












Programmer's Wizard 2 [Nápovida]

-  [Co je Programmer's Wizard 2?](#)
-  [Zásuvné moduly](#)
-  [Nástroje](#)
-  [Co najdu v adresáři](#)
-  [Licence](#)
-  [Registrace](#)
-  [Technické detaily](#)
-  [Autor](#)

Hledáte odpověď na konkrétní dotaz? Zkuste hledat zde:

-  [Průručka uživatele](#)

Tip: Kliknutím na logo na horní straně stránky se vždy dostanete zpět do obsahu.

Programmer's Wizard 2
<http://pwiz.hyperlink.cz/>



Programmer's Wizard rel. 2

Licence

Tento program je v současné verzi freeware. Autor nenese žádnou odpovědnost za případné škody, ztráty dat nebo finanční ztráty způsobené používáním tohoto programu. Program smí být dále redistribuován pouze v originálním archivu, který obsahuje funkční instalátor a soubory README. Redistribuce pomocí Internetu a CD-ROMů je možná jen se svolením autora. Při redistribuci nesmí být originální archiv jakkoliv zmíněn (tj. není možné mluvit, odebírat nebo přidávat soubory). Program nemůže být dekompilován nebo převáděn ze strojového kódu a nesmí být provedeny žádné změny v jeho binárních souborech ani v souborech README bez svolení autora. Uživatel může provádět instalace programu na libovolný počet počítačů, pokud k instalaci použije originální archivy, přičemž registrovat musí jen první kopii programu. Pokud bude program na jednom počítači používat více osob, vyplní a odešle registrační formulář každý z uživatelů.

Autor



Autor

Petr Ešner
petr.esner@atlas.cz
#ICQ 62399997



Co je Programmer's Wizard 2?

Programmer's Wizard 2 je freewareový nástroj určený programátorům, vývojářům a pokročilým uživatelům. Lze jej definovat jako univerzální editor pro soubory mnoha formátů, běžící na platformě **Win9x a WinNT**. Jedná se o MDI aplikaci s mnoha editory a monitory.

Jádro programu Programmer's Wizard 2 předpokládá spolupráci s více vnějšími zásuvnými moduly (add-ons, plugins...jak kdo chce). Distribuce programu obsahuje tyto komponenty:

- **Program samotný**
Tato část je společná pro všechny nástroje, editory a monitory. Stará se o uživatelské rozhraní, společné funkce, obsluhu skriptování apod.
- **Zásuvné moduly**
Obsahují vlastní implementace jednotlivých nástrojů. Každý zásuvný modul je reprezentován knihovnou (.dll), která může obsahovat jeden nebo více nástrojů. Tato verze programu je instalována s následujícími moduly:
 - **PWEDIT.DLL** – nástroje SyntaxEditor a HexEditor
 - **PWSYSMON.DLL** – Monitor procesů, Monitor vláken, Process Navigator
 - **PWSCRIPT.DLL** – Editor skriptů

Tato verze programu (s moduly PWEDIT, PWSYSMON a PWSCRIPT) nabízí:

SyntaxEditor:

Textový editor s barvením syntaxe, v současné verzi je podporováno 33 různých programovacích a skriptovacích jazyků (C++, Object Pascal, Java, Fortran, Visual Basic, HTML, CSS, PHP, VBScript, JavaScript, AWKScript, Perl, Tcl Tk, Python, CA-Clipper, Foxpro, SQL, Sybase SQL, x86 Assembler a jiné, také je zde podpora .INI konfiguračních souborů, .BAT dávkových souborů MS-DOSu, formulářů Delphi a C++ Builder..) SyntaxEditor podporuje několik druhů záložek, standardní operace se schránkou, operace Zpět/Opakovat, zobrazování čísel řádků, prohledávání dokumentu (hledat lze šetizce, 12 možností hledání), funkce 'Najdi a nahrať'. Je zde plná nastavitelnost barvení syntaxe / barevných schémát, editor podporuje Drag&Drop operace. Označený blok lze vložit do jiného souboru, jiný soubor lze vložit do právní upravovaného. Vždy je k dispozici kontextová nabídka s příhodnými příkazy. Upravovaný dokument lze přepnout do režimu 'Jen pro čtení', kdy je zamezeno jeho nechtěné změny.

HexEditor:

Hexa editor s vlastnostmi textového editoru. Zobrazuje pole ve formátu [Adresa>Data:ASCII data], šířka zobrazovacích polí a tím i počet bytů na řádku lze nastavit jednoduchým přetažením okna. Hexa editor může vkládat/odstraňovat znaky, měnit data v souboru pomocí metody In-place editing. Je zde standardní kurzor, jaký bývá u textových editorů. Hexa editor podporuje operace se schránkou, má neomezené množství kroků u funkcí Zpět/Opakovat. Obsahuje funkce pro hledání dat; hledat lze ASCII šetizce, binární hodnoty i proměnné, podporováno je 9 standardních datových typů. Prohledávání rozsáhlých dokumentů může vygenerovat i desítky tisíc výsledků s minimální závislostí rychlosti programu na počtu zpracovávaných výsledků. Výsledky hledání mohou zůstat zobrazeny v pomocném panelu, odkud lze přecházet na jejich místa v dokumentu nebo je upravovat. Dokument lze přepnout do režimu 'Jen pro čtení', kdy je zamezeno jeho nechtěné změny.

Editor skriptů:

Editor skriptů je určen pro úpravu skriptů programu Programmer's Wizard 2. Pomocí skriptů si můžete zcela přizpůsobit chování programu v určitých situacích. Můžete si napsat vlastní skript, který za Vás bude provádět časté operace nebo který se bude starat o Vaše dokumenty. Editor skriptů je textovým editorem s barvením syntaxe zaměřeným na tento skriptovací jazyk. Detaily o jazyku naleznete po instalaci v souboru SCRIPT.DOC nebo v této nápovědi v části Skriptování. Další funkce Editoru skriptů jsou zcela identické s funkcemi SyntaxEditoru, vše, co lze v SyntaxEditoru, je možné i tady. Editor skriptů navíc obsahuje funkce AutoComplete a CompletionProposal, které Vám pomohou s psaním Vašich

vlastních skriptů.

Monitor procesů:

Monitor procesů je nástroj určený pro sledování běžících procesů. Vypisuje jejich seznam, technické parametry a další informace. Výsledky sledování lze uložit do souboru nebo zkopírovat do schránky.

Monitor vláken:

Monitor vláken je dalším nástrojem, který má za úkol sledovat systémové pochody. Tento monitor vypisuje seznam existujících vláken (spuštěných i ukončených), jejich technické parametry a další informace. Výsledky sledování lze také uložit do souboru, popřípadě zkopírovat do schránky.

Process Navigator (nyní dostupné i na [WinNT 5.0](#) pro uživatele skupiny Guests!):

Tento nástroj může sledovat libovolný běžící proces. Zobrazí jeho vlákna a jejich stav (priorita, ID, handle); knihovny a moduly, které má program načtené, soubor, ze kterého byly načteny a další detaily. Tento monitor navíc umí pracovat s pamětí sledovaného procesu. Můžete prohledávat paměť na šestičíslicí, binární data nebo proměnné, nalezená data pak můžete upravovat přímo v paměti sledovaného procesu za jeho běhu (!). **Upozornění:** tento nástroj může být (jako jediný) skutečně nebezpečný, pokud nevíte, co děláte. Sledovaný proces nemůže zjistit, že byla nějak narušena jeho paměť a že jeho soukromá data byla zminována. Neuvážený zápis do paměti procesů může způsobit nekonzistentnost struktur ve sledovaném procesu a tím i jeho nestabilitu, případně nestabilitu celého systému!

Programmer's Wizard 2 používá uživatelské rozhraní standardu Microsoft Office; setkáte se tedy s panely nástrojů, animovanými grafickými nabídkami i dalšími prvky. Uživatelskému komfortu přispívá i vysoká nastavitelnost téměř všech vlastností programu. Mnoho dalších funkcí (jako jsou 'Automatické zálohování', 'Podpora externích programů', 'Podpora projektů'...) přispívají k plnému pohodlí a zvyšují efektivitu práce v programu.





Zásuvné moduly

Jádro programu Programmer's Wizard 2 předpokládá spolupráci s více vnějšími zásuvnými moduly (add-ons, plugins...jak kdo chce). Distribuce programu obsahuje tyto zásuvné moduly (dále jen *moduly*):

- [pwedit.dll](#)
- [pwsysmon.dll](#)
- [pwscript.dll](#)

Každý z *modulů* obsahuje jeden nebo více nástrojů. Programmer's Wizard 2 po spuštění prohledá příslušný adresář s *moduly* a najde všechny **knihovny**, které by mohly být jeho *moduly*. Pak všechny nalezené moduly inicializuje a připraví k práci. Načtení modulů může způsobit změnu nabídek programu (přidání příkazů) nebo změnu panelů nástrojů (přidání tlačítek). Po úspěšném načtení všech modulů je program připraven k práci.

Tato architektura programu má oproti staticky vytvořenému programu několik výhod:

- když bude k dispozici nová verze některého nástroje, nemusíte si stahovat program celý; stačí stáhnout jen aktualizovaný modul
- program je otevřený k novým modulům, když bude k dispozici nový nástroj, stačí stáhnout modul s nástrojem a program jej začne používat
- nemusíte mít načtené všechny moduly najednou, pokud nepoužíváte některý nástroj, můžete odstranit modul s tímto nástrojem a tím urychlit dobu načítání programu



Win9x

Windows 95, Window 98, Window98 SE

WinNT

WinNT 4.0, WinNT 5.0 (Windows 2000)

WinNT 5.0

Windows 2000

Knihovny

Binární soubory v spustitelném formátu, které obsahují kód a data. Jsou to součástí jiných programů, které je používají ke své práci. Název souboru mívá obvykle koncovku **.dll** (ale není to nutností).

.dll

Zkratka od *Dynamic Link Library* (pø. dynamicky linkovaná knihovna)



Technické požadavky

Programmer's Wizard 2 lze spustit na platformi [Win9x](#) a [WinNT](#). Program nevyžaduje žádný nestandardní hardware a pro plné využití funkcí postačí tato konfigurace:

Pentium 90

32MB RAM

Windows 95

Zobrazení min. 800x600 při 256 barvách





Technické specifikace

Program Programmer's Wizard 2 a všechny jeho zásuvné moduly jsou 32-bitové aplikace vytvořené pomocí vývojového prostředí Borland (Inprise) Delphi 5. Program nepoužívá žádné speciální verze knihoven Windows, které nejsou k dispozici na Windows95.


Program nelze spustit na Win3x s balíkem Win32s, který integruje částečnou 32bitovou podporu do Windows verzí 3x.



Technické detaily


Odkazy:


 [Technické požadavky](#)

 [Technické specifikace](#)

pwedit.dll
Zásuvný modul




Obsahuje nástroje:

 SyntaxEditor


 HexEditor

pwsysmon.dll
Zásuvný modul

Obsahuje nástroje:


-  Monitor procesů
-  Monitor vláken
-  Process Navigator

pwscrip.dll
Zásuvný modul

Obsahuje nástroje:
 Editor skriptů



SyntaxEditor (stručné shrnutí)

 (modul [pwedit.dll](#))


Textový editor s barvením syntaxe, v současné verzi je podporováno 33 různých programovacích a skriptovacích jazyků (C++, Object Pascal, Java, Fortran, Visual Basic, HTML, CSS, PHP, VBScript, JavaScript, AWKScript, Perl, Tcl Tk, Python, CA-Clipper, Foxpro, SQL, Sybase SQL, x86 Assembler a jiné, také je zde podpora .INI konfiguračních souborů, .BAT dávkových souborů MS-DOSu, formulářů Delphi a C++ Builder..) SyntaxEditor podporuje několik druhů záložek, standardní operace se schránkou, operace Zpět/Opakovat, zobrazování čísel řádků, prohledávání dokumentu (hledat lze øetizce, 12 možností hledání), funkce 'Najdi a nahrať'. Je zde plná nastavitelnost barvení syntaxe / barevných schémát, editor podporuje Drag&Drop operace. Označený blok lze vložit do jiného souboru, jiný soubor lze vložit do právi upravovaného. Vždy je k dispozici kontextová nabídka s příhodnými příkazy. Upravovaný dokument lze přepnout do režimu 'Jen pro čtení', kdy je zamezeno jeho nechtěné změny.

Odkazy:

 [Podporované programovací a skriptovací jazyky](#)




HexEditor (stručné shrnutí)

 (modul [pwedit.dll](#))

Hexa editor s vlastnosti textového editoru. Zobrazuje pole ve formátu [Adresa:Data:ASCII data], šířka zobrazovacích polí a tím i počet bytů na řádku lze nastavit jednoduchým přetažením okna. Hexa editor může vkládat/odstraňovat znaky, měnit data v souboru pomocí metody In-place editing. Je zde standardní kurzor, jaký bývá u textových editorů. Hexa editor podporuje operace se chránkou, má neomezené množství kroků u funkcí Zpět/Opakovat. Obsahuje funkce pro hledání dat; hledat lze ASCII řetězce, binární hodnoty i proměnné, podporováno je 9 standardních datových typů. Prohledávání rozsáhlých dokumentů může vygenerovat i desítky tisíc výsledků s minimální závislostí rychlosti programu na počtu zpracovávaných výsledků. Výsledky hledání můžou zůstat zobrazeny v pomocném panelu, odkud lze přecházet na jejich místa v dokumentu nebo je upravovat. Dokument lze přepnout do režimu 'Jen pro čtení', kdy je zamezeno jeho nechtěné změny.




Monitor procesů (stručné shrnutí)

 (modul [pwsysmon.dll](#))

Monitor procesů je nástroj určený pro sledování běžících procesů. Vypisuje jejich seznam, technické parametry a další informace. Výsledky sledování lze uložit do souboru nebo zkopírovat do schránky.



Monitor vláken (stručné shrnutí)


 (modul [pwsysmon.dll](#))

Monitor vláken je dalším nástrojem, který má za úkol sledovat systémové pochody. Tento monitor vypisuje seznam existujících vláken (spuštěných i ukončených), jejich technické parametry a další informace. Výsledky sledování lze také uložit do souboru, popřípadě zkopírovat do schránky.






