RecordFirst
RecordLast
RecordNext
RecordPrev
RecordSet

RecordSet

```
function RecordSet(
  rec_number : integer) : boolean;
```

Parametry: *rec_number* číslo záznamu

Popis: Funkce nastavuje zadaný záznam jako aktuální pro slučování s databází. Není-li nastaveno **Obsah polí** (menu **Zobrazit**) dojde zároveň k nastavení tohoto přepínače.

Omezení: Vyžaduje aktuální dokument.

Viz: **MailMergeType**
RecordCurrent
RecordFirst
RecordLast
RecordNext
RecordPrev
RecordsCount

RemoveIndexEntry

```
function RemoveIndexEntry(
  var entry : const string) : boolean;
```

Parametry: *entry* položka rejstříku (textový řetězec)

Popis: Funkce slouží k odstranění položky *entry* z rejstříku. Pokud se takováto položka v rejstříku nachází, dojde k jejímu odstranění a funkce vrátí TRUE. Jinak se neprovede nic a funkce vrátí FALSE.

Omezení: Potřebuje aktuální dokument.

Viz: **IndexExists**
IndexEntries
IndexEntry
IndexEntryInd
IndexEditDlg
CountOfEntryRef
GoToIndexEntry
AddIndexEntryDlg
AddIndexEntry
SelectIndex

Replace

```
function Replace(
  dlg : boolean;
  var what, by : const string;
  casesenzitiv, wholewords, forward, reserved : integer) : short;
```

Parametry:

dlg	zobrazovat dialog (lze použít jedině FALSE)
what	hledaný řetězec
by	řetězec, kterým se bude nahrazovat
casesenzitiv	rozlišovat malá/velká písmena
wholewords	hledat jen celá slova
forward	hledat dopředu
reserved	zatím nevyužito, musí platit *reserved* = 0

Implicitní hodnoty:
casesenzitiv = 0
wholewords = 0
forward = 0
reserved = 0

Popis: Funkce provádí činnost odpovídající příkazu **Zaměnit** v menu **Úpravy**.
Při hodnotě dlg = TRUE vrací IDERROR.
Hledaný řetězec *what* ani řetězec by neměly by mít větší délku než 127 znaků, jinak jsou interně na tuto délku zkráceny. Celočíselné parametry *casesenzitiv*, *wholewords*, *forward* jsou interpretovány takto:

Hodnota:	Význam:
větší než nula	ANO (TRUE)
menší než nula	NE (FALSE)
rovná nule	použít stav při předchozím hledání.

Návratová hodnota: Funkce může vrátit některou z těchto hodnot:

0	odpovídající řetězec nebyl nalezen
IDERROR	došlo k chybě
128	nejméně jedenkrát proběhlo nalezení řetězce a záměna

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru, který není v zobrazení osnovy.

Viz: **Search**
SearchAgain
SearchIsReady
ReplaceAgain

ReplaceIsReady

ReplaceAgain

```
function ReplaceAgain : boolean;
```

Popis: Funkce provádí opakování činnosti odpovídající příkazu **Zaměnit** z menu **Úpravy** s týmiž parametry jako "posledně" (je-li toto opakování možné vzhledem ke stavu zobrazení dokumentu a je-li co opakovat – viz **ReplaceIsReady**) a pokud je hledání s následnou záměnou úspěšné, vrací TRUE. V opačném případě vrací FALSE.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru, který není v zobrazení osnovy.

Viz: **Search**
SearchAgain
SearchIsReady
Replace
ReplaceIsReady

ReplaceIsReady

```
function ReplaceIsReady(
  toRepeat : boolean) : boolean;
```

Parametry: *toRepeat* zjišťovat možnost opakování

Implicitní hodnota: *toRepeat* = FALSE

Popis: Při implicitní hodnotě parametru ***toRepeat* = FALSE** funkce zjišťuje "připravenost" editoru pro hledání a záměnu textu; tj. vrací TRUE právě tehdy, když není zašedlá položka příkazu **Zaměnit** z menu **Úpravy** (což nastává při neminimalizovaném aktuálním dokument, jenž má aktivní řádkový kurzor a není v zobrazení osnovy).

Při hodnotě ***toRepeat* = TRUE** navíc zjišťuje připravenost pro zopakování posledního nahrazení; tj. zda již byla prováděna funkce **Zaměnit** (ať už prostřednictvím uživatelského rozhraní anebo voláním z makra) a z dvojice funckí **Najít** / **Zaměnit** byla funkce **Zaměnit** prováděna jako poslední.

Viz: **Search**
SearchAgain
SearchIsReady
Replace
ReplaceAgain

Reset

```
function Reset(
  var f : file;
  var filename : const string) : boolean;
```

Parametry: *f* proměnná typu soubor (file)
filename jméno otevíraného souboru (včetně cesty)

Popis: Funkce otevírá textový soubor *f* pro čtení. Parametr *filename* musí označovat jméno existujícího souboru (vytvořené podle konvencí operačního systému). Obsah souboru otevřeného funkcí **Reset** lze číst, je však možno do něj i zapisovat.

Hodnota: Funkce vrací hodnotu TRUE, pokud se soubor podařilo otevřít. Jinak vrací FALSE.

Viz: **Read**
Write
Writeln
Close
Rewrite
Seek
Eof
Filelenght

Rewrite

```
function Rewrite(
  var f : file;
  var filename : const string) : boolean;
```

Parametry:

f	proměnná typu soubor (file)
filename	jméno otevíraného souboru (včetně cesty)

Popis: Funkce vytváří nový textový soubor ***f***. Parametr *filename* označuje jméno, které se přidělí tomuto souboru (vytvořené podle konvencí operačního systému). Pokud soubor stejného jména již existuje, je zrušen a založen znovu. Do souboru otevřeného funkcí **Rewrite** lze zapisovat. Zapsané údaje je však možno i číst.

Hodnota: Funkce vrací hodnotu TRUE, pokud se soubor podařilo vytvořit. Jinak vrací FALSE.

Viz: **Read**
Write
Writeln
Close
Reset
Seek
Eof
Filelenght

RGB

```
function RGB(
  red, green, blue : short) : integer;
```

Parametry:

red	intenzita červené složky
green	intenzita zelené složky
blue	intenzita modré složky

Popis: Funkce podle intenzity zadaných barevných složek vytváří jako návratovou hodnotu jejich barevnou kombinaci. Přípustné hodnoty parametrů jsou v rozmezí 0 až 255 včetně. Hodnoty menší než 0 se interpretují jako 0, hodnoty větší než 255 se interpretují jako 255.

Poznámka: Jsou-li všechny parametry 0, výsledkem je černá. Mají-li všechny parametry hodnotu 255, výsledkem je bílá barva.

Viz: **RgbTrio**
SetRGBColor
SetRGBText

GetRGBText

RgbTrio

```
procedure RgbTrio(
  colorref : integer;
  var red, green, blue : short);
```

Parametry:
colorref barevná kombinace
red intenzita červené složky
green intenzita zelené složky
blue intenzita modré složky

Popis: Funkce zadanou hodnotu barevné kombinace *colorref* rozkládá na jednotlivé barevné složky, které jsou jejími výstupními parametry.

Viz:
RGB
SetRGBColor
SetRGBText
GetRGBText

RightOfLine

```
function RightOfLine(
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Implicitní hodnota: *shift* = FALSE

Popis: Funkce přesune kurzor na konec aktuální řádky. Pokud se řádkový kurzor pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz:
CharLeft
CharRight
LineDown
LineUp
CaretEnd
CaretHome
LeftOfLine
WordLeft
WordRight
RightOfWord
PageDown
PageUp
BottomOfScreen
TopOfScreen

RightOfWord

```
function RightOfWord(
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Implicitní hodnota: *shift* = FALSE

Popis: Funkce přesune kurzor na další znak jež následuje za koncem slova. Pokud se řádkový kurzor pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharLeft**
CharRight
LineDown
LineUp
CaretEnd
CaretHome
LeftOfLine
RightOfLine
WordLeft
WordRight
PageDown
PageUp
BottomOfScreen
TopOfScreen

Round

```
function Round(
  r : real) : integer
```

Parametry: *r* reálné číslo z intervalu hodnot typu Integer

Popis: Funkce zaokrouhlí reálné číslo na nejbližší celé číslo.

Hodnota: Funkce vrací celé číslo nejbližší hodnotě parametru. Pokud je hodnota parametru funkce mimo rozsah typu *Integer*, není hodnota funkce definována.

Rtrim

```
function Rtrim(
  var str : string) : string;
```

Parametry: *str* řetězec znaků

Popis: Funkce odstraní z řetězce *str* mezery na konci. Mezery vyskytující se mezi jinými znaky ponechá v řetězci beze změny. Též ponechá beze změny mezery vyskytující se na začátku (pokud se ovšem řetězec neskládá ze samých mezer). Řetězec *str*, předaný odkazem, je touto funkcí změněn.

Hodnota: Funkce vrací řetězec *str*.

Viz: **Ltrim**
Strtrim

SaveDbRecord

```
function SaveDbRecord : boolean;
```

Popis: Zápis záznamu (vytvořeného voláním funkce **AddToDbRecord**) do přiřazené databáze.

Hodnota: TRUE = byl-li zápis úspěšný

Viz: **SetMM**
DisconnectMM
AddToDbRecord
MailMergeType
RecordsCount
RecordFirst
RecordLast
RecordNext
RecordPrev
RecordCurrent
RecordSet
NextMMField
GetTextMM
GetTextMMI

SaveFile

```
function SaveFile : short;
```

Popis: Funkce uloží aktuální dokument. Při správném uložení vrací IDOK; pokud došlo k chybě vrací IDERROR.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **DocClose**
OpenFile

SaveFileAs

```
function SaveFileAs(
  dlg : boolean;
  var path_name : const string;
  format : integer;
  code : short) : short;
```

Parametry:

dlg	zobrazení dialogu
path_name	plné jméno souboru
format	ukládaný formát
code	typ kódování

Hodnota: TRUE – pokud se podařilo soubor uložit.

Popis: Funkce ukládá aktuální dokument ve zvoleném formátu. Je-li nastaven parametr *dlg* jsou dále zadané hodnoty pouze inicializací ukládaného dialogu.

Omezení: Vyžaduje aktuální dokument.