Process Navigator (stručné shrnutí)

 (modul [pwsysmon.dll](#))

Tento nástroj může sledovat libovolný běžící proces. Zobrazí jeho vlákna a jejich stav (priorita, ID, handle); knihovny a moduly, které má program načtené, soubor, ze kterého byly načteny a další detaily. Tento monitor navíc umí pracovat s pamětí sledovaného procesu. Můžete prohledávat paměť na šetizce, binární data nebo proměnné, nalezená data pak můžete upravovat přímo v paměti sledovaného procesu za jeho běhu (!). **Upozornění:** tento nástroj může být (jako jediný) skutečně nebezpečný, pokud nevíte, co děláte. Sledovaný proces nemůže zjistit, že byla nějak narušena jeho paměť a že jeho soukromá data byla zminina. Neuvážený zápis do paměti procesů může způsobit nekonzistentnost struktur ve sledovaném procesu a tím i jeho nestabilitu, případně nestabilitu celého systému!



Editor skriptů (stručné shrnutí)










 (modul [pwscript.dll](#))

Editor skriptů je určen pro úpravu skriptů programu Programmer's Wizard 2. Pomocí skriptů si můžete zcela přizpůsobit chování programu v určitých situacích. Můžete si napsat vlastní skript, který za Vás bude provádět časté operace nebo který se bude starat o Vaše dokumenty. Editor skriptů je textovým editorem s barvením syntaxe zaměřeným na tento skriptovací jazyk. Detaily o jazyku naleznete po instalaci v souboru SCRIPT.DOC nebo v této nápovědi v části Skriptování. Další funkce Editoru skriptů jsou zcela identické s funkcemi SyntaxEditoru, vše, co lze v SyntaxEditoru, je možné i tady. Editor skriptů navíc obsahuje funkce AutoComplete a CompletionProposal, které Vám pomohou s psaním Vašich vlastních skriptů.



Nástroje
































Odkazy:

-  [pwwedit.dll](#)
-  [SyntaxEditor](#)
-  [HexEditor](#)
-  [pwwsysmon.dll](#)
-  [Monitor procesů](#)
-  [Monitor vláken](#)
-  [Process Navigator](#)
-  [pwwscript.dll](#)
-  [Editor skriptů](#)



Podporované programovací jazyky

SyntaxEditor podporuje tyto programovací a skriptovací jazyky:

-  68HC11
-  ADSP21xx
-  AWK skript
-  Baan 4GL
-  C++
-  CA-Clipper
-  CSS (Cascading Stylesheets)
-  Delphi / C++ Builder formuláøe
-  Fortran
-  Foxpro
-  Galaxy
-  Gembase
-  HP48
-  HTML
-  INI konfiguraèní soubory
-  Java
-  JavaScript
-  KIX32
-  MS VBScript
-  MS-DOS dávkový soubor
-  ObjectPascal
-  Perl
-  PHP
-  Python
-  Resource files (soubory zdrojù .rc)
-  SQL
-  Standard ML
-  Sybase SQL
-  Tcl Tk
-  Visual Basic
-  x86 Assembly Language





Nastavení adresářů programu

Programmer's Wizard 2 pracuje s tímto výchozími adresáři:

složka s projekty

Zde se ukládají projekty, vytvořené programem. Výchozí umístění: \Projects

složka se zásuvnými moduly


V této složce jsou zásuvné moduly používané programem. Výchozí umístění: \Plugins

složka se skripty

Tady jsou skripty, vytvořené programem. Výchozí umístění: \Scripts

Umístění těchto složek lze změnit z dialogu *Souštiní a ukončování*, který lze otevřít vybráním stejnojmenného příkazu z hlavní nabídky 'Soubor' / 'Nastavení'. Změny se projeví při příštím spuštění programu.

Odkazy:

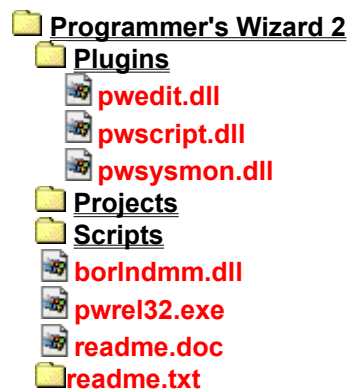
 Struktura složky programu





Struktura složky programu

Toto je adresářová struktura programu tak, jak se vytvoří po instalaci programu. Kliknutím na jednotlivé ikony získáte jejich popis.



Odkazy:

 Umístění složek **Plugins**, **Projects** a **Scripts** lze kdykoliv změnit.



readme.txt

Základní informace o programu Programmer's Wizard 2, shrnutí funkcí, licence a další informace. Vše v èistí textovém formátu.

readme.doc

Základní informace o programu Programmer's Wizard 2, shrnutí funkcí, licence a další informace. Urèeno pro MS Word 97 a vyšší.

pwrel32.exe

Vlastní program Programmer's Wizard 2.

borIndmm.dll

[Knihovna](#) 'Správce sdílené paměti'



Projekty

Programmer's Wizard 2 obsahuje podporu projektů, pomocí kterých si můžete přehledně uspořádat práci.

Projekt je seznam souborů, které patří k určité skupině. Například, projekt s WWW prezentací bude obsahovat všechny soubory, které jsou k prezentaci nutné - tedy všechny její stránky, skripty i grafiku. Všechny tyto soubory jsou nutné pro prezentaci. Analogicky, dejme je všechny do jedné *krabice* s popisem "Projekt WWW prezentace", protože pak nebude potřeba prohledávat spousty dokumentů na stole, stačí jít rovnou do krabice. Můžeme si takto rozdělit všechnu práci a popsané krabice dát .. třeba na polici.

V programu je pak touto policí nabídka 'Projekt'. Z této nabídky lze otevírat projekty, vytvářet nové nebo upravovat existující.

Při vytváření projektu se zobrazí dialog 'Upravit projekt', ve kterém je potřeba přidat do projektu soubory, které má obsahovat. Po vybrání souborů, zadání jména a popisu projektu se projekt automaticky otevře a také se zavěou všechny okna se soubory, které byly otevřeny před vytvořením projektu. Můžete začít pracovat. K otevření souboru z projektu stačí vybrat příkaz 'Otevřít dokument' z nabídky 'Projekt' - zobrazí se seznam všech souborů, které jsou v projektu. Jednoduchým kliknutím na jméno souboru je příslušný soubor otevře a přepřaví k editaci.

Až bude potřeba projekt zavěřit, nezavěrejte napřed všechny otevřené dokumenty, místo toho vyberte příkaz 'Zavěřit projekt' z nabídky 'Projekt'. Okna se zavěou všechna najednou, pokud je nikterý soubor změněn, zobrazí program upozornění a můžete změny uložit. Hlavní výhoda používání projektů spoěívá v tom, že program při zavěírání projektu uloží také jeho stav (otevřené soubory, polohu jejich oken atd.) a při příštím otevření projektu je obnoví a tak lze pokračovat v práci od místa, kde se přestalo. Další výhodou použití projektů je snadný přístup k souborům; není nutné otevírat okna a v nich hledat potřebný soubor, stačí se proklikat jednoduchou nabídkou.

Odkazy:

 [Plochy](#)





Plochy

Plochy jsou součástí projektů. Každý projekt může obsahovat několik ploch. Při vytvoření projektu je také vytvořena jeho primární plocha.

V projektu programu Programmer's Wizard 2 je plochou nazývána ta část okna, ve které se zobrazují upravované dokumenty a okna nástrojů. *Stavem plochy* je rozuměn seznam otevřených souborů, poloha jejich oken, stav nástrojů použitých k jejich editaci apod. Programmer's Wizard 2 při zavírání projektu zavře jeho primární plochu a uloží její *stav*. Při opětovném otevření projektu pak otevře primární plochu a stav obnoví, tj. otevře opět všechny dokumenty, které byly otevřeny před zavřením plochy a obnoví nástroje do takového stavu, v jakém před zavřením byly.

Projekt programu Programmer's Wizard 2 může však obsahovat více ploch najednou, každou se svým vlastním *stavem*. Pokud budeme vycházet z příkladu WWW prezentace, probraného v tématu Projekty, můžeme primární plochu nazvat 'HTML' a otevřít do ní několik HTML stránek. Pak lze vytvořit novou plochu s názvem 'Skripty', ve které budeme pracovat jen se skripty. Hlavní výhodou takového rozdělení práce je to, že *každá* plocha bude při uzavření uložena. Když tedy budete chtít pracovat se skripty, otevřete stejnojmennou plochu a upravíte skript. Pak se lze přepnout zpět na plochu 'HTML' a pokračovat v úpravě stránek tam, kde se přestalo.

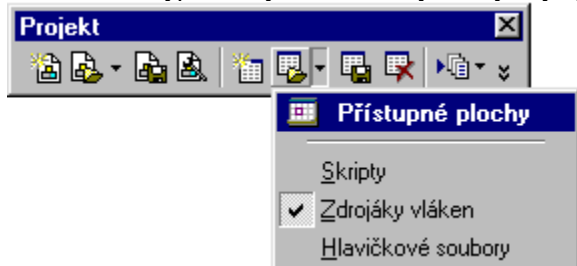
Projekty a plochy programu Programmer's Wizard 2 zřehledují práci. Při jejich použití může být zdoluhavé používání dialogu 'Otevřít' minulostí.

Pozn.: pro nejvyšší pohotovost přístupu k plochám a funkcím projektů používejte místo nabídky radij panel nástrojů 'Projekty'. Tento způsob je zvláště vhodný pro rychlé otevírání ploch a souborů, protože vedle některých tlačítek na panelu nástrojů se zobrazuje šipka, po jejímž stisknutí se zobrazí příhodná nabídka.

Hledejte tyto šipky:



Takto může vypadat výsledek, zde jde o rychlý výběr plochy:



Odkazy:

[Projekty](#)

















Panely nástrojů

Programmer's Wizard 2 obsahuje několik panelů nástrojů, které zrychlují a zpřístupňují práci s programem. Obsahují nejbližnější příkazy z nabídek a jsou uživatelsky konfigurovatelné (umístění, počet tlačítek, jejich význam). Některé funkce pro zrychlení práce jsou dostupné dokonce jen z panelu nástrojů (viz. šipky v tématu Plochy).

Vyberte panel, o kterém se chcete dozvědět více:

-  Nabídky
-  Soubor
-  Úpravy
-  Poznámky
-  Okno
-  Formát
-  Projekt
-  Záložky
-  Dokumenty
-  Skripty

-  Poznámky (panel)
-  Složka Dokumenty (panel)

Odkazy:

-  Vlastní panely nástrojů






Panel 'Soubor'

Panel nástrojů 'Soubor' dává přístup k důležitým příkazům ze stejnomené nabídky. Zde je obrázek panelu nástrojů. Kliknutím na jeho části se dozvíte více.



(kliknout zde)

Odkazy:

 [Panely nástrojů](#)



Soubor | Nový

Tento příkaz vytvoří nový soubor použitím výchozího editoru. Pokud není nainstalován žádný výchozí editor, zobrazí se okno se seznamem všech možných typů dokumentů.

Soubor | Otevřít

Tento příkaz otevře dialogové okno 'Otevřít', ve kterém lze vybrat soubor, který se má otevřít. Vybraný soubor se pak otevře v příslušném nástroji.

Soubor | Nový (šipka)

Tato šipka zobrazí po svém stisknutí nabídku se seznamem všech typů souborů, které mohou být vytvořeny. Obsah nabídky záleží na nainstalovaných modulech.

Soubor | Uložit

Po vybrání tohoto příkazu se uloží právě upravovaný dokument.

Soubor | Uložit vše

Po vybrání tohoto příkazu se uloží všechny právi upravované zmíniné soubory.

Soubor | Tisk

Po vybrání tohoto příkazu se vytiskne právě upravovaný dokument, pokud to daný nástroj umožňuje.

Soubor | Vlastnosti

Po vybrání tohoto souboru se zobrazí vlastnosti právi upravovaného souboru (jméno, velikost, atributy). Ze zobrazeného okna lze také vlastnosti souboru minit.



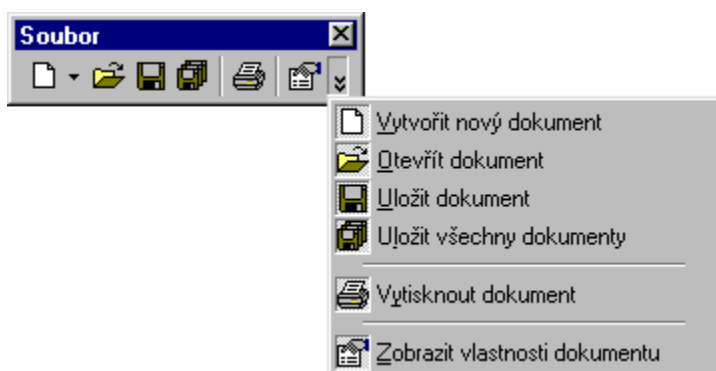
Vlastní panely nástrojů

Panely nástrojů programu Programmer's Wizard 2 mohou být přizpůsobeny každému (téměř každému :-)) uživateli. K nastavení panelů lze použít dvě techniky:

☞ Volba tlačítek panelu vybráním tlačítka 'Šipka' (

☑, viz např. panel Soubor), které je na konci každého panelu.

Po stisku šipky se zobrazí nabídka se seznamem všech příkazů, které jsou na panelu k dispozici.



Odznačením jednotlivých příkazů se příslušná tlačítka ukryjí, jejich opitným označením se tlačítka zviditelní.

☞ Dialog 'Upravit panely nástrojů'

Tento dialog nabízí nastavení všech vlastností všech panelů z jednoho místa. Dialog 'Upravit panely nástrojů' lze otevřít vybráním stejnojmenného příkazu z nabídky, které se otevře po kliknutí pravým myšítkem na kterýkoliv panel.






Panel 'Úpravy'

Panel nástrojů 'Úpravy' dává přístup k důležitým příkazům ze stejnomené nabídky. Zde je obrázek panelu nástrojů. Kliknutím na jeho části se dozvíte více.



(kliknout zde)

Odkazy:

 [Panely nástrojů](#)



Úpravy | Vymout

Vyjme vybraná data do schránky.

Úpravy | Kopírovat

Zkopíruje vybraná data do schránky.

Úpravy | Vložit

Vloží do dokumentu data ze schránky.

Úpravy | Zpět

Vrátí poslední provedenou operaci.

Úpravy | Zpět (šipka)

Zobrazí seznam provedených operací, které lze vrátit, pokud to příslušný nástroj dovoluje.

Úpravy | Opakovat

Zopakuje poslední vrácenou operaci.

Úpravy | Opakovat (šipka)

Zobrazí seznam vrácených operací, které lze zopakovat, pokud to příslušný nástroj dovoluje.

Úpravy | Najít

Zobrazí okno, pomocí kterého je možné začít prohledávání dokumentu.

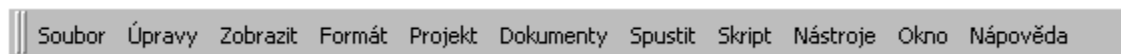
Úpravy | Nahradit

Zobrazí okno, pomocí kterého je možné nahradit určitá data v dokumentu.














Panel 'Nabídky'

Panel nástrojů 'Nabídky' dává přístup ke všem pod-nabídkám programu. Zde je obrázek panelu nástrojů. Kliknutím na jeho části se dozvíte více.



(kliknout zde)

..nebo můžete kliknout na tyto názvy jednotlivých nabídek:

-  [Soubor](#)
-  [Úpravy](#)
-  [Zobrazit](#)
-  [Formát](#)
-  [Projekt](#)
-  [Dokumenty](#)
-  [Spustit](#)
-  [Skript](#)
-  [Nástroje](#)
-  [Okno](#)
-  [Nápověda](#)

Odkazy:

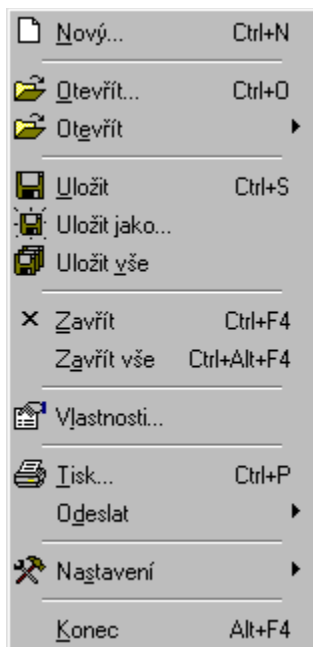
-  [Panely nástrojů](#)






Nabídka 'Soubor'

Nabídka 'Soubor' se otevře po kliknutí na stejnojmenný příkaz na panelu nabídek. Zde je obrázek nabídky. Kliknutím na jednotlivé části se dozvíte více.



(kliknout zde)

Odkazy:

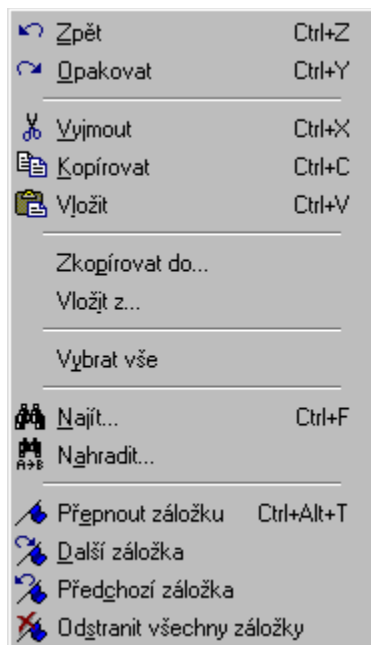
 [Panely nástrojů](#)





Nabídka 'Úpravy'

Nabídka 'Úpravy' se otevře po kliknutí na stejnojmenný příkaz na panelu nabídek. Zde je obrázek nabídky. Kliknutím na jednotlivé části se dozvíte více.



(kliknout zde)

Odkazy:

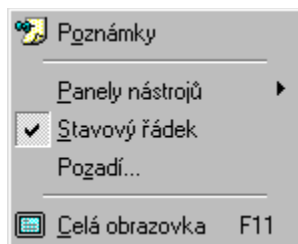
 [Panely nástrojů](#)






Nabídka 'Zobrazit'

Nabídka 'Zobrazit' se otevře po kliknutí na stejnojmenný příkaz na panelu nabídek. Zde je obrázek nabídky. Kliknutím na jednotlivé části se dozvíte více.



(kliknout zde)

Odkazy:

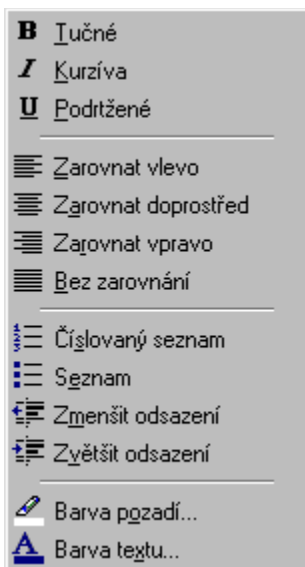
 Panely nástrojů






Nabídka 'Formát'

Nabídka 'Formát' se otevře po kliknutí na stejnojmenný příkaz na panelu nabídek. Zde je obrázek nabídky. Kliknutím na jednotlivé části se dozvíte více.



(kliknout zde)

Odkazy:

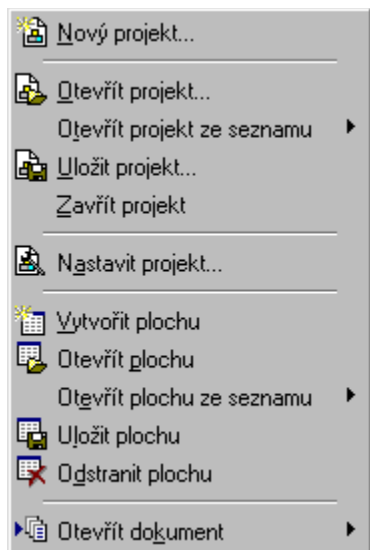
 [Panely nástrojů](#)






Nabídka 'Projekt'

Nabídka 'Projekt' se otevře po kliknutí na stejnojmenný příkaz na panelu nabídek. Zde je obrázek nabídky. Kliknutím na jednotlivé části se dozvíte více.



(kliknout zde)

Odkazy:

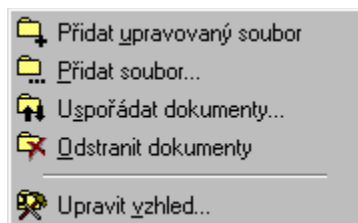
 [Panely nástrojů](#)






Nabídka 'Dokumenty'

Nabídka 'Dokumenty' se otevře po kliknutí na stejnojmenný příkaz na panelu nabídek. Zde je obrázek nabídky. Kliknutím na jednotlivé části se dozvíte více.



(kliknout zde)

Odkazy:

 [Panely nástrojů](#)






Nabídka 'Spustit'

Tato nabídka je portálem nainstalovaných zásuvných modulů. Obsahuje všechny nástroje, které lze spustit. Kliknutím na libovolný příkaz v této nabídce se otevře příslušný nástroj.

Odkazy:

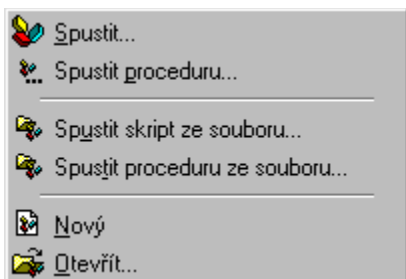
 Panely nástrojů






Nabídka 'Skript'

Nabídka 'Skript' se otevře po kliknutí na stejnojmenný příkaz na panelu nabídek.
Zde je obrázek nabídky. Kliknutím na jednotlivé části se dozvíte více.



(kliknout zde)

Odkazy:

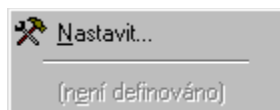
 Panely nástrojů





Nabídka 'Nástroje'


Nabídka 'Nástroje' se otevře po kliknutí na stejnojmenný příkaz na panelu nabídek. Zde je obrázek nabídky. Kliknutím na jednotlivé části se dozvíte více.




(kliknout zde)

Tato nabídka je uživatelsky definovaná, viz. Externí nástroje.

Odkazy:

 Externí nástroje

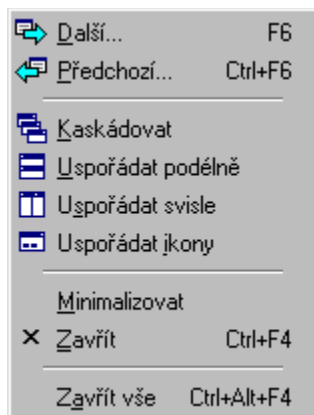
 Panely nástrojů






Nabídka 'Okno'

Nabídka 'Okno' se otevře po kliknutí na stejnojmenný příkaz na panelu nabídek. Zde je obrázek nabídky. Kliknutím na jednotlivé části se dozvíte více.



(kliknout zde)

Odkazy:

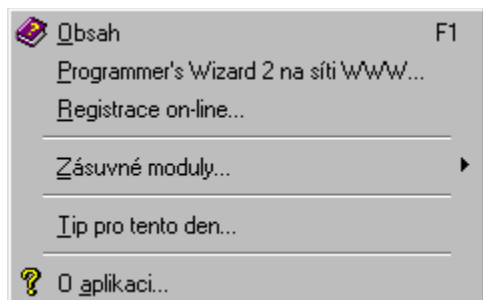
 [Panely nástrojů](#)






Nabídka 'Nápovida'

Nabídka 'Nápovida' se otevře po kliknutí na stejnojmenný příkaz na panelu nabídek. Zde je obrázek nabídky. Kliknutím na jednotlivé části se dozvíte více.



(kliknout zde)

Odkazy:

 [Panely nástrojů](#)



Soubor | Otevřít (nedávno otevřené soubory)

Zobrazí nabídku se seznamem naposledy použitých dokumentů. Kliknutím na některý dokument se tento otevře v příslušném nástroji.

Soubor | Uložit jako

Uloží právi upravovaný dokument pod novým, vybraným názvem.

Soubor | Zavřít

Umožní uložení změn v právi otevřeném dokumentu pak jej uzavře.

Soubor | Zavřít vše

Umožní uložení změn ve všech právi otevřených dokumentech pak je uzavře.

Soubor | Odeslat

Zobrazí podnabídku se seznamem míst, na která lze odeslat právě otevřený dokument.

Soubor | Nastavení

Otevře podnabídku se seznamem funkcí, které lze nastavit.

Soubor | Konec

Uzavře projekt (pokud je otevřen), všechna okna a pak ukončí program.

Úpravy | Vybrat vše

Označí všechna data v dokumentu.

Úpravy | Zkopírovat do

Zkopíruje vybraná data do jiného dokumentu.

Úpravy | Vložit z

Vloží do dokumentu data z jiného souboru.

Úpravy | Pøepnout záložku

Pøepne záložku na aktuální pozici.

Úpravy | Další záložka

Přejde na další nastavenou záložku.

Úpravy | Pøedchozí záložka

Pøejde na pøedchozí nastavenou záložku.

Úpravy | Odstranit všechny záložky

Odstraní všechny záložky v dokumentu.

Zobrazit | Poznámky

Zobrazí nebo ukryje panel Poznámky.

Zobrazit | Panely nástrojů

Zobrazí nabídku se seznamem všech panelů nástrojů, pomocí které lze zapnout nebo vypnout kterýkoliv panel nástrojů. Tato nabídka se také zobrazí po kliknutí pravým myšítkem na kterýkoliv panel nástrojů.

Zobrazit | Stavový øádek

Zobrazí nebo ukryje stavový øádek.

Zobrazit | Pozadí

Zobrazí dialog, ve kterém lze nastavit barvu nebo tapetu pozadí plochy programu.

Zobrazit | Celá obrazovka

Přepíná mezi zobrazováním přes celou obrazovku a normálním režimem.

Formát | Tuèné

Nastaví tuèné písmo.

Formát | Kurzíva

Nastaví písmo stylu kurzíva.

Formát | Podtržené

Nastaví podtržené písmo.

Formát | Zarovnat vlevo

Zarovná blok textu nebo dat vlevo.

Formát | Zarovnat doprostřed

Zarovná blok textu nebo dat doprostřed.

Formát | Zarovnat vpravo

Zarovná blok textu nebo dat vpravo.

Formát | Bez zarovnání

Zruší zarovnání bloku textu nebo dat.

Formát | ěíslovaný seznam

Vytvoří ěíslovaný seznam.

Formát | Seznam

Vytvoří obyèejný seznam.

Formát | Zmenšit odsazení

Zmenší odsazení aktuálních nebo vybraných dat.

Formát | Zvitšit odsazení

Zvitšit odsazení aktuálních nebo vybraných dat.

Formát | Barva pozadí

Umožní nastavit jinou barvu pozadí.

Formát | Barva textu

Umožní nastavit jinou barvu textu.

Projekt | Nový projekt

Otevře okno, ve kterém lze vytvořit a nastavit projekt. Vytvořený projekt pak otevře.

Projekt | Otevřít projekt

Zobrazí dialog, ve kterém lze vybrat existující projekt. Vybraný projekt otevře.

Projekt | Otevřít projekt ze seznamu

Zobrazí nabídku se seznamem všech existujících projektů. Po kliknutí na některý z názvů se otevře příslušný projekt.

Projekt | Uložit projekt

Uloží právi otevřený projekt.

Projekt | Zavřít projekt

Uzavře právi otevřený projekt.

Projekt | Nastavit projekt

Zobrazí okno umožňující nastavit vlastnosti právi otevřeného projektu.

Projekt | Vytvořit plochu

Vytvoří novou plochu právi otevřeného projektu.

Projekt | Otevřít plochu

Umožní otevřít nikterou z ploch právi otevřeného projektu.

Projekt | Otevřít plochu ze seznamu

Zobrazí nabídku se seznamem všech existujících ploch právi otevřeného projektu. Po kliknutí na některou z položek pak příslušnou plochu otevře.

Projekt | Uložit plochu

Uloží právi otevřenou plochu. Plocha je také automaticky uložena při jejím zavírání, tj. i při zavírání projektu.

Projekt | Odstranit plochu

Odstraní právi otevřenou plochu.