Viz: **OpenFile**
SaveFile

ScrollLineDown

```
function ScrollLineDown(
  lines : integer) : boolean;
```

Parametry: *lines* počet řádek

Implicitní hodnota: *lines* = 1

Popis: Funkce posune text dolů o specifikovaný počet řádků a zanechá pozici řádkového kurzor beze změny. Pokud se obsah okna pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **FixScreenPos**
ScrollLineUp
ScrollPageDown
ScrollPageUp

ScrollLineUp

```
function ScrollLineUp(
  lines : integer) : boolean;
```

Parametry: *lines* počet řádek

Implicitní hodnota: *lines* = 1

Popis: Funkce posune zobrazený text nahoru o specifikovaný počet řádků; zanechá pozici řádkového kurzoru beze změny. Pokud se obsah okna pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **FixScreenPos**
ScrollLineDown
ScrollPageDown
ScrollPageUp

ScrollPageDown

```
function ScrollPageDown : boolean;
```

Popis: Funkce posune zobrazený text dolů o obsah jednoho okna a zanechá pozici řádkového kurzoru beze změny. Pokud se obsah okna pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **FixScreenPos**
ScrollLineUp
ScrollLineDown
ScrollPageUp

ScrollPageUp

```
function ScrollPageUp : boolean;
```

Popis: Funkce posune zobrazený text nahoru o obsah jednoho okna a zanechá pozici řádkového kurzoru beze změny. Pokud se obsah okna pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **FixScreenPos**
ScrollLineUp
ScrollLineDown
ScrollPageDown

Search

```
function Search(
  dlg : boolean;
  var what : const string;
  casesenzitiv, wholewords, forward, reserved : integer) : short;
```

Parametry:

dlg	zobrazovat dialog (lze použít jedině FALSE) *what* hledaný řetězec
casesenzitiv	rozlišovat malá/velká písmena
wholewords	hledat jen celá slova
forward	hledat dopředu
reserved	zatím nevyužito, musí platit *reserved* = 0

Implicitní hodnota:
casesenzitiv = 0
wholewords = 0
forward = 0
reserved = 0

Popis: Funkce provádí činnost odpovídající příkazu **Najít** z menu **Úpravy**.
Při hodnotě *dlg* = TRUE vrací IDERROR.

Hledaný řetězec *what* by neměl mít větší délku než 127 znaků, jinak je interně na tuto délku zkrácen. Celočíselné parametry *casesenzitiv*, *wholewords*, *forward* jsou interpretovány takto:

Hodnota	Význam
větší než nula	ANO (TRUE)
menší než nula	NE (FALSE)
rovná nule	použít stav při předchozím hledání

Návratová hodnota: Funkce může vrátit některou z těchto hodnot:

0	odpovídající řetězec nebyl nalezen
IDERROR	došlo k chybě
IDGOTO	hledaný řetězec nalezen

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru, který není v zobrazení osnovy.

Viz: **SearchAgain**
SearchIsReady
Replace
ReplaceAgain
ReplaceIsReady

SearchAgain

```
function SearchAgain : boolean;
```

Popis: Funkce provádí opakování posledního hledání (je-li toto opakování možné vzhledem ke stavu zobrazení dokumentu a je-li co opakovat – viz **SearchIsReady**) a je-li toto hledání úspěšné, vrací TRUE. V opačném případě vrací FALSE.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru, který není v zobrazení osnovy.

Viz: **Search**
SearchIsReady
Replace
ReplaceAgain
ReplaceIsReady

SearchIsReady

```
function SearchIsReady(
  toRepeat : boolean) : boolean;
```

Parametry: *toRepeat* zjišťovat možnost opakování

Implicitní hodnota: *toRepeat* = FALSE

Popis: Při implicitní hodnotě parametru ***toRepeat* = FALSE** funkce zjišťuje připravenost editoru pro hledání textu; tj. vrací TRUE právě tehdy, když není zašedlý příkaz **Najít** z menu **Úpravy** (což nastává při neminimalizovaném aktuálním dokumentu, jenž má aktivní řádkový kurzor a není v zobrazení osnovy).

Při hodnotě ***toRepeat* = TRUE** navíc zjišťuje připravenost pro zopakování posledního hledání; tj. zda již byla prováděna funkce **Najít** (ať už prostřednictvím uživatelského rozhraní anebo voláním z makra) a z dvojice funcí **Najít** / **Zaměnit** byla funkce **Najít** prováděna jako poslední.

Viz: **Search**
SearchAgain
Replace
ReplaceAgain
ReplaceIsReady

Sec1000

```
function Sec1000(
  tm : time) : short;
```

Parametry: *tm* údaj o čase ve vnitřním formátu 602Text

Popis: Funkce extrahuje počet tisícin sekundy z časového údaje.

Hodnota: Funkce vrací počet tisícin sekundy z intervalu 0 až 999.

Viz: **Hours**
Day
Month
Year
Today
Make_time
Make_date
Minutes
Now
Seconds

Seconds

```
function Seconds(
  tm : time) : short;
```

Parametry: *tm* údaj o čase ve vnitřním formátu 602Text

Popis: Funkce extrahuje počet sekund z časového údaje.

Hodnota: Funkce vrací počet sekund z intervalu 0 až 59.

Viz: **Hours**
Day
Month
Year
Today
Make_time
Make_date
Minutes
Now
Sec1000

Seek

```
function Seek(
  f : file;
  position : integer) : boolean;
```

Parametry: *f* proměnná typu soubor (file)
position pozice v souboru (nezáporné číslo)

Popis: Funkce nastaví pozici v souboru ***f*** na hodnotu zadanou druhým parametrem. Soubor musí být otevřen. Pozice v souboru určuje místo, kam se do souboru bude zapisovat následující funkcí **Write** (**Writeln**), resp. odkud se bude číst funkcí **Read**.

Hodnota: Funkce vrátí TRUE pokud soubor ***f*** je otevřen. Jinak vrací FALSE.

Viz: **Read**
Write
Writeln
Close
Reset
Rewrite
Eof
Filelenght

SelectBlock

```
function SelectBlock(
  dlg : boolean;
  var name : const string) : short;
```

Parametry: *dlg* zobrazovat dialog
name jméno pojmenovaného bloku

Popis: Pro ***dlg*** **= TRUE** vyvolá dialog **Pojmenované bloky.** Jeho vstupní pole se inicializuje druhým parametrem. Dle volby vrátí IDGOTO, IDINSERT, IDCANCEL.

Pro ***dlg*** **= FALSE** pokud blok existuje označí ho a vrátí IDGOTO. Pokud žádný takový není vrátí IDERROR.

Poznámka: Funkce zohledňuje vůči parametru *name* rozdíl mezi malými a velkými písmeny.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **EditBlocksDlg**
LabelBlock
UnlabelBlock
DoesBlockExist
BlocksCount
BlockName
CopyBlock

SelectContents

```
function SelectContents : boolean;
```

Popis: Funkce označí jako blok dříve vygenerovaný obsah (pokud tento existuje) a vrátí TRUE. Nebyl-li obsah dosud vytvořen, neprovede se nic a návratová hodnota je FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **ContentsExist**
SelectIndex

SelectIndex

```
function SelectIndex : boolean;
```

Popis: Funkce označí jako blok dříve vygenerovaný rejstřík (pokud tento existuje) a vrátí TRUE. Nebyl-li rejstřík dosud vygenerován, neprovede se nic a návratová hodnota je FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **IndexExists**
AddIndexEntryDlg
IndexEntries
CountOfEntryRef
IndexEditDlg
AddIndexEntry
GoToIndexEntry
RemoveIndexEntry
IndexEntry
IndexEntryInd

SelectObject

```
function SelectObject(
  id : integer) : boolean;
```

Parametry: *id* identifikátor objektu

Popis: Funkce pro objekt zadaného identifikátoru tento objekt selektuje.

Hodnota: Pokud aktuální dokument obsahuje objekt zadaného identifikátoru funkce jej selektuje a vrací TRUE. Jinak vrací FALSE.

Omezení: Vyžaduje neminimalizovaný aktuální dokument.

Viz: **ObjectsCount**
ObjectId
DoesObjectExist
UnselectObject

SetAlign

```
procedure SetAlign(
  align_type : short);
```

Parametry: *align_type* typ zarovnání

Popis: Procedura nastavuje zarovnání dle parametru *align_type – alLeft, alCenter, alRight, alJustify*. Pro jiné hodnoty parametru nastane chyba za běhu makra.

Poznámka: Viz Poznámka 6.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **SetAlignLeft**
SetAlignCenter
SetAlignRight
SetAlignJustify
GetAlignType

SetAlignCenter

```
procedure SetAlignCenter;
```

Popis: Procedura nastavuje centrované zarovnávání.

Poznámka: Viz Poznámka 6.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **IsAlignCenter**
SetAlign
GetAlignType

SetAlignJustify

```
procedure SetAlignJustify;
```

Popis: Procedura nastavuje oboustranné zarovnávání.

Poznámka: Viz Poznámka 6.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **IsAlignJustify**
SetAlign
GetAlignType

SetAlignLeft

```
procedure SetAlignLeft;
```

Popis: Procedura nastavuje zarovnávání nalevo.

Poznámka: Viz Poznámka 6.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **IsAlignLeft**
SetAlign
GetAlignType

SetAlignRight

```
procedure SetAlignRight;
```

Popis: Procedura nastavuje zarovnávání napravo.

Poznámka: Viz Poznámka 6.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **IsAlignRight**
SetAlign
GetAlignType

SetBold

```
procedure SetBold(
  on : boolean);
```

Parametry: *on* nastavení nebo potlačení

Popis: Procedura nastavuje nebo vypíná tučné písmo.

Poznámka: Viz Poznámka 3.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **IsBold**
SetFormatFont

SetBottomMargin

```
function SetBottomMargin(
  value : real) : boolean;
```

Parametry: *value* hodnota spodního okraje

Popis: Funkce nastavuje spodní okraj a vrací TRUE, pokud lze takovou hodnotu nastavit (ne všechny jsou přípustné). Jinak vrací FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **SetMargins**
GetLeftMargin
SetLeftMargin
GetRightMargin
SetRightMargin
GetTopMargin
SetTopMargin
GetBottomMargin

SetBrush

```
procedure SetBrush(
  on : boolean);
```

Parametry: *on* aktivovat/deaktivovat štěteček

Implicitní hodnota: *on* = TRUE

Popis: Podle hodnoty parametru *on* procedura aktivuje nebo deaktivuje v aktuálním dokumentu štěteček.

Omezení: Vyžaduje neminimalizovaný aktuální dokument v textovém modu, který není v zobrazení osnovy.