Projekt | Otevřít dokument

Zobrazí seznam všech souborů, které jsou v právě otevřeném projektu. Po kliknutí na některý z názvů se příslušný dokument otevře.

Dokumenty | Pøidat upravovaný soubor

Pøidá právi upravovaný dokument do složky dokumentů.

Dokumenty | Přidat soubor

Zobrazí dialog, ve kterém je možné vybrat jeden nebo více dokumentů, které se mají do složky dokumentů přidat.

Dokumenty | Uspořádat dokumenty

Umožní uspořádat obsah složky dokumentů.

Dokumenty | Odstranit dokumenty

Odstraní všechny zástupce ze složky dokumentů.

Dokumenty | Upravit vzhled

Zobrazí okno umožňující změnit vzhled složky dokumentů.

Skript | Spustit

Zobrazí okno se seznamem nalezených skriptů, vybraný skript pak spustí.

Skript | Spustit proceduru

Zobrazí okno se seznamem nalezených skriptů, po vybrání skriptu zobrazí dotaz na název procedury, která se má provést. Vybranou proceduru pak spustí.

Skript | Spustit skript ze souboru

Zobrazí dialog, ve kterém lze najít a vybrat skript. Vybraný skript pak spustí.

Skript | Spustit proceduru ze souboru

Zobrazí dialog, ve kterém lze najít a vybrat skript, po vybrání skriptu zobrazí dotaz na název procedury, která se má provést. Vybranou proceduru pak spustí.

Skript | Nový

Vytvoří nový skript nainstalovaným nástrojem pro práci se skripty.

Skript | Otevřít

Zobrazí dialog, ve kterém lze najít a vybrat skript, vybraný skript pak otevře v nainstalovaném nástroji pro práci se skripty.

Nástroje | Nastavit

Zobrazí dialog umožňující nastavení externích nástrojů.

Okno | Další

Pøenese do popøedí další okno.

Okno | Pøedchozí

Pøenese do popøedí pøedchozí okno.

Okno | Kaskádovat

Kaskáduje všechna otevřená okna.

Okno | Uspořádat podél ní

Uspořádá všechna okna podél ní.

Okno | Uspořádat svisle

Uspořádá všechna okna svisle.

Okno | Uspořádat ikony

Uspořádá všechna minimalizovaná okna.

Okno | Minimalizovat

Minimalizuje právě otevřené okno.

Okno | Zavřít

Zavře právi otevřené okno.

Okno | Zavřít vše

Zavře všechna otevřená okna.

Nápovida | Obsah

Zobrazí seznam nápovidy.

Nápovida | Programmer's Wizard 2 na síti WWW

Pøejde na WWW stránky programu Programmer's Wizard 2.

Nápovida | Registrace on-line...

Zobrazí registraèní formulář programu Programmer's Wizard 2.

Nápovida | Zásuvné moduly

Zobrazí informace o nainstalovaných zásuvných modulech.

Nápovida | Tip pro tento den

Zobrazí okno Tip dne.

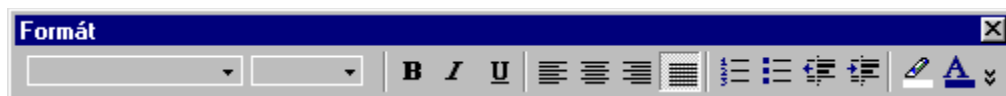
Nápovida | O aplikaci

Zobrazí informace o aplikaci (verze, licence)




Panel 'Formát'

Panel nástrojů 'Formát' dává přístup k důležitým příkazům ze stejnojmenné nabídky. Zde je obrázek panelu nástrojů. Kliknutím na jeho části se dozvíte více.



(kliknout zde)

Odkazy:

 [Panely nástrojů](#)



Formát | Písmo

Umožní vybrat písmo.

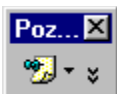
Formát | Velikost písma

Umožní vybrat velikost písma.




Panel 'Poznámky'

Panel nástrojů 'Poznámky' dává přístup k důležitým příkazům z nabídky 'Zobrazit'. Zde je obrázek panelu nástrojů. Kliknutím na jeho části se dozvíte více.



(kliknout zde)

Odkazy:

 [Panely nástrojů](#)



Zobrazit | Poznámky (šipka)

Zobrazí nabídku s příhodnými příkazy pro otevření nebo vymazání poznámek.




Panel 'Záložky'

Panel nástrojů 'Záložky' dává přístup k důležitým příkazům z nabídky 'Úpravy'.
Zde je obrázek panelu nástrojů. Kliknutím na jeho části se dozvíte více.



(kliknout zde)

Odkazy:

 [Panely nástrojů](#)






Panel 'Okno'

Panel nástrojů 'Okno' dává přístup k důležitým příkazům ze stejnomené nabídky. Zde je obrázek panelu nástrojů. Kliknutím na jeho části se dozvíte více.



(kliknout zde)

Odkazy:

 [Panely nástrojů](#)





Panel 'Projekt'

Panel nástrojů 'Projekt' dává přístup k důležitým příkazům ze stejnojmenné nabídky. Zde je obrázek panelu nástrojů. Kliknutím na jeho části se dozvíte více.



(kliknout zde)

viz. [Panely nástrojů](#)

Projekt | Otevřít (šipka)

Zobrazí seznam nalezených skriptů. Vybraný skript otevře v nainstalovaném nástroji pro práci se skripty.

Projekt | Spustit skript (šipka)

Zobrazí seznam nalezených skriptů. Vybraný skript spustí.

Projekt | Spustit proceduru (šipka)

Zobrazí seznam nalezených skriptů. Po vybrání jednoho otevře dialog, do kterého lze vložit název procedury. Pak proceduru spustí.




Panel 'Dokumenty'

Panel nástrojů 'Dokumenty' dává přístup k důležitým příkazům ze stejnomené nabídky. Zde je obrázek panelu nástrojů. Kliknutím na jeho části se dozvíte více.



(kliknout zde)

Odkazy:

 [Panely nástrojů](#)






Panel 'Skript'

Panel nástrojů 'Skript' dává přístup k důležitým příkazům ze stejnomené nabídky. Zde je obrázek panelu nástrojů. Kliknutím na jeho části se dozvíte více.



(kliknout zde)

Odkazy:

 [Panely nástrojů](#)





Dokumenty

Programmer's Wizard 2 obsahuje složku *Dokumenty*, která je určena pro pohodlný přístup k dokumentům, které jsou otevírány často a musí být vždy po ruce. Složka *Dokumenty* nemá nic společného se stejnojmennou složkou ve Windows. Složka *Dokumenty* Windows je fyzický adresář a obsahuje fyzické soubory zatímco složka *Dokumenty* programu Programmer's Wizard 2 je virtuální a obsahuje virtuální zástupce fyzických souborů.

Odkazy:



[Více informací o Dokumentech](#)



[Složka Historie](#)



[Jak zobrazit obsah složky Dokumentů](#)



[Nabídka 'Dokumenty'](#)



[Panel nástrojů 'Dokumenty'](#)





Více o složce Dokumenty

Složka Dokumenty programu Programmer's Wizard 2 může obsahovat zástupce libovolného dokumentu. Obsah této složky může být zobrazen jako panel Dokumenty v hlavním okně programu. Panel Dokumenty zobrazuje seznam všech zástupců uvnitř složky, kliknutím na některého z těchto zástupců se otevře příslušný dokument v programu. Složka Dokumenty tak umožňuje rychlý přístup k dokumentům, které jsou potřeba vždy 'u ruky' kdy i proklikávání se nabídkami je zdržující.



Panel Dokumenty zobrazuje kromě dokumentů i složku Historie, která vždy obsahuje několik naposledy otevřených dokumentů.







Složka Historie

Složka *Historie* vždy obsahuje několik naposledy použitých dokumentů, nejvíce 16 (pro přehlednost). Do této složky je přidán odkaz vždy, když Programmer's Wizard 2 otevírá nějaký dokument, lhostejno odkud. Když je použit dialog 'Otevřít', otevřen dokument z projektu nebo byl nějaký dokument přetažen na okno programu, vždy bude zástupce otevíraného dokumentu vložen na první místo ve složce Historie. Ke složce Historie se lze dostat dvěma způsoby:

-  Vybráním příkazu 'Otevřít...' z nabídky Soubor
-  Zobrazením složky Historie pomocí panelu Dokumenty (více...)

Odkazy:

-  Složka Dokumenty
-  Jak otevřít složku Dokumenty







Jak zobrazit složku Dokumenty

Složky Dokumenty a Historie lze zobrazit podobně jako ostatní panely nástrojů.

- 1) Klikněte pravým myšítkem na libovolný panel nástrojů
- 2) Z nabídky vyberte možnost *Panel 'Složka Dokumenty'*

Odkazy:

 Složka Dokumenty



















 Složka Historie






Pøíruèka užívatelè

Vyberte téma, o kterém se chcete dozvědít více.

-  [Editor skriptù](#)
-  [Externí nástroje](#)
-  [HexEditor](#)
-  [Klávesové zkratky](#)
-  [Monitor procesù](#)
-  [Monitor vláken](#)
-  [Nabídky](#)
-  [Panely nástrojù](#)
-  [Pascal Script](#)
-  [Plochy](#)
-  [Process Navigator](#)
-  [Projekty](#)
-  [Složka Dokumenty](#)
-  [Složka Historie](#)
-  [Skriptování](#)
-  [SyntaxEditor](#)
-  [Vlastní panely nástrojù](#)
-  [Zálohování](#)

Hledáte obecnější téma? Zkuste hledat v obsahu.

-  [Obsah](#)





Automatické zálohování

Programmer's Wizard 2 obsahuje funkce pro zálohování upravovaných souborů. Program může být nastaven tak, aby před uložením upravovaného dokumentu (nezáleží na **nástroji**, který je použit) vytvořil jeho záložní kopii. Můžete si zvolit kam a jak má být záložní kopie uložena. Program může zmínit jméno souboru, uložit kopii do jiné složky apod.

Nastavení této funkce je možné z dialogového okna Automatické zálohování, které se otevře po vybrání stejnojmenného příkazu z nabídky Nastavení v hlavní nabídce Soubor.

Výchozí nastavení:

Po instalaci je tato funkce zcela vypnuta, po jejím spuštění se nastaví zálohování podle klíče klíče `%name.~%ext`.

Odkazy:



Klíče Automatického zálohování



Nástroje

Nástroje jsou tvořeny moduly. Nástrojem je například SyntaxEditor nebo HexEditor.



Klíče Automatického zálohování

Klíče Automatického zálohování definují způsob, kterým probíhá zálohování. Pomocí těchto klíčů si můžete zvolit jméno souboru se záložní kopí.

Klíč je řetězec znaků, který může obsahovat tyto makra:

%name% - název originálního dokumentu

%ext% - koncovka originálního dokumentu

Příklad: zálohujeme dokument s názvem "*SOUBOR.HTM*"

..pak bude makro **%name%** znamenat "*SOUBOR*"

..a makro **%ext%** bude rovno řetězci "*HTM*".

Pomocí kombinací normálních znaků a maker lze nastavit jakékoliv jméno výsledného souboru.

Příklady (originálním souborem bude "*SOUBOR.HTM*"):

Klíč: "*%name%.~%ext%*"

Kopie bude mít název: "*SOUBOR.~HTM*"

Klíč: "*%name%-%ext%.tmp*"

Kopie bude mít název: "*SOUBOR-HTM.tmp*"

Specifikace cesty

Klíč může obsahovat i specifikaci cesty, oddělené zprávným lomítkem.

Příklad:

Klíč: "*%ext%\%name%.%ext%*"

Kopie bude vytvořena ve složce "*HTM*" a bude mít název "*SOUBOR.HTM*".

Odkazy:

[Automatické zálohování](#)





Klávesové zkratky


Programmer's Wizard 2 podporuje plnou nastavitelnost všech klávesových zkratk programů. Můžete si přizpůsobit zkratky pro jakýkoliv příkaz z nabídek, změny se projeví okamžitě po nadefinování. Tyto vlastnosti ovládání programu lze nastavit z dialogového okna *Klávesové zkratky*, které se otevře po vybrání stejnojmenného příkazu z podnabídky 'Nastavení' nabídky Soubor.

Jak na to:

- 1) otevřete nabídku 'Soubor' z panelu nabídek
- 2) vyberte nabídku 'Nastavení'
- 3) klikněte na příkaz *Klávesové zkratky*

Zobrazí se okno *Nastavení klávesových zkratk*, hledáte-li nápovědu k tomuto oknu, naleznete ji zde.

Odkazy:

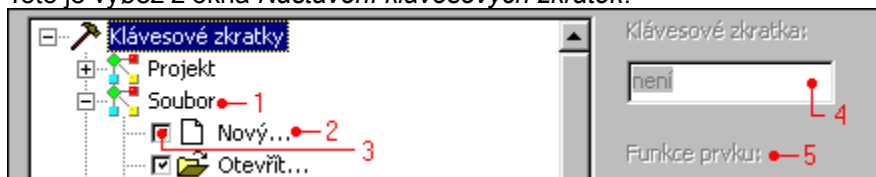
 Jak si přizpůsobit klávesové zkratky





Jak si pøizpůsobit klávesové zkratky


Toto je výøez z okna *Nastavení klávesových zkratk*:



Legenda:

- 1) Skupina pøíkazù. Pøíkazy jsou dïleny do skupin podle toho, k jaké pøísluší nabídce.
- 2) Název a ikona pøíkazu tak, jak je vidít v pøíslušné nabídce.
- 3) Zaškrťovací tlačítko, zaškrtnutí indikuje pøítomnost klávesové zkratky.
- 4) Zde se zobrazuje klávesová zkratka právi vybraného pøíkazu. Pomocí tohoto pole lze také zkratku pøedefinovat. Stačí kliknout do pole a stisknout novou požadovanou klávesovou zkratku.
- 5) Pomocné pole, zde se zobrazí popis (krátká nápovída) k vybranému pøíkazu.

Odkazy:

 [Klávesové zkratky](#)





Externí nástroje

Programmer's Wizard 2 podporuje externí nástroje. *Externí nástroj* je libovolný program, který používáte při práci. Portálem k externím nástrojům je nabídka Nástroje.

Externím nástrojem může být například speciální kalkulátor nebo program, který nijak zpracuje upravovaný soubor. Můžete si vytvořit seznam nástrojů, které používáte a k nim pak přistupovat jednoduchým vybráním jejich názvu z nabídky Nástroje.

Externím nástrojům lze předat při spuštění některé parametry, více informací naleznete přímo v programu.

Vytvořit seznam externích nástrojů můžete vybráním příkazu 'Nastavit' z nabídky Nástroje.

Odkazy:

 Nabídka 'Nástroje'





Skriptování

Tato část nápovědy obsahuje popis [skriptovacího](#) jazyka [Pascal Script](#), integrovaného do prostředí programu [Programmer's Wizard 2](#). Je zde základní popis syntaxe jazyka, podporované datové typy a externí funkce. Je zde také několik málo příkladů, které Vám pomohou před psaním vlastních skriptů.

Následující stránky **nepředkládají** definice a vysvětlení jazyka Pascal, to je za hranici této nápovědy. Naopak jsou zde popisy odlišností [Pascal Scriptu](#) od Pascalu (nebo ObjectPascalu), seznam podporovaných datových typů a seznam s vysvětlením všech funkcí a procedur pro komunikaci s programem Programmer's Wizard 2. Znalost Pascalu před čtením následujících stránek je předpokládána.

Skriptování v programu Programmer's Wizard 2 umožňuje využívání skriptů, které za Vás budou provádět mnoho opakovaných operací. Co je lepší než nejlepší dialogová okna pro nastavení chování programu? Skript, resp. Váš vlastní skript. Skript, který provede sérii operací a tím ušetří Váš čas, skript, který se může navázat na klíčové události programu (jako je spouštění, načítání [modulů](#) nebo ukončování) a přizpůsobit je přesně Vaším potřebám.

Standardní instalace programu Programmer's Wizard 2 instaluje i modul [pwscript.dll](#), který obsahuje nástroj [Editor skriptů](#). Tento editor s barvením syntaxe a funkcemi *Completion Proposal* a *AutoComplete* pomůže s psaním skriptů.

Odkazy:



[Skript](#)



[Pascal Script](#)





Pascal Script

Programmer's Wizard 2 má podporu skriptování v jazyce Pascal; Pascal proto, že je nejznámijším jazykem, se kterým má zkušenosti nejširší vrstva IT veřejnosti a to hlavně díky jeho moderní inkarnaci ObjectPascal, kterou nabízí Borland (Inprise) Delphi. Tvorba skriptu je díky Pascal Scriptu rychlá a přehledná.

Tato verze programu nemá ještě kompletní podporu všech prvků Pascalu, jsou zde zatím jen ty, které jsou nutné pro potřeby skriptování v programu. Další prvky budou podle potřeby přidávány později.

Odkazy:



[Skriptování](#)



[Datové typy](#)



[Cykly a příkazy](#)



[Matematické operace](#)



[Knihovna funkcí a procedur](#)



[Příklady](#)





Datové typy

Pascal Script podporuje tyto datové typy:

Datový typ	Popis a rozsah
Byte	byte, neznaménkované, rozsah 0..255
ShortInt	1 byte, znaménkované, rozsah -128..127
Word	2b, neznaménkované, rozsah 0..65535
Integer	2b, znaménkované, -32768..32767
LongInt	4b, znaménkované, -2147483648..2147483647
Char	1 byte, neznaménkované, rozsah ASCII znak 0..255
String	dynamický řetězec složený z prvků Char
Real	8b, znaménkované, plovoucí čárka, $5.0 \times 10^{-324} .. 1.7 \times 10^{308}$
Single	4b, znaménkované, plovoucí čárka, $1.5 \times 10^{-45} .. 3.4 \times 10^{38}$
Double	8b, znaménkované, plovoucí čárka, $5.0 \times 10^{-324} .. 1.7 \times 10^{308}$
Extended	10b, znaménkované, plovoucí čárka, $3.6 \times 10^{-4951} .. 1.1 \times 10^{4932}$
Comp	8b, znaménkované, $-2^{63}+1 .. 2^{63}-1$
Boolean	FALSE(0)..TRUE(1)
Array	dynamické pole z prvků stejného typu

Některé z těchto datových typů mají svá specifika, více informací naleznete v odkazech.

Odkazy:

[Skriptování](#)

[Pascal Script](#)

[Matematické operace](#)

[Cykly a příkazy](#)

[Knihovna funkcí a procedur](#)

[Příklady](#)

Specifika datových typů:

[Specifika datového typu Array](#)

[Specifika datového typu String](#)



Skript

Zdrojový text Pascal Scriptu není před spuštěním překládán; místo toho je interpretován programem Programmer's Wizard 2 řádek po řádku přímo za běhu.



Specifika datového typu Array (Pascal Script)

Typ Array je polem prvků stejného typu, pole může být jen jedno-rozměrné a je dynamické, tj. může obsahovat libovolný počet prvků. Velikost pole se mění za běhu skriptu voláním speciální funkce. **První prvek v poli má index [0]**. Deklarace dynamického pole může být taková:

```
var MyArray: Array of Char; //vytvoří pole prvků Char, počáteční velikost je 0
```

Dále lze ve skriptu nastavit velikost pole voláním funkce SetArrayLength:

```
//deklarace:  
//procedure SetArrayLength(var Arr: Array; NewSize: LongInt);  
SetArraySize(MyArray, 10); //nastaví velikost pole na 10 prvků
```

Zjištění velikosti pole je možné obdobnou funkcí GetArrayLength:

```
//deklarace:  
//function GetArrayLength(var Arr: Array): LongInt;  
ArraySize := GetArraySize(MyArray);
```

Odkazovat na jednotlivé prvky pole se lze standardní:

```
//prohodit obsah prvního a druhého prvku pole  
mTempChar := MyArray[0];  
MyArray[0] := MyArray[1];  
MyArray[1] := mTempChar;
```

Další odlišností pole Pascal Scriptu je jeho použití jako parametru funkcí a procedur. Pokud má být pole použito jako parametr, musí to být **vždy** parametr typu *var* a nesmí být v parametru deklarován jeho datový typ, tedy místo kompletní deklarace "*Array of String*" musí být deklarováno pole "*Array*".

```
//chybné deklarace  
function SumBooleanArray(ABooleanArray: Array of Boolean): Boolean;  
function SumBooleanArray(ABooleanArray: Array): Boolean;  
function SumBooleanArray(var ABooleanArray: Array of Boolean): Boolean;  
  
//správná deklarace  
function SumBooleanArray(var ABooleanArray: Array): Boolean;
```

Odkazy:

- [Skriptování](#)
- [Pascal Script](#)
- [Datové typy](#)
- [Cykly a příkazy](#)
- [Matematické operace](#)
- [Specifika datového typu String](#)
- [Knihovna funkcí a procedur](#)
- [Příklady](#)





Specifika datového typu String (Pascal Script)









Datový typ String je v Pascal Scriptu dynamickým a nelze nastavit jeho délku při deklaraci. Nelze přistupovat k jednotlivým prvkům řetězce pomocí indexování, tento příkaz nebude pracovat:

```
MyString[1] := 'A'; //<-- chyba zde  
MyChar := MyString[1]; //<-- a zde další
```

Místo toho je nutné použít zápis využívající funkce StrGet a StrSet:

```
StrSet('A', 1, MyString);  
MyChar := StrGet(MyString, 1);
```

Odkazy:

-  [Skriptování](#)
-  [Pascal Script](#)
-  [Datové typy](#)
-  [Cykly a příkazy](#)
-  [Matematické operace](#)
-  [Specifika datového typu Array](#)
-  [Knihovna funkcí a procedur](#)
-  [Příklady](#)





Podporované příkazy a cykly (Pascal Script)

PascalScript podporuje v současné verzi tyto příkazy a cykly (příkazy v hranatých závorkách mohou, ale nemusí být uvedeny):

```
//zakladni rozvinute vetveni
if BooleanValue then [Begin]
    Statement
[end] else [Begin]
    Statement;
[end;]

//slozeny prikaz vetveni
case OrdinalVariable of
    OrdinalValue1:
        [Begin]
            Statement;
        [end;]
    OrdinalValue2:
        [Begin]
            Statement;
        [end;]
    ...
else
        [Begin]
            Statement;
        [end;]
end;

//cyklus FOR
for OrdinalVariable := OrdinalInitialValue {to/downto} OrdinalValue do
[Begin]
    Statement;
[end;]


//cyklus WHILE
while BooleanValue do
[Begin]
    Statement;
[end;]


//cyklus REPEAT
REPEAT
    Statement;
until BooleanValue;
```

Syntaxe těchto příkazů ("if x then / else", "case x of.." "for x := y to/downto z do", "while x do" a "REPEAT until x") je zcela shodná se syntaxí jazyka Pascal, nejsou zde žádné změny ani omezení.

Odkazy:

- [Skriptování](#)
- [Pascal Script](#)
- [Datové typy](#)
- [Matematické operace](#)

 Knihovna funkcí a procedur

 Pøíklady





Matematické operace (Pascal Script)

Pascal Script podporuje v současné verzi tyto matematické operátory:

+ - * / ..standardní operátory ordinálních typů (op. "+" určen i pro řetězce (String))
div mod ..standardní operátory dělení
= > < <> >= <= ..porovnávací operátory (výsledkem je vždy Boolean)
and or xor not ..binární operátory

Tyto operátory se chovají stejně jako v Pascalu.







Pozor: Pascal Script vyžaduje u některých operací zavírání proměnných do závorek. Zde je seznam těchto chybných operací a způsob nápravy:

Chybní: **Řešení:**
-OrdinalVariable *-(OrdinalValue)*
not Variable *not (Variable)*

Pokud bude vynechána tato závorková konvence, bude operátor "-" nebo "not" ignorován!

Toto je problém Pascal Scriptu a bude brzy odstraněn. Zde je příklad skriptu, který obchází tuto chybu: [zde](#).

Odkazy:

-  [Příklad závorkové konvence "-" a "not"](#)
-  [Skriptování](#)
-  [Datové typy](#)
-  [Cykly a příkazy](#)
-  [Knihovna funkcí a procedur](#)
-  [Příklady](#)



Chyba operátorů "-" a "not" (více..) (seznam příkladů)

(zkopírovat do Editoru skriptů)

Tento skript demonstruje závorkovou konvenci.

{button Zkopířovat,CT()} {button Vytisknout,Print()}

```
{*****}
{
{   Programmer's Wizard 2 Pascal Script   }
{   (popis jazyka v souboru SCRIPT.DOC)   }
{
{*****}
uses ScriptUtils;

var mBooleanVariable: Boolean;
Begin
    mBooleanVariable := false;
    ShowMessage('mBooleanVariable nastaveno na FALSE.');
```

mBooleanVariable := not mBooleanVariable; //poruseni zavorkove konvence

```
    if mBooleanVariable then Begin
        ShowMessage('Nebyl zjiřtěn žádný problém.');
```

Halt;

```
    end else
        ShowMessage('Po provedení mBooleanVariable := not mBooleanVariable je ' +
            'hodnota stále FALSE.');
```

mBooleanVariable := not (mBooleanVariable); //oprava zavorkove konvence

```
    if mBooleanVariable then
        ShowMessage('Po provedení mBooleanVariable := not (mBooleanVariable) je '
+
            'jiř vše v pořádku, hodnota je TRUE.')
```

else //???

```
        ShowMessage('..ale stále pracuje funkce ShowMessage :-)');
```

end.

Minimální skript ([seznam příkladů](#))

(zkopírovat do [Editoru skriptů](#))

Toto je minimální skript, který obsahuje jen nutné části - fragment *Uses* a proceduru *Main*.

{button Zkopířovat,CT()} {button Vytisknout,Print()}

```
{*****}
{
{      Programmer's Wizard 2 Pascal Script      }
{      (popis jazyka v souboru SCRIPT.DOC)      }
{
{*****}
```

```
uses ScriptUtils;
```

```
Begin
```

```
end.
```

Základní skript (seznam příkladů)

(zkopírovat do [Editoru skriptů](#))

Tento skript je generován Editorem skriptů při vytvoření nového skriptu. Zjišťuje nainstalovanou verzi interpreta (Programmer's Wizard 2) a pokud se verze liší od požadované, zobrazí upozornění.

```
{button Zkopířovat,CT()} {button Vytisknout,Print()}
```

```
{*****}
{
  Programmer's Wizard 2 Pascal Script
  (popis jazyka v souboru SCRIPT.DOC)
}
{*****}
uses ScriptUtils;

function GetVerStr: string;
var
  mMajor, mMinor, mRel: Integer;
Begin
  GetVersionEx(mMajor, mMinor, mRel);
  result := 'v' + IntToStr(mMajor) + '.' + IntToStr(mMinor) + '.' +
    IntToStr(mRel);
end;

function GetValidVerStr: string;
Begin
  result := 'v2.0.1'; //požadovana verze PW
end;

Begin
  if GetValidVerStr <> GetVerStr then //nespravna verze?
    ShowMessage('Upozornění: tento skript byl napsán pro Programmers Wizard '
+
      GetValidVerStr + '. Momentálně spuštěná verze: ' + GetVerStr);

  //zde vložte vas skript
end.
```




Přklady (Pascal Script)


Toto je seznam ukázkových skriptů, které jsou obsaženy v této nápovědi. Kliknutím na některý z nich se zobrazí jeho zdrojový text. Vkládat skripty do Editoru skriptů můžete stiskem tlačítka Kopírovat a následným vložením do programu pomocí funkce Vložit (nabídka nebo panel Úpravy).


 [Minimální skript](#)


 [Základní skript](#)


 [Chyba operátorů operátorů "-" a "not"](#)

Odkazy:

 [Skriptování](#)

 [Datové typy](#)

 [Cykly a příkazy](#)

 [Matematické operace](#)

 [Knihovna funkcí a procedur](#)






















Knihovna funkcí a procedur (Pascal Script)

Zde je seznam všech funkcí a procedur, které lze používat ve skriptovacím jazyce Pascal Script programu Programmer's Wizard 2. Funkce jsou seřazeny do kategorií, uvnitř kategorií abecedně. V některých kategoriích je i odkaz "Termíny použité v této kategorii"; stránka, na kterou tento link odkazuje, obsahuje důležité informace o použitých pojmech a neměla by být před použitím funkcí příslušné kategorie v žádném případě vynechána.