Viz: **IsBrushOn**

SetDbForSave

```
function SetDbForSave : boolean;
```

Popis: Nastaví připojenou databázi pro zápis.

SetDocWndHeight

```
function SetDocWndHeight(
  pixels : short) : boolean;
```

Parametry: *pixels* vertikální rozměr

Popis: Funkce nastavuje výšku okna s dokumentem. Souřadnice jsou zde vždy v bodech (nikoli v aktuálních jednotkách).

Omezení: Nutno mít aktuální dokument.

Viz: **GetDocWndWidth**
SetDocWndWidth

SetDocWndWidth

```
function SetDocWndWidth(
  pixels : short) : boolean;
```

Parametry: *pixels* horizontální rozměr

Popis: Funkce nastavuje šířku okna s dokumentem. Souřadnice jsou zde vždy v bodech, nikoli v aktuálních jednotkách.

Omezení: Nutno mít aktuální dokument.

Viz: **GetDocWndWidth**
SetDocWndHeight

SetFormatFont

```
procedure SetFormatFont(
  kind, value : short);
```

Parametry: *kind* druh nastavované vlastnosti písma
value nastavovaná hodnota

Popis: Procedura zajišťuje nastavení vlastností písma. Dle parametru *kind* se interpretuje příslušná hodnota *value*:

***kind* = kCHPsize** - nastaví velikost písma v bodech a to pro hodnoty *value*: (4 <= *value*) && (*value* <= 128). Pro jiné hodnoty *value* nastane běhová chyba.

***kind* = kCHPbold** - pro hodnoty parametru *value kOn* nebo *kOff* nastaví či zruší vlastnost písma "tučné". Pro jiné hodnoty *value* nastane běhová chyba.

***kind* = kCHPitalic** - pro hodnoty parametru *value kOn* nebo *kOff* nastaví či zruší vlastnost písma "kurzíva". Pro jiné hodnoty *value* nastane běhová chyba.

***kind* = kCHPunderline** - podle následujícího výčtu nastaví dle parametru *value* příslušné podtržení textu:

0	bez podtržení
1	podtržení všeho
2	podtržení slov

Pro jiné hodnoty *value* nastane běhová chyba.

***kind* = kCHPsupersub** - podle následujícího výčtu nastaví dle parametru *value* příslušnou vlastnost textu normální – superscript – subscript:

0	(tSNormal)	normální
1	(tSuper)	horní index
2	(tSub)	dolní index

Pro jiné hodnoty *value* nastane běhová chyba.

***kind* = kCHPcaps** - pro hodnoty parametru *value kOn* nebo *kOff* nastaví či zruší vlastnost písma "všechna písmena velká". Pro jiné hodnoty *value* nastane běhová chyba.
Pro jiné hodnoty parametru *kind* nastane běhová chyba.

Poznámka: Viz Poznámka 3.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetFormatFont**

SetFormatPara

```
function SetFormatPara(
  kind : short;
  value : real) : boolean;
```

Parametry: *kind* druh nastavované vlastnosti odstavce
value nastavovaná hodnota

Popis: Funkce zajišťuje nastavení vlastností odstavce, resp. vybraných odstavců textu. Dle parametru *kind* se interpretuje příslušná hodnota *value*. Je-li hodnota parametru *kind* přípustná, ale hodnota *value* nikoliv, funkce neprovede nic a vrací FALSE. Při "správných" vstupních hodnotách vrací TRUE.

kind = kPPAlign - funkce nastavuje typ zarovnání odstavce. Přípustné hodnoty parametru *value* jsou celočíselné konstanty *alLeft, alCenter, alRight, alJustify*. Pro jiné hodnoty funkce neprovede nic.

kind = kPPLeftIndent - slouží pro nastavení odsazení odstavce nalevo. Hodnota *value* je udává v aktuálních jednotkách (centimetry, palce, atd.). Pro nepřípustné hodnoty *value* (záporné anebo příliš velké) funkce neprovede nic a vrátí FALSE.

kind = kPPRightIndent - slouží pro nastavení odsazení odstavce napravo. Hodnota *value* je udává v aktuálních jednotkách (centimetry, palce, atd.). Pro nepřípustné hodnoty *value* (záporné anebo příliš velké) funkce neprovede nic a vrátí FALSE.

kind = kPPFirstIndent - slouží pro nastavení odsazení první řádky odstavce. Hodnota *value* udává velikost v aktuálních jednotkách. Pro nepřípustné hodnoty *value* funkce neprovede nic a vrátí FALSE.

kind = kPPUpperSpace - nastavuje místo před odstavcem. Hodnota *value* udává velikost v aktuálních jednotkách. Přípustné hodnoty jsou reálná čísla od nuly do hodnoty odpovídající dvěma palcům včetně.

kind = kPPLowerSpace - nastavuje místo za odstavcem. Hodnota *value* udává velikost v aktuálních jednotkách. Přípustné hodnoty jsou reálná čísla od nuly do hodnoty odpovídající dvěma palcům včetně.

kind = kPPShade - slouží pro nastavení stínování odstavce. Parametr *value* může nabývat dvaceti celočíselných hodnot od 0 do 19 včetně s tímto významem:

0		žádné stínování
1	'/'	řídké šrafování vlasovou čarou vpravo
2	'\'	řídké šrafování vlasovou čarou vlevo
3	'X'	řídké šrafování vlasovou čarou vpravo i vlevo
4	'—'	řídké vodorovné šrafování vlasovou čarou
5	'|'	řídké svislé šrafování vlasovou čarou
6	'+'	řídké svislé i vodorovné šrafování vlasovou čarou
7	'\'	husté šrafování vlasovou čarou vlevo
8	'/'	husté šrafování vlasovou čarou vpravo
9	'—'	husté i řídké vodorovné šrafování vlasovou čarou
10	'|'	husté svislé šrafování vlasovou čarou
11	'+'	husté svislé i vodorovné šrafování vlasovou čarou
12		celistvá černá plocha
13		černá s 25 % světlostí
14		černá s 37, 5 % světlostí
15		černá s 50 % světlostí – tmavě šedá celistvá plocha
16		černá s 62, 5 % světlostí
17		černá s 75 % světlostí
18		černá s 87, 5 % světlostí
19		zcela bílá plocha

kind = kPPLine - slouží pro nastavení typu čar kolem odstavce. Parametr *value* může nabývat dvanácti celočíselných hodnot od 0 do 11 včetně, s tímto významem:

0	žádná čára

1 vlasová čára
2 čára o síle 0, 5 bodu
3 čára o síle 1 bod
4 čára o síle 2 body
5 čára o síle 4 body
6 čára o síle 6 bodů
7 čára o síle 8 bodů
8 čára o síle 12 bodů
9 dvojitá (vlasová) čára
10 dvojitá (vlasová + 2 body) čára
11 dvojitá (2 body + vlasová) čára

kind = kPPBorder - slouží pro nastavení typu orámování odstavce, tj. přítomnosti či nepřítomnosti jednotlivých čar kolem odstavce.

Přípustnými hodnotami parametru *value* jsou nezáporná celá čísla. O kreslení či nekreslení jednotlivých čar rozhoduje nastavení či nenastavení jednotlivého bitu v dvojkovém vyjádření parametru *value* s tímto přiřazením:

nejnižší bit $(0001)_2$	čára nalevo
druhý nejnižší bit $(0010)_2$	čára napravo
třetí nejnižší bit $(0100)_2$	čára nahoře
čtvrtý nejnižší bit $(1000)_2$	čára dole

Příklad:
value = 6 = $(0110)_2$ čára bude nahoře a napravo

kind = kPPLevel - slouží pro nastavení úrovně osnovy; přípustné hodnoty parametru value jsou nezáporná celá čísla mezi 0 a 9 včetně; přitom hodnota 0 má význam "všechny".

kind = kPPNumber - slouží pro nastavení druhu číslování osnovy. Přípustné hodnoty parametru *value* jsou tato nezáporná celá čísla:
0 žádný
1 legální (1.1 apod.)
2 osnova (1, A apod.)

kind = kPPLeading - nastavuje vzdálenost řádek v procentech. Přípustné hodnoty parametru *value* jsou reálná čísla mezi 100 a 240 včetně.

kind = kPPHyphenation - nastavuje druh dělení slov. Přípustné hodnoty parametru *value* jsou tato nezáporná celá čísla:
0 žádné
1 všechny řádky
2 ob jeden řádek
3 ob dva řádky

kind = kPPBorderOffset - nastavuje oddělení (vzdálenost) písma a čar kolem odstavce v aktuálních jednotkách.

kind = kPPBullets - jedná se o vlastnost odstavec - odrážky. Pro návratovou (resp.nastavovanou) hodnotu platí tyto možnosti:
0 žádné odrážky
1-6 odrážky dle pořadí v dialogu
100-104 číslování dle pořadí v dialogu

kind = kPPSkipBullets - nastavuje příznak indikující, zda se mají nebo nemají použít odrážky. Přípustné hodnoty jsou tato celá čísla:
kOn = 1 použít
kOff = 0 nepoužít

Pro jiné hodnoty parametru *kind* nastane běhová chyba.

Poznámka: Viz Poznámka 3.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetFormatPara**
GetFormatFont
GetFormatSect
FormatParaDlg

SetFormatSect
SetFormatFont

SetFormatSect

```
function SetFormatSect(
  kind : short;
  value : real) : boolean;
```

Parametry: *kind* druh nastavované vlastnosti sekce
value nastavovaná hodnota

Popis: Funkce zajišťuje nastavení vlastností sekce, resp. vybraných sekcí textu. Dle parametru *kind* se interpretuje příslušná hodnota *value*. Je-li hodnota parametru *kind* přípustná, ale hodnota *value* nikoliv, funkce neprovede nic a vrací FALSE. Při "správných" vstupních hodnotách vrací TRUE.

kind = kSPColSpace - nastavuje velikost mezery mezi sloupci v aktuálních jednotkách. Pro nepřípustné hodnoty *value* (příliš malé nebo příliš velké) funkce neprovede nic a vrátí FALSE.

kind = kSPMaxCol - slouží pro nastavení počtu sloupců. Přípustné hodnoty parametru *value* jsou celá čísla mezi 1až 8 včetně.

kind = kSPColHeight - nastavuje výšku sloupce v aktuálních jednotkách. Přípustné hodnoty jsou nezáporná celá čísla, přičemž horní mez závisí na výšce strany a na okrajích.

kind = kSPColInc - nastavuje přírůstek sloupce v aktuálních jednotkách. Přípustné hodnoty jsou nezáporná celá čísla, přičemž horní mez závisí na výšce strany a na okrajích.

kind = kSPColLine - nastavuje typ čáry mezi sloupci. Přípustné hodnoty i jejich význam jsou přesně tytéž jako pro *kind = kPPLine* ve funkci **SetFormatPara** (nastavení typu čar kolem odstavce).