Seznam kategorií:

-  [Obecné](#) (vše, co není jinde, je zde)
-  [Přímá komunikace s uživatelem](#)
-  [Stavový řádek](#)
-  [Projekt](#)
-  [Plochy](#)
-  [Okna](#)
-  [Zásuvné moduly](#)
-  [Dokumenty](#)
-  [FS \(soubory a složky\)](#)
-  [Utility a systémové operace](#)
-  [Registr](#)
-  [Nabídky](#)

Odkazy:

-  [Skriptování](#)
-  [Datové typy](#)
-  [Cykly a příkazy](#)
-  [Matematické operace](#)
-  [Příklady](#)





Knihovna funkcí a procedur (Pascal Script)

(zpit do obsahu knihovny tudy)

Kategorie Obecné

- EnumFiles
- GetCurrentScript
- GetPluginsDir
- GetProjectsDir
- GetScriptsDir
- GetVersionEx
- IsSafeScriptingEnabled
- OpenFile
- OpenFileEx
- SendMail
- SendMailEx
- Sleep

Odkazy:

- Skriptování
- Datové typy
- Cykly a příkazy
- Matematické operace
- Příklady


















Knihovna funkcí a procedur (Pascal Script)

(zpět do obsahu knihovny [tady](#))

Kategorie **Přímá komunikace s uživatelem**

-  [AskUser](#)
-  [InputBox](#)
-  [InputQuery](#)
-  [SelectFiles](#)
-  [SelectDirectory](#)
-  [ShowCheckListDialog](#)
-  [ShowListDialog](#)
-  [ShowMessage](#)

Odkazy:

-  [Skriptování](#)
-  [Datové typy](#)
-  [Cykly a příkazy](#)
-  [Matematické operace](#)
-  [Příklady](#)





Knihovna funkcí a procedur (Pascal Script)

(zpit do obsahu knihovny [tudy](#))

Kategorie Stavový oádek

[CancelStatusBarMessage](#)

[GetProgressBarStatus](#)

[ResetProgressBar](#)

[SetProgressBarStatus](#)

[SetStatusBarMessage](#)

Odkazy:

[Skriptování](#)

[Datové typy](#)

[Cykly a pøíkazy](#)

[Matematické operace](#)

[Pøíklady](#)








Knihovna funkcí a procedur (Pascal Script)


(zpit do obsahu knihovny tudy)


 Termíny použité v této kategorii


 **Kategorie Projekt**

 CloseProject


 CreateProject


 EnumProjectFiles


 GetCurrentProject

 GetProjectCount

 GetProjectInfo


 IsProjectOpened


 OpenProject


 SetProjectFiles


Odkazy:

 Skriptování

 Datové typy

 Cykly a příkazy

 Matematické operace

 Příklady





Knihovna funkcí a procedur (Pascal Script)

(zpět do obsahu knihovny [tudy](#))

[Termíny použité v této kategorii](#)

Kategorie Plochy

[CloseWorkspace](#)

[CreateWorkspace](#)

[CreateWorkspaceEx](#)

[GetCurrentWorkspace](#)

[GetWorkspaceCount](#)

[GetWorkspaceCountEx](#)

[GetWorkspaceFile](#)

[GetWorkspaceFileEx](#)

[GetWorkspaceInfo](#)

[GetWorkspaceInfoEx](#)

[OpenWorkspace](#)

Odkazy:

[Skriptování](#)

[Datové typy](#)

[Cykly a příkazy](#)

[Matematické operace](#)

[Příklady](#)


















Knihovna funkcí a procedur (Pascal Script)

(zpět do obsahu knihovny [tady](#))

Kategorie FS (soubory a složky)

-  [CopyFile](#)
-  [DeleteFile](#)
-  [ExecuteFile](#)
-  [ExtractFileExt](#)
-  [ExtractFileName](#)
-  [ExtractFilePath](#)
-  [FileExists](#)
-  [RenameFile](#)

Odkazy:

-  [Skriptování](#)
-  [Datové typy](#)
-  [Cykly a příkazy](#)
-  [Matematické operace](#)
-  [Příklady](#)










Knihovna funkcí a procedur (Pascal Script)

(zpit do obsahu knihovny [tudy](#))

Kategorie Utility

-  [Copy](#)
-  [Cos](#)
-  [Dec](#)
-  [Delete](#)
-  [GetArrayLength](#)
-  [Halt](#)
-  [Chr](#)
-  [Inc](#)
-  [Insert](#)
-  [IntToStr](#)
-  [Length](#)
-  [LowerCase](#)
-  [Ord](#)
-  [Pi](#)
-  [Pos](#)
-  [Random](#)
-  [Randomize](#)
-  [Round](#)
-  [SetArrayLenght](#)
-  [Sin](#)
-  [StrGet](#)
-  [StrSet](#)
-  [StrToInt](#)
-  [Trunc](#)
-  [UpperCase](#)

Odkazy:

-  [Skriptování](#)
-  [Datové typy](#)
-  [Cykly a příkazy](#)
-  [Matematické operace](#)
-  [Příklady](#)








Knihovna funkcí a procedur (Pascal Script)


(zpit do obsahu knihovny [tudy](#))

Kategorie Registr


 [RegReadInt](#)


 [RegReadString](#)


 [RegWriteInt](#)


 [RegWriteString](#)


Odkazy:

 [Skriptování](#)

 [Datové typy](#)

 [Cykly a příkazy](#)

 [Matematické operace](#)

 [Přklady](#)



Knihovna funkcí a procedur (Pascal Script)

(zpit do obsahu knihovny tudy)







































Kategorie **Nabídky** má odlišnou strukturu od ostatních, nejsou zde detailní popisy funkcí. Všechny funkce v této kategorii jsou určeny pro simulaci kliknutí myšítkem do nabídek programu. Každému jednotlivému příkazu z nabídek odpovídá jedna konkrétní funkce.










































Tyto funkce mají shodnou deklaraci:

```
function Menu_x: Boolean;
```






Návratová hodnota:

Funkce vrací *TRUE* pokud byl příkaz úspěšně simulován, jinak vrací *FALSE*.

-  **Kategorie Nabídky**
-  [Menu_DocsAddCurrFile](#)
-  [Menu_DocsAddFile](#)
-  [Menu_DocsArrange](#)
-  [Menu_DocsClear](#)
-  [Menu_DocsCustomize](#)
-  [Menu_EditCopy](#)
-  [Menu_EditCopyTo](#)
-  [Menu_EditCut](#)
-  [Menu_EditFind](#)
-  [Menu_EditNextBookmark](#)
-  [Menu_EditPaste](#)
-  [Menu_EditPasteFrom](#)
-  [Menu_EditPrevBookmark](#)
-  [Menu_EditRedo](#)
-  [Menu_EditRemoveAllBookmarks](#)
-  [Menu_EditReplace](#)
-  [Menu_EditSelectAll](#)
-  [Menu_EditToggleBookmark](#)
-  [Menu_EditUndo](#)
-  [Menu_FileExit](#)
-  [Menu_FileNew](#)
-  [Menu_FileOpen](#)
-  [Menu_FilePrint](#)
-  [Menu_FileProperties](#)
-  [Menu_FileSave](#)
-  [Menu_FileSaveAll](#)
-  [Menu_FileSaveAs](#)
-  [Menu_FileSendToDesktop](#)
-  [Menu_FileSendToMailRecipient](#)
-  [Menu_FileSettingsAutoBackup](#)
-  [Menu_FileSettingsScripting](#)
-  [Menu_FileSettingsShortcuts](#)
-  [Menu_FileSettingsStartup](#)
-  [Menu_FormatBgColor](#)
-  [Menu_FormatBold](#)
-  [Menu_FormatBullets](#)
-  [Menu_FormatCenterAlign](#)

-  [Menu_FormatDeclindent](#)
-  [Menu_FormatIncindent](#)
-  [Menu_FormatItalic](#)
-  [Menu_FormatLeftAlign](#)
-  [Menu_FormatNoAlign](#)
-  [Menu_FormatNumBullets](#)
-  [Menu_FormatRightAlign](#)
-  [Menu_FormatTextColor](#)
-  [Menu_FormatUnderline](#)
-  [Menu_HelpAbout](#)
-  [Menu_HelpRegister](#)
-  [Menu_HelpTip](#)
-  [Menu_HelpWWW](#)
-  [Menu_ProjectClose](#)
-  [Menu_ProjectDeleteWorkspace](#)
-  [Menu_ProjectNew](#)
-  [Menu_ProjectNewWorkspace](#)
-  [Menu_ProjectOpen](#)
-  [Menu_ProjectOpenWorkspace](#)
-  [Menu_ProjectSave](#)
-  [Menu_ProjectSaveWorkspace](#)
-  [Menu_ProjectSetup](#)
-  [Menu_ScriptExecute](#)
-  [Menu_ScriptExecuteProc](#)
-  [Menu_ScriptOpenExecute](#)
-  [Menu_ScriptOpenExecuteProc](#)
-  [Menu_ScriptNew](#)
-  [Menu_ScriptOpen](#)
-  [Menu_ToolsSetup](#)
-  [Menu_ViewDesktopOptions](#)
-  [Menu_ViewFullscreen](#)
-  [Menu_ViewStatusBar](#)
-  [Menu_WindowArrange](#)
-  [Menu_WindowArrangeIcons](#)
-  [Menu_WindowArrangeV](#)
-  [Menu_WindowCascade](#)
-  [Menu_WindowClose](#)
-  [Menu_WindowCloseAll](#)
-  [Menu_WindowMinimize](#)
-  [Menu_WindowNext](#)
-  [Menu_WindowPrevious](#)

Odkazy:


-  [Skriptování](#)
-  [Datové typy](#)
-  [Cykly a příkazy](#)
-  [Matematické operace](#)
-  [Příklady](#)





Knihovna funkcí - prohlížení

 *Kategorie: Obecné*

 *Téma: OpenFile*

Deklarace:

function OpenFile(FileName: String): Boolean

Parametry:

FileName: String

Název dokumentu k otevření

Popis:

Funkce *OpenFile* otevírá dokument udaný parametrem *FileName* v programu Programmer's Wizard 2. Dokument se otevře ve výchozím nástroji tak, jako by byl otevřen dialogem Otevřít a bude přidán do složky Historie. Pokud je dokument již otevřený, přepne se jeho okno do popředí. Tato funkce vždy otevře soubor v programu Programmer's Wizard 2, pro spuštění programu nebo otevření dokumentu jeho asociovaným program je vhodnější funkce ExecuteFile.

Návratový výraz:


Funkce vrací *TRUE* pokud byl soubor úspěšně otevřen, jinak vrací *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: Přímá komunikace s uživatelem*

 *Téma: AskUser*

Deklarace:

function AskUser(Question: String): Boolean

Parametry:

Question: String

Dotaz na uživatele

Popis:

Funkce *AskUser* zobrazuje dotazovací dialog. Parametr *Question* musí obsahovat otázku, které má být zobrazena uživateli, uživatel pak může vybrat odpověď *Ano* nebo *Ne*. Odpověď uživatele vrací funkce jako typ Boolean.

Návratový výraz:


Funkce vrací *TRUE* pokud bylo stisknuto tlačítko *Ano*, jinak vrací *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: Přímá komunikace s uživatelem*

 *Téma: InputDialog*

Deklarace:

function InputBox(Caption: String; Prompt: String; Default: String): String

Parametry:

Caption: String

Titulek okna

Prompt: String

Text uvnitř okna

Default: String

Text v editačním poli po zobrazení dialogu

Popis:

Funkce *InputDialog* zobrazuje dialog s dvěma tlačítky (*OK* a *Storno*) a jedním editačním polem, do kterého lze zadávat text. Okno bude mít nadpis definovaný parametrem *Caption*, popis editačního pole bude nastaven na parametr *Prompt*. Editace pole bude obsahovat text udaný parametrem *Default*. Funkce vrací obsah editačního pole po uzavření okna.

Návratový výraz:


Funkce vrací vždy obsah editačního pole, nezávisle na tom, zda uživatel stiskl *OK* nebo *Storno*. Pro případy kde je nutné znát i způsob, jakým bylo okno uzavřeno (jestli tlačítkem *OK* nebo *Storno*), je vhodnější funkce [InputQuery](#).





Knihovna funkcí - prohlížení

 *Kategorie: Přímá komunikace s uživatelem*

 *Téma: InputQuery*

Deklarace:

function InputQuery(Caption: String; Prompt: String; var Default: String): Boolean

Parametry:

Caption: String

Titulek okna

Prompt: String

Text uvnitř okna

var Default: String

Text v editačním poli po zobrazení dialogu

Popis:

Funkce *InputQuery* zobrazuje dialog s dvěma tlačítky (*OK* a *Storno*) a jedním editačním polem, do kterého lze zadávat text. Okno bude mít nadpis definovaný parametrem *Caption*, popis editačního pole bude nastaven na parametr *Prompt*. Editační pole bude obsahovat text udaný parametrem *Default*. Funkce vrací obsah editačního pole po uzavření okna v parametru *Default*.

Návratový výraz:


Funkce vrací *TRUE* pokud bylo okno zavřeno stiskem tlačítka *OK*, při stisku *Storno* vrací *FALSE*. Obsah editačního pole po zavření okna je vrácen do parametru *Default*. Pro případy kde není nutné znát způsob, jakým bylo okno uzavřeno (jestli tlačítkem *OK* nebo *Storno*), je vhodnější funkce [InputBox](#).





Knihovna funkcí - prohlížení

 *Kategorie: Přímá komunikace s uživatelem*

 *Téma: ShowListDialog*

Deklarace:

function ShowListDialog(Caption, Prompt: String; var Lines: Array of string): LongInt

Parametry:

Caption: String

Titulek okna

Prompt: String

Text uvnitř okna

var Lines: Array of string

Seznam řetězců, které se mají zobrazit jako jednotlivé položky

Popis:

Funkce *ShowListDialog* umožňuje uživateli vybrat si ze seznamu řetězcových výrazů. Funkce zobrazuje okno s nadpisem udaným parametrem *Caption*. Okno obsahuje tlačítka *OK* a *Storno*, také obsahuje pole se seznamem řetězců, ze kterých může uživatel jeden vybrat a stisknout tlačítko *OK*. Obsah seznamu je definován parametrem *Lines*. Funkce vrací index vybraného řetězce.