Pro jiné hodnoty parametru *kind* nastane běhová chyba.

Poznámka: Viz Poznámka 3.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetFormatPara**
SetFormatPara
GetFormatFont
SetFormatFont
GetFormatSect

SetInitialOpen

```
procedure SetInitialOpen(
  on : boolean);
```

Parametry: *on* nový stav interního indikátoru

Popis: Speciální procedura ovlivňující probíhající start 602Text a tedy použitelná v makru **AutoExec** (jinde je bez účinku). Nastaví interní indikátor, který rozhoduje, zda se má po ukončení makra **AutoExec** otvírat úvodní implicitní prázdný dokument, resp. dokumenty, zadané na příkazové řádce. Stav indikátoru při startu je TRUE (tj. ANO). Zavolá-li se v **AutoExec**: **SetInitialOpen** (FALSE), budou po startu 602Text

otevřené pouze ty dokumenty, které makro **AutoExec** samo otevřelo nebo založilo jako nové.

Viz: **ErasethisMacro**

SetInsertMode

```
procedure SetInsertMode;
```

Popis: Procedura nastaví režim vkládání.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **IsInsertMode**
SetOverWriteMode
ToggleInsert

SetIsMacro

```
procedure SetIsMacro(
  on : boolean);
```

Parametry: *on* platnost režimu editace maker

Popis: Pro parametr ***on*** = **TRUE** nastavuje aktuální dokument do režimu editace maker.
Pro parametr ***on*** = **FALSE** nastavuje standardní režim editace.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **IsMacro**

SetItalic

```
procedure SetItalic(
  on : boolean);
```

Parametry: *on* nastavení nebo potlačení

Popis: Procedura nastaví nebo zruší vlastnost textu “kurzíva”.

Poznámka: Viz Poznámka 3.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **IsItalic**
SetFormatFont

SetLeftMargin

```
function SetLeftMargin(
  value : real) : boolean;
```

Parametry: *value* hodnota levého okraje

Popis: Funkce nastavuje levý okraj a vrací TRUE, pokud lze takovou hodnotu nastavit (ne všechny jsou přípustné). Jinak vrací FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument v textovém režimu (který není v zobrazení osnovy).

Viz: **SetMargins**
GetLeftMargin
GetRightMargin
SetRightMargin
GetTopMargin
SetTopMargin
GetBottomMargin
SetBottomMargin

SetMargins

```
function SetMargins(
  dlg : boolean;
  left, right, top, bottom : real) : short;
```

Parametry:

dlg	zobrazovat dialog
left	hodnota levého okraje
right	hodnota pravého okraje
top	hodnota horního okraje
bottom	hodnota spodního okraje

Popis: Odpovídá menu **Formát** a příkazu **Okraje.** Zadávané hodnoty se rozumí v aktuálních jednotkách (např. cm):
Pro ***dlg*** **= TRUE** - zavolá se dialog pro nastavení okrajů. Vstupní pole se inicializují parametry funkce a dle způsobu opuštění dialogu vrátí IDOK / IDCANCEL **.**
Pro ***dlg*** **= FALSE** - rovnou se použijí hodnoty parametrů, ovšem ne všechny jsou přípustné (toto závisí na mnoha faktorech **–** aktuální jednotky, rozměr stránky, aj.).
Pro nepřípustné hodnoty funkce vrátí IDERROR a neprovede nic. Jinak vrací IDOK.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **GetLeftMargin**
SetLeftMargin
GetRightMargin
SetRightMargin
GetTopMargin
SetTopMargin
GetBottomMargin
SetBottomMargin

SetMM

```
function SetMM : boolean;
```

Popis: Otevře dialog pro nastavení databáze. Vrací TRUE v případě úspěšného nastavení.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **MailMergeType**
RecordsCount
RecordFirst
RecordLast
RecordNext
RecordPrev

RecordCurrent
RecordSet
NextMMField
GetTextMMI
GetTextMM

SetNewWinState

```
procedure SetNewWinState(
  on : boolean);
```

Parametry: *on* nový stav tlačítka

Popis: Procedura nastavuje stav tlačítka **Nové okno**:
TRUE = stisknuto
FALSE = ne

Viz: **GetNewWinState**

SetObjectHTMLStr

```
procedure SetObjectHTMLStr(
  id : integer;
  var str : const string);
```

Parametry: *id* identifikátor objektu
str řetězec, který bude přiřazen jako další HTML atribut objektu

Popis: Funkce přiřadí objektu specielní HTML atribut.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **FieldHTML**
GetObjectHTMLStr

SetOutlineLevel

```
function SetOutlineLevel(
  dlg : boolean;
  level : short) : short;
```

Parametry: *dlg* zobrazovat dialog
level nová úroveň osnovy

Popis: Funkce nastavuje úroveň osnovy 0 – 9; 0 = všechny.
Pro ***dlg* = TRUE** - zavolá se dialog **Úrovně osnovy** a inicializuje se parametrem *level*.
Pro špatné hodnoty *level* (*level* > 9) se inicializuje jako pro *level* = 0. Vrací IDOK (a nastaví příslušnou úroveň) či IDCANCEL při **Esc**.
Pro ***dlg* = FALSE** - pro *level* <=9 se nastaví požadovaná úroveň a funkce vrátí IDOK. Jinak vrací IDERROR.

Omezení: Potřebuje neminimalizovaný aktuální dokument v zobrazení osnovy.

Viz: **GetOutlineLevel**
DisplayOutline

SetOverwriteMode

```
procedure SetOverwriteMode;
```

Popis: Procedura nastaví režim přepisu.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **IsInsertMode**
SetInsertMode
ToggleInsert

SetPreferences

```
function SetPreferences(
  kind, value : short) : boolean;
```

Parametry: *kind* druh nastavované hodnoty
value nastavovaná hodnota

Popis: Funkce nastavuje hodnoty, které jsou v dialogu **Předvolba** v menu **Pomůcky**; (parametr *kind* sděluje "co zadáváme"; *value* je zadávaná hodnota):

pro *kind* = kPREF_DEV_FONTS – **Nabízet písma tiskáren**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE

pro *kind* = kPREF_B_HLP – **Zobrazovat "bublinovou" nápovědu**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE

pro *kind* = kPREF_SUPP_SUMM – **Nabízet popis při ukládání**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE

pro *kind* = kPREF_S_WINDOWS – **Ukládat rozložení oken**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE

pro *kind* = kPREF_TEMPL_NEW – **Nabízet šablony pro nové dokumenty**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE

pro *kind* = kPREF_AUTOSAVE – **Automatické ukládání**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE

pro *kind* = kPREF_AUTO_INT – **Automatické ukládání (interval)**: pro hodnoty value od 0 do 99 tuto hodnotu použije pro nastavení intervalu ukládání a vrací TRUE; pro jiné hodnoty se neprovede nic a funkce vrací FALSE

pro *kind* = kPREF_IMG_DITHER **– Výpočet odstínů šedi pro tisk obrázků**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE

pro *kind* = kPREF_GLOB_UNITS – **Jednotky**: pro hodnoty (1<= value) && (value <=5): 1– palce, 2 – cm, 3 – body, 4 – pica, 5 – dekadické palce funkce vrací TRUE; pro jiné hodnoty parametru value je bez efektu a vrací FALSE

pro *kind* = kPREF_HORIZONTAL – **Preferovat vodorovné uspořádání mozaiky**: nenulová hodnota (ano) nebo 0 (ne) vrací TRUE.

Pro jiné než vyjmenované hodnoty parametru *kind* je funkce bez efektu a vrací FALSE.

Viz: **PreferencesDlg**
GetPrefFootnStr
SetPrefFootnStr
GetPrefFaxName
GetPrefFax
SetPrefFax

SetPrefFaxName

SetPrefFax

```
procedure SetPrefFax(
  var str : const string);
```

Parametry: *str* textový řetězec

Popis: Nastavuje jméno položky databáze, kde se bude při faxování se slučováním hledat faxové číslo adresáta.

Viz: **GetPrefFax**
GetPrefFaxName
SetPrefFaxName
GetPrefFootnStr
SetPrefFootnStr

SetPrefFaxName

```
procedure SetPrefFaxName(
  var name : const string);
```

Parametry: *name* textový řetězec

Popis: Nastavuje jméno položky databáze, kde se bude při faxování se slučováním hledat jméno adresáta.

Viz: **GetPrefFax**
GetPrefFaxName
GetPrefFootnStr
SetPrefFootnStr
SetPrefFax

SetPrefFootnStr

```
procedure SetPrefFootnStr(
  var str : const string);
```

Parametry: *str* textový řetězec

Popis: Procedura nastavuje jako implicitní značku poznámky pod čarou textový řetězec *str* (tato položka předvoleb se volí v menu **Pomůcky** příkaz **Předvolby**).

Viz: **GetPrefFootnStr**
GetPreferences
SetPreferences
GetPrefFax
PreferencesDlg
SetPrefFax
GetPrefFaxName
SetPrefFaxName

SetRGBColor

```
procedure SetRGBColor(
  bRed, bGreen, bBlue : short);
```

Parametry: *bRed* intenzita červené složky
bGreen intenzita zelené složky
bBlue intenzita modré složky

Popis: Procedura zajišťuje nastavení barvy písma. Intenzita zastoupení barev se zadává v parametrech (jsou sice deklarovány short, ale za přípustné se považují jen hodnoty v rozsahu 0 až 255 včetně). Hodnoty menší než 0 se interpretují jako 0, hodnoty větší než 255 se interpretují jako 255. Nelze však zvolit všechny barevné kombinace (tzn. že ne všechny barevné kombinace zadané těmito parametry budou skutečně realizovány). 602Text zvolí pro danou barevnou kombinaci takovou položku z palety barev, která je nejbližší. (Paleta barev je přístupná ve funkci **ColorDlg**).

Poznámka: Viz Poznámka 3.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **ColorDlg**

SetRGBText

```
procedure SetRGBText(
  colorref : integer);
```

Parametry: *colorref* barevná kombinace

Popis: Procedure zajišťuje nastavení barvy písma, podobně jako procedura **SetRGBColor** , u které se však barva zadává po složkách.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **SetRGBColor**
GetRGBText
RGB
RgbTrio

SetRightMargin

```
function SetRightMargin(
  value : real) : boolean;
```

Parametry: *value* hodnota pravého okraje

Popis: Funkce nastavuje pravý okraj a vrací TRUE, pokud lze takovou hodnotu nastavit (ne všechny jsou přípustné). Jinak vrací FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **SetMargins**
GetLeftMargin
SetLeftMargin
GetRightMargin
GetTopMargin
SetTopMargin
GetBottomMargin

SetBottomMargin

SetScale

```
procedure SetScale(
  percent : short);
```

Parametry: *percent* velikost zobrazení v procentech

Popis: Procedura nastavuje přímo (bez dialogu) zvětšení zadané v procentech – ovšem jen pokud (50 <= *percent*) && (*percent* <= 400).

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **GetScale**
SetScaleFullPage
ManualScaleDlg

SetScaleFullPage

```
procedure SetScaleFullPage;
```

Popis: Procedura odpovídá výběru položky menu **Zobrazit** příkaz **Celou stránku**.

Omezení: Vyžaduje neminimalizovaný aktuální dokument.

Viz: **GetScale**
SetScale
ManualScaleDlg

SetStatusStrip

```
procedure SetStatusStrip(
  on : boolean);
```

Parametry: *on* zobrazení nebo potlačení

Popis: Procedura zapíná nebo vypíná stavový řádek.

Viz: **IsStatusStrip**
SetStatusText

SetStatusText

```
function SetStatusText(
  var text : const string) : boolean;
```

Parametry: *text* vypisovaný text

Popis: Je-li zapnutý stavový řádek (status bar), funkce na něj zapíše text a vrátí TRUE. Jinak vrací FALSE.

Viz: **IsStatusStrip**
SetStatusStrip

SetTextDefault

```
procedure SetTextDefault;
```

Popis: Procedura nastaví text daný stylem odstavce.

Poznámka: Viz Poznámka 3.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **IsTextDefault**

SetTextRegular

```
procedure SetTextRegular;
```

Popis: Procedura nastaví text na obyčejný.

Poznámka: Viz Poznámka 3.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **IsTextRegular**

SetToolbar

```
procedure SetToolbar(
  on : boolean;
  index : short);
```

Parametry: *on* zobrazení nebo potlačení
index index označující nástrojovou lištu

Implicitní parametr: *index* = 2

Popis: Procedura zapne nebo vypne nástrojovou lištu danou parametrem *index*. Lišty jsou očíslovány od indexu 1 do 13 včetně, s tímto významem:

1	Osnova
2	Formát
3	Objekt
4	Tabulka
5	Standardní
6	Pomůcky
7	Pole a databáze
8	Náhled
9	HTML
10	Zobrazení
11	Formulářové objekty
12	Štítky
13	WEB

Viz: **IsToolbar**
ToolbarCrgDlg

SetTopMargin

```
function SetTopMargin(
  value : real) : boolean;
```

Parametry: *value* hodnota horního okraje

Popis: Funkce nastavuje horní okraj a vrací TRUE, pokud lze takovou hodnotu nastavit (ne všechny jsou přípustné). Jinak vrací FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru (který není v zobrazení osnovy).

Viz: **SetMargins**
GetLeftMargin
SetLeftMargin
GetRightMargin
SetRightMargin
GetTopMargin
GetBottomMargin
SetBottomMargin

SetUnderlineType

```
procedure SetUnderlineType(
  underline_type : short);
```

Parametry: *underline_type* druh podtržení

Popis: Procedura nastaví způsob podtrženého textu pro *underline_type*:
0 bez podtržení
1 vše podtrženo
2 podtržení slov
Pro jiné hodnoty je bez efektu.

Poznámka: Viz Poznámka 3.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetUnderlineType**
SetFormatFont

SetWordOverFlow

```
procedure SetWordOverFlow(
  on : boolean);
```

Parametry: *on* zobrazení nebo potlačení

Popis: Procedura nastavuje nebo ruší nastavení přetékání zvolených slov pro aktuální dokument.

Viz: **IsWordOverFlow**

Sgn

```
function Sgn(
  r : real) : short;
```

Parametry: *r* reálné číslo

Popis: Funkce počítá znaménko zadaného čísla:
1 pro *r* kladné
0 pro *r* = 0
–1 pro *r* záporné

Hodnota: Znaménko parametru *r*.

Viz: **Abs**

Sin

```
function Sin(
  r : real) : real;
```

Parametry: *r* reálné číslo

Popis: Funkce počítá sinus úhlu zadaného v radiánech.

Hodnota: Funkce vrátí hodnotu sinu zadaného úhlu.

sizeof

```
function sizeof(
  var v : const anytype) : integer;
```

Parametry: *v* proměnná libovolného typu

Popis: Funkce pro proměnnou libovolného typu vrací její velikost v bajtech; tj. kolik bajtů zabírá proměnná v paměti.

Poznámka: Namísto identifikátoru proměnné může vystupovat v zápisu argumentu této funkce též identifikátor typu. Funkce pak vrací velikost libovolné proměnné tohoto typu.

Příklad:

```
   type
str30 = string[30];
str40 = string[40];
person = record
   prijmeni : str30;
   ulice : str40;
   narozen : date;
end;
   var
Petr, Pavel : person;
   begin
Petr.prijmeni := "Strnad";
Petr.ulice := "Pod lesem"
Petr.narozen := 1.2.2006;
Memcpy(Pavel, Petr, sizeof(str30) + sizeof(str40));
Pavel.narozen := 6.7.1415;
```

Viz: **Memcpy**

SpellEnabled

```
function SpellEnabled : boolean;
```

Popis: Funkce vrací přístupnost položky menu **Pomůcky** příkaz **Kontrola pravopisu.**

Viz: **SpellRun**
HyphenDlgEnabled
DatabaseEnabled
ContentsEnabled

IndexMarkEnabled
IndexEditEnabled
IndexGenEnabled
LangSetupEnabled
TranslateEnabled
ThesaurusEnabled

SpellRun

```
function SpellRun(
  dlg : boolean;
  check_type : short) : short;
```

Parametry: *dlg* zobrazovat dialog
check_type specifikace části textu ke kontrole

Popis: Funkce odpovídá příkazu **Kontrola pravopisu** z menu **Pomůcky**.
Pro dlg = TRUE funkce vrací IDERROR.
Parametr *check_type* má následující význam:

kCheckFromCaret – kontrola od pozice kurzoru
kCheckEntireDoc – kontrolovat celý dokument

Jestliže je dokument bez pravopisných chyb - vrátí funkce IDOK. V opačném případě označí první chybný výraz a vrátí IDERROR.

Poznámka: Funkce vrátí IDERROR (a kontrola pravopisu neproběhne) i v případě, pokud se nepodaří inicializace příslušných slovníků.

Omezení: Funkce vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **SpellEnabled**

Sqr

```
function Sqr(
  r : real) : real;
```

Parametry: *r* reálné číslo

Popis: Funkce spočte druhou mocninu reálného čísla.

Hodnota: Funkce vrací druhou mocninu čísla zadaného jako parametr. Na rozdíl od podobné funkce v Pascalu je typ výsledku vždy *real***.**

Sqrt

```
function Sqrt(
  r : real) : real;
```

Parametry: *r* reálné nezáporné číslo

Popis: Funkce počítá druhou odmocninu. Parametrem musí být nezáporné číslo, jinak dojde k chybě při výpočtu.

Hodnota: Funkce vrací druhou odmocninu svého parametru.

Str2date

```
function Str2date(
  var str : const string) : date;
```

Parametry: *str* řetězec znaků obsahující zápis data ve tvaru den.měsíc nebo den.měsíc.rok

Popis: Funkce převede zápis data v řetězci *str* na hodnotu typu *date* a vrátí ji. Prázdný řetězec se převede na hodnotu NONEDATE.

Hodnota: Funkce vrátí datum, pokud jeho zápis je (jediným) obsahem řetězce *str*. Jinak vrací NONEDATE.

Viz: **Int2str**
Str2int
Real2str
Str2real
Money2str
Str2money
Date2str
Time2str
Str2time

Str2int

```
function Str2int(
  var str : const string) : integer;
```

Parametry: *str* řetězec znaků obsahující zápis čísla typu integer

Popis: Funkce převede zápis čísla v řetězci *str* na hodnotu typu *integer* a vrátí ji. Prázdný řetězec se převede na hodnotu NONEINTEGER.

Hodnota: Funkce vrátí celočíselnou hodnotu, pokud její zápis je (jediným) obsahem řetězce *str*. Jinak vrací NONEINTEGER.

Viz: **Int2str**
Real2str
Str2real
Money2str
Str2money
Date2str
Str2date
Time2str
Str2time

Str2money

```
function Str2money(
  var str : const string) : money;
```

Parametry: *str* řetězec znaků obsahující zápis čísla typu money

Popis: Funkce převede zápis čísla v řetězci *str* na hodnotu typu *money* a vrátí ji. Prázdný řetězec se převede na hodnotu NONEMONEY.

Hodnota: Funkce vrátí hodnotu typu *money*, pokud její zápis je (jediným) obsahem řetězce *str*. Jinak vrací NONEMONEY.

Viz: **Int2str**
Str2int
Real2str
Str2real
Money2str
Date2str
Str2date
Time2str
Str2time

Str2real

```
function Str2real(
  var str : const string) : real;
```

Parametry: *str* řetězec znaků obsahující zápis čísla typu real

Popis: Funkce převede zápis čísla v řetězci *str* na hodnotu typu *real* a vrátí ji. Prázdný řetězec se převede na hodnotu NONEREAL.

Hodnota: Funkce vrátí hodnotu typu *real*, pokud její zápis je (jediným) obsahem řetězce *str*. Jinak vrací NONEREAL.

Viz: **Int2str**
Str2int
Real2str
Money2str
Str2money
Date2str
Str2date
Time2str
Str2time

Str2time

```
function Str2time(
  var str : const string) : time;
```

Parametry: ***str*** řetězec znaků obsahující zápis času, který může být ve tvaru:
hodina : minuta
nebo
hodina : minuta : sekunda
nebo
hodina : minuta : sekunda : tisíciny.

Popis: Funkce převede zápis času v řetězci *str* na hodnotu typu *time* a vrátí ji. Prázdný řetězec se převede na hodnotu NONETIME.