Návratový výraz:


Funkce vrací index vybraného řetězce. Tento index lze přímo použít do pole *Lines* takto: "Lines[result]" kde *result* je hodnota vrácená funkcí *ShowListDialog*. Když uživatel zruší výběr prvku tlačítkem *Storno*, funkce vrací hodnotu -1. Skript musí vždy ověřit, zda není výsledná hodnota -1 protože bez tohoto testu by mohla být provedena tato instrukce: "Lines[-1]". Instrukce je neplatná protože první prvek z pole má index 0 - skript by byl okamžitě ukončen s chybovým hlášením.





Knihovna funkcí - prohlížení

 *Kategorie: Přímá komunikace s uživatelem*

 *Téma: ShowMessage*

Deklarace:

procedure ShowMessage(Message: String)

Parametry:

Message: String

Informace pro uživatele

Popis:

Funkce *ShowMessage* zobrazuje informační okno s titulkem *Programmer's Wizard 2* a textem uvnitř okna definovaným parametrem *Message*. Okno má jediné tlačítko; tlačítko *OK*. Po jeho stisku se okno uzavře a skript může pokračovat.


Návratový výraz:

Nevrací žádný výraz.





Knihovna funkcí - prohlížení

 *Kategorie:* Stavový řádek

 *Téma:* CancelStatusBarMessage

Deklarace:

procedure CancelStatusBarMessage

Popis:

Funkce *CancelStatusBarMessage* ruší zprávu na stavovém řádku vytvořenou funkcí SetStatusBarMessage. Skript by měl zavolat tuto funkci vždy před ukončením, pokud tyto funkce používal a nemá jiný důvod pro zanechání své zprávy na stavovém řádku.

Poznámka:

Jako bezpečnostní opatření proti skriptům které po sobě neodstraní zprávu na stavovém řádku je zpráva vytvořená pomocí funkce SetStatusBarMessage omezená časem 10 vteřin. Po uplynutí této doby bude zpráva odstraněna automaticky.


Návratový výraz:


Nevrací žádný výraz.





Knihovna funkcí - prohlížení

 *Kategorie: Stavový řádek*

 *Téma: GetProgressBarStatus*

Deklarace:

procedure GetProgressBarStatus(var Position: LongInt; var Maximum: LongInt)

Parametry:

var Position: LongInt

Aktuální hodnota ukazatele průběhu

var Maximum: LongInt

Maximální hodnota ukazatele průběhu

Popis:

Funkce *GetProgressBarStatus* vrací aktuální stav ukazatele průběhu. Tento ukazatel se nachází na stavovém řádku a je sdílen jak skripty, tak nástroji. Ukazatel lze nastavit obdobnou funkcí SetProgressBarStatus.

Návratový výraz:

Funkce vrací aktuální a maximální hodnotu ukazatele průběhu pomocí parametrů *Position* a *Maximum*. Pokud se obě hodnoty rovnají nule (0), není ukazatel zobrazen.


Návratový výraz:


Nevrací žádný výraz.





Knihovna funkcí - prohlížení

 *Kategorie:* Stavový oádek

 *Téma:* ResetProgressBar

Deklarace:

procedure ResetProgressBar

Popis:

Funkce *ResetProgressBar* nastavuje stavový oádek zpít do klidového stavu. Ukazatel prùbìhu bude skryt a jeho hodnoty *Position* a *Maximum* budou nastaveny na nulu (0). Skript by mìl zavolat tuto funkci pøed svým ukonèením vždy pøed svým ukonèením, pokud ukazatel používal, jinak zůstane ukazatel zobrazen i po ukonèení skriptu.

Návratový výraz:

Nevrací žádný výraz.

Návratový výraz:

Nevrací žádný výraz.


Poznámka:

Jako bezpečnostní opatøení proti skriptùm, které po sobì neodstraní ukazatel prùbìhu, je ukazatel vytvoøený pomocí funkce *SetProgressBarStatus* omezen èasem 8 vteøin. Pokud se stav ukazatele prùbìhu za tuto dobu nezminí (nezáleží jak; technicky by však mìlo jít o pøiblížení hodnoty *Position* parametru *Maximum*), bude ukazatel odstranìn automaticky.





Knihovna funkcí - prohlížení

 *Kategorie:* Stavový řádek

 *Téma:* SetStatusBarMessage

Deklarace:

procedure SetStatusBarMessage(Message: String)

Parametry:

Message: String

Nová zpráva pro zobrazení na stavovém řádku

Popis:

Funkce SetStatusBarMessage nastavuje zprávu na stavovém řádku. Všechny druhotné panely na stavovém řádku budou ukryty a celý jeho prostor bude vyhrazen zprávě skriptu. Ta bude vykreslena tučným písmem tak, aby byla kontrastní k ostatní části okna. Zpráva bude zobrazena až do zavolání funkce CancelStatusBarMessage.

Návratový výraz:

Nevrací žádný výraz.


Poznámka:

Jako bezpečnostní opatření proti skriptům které po sobě neodstraní zprávu na stavovém řádku je zpráva vytvořená pomocí funkce SetStatusBarMessage omezená časem 10 vteřin. Po uplynutí této doby bude zpráva odstraněna automaticky.





Knihovna funkcí - prohlížení

 *Kategorie:* Stavový řádek

 *Téma:* SetProgressBarStatus

Deklarace:

procedure SetProgressBarStatus(Position, Maximum: LongInt)

Parametry:

Position: LongInt

Aktuální hodnota ukazatele průběhu

Maximum: LongInt

Maximální hodnota ukazatele průběhu

Popis:

Funkce *SetProgressBarStatus* nastavuje sdílený ukazatel průběhu. Tento ukazatel se zobrazuje na stavovém řádku. Vykreslován je jako standardní ukazatel Windows. Parametry *Position* a *Maximum* jsou interpretovány jako "*Position* hotovo z *Maximum* celkem", tedy skript by měl nastavit hodnotu *Position* na nulu (0) při začátku dlouhé operace a postupně ji zvyšovat tak, aby při dokončení byla rovna hodnotě *Maximum*. Tento ukazatel je nutné po dokončení operaci odstranit voláním funkce ResetProgressBar. Skript by měl tento ukazatel používat při provádění časově náročných operací.

Návratový výraz:

Nevrací žádný výraz.

Poznámka:


Jako bezpečnostní opatření proti skriptům, které po sobě neodstraní ukazatel průběhu, je ukazatel vytvořený pomocí funkce *SetProgressBarStatus* omezen časem 8 vteřin. Pokud se stav ukazatele průběhu za tuto dobu nezminí (nezáleží jak; technicky by však mělo jít o přiblížení hodnoty *Position* parametru *Maximum*), bude ukazatel odstraněn automaticky.





Knihovna funkcí - prohlížení

 *Kategorie: Projekt*

 *Téma: CloseProject*

Deklarace:

function CloseProject: Boolean

Popis:

Funkce *CloseProject* uzavírá projekt, pokud je nějaký otevřen. Zavření projektu způsobí zavření plochy a tím i všech otevřených oken. Uživateli bude dána možnost uložit změny, pokud nějaké existují. Když uživatel neuloží svou práci a místo toho stiskne tlačítko *Storno*, nebude nikteré z oken uzavřeno, neuzavře se plocha a ani projekt; v takovém případě funkce vrací *FALSE*. Po provedení této funkce s výsledkem *TRUE* lze předpokládat, že je otevřena základní plocha programu a není otevřen jakýkoliv dokument.

Návratový výraz:


Funkce vrací *TRUE* pokud byl úspěšně zavřen projekt, jeho plocha s ní i všechna otevřená okna. Funkce vrací *TRUE* i tehdy, pokud nebyl žádný projekt otevřen; v takovém případě jednoduše uzavře všechna okna a plochu.





Knihovna funkcí - prohlížení

 *Kategorie: Projekt*

 *Téma: GetCurrentProject*

Deklarace:

function GetCurrentProject: LongInt

Popis:

Funkce *GetCurrentProject* vrací ID právi otevřeného projektu. Toto ID může být použito v dalších funkcích této kategorie. ID je garantováno jako hodnota rozsahu $0 \leq ID < \text{GetProjectCount}$ a je indexem do vnitřní tabulky projektů programu. Skript by měl před voláním této funkce vždy napřed zjistit, zda je nějaký projekt otevřen a to funkcí *IsProjectOpened*.

Návratový výraz:

Funkce vrací vždy ID právi otevřeného projektu. Pokud není žádný projekt otevřen, vrací funkce hodnotu *-1*, což je pro všechny další funkce neplatné ID.





Knihovna funkcí - prohlížení

 *Kategorie: Projekt*

 *Téma: GetProjectCount*

Deklarace:

function GetProjectCount: LongInt

Popis:

Funkce *GetProjectCount* vrací počet existujících projektů. Tuto informaci lze použít pro procházení projektů, protože ID projektu je garantováno jako hodnota rozsahu $0 \leq ID < GetProjectCount$ a je indexem do vnitřní tabulky projektů programu.

Návratový výraz:


Funkce vrací vždy počet existujících projektů.





Knihovna funkcí - prohlížení

 *Kategorie: Projekt*

 *Téma: GetProjectCount*

Deklarace:

function GetProjectInfo(ProjectID: LongInt; var FileName, Name, Description: string): Boolean;

Parametry:

ProjectID: LongInt

ID projektu

var FileName: string

Do tohoto parametru je zkopírován název souboru s projektem

var Name: string

Do tohoto parametru je zkopírován název projektu

var Description: string

Do tohoto parametru je zkopírován popis projektu, pokud byl nijaký zadán

Popis:

Funkce *GetProjectInfo* se používá ke zjištění informací o projektu. Projekt je identifikován parametrem *ProjectID*, informace o projektu jsou zkopírovány do parametrů *FileName*, *Name* a *Description*. *ProjectID* musí být v rozsahu $0 \leq ProjectID < \underline{GetProjectCount}$.

Návratový výraz:


Funkce vrací *TRUE* pokud byly informace zkopírovány do parametrů, jinak vrací *FALSE*. Funkce většinou vrátí *FALSE* jako následek nesprávného parametru *ProjectID*.





Knihovna funkcí - prohlížení

 *Kategorie: Projekt*

 *Téma: IsProjectOpened*

Deklarace:

function IsProjectOpened: Boolean

Popis:

Funkci *IsProjectOpened* lze použít pro získání informace, zda je právě otevřen nějaký projekt. ID otevřeného projektu lze zjistit voláním funkce *GetCurrentProject*. Skript by před voláním *GetCurrentProject* vždy měl zavolat napřed tuto funkci a ujistit se, že je nějaký projekt otevřen.

Návratový výraz:


Funkce vrací *TRUE* pokud je v době jejího volání otevřen nějaký projekt, jinak vrací *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: Projekt*

 *Téma: OpenProject*

Deklarace:

function OpenProject(ProjectID: LongInt): Boolean

Parametry:

ProjectID: LongInt

ID projektu, který má být otevřen

Popis:

Funkce *OpenProject* otevírá projekt s ID udaným parametrem *ProjectID*. Pokud je již nějaký projekt otevřen, funkce se ho napřed pokusí zavřít, může také zobrazit dialog s upozorněním pro uživatele na změny v dokumentech, které se zavírají a které je nutné uložit, funkce prostě udělá vše co funkce *CloseProject* a teprve pak projekt otevře.

Návratový výraz:


Funkce vrací *TRUE* pokud byl nový projekt úspěšně otevřen, jinak vrací *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: Plochy*

 *Téma: CloseWorkspace*

Deklarace:

function CloseWorkspace: Boolean

Popis:

Funkce *CloseWorkspace* uzavírá plochu právě otevřeného projektu, pokud je nějaký otevřen. Zavření plochy způsobí i zavření všech otevřených oken. Uživateli bude dána možnost uložit změny, pokud nějaké existují. Když uživatel neuloží svou práci a místo toho stiskne tlačítko *Storno*, nebude nikteré z oken uzavřeno, neuzavře se plocha; v takovém případě funkce vrátí *FALSE*. Po provedení této funkce s výsledkem *TRUE* lze předpokládat, že není otevřen jakýkoliv dokument.

Návratový výraz:


Funkce vrátí *TRUE* pokud byla úspěšně zavřena plocha a s ní i všechna otevřená okna. Funkce vrátí *TRUE* i tehdy, pokud nebyla žádná plocha otevřena; v takovém případě jednoduše uzavře všechna okna.





Knihovna funkcí - prohlížení

 *Kategorie: Plochy*

 *Téma: GetCurrentWorkspace*

Deklarace:

function GetCurrentWorkspace: LongInt

Popis:

Funkce *GetCurrentWorkspace* vrací ID právní otevřené plochy otevřeného projektu, pokud je nějaký otevřen. Toto ID může být použito v dalších funkcích této kategorie. ID je garantováno jako hodnota rozsahu $0 \leq ID < GetWorkspaceCount$ a je indexem do vnitřní tabulky ploch programu. Skript by měl před voláním této funkce vždy napřed zjistit, zda je nějaký projekt otevřen a to funkcí *IsProjectOpened*.

Návratový výraz:


Funkce vrací vždy ID právní otevřené plochy. Pokud není žádný projekt otevřen (resp. není otevřena žádná plocha), vrací funkce hodnotu *-1*, což je pro všechny další funkce neplatné ID.





Knihovna funkcí - prohlížení

 *Kategorie: Plochy*

 *Téma: GetWorkspaceCount*

Deklarace:

function GetWorkspaceCount: LongInt

Popis:

Funkce *GetWorkspaceCount* zjišťuje počet ploch v právě otevřeném projektu. Vrácenou hodnotu lze použít pro procházení plochami, protože identifikátor plochy *WorkspaceID* je garantován jako hodnota $0 \leq \text{WorkspaceID} < \text{GetWorkspaceCount}$. Skript by měl před voláním této funkce vždy napřed zjistit, zda je nějaký projekt otevřen a to funkcí *IsProjectOpened*.

Návratový výraz:

Funkce vrací vždy počet ploch v otevřeném projektu, pokud není otevřen žádný projekt, vrací hodnotu 0.