Hodnota: Funkce vrátí údaj o čase, pokud jeho zápis je (jediným) obsahem řetězce *str*. Jinak vrací NONETIME.

Viz: **Int2str**
Str2int
Real2str
Str2real
Money2str
Str2money
Date2str
Str2date

Time2str

Strcat

```
function Strcat(
  var strl, str2 : const string) : string;
```

Parametry:
str1 libovolný řetězec znaků
str2 libovolný řetězec znaků

Popis: Funkce spojí řetězce *str1* a *str2* do jednoho řetězce (*str2* připojí na konec *str1*) a vrátí výsledný řetězec. Přitom výsledek nesmí být delší než 255 znaků, jinak se vrátí pouze jeho prefix.
Tato funkce je zahrnuta do 602Text pouze v zájmu kompatibility s jazykem WinBase602. Spojování řetězců lze nyní provádět operátorem +, místo **Strcat** (*str1*, *str2*) stačí tedy psát pouze *str1+str2*.

Hodnota: Funkce vrátí zřetězení (konkatenaci) *str1* a *str2*.

Poznámka: Narozdíl od obdobné funkce v jazyku C jsou zde oba parametry pouze vstupní. Tedy řetězec *str1* není modifikován; zřetězení je přístupné přes návratovou hodnotu funkce.

Viz:
Strcopy
Strinsert
Strdelete
Strpos
Strlenght
Strtrim
Pref
Like
Substr
Upcase

Strcopy

```
function Strcopy(
  var str : const string;
  index, count : short) : string;
```

Parametry:
str řetězec znaků
index pozice v řetězci str
count počet znaků

Popis: Funkce vybere z řetězce *str* podřetězec začínající v pozici *index* délky *count*. Řetězce se indexují od 1. Funkce je obdobou funkce **Copy** z Turbo Pascalu.

Hodnota: Funkce vrací podřetězec vybraný z řetězce *str*.

Viz:
Strcat
Strinsert
Strdelete
Strpos
Strlenght
Strtrim
Substr
Pref
Like
Upcase

Strdelete

```
procedure Strdelete(
  var str : string;
  index, count : short);
```

Parametry:
str řetězec znaků
index pozice v řetězci str
count počet znaků

Popis: Procedura zruší v řetězci *str* jeho podřetězec začínající v pozici *index* délky *count*. Řetězce se indexují od 1. Procedura je obdobou procedury **Delete** z Turbo Pascalu.

Viz:
Strcat
Strcopy
Strinsert
Strpos
Strlenght
Strtrim
Pref
Like
Substr
Upcase

Strinsert

```
procedure Strinsert(
  var source : const string;
  var str : string;
  index : short);
```

Parametry:
source vkládaný řetězec znaků
str řetězec znaků, do něhož se vkládá
index pozice v řetězci *str*

Popis: Procedura vloží do řetězce *str* od pozice *index* řetězec *source*. Řetězce se indexují od 1. Výsledná délka řetězce *str* nesmí překročit maximální délku zadanou v jeho deklaraci, jinak může dojít ke zhroucení systému. Procedura je obdobou procedury **Insert** z Turbo Pascalu.

Viz:
Strcat
Strcopy
Strdelete
Strpos
Strlenght
Strtrim
Pref
Like
Substr
Upcase

Strlength

```
function Strlength(
  var str : const string) : short;
```

Parametry: *str* řetězec znaků

Popis: Funkce zjišťuje skutečnou délku řetězce *str*. Funkce je obdobou funkce **Length** z Turbo Pascalu.

Hodnota: Funkce vrací délku řetězce *str*.

Viz: **Strcat**
Strcopy
Strinsert
Strdelete
Strpos
Strtrim
Pref
Like
Substr
Upcase

StrIsRC

```
function StrIsRC(var par : const string) : boolean;
```

Parametry: *par* rodné číslo

Popis: Funkce zjišťuje, je-li řetězec rodné číslo.

Hodnota: TRUE = proběhla-li kontrola úspěšně

Strpos

```
function Strpos(
  var substr, str : const string) : short;
```

Parametry: *substr* řetězec znaků
str řetězec znaků

Popis: Funkce zjišťuje, zda řetězec *substr* je podřetězcem řetězce *str*. Funkce je obdobou funkce **Pos** z Turbo Pascalu.

Hodnota: Funkce vrací index prvního znaku prvního výskytu podřetězce *substr* v řetězci *str*, pokud nějaký výskyt existuje (řetězce se indexují od 1). Jinak vrací 0.

Viz: **Strcat**
Strcopy
Strinsert
Strdelete
Strlenght
Strtrim
Pref
Like
Substr
Upcase

Strtrim

```
function Strtrim(
  var str : string) : string;
```

Parametry: *str* řetězec znaků

Popis: Funkce odstraní z řetězce *str* mezery na začátku a na konci. Mezery vyskytující se mezi jinými znaky v řetězci ponechá beze změny. Řetězec *str*, předaný odkazem, je touto funkcí změněn.

Hodnota: Funkce vrací řetězec *str*.

Viz: **Strcat**
Strcopy
Strinsert
Strdelete
Strpos
Strlenght
Pref
Like
Substr
Upcase

StyleIndex

```
function StyleIndex(
  var style_name : const string) : short;
```

Parametry: *style_name* název stylu

Popis: Návratovou hodnotou je číslo (index) stylu, který je označen jménem *style_name*.

Poznámka: Pokud parametr *style_name* neoznačuje jméno žádného existujícího stylu – návratovou hodnotou funkce bude NONESHORT.

Omezení: Potřebuje aktuální dokument.

Viz: **GetStyleCount**
GetStyleName
StyleIsBased

StyleIsBased

```
function StyleIsBased(
  style_index : short) : short;
```

Parametry: *style_index* číslo stylu

Popis: Návratovou hodnotou je číslo (index) stylu z něhož je styl daný parametrem *style_index* odvozen.

Poznámka: Pokud styl daný parametrem *style_index* není z žádného stylu odvozen nebo dokonce parametr *style_index* neodkazuje na žádný existující styl návratovou hodnotou funkce bude NONESHORT.

Omezení: Potřebuje aktuální dokument.

Viz: **GetStyleCount**
GetStyleName
StyleIndex

StyleUpdate

```
function StyleUpdate(
  var template : const string;
  style_index : short) : short;
```

Parametry:
template jméno šablony nebo dokumentu podle kterého dojde k aktualizaci stylu; při zadání prázdného řetězce dojde k aktualizaci dle šablony ze které dokument vznikl
style_index číslo stylu

Popis: Funkce načte vlastnosti příslušných stylů ze zadané šablony a zapíše je do aktuálního dokumentu. Tato funkce umožňuje udržovat a měnit jednotný vzhled více dokumentů. V případě, že při aktualizaci nedošlo k žádné změně vrací funkce 128; jinak vrací IDOK či IDERROR.

Poznámka: Je-li *style_index* rovno 0 dojde k aktualizaci všech stylů.

Omezení: Potřebuje aktuální dokument.

Substr

```
function Substr(
  var substr, str : const string) : boolean;
```

Parametry:
sub řetězec znaků, který má být obsažen v řetězci *str*
str řetězec znaků, který má obsahovat řetězec *sub*

Popis: Funkce zjišťuje, zda řetězec *sub* je obsažen v řetězci *str*. Místo, kde je obsažen, zde nehraje roli, může být na začátku, uvnitř nebo na konci *str*.

Hodnota: Funkce vrátí TRUE, pokud je *sub* obsažen v *str*. Jinak vrátí FALSE.

Viz:
Strcat
Strcopy
Strinsert
Strdelete
Strpos
Strlenght
Strtrim
Pref
Like
Substr
Upcase

Succ

```
function Succ(
  t : ordinaltype) : ordinaltype;
```

Parametry:
t výraz libovolného ordinálního typu

Popis: Funkce vrací pro hodnotu *t* libovolného ordinálního typu hodnotu jejího následníka, je-li ovšem tento definován. V opačném případě hodnota výsledku není definována.

Poznámka: Další podrobnosti naleznete v popisu vnitřního jazyka (v části týkající se ordinálních typů).

Hodnota: Hodnotou funkce je následník výrazu zadaného parametrem; je téhož typu jako tento parametr.

Viz: **Ord**
Pred
Chr

SummaryDlg

```
function SummaryDlg : short;
```

Popis: Funkce vyvolá dialog **Vlastnosti**. Vrací IDOK, IDCANCEL.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

SummaryGetStr

```
function SummaryGetStr(
  str_id : short) : string;
```

Parametry: *str_id* specifikace položky popisu

Popis: Podle hodnoty parametru *str_id* vrací funkce některou z položek popisu aktuálního dokumentu:
1 – titulek
2 – předmět
3 – autor
4 – klíčová slova
5 – poznámky
Pro jiné hodnoty *str_id* vrátí funkce prázdný řetězec.

Omezení: Vyžaduje aktuální dokument.

Viz: **SummarySetStr**
SummaryDlg

SummarySetStr

```
procedure SummarySetStr(
  var str : const string;
  str_id : short);
```

Parametry: Podle hodnoty parametru *str_id* funkce mění některou z položek popisu aktuálního dokumentu na hodnotu *str*:
1 – titulek
2 – předmět
3 – autor
4 – klíčová slova
5 – poznámky
Pro jiné hodnoty *str_id* funkce neprovede nic.

Omezení: Vyžaduje aktuální dokument.

Viz: **SummaryGetStr**
SummaryDlg

SwitchToDoc

```
function SwitchToDoc(
  var path_name : const string) : boolean;
```

Parametry: *path_name* název souboru včetně cesty

Popis: Funkce hledá okno s dokumentem dle specifikace odpovídajícího souboru. Najde-li je, přepne do něj (nutno zadat kompletně diskovou jednotku i přístupovou cestu, např. `"c:\tmp\readme.wpd"`).
Tato funkce nečiní rozdílu mezi velkými a malými písmeny, tj. `"C:\TMP\README.WPD"` má stejný efekt.

Viz: **IsDocumentOpened**
SwitchToWindow
GetDocWndTittle
GetDocFileName

SwitchToWindow

```
function SwitchToWindow(
  var title : const string;
  case_senzitive : boolean) : boolean;
```

Parametry: *tittle* titulní řetězec
case_senzitive rozlišovat malá/velká písmena

Implicitní hodnota: *case_senzitive* = FALSE

Popis: Funkce hledá takové okno dokumentu, které má v názvu řetězec *title*. Druhý parametr *case_senzitive* udává, zda se mají při hledání rozlišovat velká a malá písmena. Pokud hledané okno existuje, stane se aktivní a funkce vrací TRUE. Jinak se neprovede nic a návratová hodnota je FALSE.

Viz: **DoesWindowExist**
NextWindow
PreviousWindow
SwitchToDoc
GetDocWndTittle
GetDocFileName

ThesaurusDlg

```
function ThesaurusDlg : boolean;
```

Popis: Je-li v menu **Pomůcky** přístupna položka příkaz **Synonyma**, funkce zavolá dialog **Synonyma** a po jeho ukončení vrátí TRUE. Jinak vrací FALSE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **ThesaurusEnabled**

ThesaurusEnabled

```
function ThesaurusEnabled : boolean;
```

Popis: Funkce vrací přístupnost položky menu **Pomůcky** příkaz **Synonyma.**

Viz: **ThesaurusDlg**
HyphenDlgEnabled
DatabaseEnabled
ContentsEnabled
IndexMarkEnabled
IndexEditEnabled
IndexGenEnabled
LangSetupEnabled
TranslateEnabled
SpellEnabled

Time2str

```
function Time2str(
  tm : time;
  prez : short) : string;
```

Parametry: *tm* hodnota, která se má konvertovat
prez způsob konverze

Popis: Funkce převede hodnotu zadanou prvním parametrem na řetězec znaků. Při konverzi se použije hodnota parametru *prez* takto:
je-li *prez* = **0** – zápis času bude obsahovat pouze hodiny a minuty
1 – zápis času bude obsahovat hodiny, minuty a sekundy
2 – zápis času bude obsahovat hodiny, minuty, sekundy a tisíciny sekundy.

Hodnota: Funkce vrací textový zápis času *tm*.

Viz: **Int2str**
Str2int
Real2str
Str2real
Money2str
Str2money
Date2str
Str2date
Str2time

Today

```
function Today : date;
```

Popis: Funkce je funkce bez parametrů a zjišťuje dnešní datum.

Hodnota: Funkce vrátí datum ve vnitřním formátu 602Text, získané dotazem u operačního systému počítače.

Viz: **Minutes**
Hours
Day
Month
Year
Make_time
Make_date
Seconds
Sec1000
Now

Day_of_week

ToggleInsert

```
procedure ToggleInsert;
```

Popis: Procedura přepíná mezi režimy přepisu a vkládání.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **IsInsertMode**
SetInsertMode
SetOverWriteMode

ToolbarCfgDlg

```
function ToolbarCfgDlg : short;
```

Popis: Funkce zobrazí dialog **Pravítka a lišty** (karta **Nástrojové lišty**).

Vrací IDOK nebo IDCANCEL.

Viz: **IsToolbar**
SetToolbar

TopOfScreen

```
function TopOfScreen(
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Implicitní hodnota: *shift* = FALSE

Popis: Funkce přesune kurzor na horní řádek aktuálně zobrazený v okně a zanechá sloupec beze změny (je-li to možné). Pokud se řádkový kurzor pohnul vrací TRUE.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharLeft**
CharRight
LineDown
LineUp
CaretEnd
CaretHome
LeftOfLine
RightOfLine
WordLeft
WordRight
RightOfWord
PageDown
PageUp
BottomOfScreen

TranslateEnabled

```
function TranslateEnabled : boolean;
```

Popis: Funkce vrací přístupnost položky menu **Pomůcky** příkaz **Překlad slova**.

Viz: **HyphenDlgEnabled**
DatabaseEnabled
ContentsEnabled
IndexMarkEnabled
IndexEditEnabled
IndexGenEnabled
LangSetupEnabled
ThesaurusEnabled
SpellEnabled

Trunc

```
function Trunc(
  r : real) : integer;
```

Parametry: *r* reálné číslo z intervalu hodnot typu *integer*

Popis: Funkce převede reálné číslo na celé číslo odříznutím desetinné části.

Hodnota: Funkce vrací celé číslo vzniklé seříznutím z hodnoty parametru. Pokud je hodnota parametru funkce mimo rozsah typu *integer*, není hodnota funkce definována.

Undo

```
function Undo : boolean;
```

Popis: Funkce spustí příkaz **Odvolat,** jejíž výsledek závisí na okamžitých podmínkách.

Omezení: Potřebuje neminimalizovaný aktuální dokument.

Viz: **CanUndo**

UnlabelBlock

```
function UnlabelBlock(
  var name : const string) : boolean;
```

Parametry: *name* jméno bloku

Popis: Funkce vypustí pojmenovaný blok ze seznamu pojmenovaných bloků (je-li v něm takový) a vrátí TRUE. Neexistuje-li blok jména *name* vrátí FALSE.

Poznámka: Funkce zohledňuje vůči parametru *name* rozdíl mezi malými a velkými písmeny.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **EditBlocksDlg**
SelectBlock
LabelBlock
DoesBlockExist

BlocksCount
BlockName
CopyBlock

UnselectBlock

```
function UnselectBlock : boolean;
```

Popis: Funkce zruší označení bloku textu beze změny pozice řádkového kurzoru. Není-li blok označený vrací FALSE.

Omezení: Vyžaduje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **IsBlockSelected**

UnselectObject

```
function UnselectObject : boolean;
```

Viz: **SelectObject**

Upcase

```
function Upcase(
  var str : string) : string;
```

Parametry: *str* řetězec znaků

Popis: Funkce konvertuje všechna malá písmena v řetězci *str* na velká. Umí konvertovat i národní znaky.

Hodnota: Funkce vrací řetězec *str*.

Viz: **Lcase**
ChangeCase
Strcat
Strcopy
Strinsert
Strdelete
Strpos
Strlenght
Pref
Like
Substr

UseFont

```
procedure UseFont(
  index : short);
```

Parametry: *index* index fontu

Popis: Procedura nastavuje typ písma - font buďto pro označený blok textu anebo (není-li žádný blok označen) pro následující vkládaný text. Font je identifikován svým indexem specifikujícím pořadí v seznamu aktuálně dostupných fontů.

Viz: **UseFontOfName**
CountFonts
FontName
CurrentFont
UseFontOfNameExt

UseFontOfName

```
procedure UseFontOfName(
  var str : const string);
```

Parametry: *str* jméno fontu

Popis: Procedura nastavuje font stejným způsobem jako procedura **UseFont**; ovšem s tím rozdílem, že font je identifikován svým jménem. Kromě fontů aktuálně dostupných na systému můžete tedy touto procedurou nastavit font, který se vyskytuje v dokumentu, protože byl použit při vytváření dokumentu na jiném počítači apod.

Viz: **UseFont**
CountFonts
FontName
CurrentFont
UseFontOfNameExt

UseFontOfNameExt

```
procedure UseFontOfNameExt(
  var str : const string;
  charset : short);
```

Parametry: *str* jméno fontu

charset skript - jazyková specifikace:

0	základní
1	dle typu Windows
2	symboly
161	řecký
204	ruský
238	středoevropský

Popis: Procedura nastavuje font stejným způsobem jako procedura **UseFont** ovšem s tím rozdílem, že font je identifikován svým jménem a jazykovou specifikací.

Omezení: Potřebuje nemimimalizovaný aktuální dokument při aktivním řádkovém kurzoru, který není v režimu zobrazení osnovy.

Viz: **UseFontOfName**
UseFont
CountFonts
FontName
CurrentFont

UseParaStyleNum

```
function UseParaStyleNum(
  dlg : boolean;
  style_index : short;
  text_only : boolean) : short;
```

Parametry: *dlg* zobrazovat dialog
style_index číslo stylu
text_only použít pouze na text

Popis: Funkce odpovídá příkazu **Použít styl odstavce** z menu **Formát**.
Příslušný styl odstavce je zadaný indexem *style_index* dle interního setřídění.
Pro ***dlg* = TRUE** - zavolá se příslušný dialog, který se inicializuje parametry *style_index, text_only*. Funkce vrátí IDOK/IDCANCEL.
Pro ***dlg* = FALSE** - použijí se rovnou příslušné hodnoty parametrů (obdobně jako by styl zadaný indexem *style_index* byl vybrán na uživatelské liště **Formát**).

Poznámka: Pokud neplatí (1 <= *style_index) && (style_index* <=**GetStyleCount**), funkce vrátí IDERROR.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **GetStyleCount**
GetStyleName
DefParaStyleDlg

ViewFieldCnts

```
procedure ViewFieldCnts(
  on : boolean);
```

Parametry: *on* zobrazení nebo potlačení

Popis: Procedura nastavuje nebo potlačuje zobrazení obsahu polí.

Omezení: Vyžaduje neminimalizovaný aktuální dokument.

Viz: **IsFieldCntsOn**

ViewFormat

```
procedure ViewFormat(
  on : boolean);
```

Parametry: *on* zobrazení nebo potlačení

Popis: Procedura nastavuje nebo potlačuje zobrazení skrytých znaků.

Omezení: Vyžaduje neminimalizovaný aktuální dokument.

Viz: **IsFormatOn**

ViewFrameMargins

```
procedure ViewFrameMargins(
  on : boolean);
```

Parametry: *on* zobrazení nebo potlačení

Popis: Procedura nastavuje nebo potlačuje zobrazení okrajů textových rámců.

Omezení: Vyžaduje neminimalizovaný aktuální dokument, jež není v zobrazení osnovy.

Viz: **FrameMarginsOn**

ViewObjects

```
procedure ViewObjects(
  on : boolean);
```

Parametry: *on* zobrazení nebo potlačení

Popis: Procedura nastavuje nebo potlačuje zobrazení objektů.

Omezení: Vyžaduje neminimalizovaný aktuální dokument, jež není v zobrazení osnovy.

ViewScrollBars

```
procedure ViewScrollBars(
  on : boolean);
```

Parametry: *on* zobrazení nebo potlačení

Popis: Procedura nastavuje zobrazení nebo potlačení posuvníků.

Omezení: Vyžaduje neminimalizovaný aktuální dokument.

Viz: **AreScrollbarsOn**

ViewTabRuler

```
procedure ViewTabRuler(
  on : boolean);
```

Parametry: *on* zobrazení nebo potlačení

Popis: Procedura nastavuje zobrazení nebo potlačení pravítka s tabelátory.

Omezení: Vyžaduje neminimalizovaný aktuální dokument, jež není v zobrazení osnovy.

Viz: **IsTabRulerOn**

ViewVertRuler

```
procedure ViewVertRuler(
  on : boolean);
```

Parametry: *on* zobrazení nebo potlačení

Popis: Procedura nastavuje zobrazení nebo potlačení svislého pravítka.

Omezení: Vyžaduje neminimalizovaný aktuální dokument, jež není v zobrazení osnovy.

Viz: **IsVertRulerOn**

Wait

```
procedure Wait(
  seconds : short);
```

Parametry: *seconds* počet sekund

Popis: Při zavolání procedura pozastaví běh makra na zadaný počet sekund.

Viz: **Wait_box_show**
Wait_box_hide

Wait_box_hide

```
function Wait_box_hide : boolean;
```

Popis: Funkce zruší okno, jež bylo předtím zobrazeno funkcí **Wait_box_show .**

Hodnota: Pokud bylo takové okno vůbec zobrazeno návratová hodnota bude TRUE. Jinak se neprovede nic a funkce vrátí FALSE.

Viz: **Wait_box_show**
Wait

Wait_box_show

```
function Wait_box_show(
  var caption, text : const string) : boolean;
```

Parametry: *caption* titulek okna
text text okna

Popis: Funkce zobrazí okno s ikonou přesýpacích hodin a nadpis s textem zadanými parametry *caption*, *text*. Na rozdíl od funkcí zobrazujících dialog (**Info_box, Yesno_box,** ...) u této funkce běh makra bezprostředně pokračuje a okno je zobrazeno až do zavolání funkce **Wait_box_hide** nebo do ukončení běhu samotného makra. Funkce tedy nalezne použití například v makru provádějícím delší výpočet před začátkem tohoto výpočtu.

Při "úspěšném" zobrazení okna vrátí funkce TRUE. Je-li zavolána podruhé, aniž mezitím byla volána funkce **Wait_box_hide**, neprovede se nic (tj. okno zůstane zobrazeno s titulkem a textem z předchozího volání) a návratová hodnota bude FALSE.

Hodnota: Při "úspěšném" zobrazení okna funkce vrátí TRUE. Jinak vrací FALSE.

Viz: **Wait_box_hide**
Wait

WindowsCascade

```
procedure WindowsCascade;
```

Popis: Procedura spustí příkaz **Kaskáda** z menu **Okno**.

Viz: **WindowsTile**
ArrangeIcons

WindowsTile

```
procedure WindowsTile;
```

Popis: Procedura spustí příkaz **Mozaika** z menu **Okno**.

Viz: **WindowsCascade**
ArrangeIcons

WordLeft

```
function WordLeft(
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Implicitní hodnota: *shift* = FALSE

Popis: Funkce přesune kurzor na začátek předchozího slova. Pokud se řádkový kurzor pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharLeft**
CharRight
LineDown
LineUp
CaretEnd
CaretHome
LeftOfLine
RightOfLine
WordRight
RightOfWord
PageDown
PageUp
TopOfScreen
BottomOfScreen

WordRight

```
function WordRight(
  shift : boolean) : boolean;
```

Parametry: *shift* označení textu při pohybu

Implicitní hodnota: *shift* = FALSE

Popis: Funkce přesune kurzor na začátek dalšího slova. Pokud se řádkový kurzor pohnul vrací TRUE.

Omezení: Potřebuje neminimalizovaný aktuální dokument při aktivním řádkovém kurzoru.

Viz: **CharLeft**

CharRight
LineDown
LineUp
CaretEnd
CaretHome
LeftOfLine
RightOfLine
WordLeft
RightOfWord
PageDown
BottomOfScreen
TopOfScreen
PageUp

Write

```
function Write(
  var f : file; ...) : boolean;
```

Parametry:

f	proměnná typu soubor (file) označující otevřený soubor
ostatní parametry	libovolné výrazy typu boolean, char, short, integer, real, money, date, time, String, CSString nebo CSIString

Popis: Funkce **Write** konvertuje hodnoty výrazů, které obdrží jako druhý a další parametry, na znakové řetězce a zapisuje je do souboru, který je jejím prvním parametrem. Soubor musí být předem otevřen.
Za druhým a každým dalším parametrem **Write** smí být uvedena dvojtečka a celočíselný výraz. Je-li takový údaj uveden, pak vyčíslí hodnotu tohoto výrazu a předá se konverzní proceduře. Význam této hodnoty není počet vypsaných znaků (jako ve standardním Pascalu); hodnota říká, jakým způsobem se provede konverze příslušného parametru na řetězec znaků.

- Pro parametr typu *real* má výraz za dvojtečkou tentýž význam jako má ve funkci **Real2str** celočíselný parametr udávající způsob konverze.
- Pro parametr typu *money* má výraz za dvojtečkou tentýž význam jako má ve funkci **Money2str** celočíselný parametr udávající způsob konverze.
- Pro parametr typu *date* má výraz za dvojtečkou tentýž význam jako má ve funkci **Date2str** celočíselný parametr udávající způsob konverze.
- Pro parametr typu *time* má výraz za dvojtečkou tentýž význam jako má ve funkci **Time2str** celočíselný parametr udávající způsob konverze.

Pro ostatní přípustné typy parametru nemá výraz za dvojtečkou na způsob konverze vliv.

Hodnota: Hodnota funkce **Write** je TRUE, pokud se povedlo zapsat do souboru všechny konvertované hodnoty parametrů od druhého parametru včetně. Jinak bude hodnota FALSE.

Viz:
Close
Read
Writeln
Reset
Rewrite
Seek
Eof
Filelenght

Writeln

```
function Writeln(
  var f : file; ...) : boolean;
```

Parametry: *f* proměnná typu soubor (file) označující otevřený soubor
ostatní parametry libovolné výrazy typu boolean, char, short, integer, real, money, date, time, String, CSString nebo CSIString

Popis: Funkce **Writeln** konvertuje hodnoty výrazů, které obdrží jako druhý a další parametry, na znakové řetězce a zapisuje je do souboru, který je jejím prvním parametrem. Za hodnotou posledního parametru funkce zapíše do souboru přechod na nový řádek (znaky CR a LF s kódy 13 a 10). Soubor musí být předem otevřen.
Za druhým a každým dalším parametrem **Writeln** smí být uvedena dvojtečka a celočíselný výraz. Je-li takový údaj uveden, pak vyčíslí hodnotu tohoto výrazu a předá se konverzní proceduře. Význam této hodnoty není počet vypsaných znaků (jako ve standardním Pascalu); hodnota říká, jakým způsobem se provede konverze příslušného parametru na řetězec znaků. Ve všech případech význam této hodnoty tentýž jako ve funkci **Write**.

Hodnota: Hodnota funkce **Writeln** je TRUE, pokud se povedlo zapsat do souboru všechny konvertované hodnoty parametrů od druhého parametru včetně a povedlo se zapsat přechod na nový řádek. Jinak bude hodnota FALSE.

Viz: **Close**
Read
Write
Reset
Rewrite
Seek
Eof
Filelenght

Year

```
function Year(
  dt : date) : short;
```

Parametry: *dt* datum ve vnitřním formátu 602Text

Popis: Funkce extrahuje rok z data.

Hodnota: Funkce vrací nezáporné číslo označující rok (v našem letopočtu).

Viz: **Minutes**
Hours
Day
Month
Today
Make_time
Make_date
Seconds
Sec1000
Now
Day_of_week

Yesno_box

```
function Yesno_box(
```

```
var caption, text : const string) : boolean;
```

Parametry: *caption* nadpis dotazovacího okna
text text v dotazovacím okně

Popis: Funkce otevře na obrazovce modální okno obsahující nadpis (*caption)*, text (*text)*, ikonu s otazníkem a tlačítka **Ano** a **Ne**. Funkce skončí poté, co stisknete některé z těchto tlačítek.

Hodnota: Pokud jste stiskli tlačítko **Ano** funkce vrátí TRUE. Jinak vrací FALSE.

Poznámka: Použijete-li namísto popsaných způsobů opuštění tohoto okna kombinaci kláves **Ctrl+Break**, přeruší se tím běh programu ve vnitřním jazyku. Podobně jako u ostatních dialogů lze takto ukončit věčný cyklus, v němž se neustále otevírá dialogový rámeček.

Příklad:

```
function konci( L : integer) : boolean;
var
  S : string[40];
    begin
    S := "Hodnota L : " + Int2Str(L);
    S := S + #13#10"Pokračovat ?";
    konci := not YesNo_box("", S);
```

Poznámka 1.

Boolovské funkce zjišťující některé vlastnosti písma (zda je tučné, kurzívou, apod.) vracejí, není-li označen blok, hodnotu dle textu na pozici řádkového kurzoru. Je-li označen blok, vracejí TRUE v tom případě, když zjišťovanou vlastnost má celý označený text.

Poznámka 2.

Funkce, které zjišťují některé vlastnosti písma a jejichž návratová hodnota je typu s*hort*, vracejí platnou hodnotu buď není-li označen blok (pak se tato hodnota týká textu na pozici řádkového kurzoru) anebo má-li v celém označeném bloku zmíněná vlastnost (např. velikost písma) stejnou hodnotu. V opačném případě (je označen blok a různé části označeného bloku jsou vůči této vlastnosti různé) vrací funkce hodnotu *kAmbiguous*.

Poznámka 3.

Funkce nastavující vlastnosti písma (velikost, horní/dolní index apod.) ji nastavují buď v označeném bloku anebo není-li blok označen, bude nastavená vlastnost platit pro následující vkládaný text.

Poznámka 4.

Boolovské funkce zjišťující některou vlastnost odstavce (např. zarovnávání) vracejí (není-li označen žádný blok) hodnotu podle odstavce, ve kterém je řádkový kurzor. Je-li označen blok vracejí TRUE v případě, že inkriminovanou vlastnost mají všechny odstavce označeného textu.

Poznámka 5.

Funkce, které zjišťují některé vlastnosti odstavce a jejichž návratová hodnota je typu s*hort*, vracejí platnou hodnotu buď tehdy, není-li označen blok (pak se tato hodnota týká odstavce příslušejícího pozici řádkového kurzoru) anebo má-li pro všechny odstavce v označeném bloku zmíněná vlastnost stejnou hodnotu. V opačném případě vrací funkce hodnotu *kAmbiguous*.

Poznámka 6.

Funkce nastavující vlastnost odstavce (například zarovnání doleva) ji nastavují buď pro aktuální odstavec (není-li označen blok) anebo pro všechny odstavce označeného bloku.