Knihovna funkcí - prohlížení

 *Kategorie: Plochy*

 *Téma: GetWorkspaceCountEx*

Deklarace:

function GetWorkspaceCountEx(ProjectID: LongInt): LongInt

Parametry:

ProjectID: LongInt

ID projektu, na který je požadavek směřován

Popis:

Funkce *GetWorkspaceCountEx* pracuje stejně jako funkce *GetWorkspaceCount* s tím rozdílem, že může zjistit počet ploch jakéhokoliv, tedy ne jen právě otevřeného projektu. Více informací lze nalézt v popisu funkce *GetWorkspaceCount*.

Návratový výraz:


Funkce vrací vždy počet ploch v projektu definovaném parametrem *ProjectID*, při neúspěchu vrací hodnotu 0.





Knihovna funkcí - prohlížení

 *Kategorie: Plochy*

 *Téma: GetWorkspaceFile*

Deklarace:

function GetWorkspaceFile(WorkspaceID, FileID: LongInt): String

Parametry:

WorkspaceID: LongInt

ID plochy, na kterou je požadavek směřován

FileID: LongInt

ID dokumentu, jehož plnou cestu má funkce vrátit

Popis:

Funkce *GetWorkspaceFile* zjišťuje plnou cestu *FileID*-tého dokumentu na ploše s ID definovaným parametrem *WorkspaceID* právní otevřeného projektu. Parametr *FileID* musí být hodnota v rozsahu $0 <= FileID < \text{počet dokumentů na ploše}$. Počet dokumentů na ploše lze zjistit voláním funkce *GetWorkspaceInfo*.

Návratový výraz:


Funkce vrací plnou cestu k vyžádanému dokumentu, pokud je parametr *WorkspaceID* nebo *FileID* neplatný, vrací prázdný řetězec.





Knihovna funkcí - prohlížení

 *Kategorie: Plochy*

 *Téma: GetWorkspaceFileEx*

Deklarace:

function GetWorkspaceFileEx(ProjectID, WorkspaceID, FileID: LongInt): String

Parametry:

ProjectID: LongInt

ID projektu, jehož plocha má být definovaná parametrem *WorkspaceID*

WorkspaceID: LongInt

ID plochy, na kterou je požadavek směřován

FileID: LongInt

ID dokumentu, jehož plnou cestu má funkce vrátit

Popis:

Funkce *GetWorkspaceFileEx* zjišťuje plnou cestu *FileID*-tého dokumentu na ploše s ID definovaným parametrem *WorkspaceID* v projektu *ProjectID*. Parametr *FileID* musí být hodnota v rozsahu $0 \leq FileID < \text{počet dokumentů na ploše}$. Počet dokumentů na ploše lze zjistit voláním funkce *GetWorkspaceInfoEx*.

Návratový výraz:


Funkce vrací plnou cestu k vyžádanému dokumentu, pokud je některý z parametrů neplatný, vrací prázdný řetězec.





Knihovna funkcí - prohlížení

 *Kategorie: Plochy*

 *Téma: GetWorkspaceInfo*

Deklarace:

function GetWorkspaceInfo(WorkspaceID: LongInt; var Name, Description: String; var FileCount: LongInt): Boolean

Parametry:

WorkspaceID: LongInt

ID plochy, na kterou je požadavek směřován

var Name: String

Do tohoto parametru je vrácen název plochy

var Description: String

Do tohoto parametru je vrácen název plochy, pokud nějaký existuje

var FileCount: LongInt

Do tohoto parametru je vrácen počet otevřených dokumentů na ploše

Popis:

Funkce *GetWorkspaceInfo* se používá ke zjištění informací o některé z ploch právní otevřeného projektu. Právní otevřenou plochu lze zjistit voláním funkce *GetCurrentWorkspace*. Procházení plochy lze voláním *GetWorkspaceCount* a následným procházením ID ploch podle klíče "*for WorkspaceID := 0 to GetWorkspaceCount - 1 do...*". Funkce vrací název a popis plochy, vrací také počet otevřených dokumentů. Cesty k jednotlivým dokumentům lze zjistit funkcí *GetWorkspaceFile*.

Návratový výraz:


Funkce vrací *TRUE* pokud vše proběhlo v pořádku a údaje byly zkopírovány do parametrů, jinak vrací *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: Plochy*

 *Téma: GetWorkspaceInfoEx*

Deklarace:

```
function GetWorkspaceInfoEx(ProjectID, WorkspaceID: LongInt; var Name, Description: String;  
var FileCount: LongInt): Boolean
```

Parametry:

ProjectID: LongInt

ID projektu, jehož plocha je definována parametrem *WorkspaceID*

WorkspaceID: LongInt

ID plochy, na kterou je požadavek smírován

var Name: String

Do tohoto parametru je vrácen název plochy

var Description: String

Do tohoto parametru je vrácen název plochy, pokud nějaký existuje

var FileCount: LongInt

Do tohoto parametru je vrácen počet otevřených dokumentů na ploše

Popis:

Funkce *GetWorkspaceInfo* se používá ke zjištění informací o některé z ploch projektu definovaného parametrem *ProjectID*. Procházet plochy lze voláním *GetWorkspaceCountEx* a následným procházením ID ploch podle klíče "*for WorkspaceID := 0 to GetWorkspaceCountEx(ProjectID) - 1 do...*". Funkce vrátí název a popis plochy, vrátí také počet otevřených dokumentů. Cesty k jednotlivým dokumentům lze zjistit funkcí *GetWorkspaceFileEx*.

Návratový výraz:


Funkce vrátí *TRUE* pokud vše proběhlo v pořádku a údaje byly zkopírovány do parametrů, jinak vrátí *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: Plochy*

 *Téma: OpenWorkspace*

Deklarace:

function OpenWorkspace(WorkspaceID: LongInt): Boolean

Parametry:

WorkspaceID: LongInt

ID plochy, která se má otevřít

Popis:

Funkce *OpenWorkspace* otvírá plochu právi otevřeného projektu. Plocha je určena parametrem *WorkspaceID*. Při provádění této funkce se napřed uzavře existující plocha a s ní i všechny otevřené dokumenty, uživateli bude dána možnost uložit změny v dokumentech, pokud nějaké existují. Po uzavření všech oken a plochy se otevře plocha *WorkspaceID*.


Návratový výraz:


Funkce vrací *TRUE* pokud byla plocha úspěšně otevřena, jinak vrací *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: FS (soubory a složky)*

 *Téma: DeleteFile*

Deklarace:

function DeleteFile(FileName: String): Boolean

Parametry:

FileName: String

Plná cesta k souboru, který se má smazat

Popis:

Funkce *DeleteFile* maže fyzický soubor udaný parametrem *FileName*.

Návratový výraz:

Funkce vrací *TRUE* pokud byl soubor smazán, jinak vrací *FALSE*.


Poznámky:


Tato funkce je jednou z chráněných funkcí a může být v nastavení skriptování zakázána. Skript by si měl vždy ověřit zda jsou tyto funkce povoleny voláním *IsSafeScriptingEnabled*.





Knihovna funkcí - prohlížení

 *Kategorie: FS (soubory a složky)*

 *Téma: ExecuteFile*

Deklarace:

function ExecuteFile(FileName: String; Silent: Boolean): Boolean

Parametry:

FileName: String

Plná cesta k souboru, který se má spustit

Silent: Boolean

Udává, zda má funkce při neúspěchu jen vrátit *FALSE*, nebo zobrazit upozornění.

Popis:

Funkce *ExecuteFile* spustí program nebo otevře dokument udaný parametrem *FileName*. Pokud se jedná o dokument, bude tento otevřen ve svém asociovaném programu, ne v programu Programmer's Wizard 2 jak by se stalo při použití funkce *OpenFile*. Pokud je parametr *Silent* nastaven na *FALSE* a program nebo dokument se nepodaří spustit, zobrazí funkce i upozornění viditelné uživateli.


Návratový výraz:

Funkce vrací *TRUE* pokud byl program nebo dokument spuštěn, jinak vrací *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: FS (soubory a složky)*

 *Téma: ExtractFileExt*

Deklarace:

```
function ExtractFileExt(FileName: String): String
```

Parametry:

FileName: String

Plná cesta k souboru, jehož koncovka se má zjistit

Popis:

Funkce *ExtractFileExt* získává z názvu souboru udaného parametrem *FileName* jeho koncovku a tu vrací jako řetězec. ("C:\SOUBOR.HTM" bude převedeno na ".HTM")


Návratový výraz:


Funkce vždy vrací koncovku souboru udaného parametrem *FileName*.





Knihovna funkcí - prohlížení

 *Kategorie: FS (soubory a složky)*

 *Téma: ExtractFileName*

Deklarace:

```
function ExtractFileName(FileName: String): String
```

Parametry:

FileName: String

Plná cesta k souboru, jehož jméno se má zjistit

Popis:

Funkce *ExtractFileName* získává z plné cesty souboru udaného parametrem *FileName* jeho název s koncovkou a výsledek vrací jako řetizec. ("C:\SOUBOR.HTM" bude převedeno na "SOUBOR.HTM")


Návratový výraz:


Funkce vždy vrací název souboru udaného parametrem *FileName*.





Knihovna funkcí - prohlížení

 *Kategorie: FS (soubory a složky)*

 *Téma: ExtractFilePath*

Deklarace:

```
function ExtractFilePath(FileName: String): String
```

Parametry:

FileName: String

Plná cesta k souboru, jehož cesta se má zjistit

Popis:

Funkce *ExtractFileName* získává z plné cesty a názvu souboru udaného parametrem *FileName* plnou cestu a výsledek vrací jako řetězec. ("C:\SOUBOR.HTM" bude převedeno na "C:\")


Návratový výraz:


Funkce vždy vrací název souboru udaného parametrem *FileName*.





Knihovna funkcí - prohlížení

 *Kategorie: FS (soubory a složky)*

 *Téma: FileExists*

Deklarace:

function FileExists(FileName: String): Boolean

Parametry:

FileName: String

Plná cesta k souboru, který se má ověřit

Popis:

Funkce *FileExists* zjišťuje, jestli soubor udaný parametrem *FileName* existuje a vrací *TRUE* pro existující soubor, *FALSE* pro neexistující.


Návratový výraz:

Funkce vrací *TRUE* pokud byl soubor udaný parametrem *FileName* existuje, jinak vrací *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: FS (soubory a složky)*

 *Téma: RenameFile*

Deklarace:

function RenameFile(OldName, NewName: String): Boolean

Parametry:

OldName: String

Plná cesta k souboru, který má být přejmenován

NewName: String

Nový název souboru

Popis:

Funkce *RenameFile* přejmenuje fyzický soubor udaný parametrem *OldName* na název *NewName*.

Návratový výraz:

Funkce vrací *TRUE* pokud byl soubor smazán, jinak vrací *FALSE*.

Poznámky:

Tato funkce je jednou z chráněných funkcí a může být v nastavení skriptování zakázána. Skript by si měl vždy ověřit zda jsou tyto funkce povoleny voláním *IsSafeScriptingEnabled*.





Knihovna funkcí - prohlížení

 *Kategorie: Obecné*

 *Téma: IsSafeScriptingEnabled*

Deklarace:

function IsSafeScriptingEnabled: Boolean

Popis:

Funkce *IsSafeScriptingEnabled* zjišťuje stav skriptování a vrátí *TRUE*, pokud skript běží při zapnutém bezpečném skriptování. Bezpečné skriptování lze zapnout a vypnout z dialogu *Nastavení skriptování*, které se zobrazí po vybrání stejnojmenného příkazu z podnabídky *Nastavení* nabídky *Soubor*. Při zapnutém bezpečném skriptování nelze provádět funkce jako jsou *DeleteFile* a *RenameFile*.

Návratový výraz:


Funkce vrátí *TRUE* pokud je bezpečné skriptování povoleno, jinak vrátí *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Copy*

Deklarace:

function Copy(S: String; Index, Length: LongInt): String

Parametry:

S: String

Zdrojový řetězec

Index: LongInt

Index prvního znaku (první v řetězci je index 1)

Length: LongInt

Délka výsledného řetězce

Popis:

Funkce *Copy* kopíruje *Length* znaků z řetězce *S*, prvním znakem je znak s indexem udaným parametrem *Index*. Výsledek vrací jako řetězec.

Návratový výraz:


Funkce vrací *Length* znaků z řetězce *S*, prvním znakem je znak na pozici *Index*.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Cos*

Deklarace:

function Cos(X: Extended): Extended

Parametry:

X: Extended

Parametr funkce Cos

Popis:

Funkce provádí standardní operaci *Cos*, parametrem je *X*. Parametr i výsledek jsou typu *Extended* a jsou chápány v radiánech.

Návratový výraz:


Funkce vždy vrací výsledek operace *Cos X*.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Dec*

Deklarace:

procedure Dec(var X: LongInt);

Parametry:

var X: LongInt

Proměnná, jejíž obsah má být dekrementován.

Popis:

Funkce *Dec* snižuje hodnotu proměnné *X* o jednu a je ekvivalentní zápisu " $X := X - 1$ ".

Návratový výraz:


Nevrací žádný výraz přímo, mění hodnotu parametru *X*.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Delete*

Deklarace:

procedure Delete(var S: String; Index, Length: LongInt);

Parametry:

var S: String

Zdrojový řetězec

Index: LongInt

Index prvního znaku (první v řetězci je index 1)

Length: LongInt

Počet znaků, které se mají odstranit.

Popis:

Funkce *Delete* odstraňuje *Length* znaků z řetězce *S*, index prvního znaku je určen parametrem *Index*. Délka výsledného řetězce se bude rovnat hodnotě výrazu "*Length(S) - Length*".

Návratový výraz:


Nevrací žádný výraz přímo, miní hodnotu parametru *S*.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: GetArrayLength*

Deklarace:

```
function GetArrayLength(var Arr: Array): LongInt;
```

Parametry:

var Arr: Array

Pole, jehož velikost se má zjistit.

Popis:

Funkce *GetArrayLength* zjišťuje délku dynamického *Arr*. Prvky pole lze pak indexovat podle klíče " $0 \leq index < GetArrayLength$ ". Délku pole lze měnit obdobnou funkcí [SetArrayLength](#).

Návratový výraz:


Vrací vždy délku pole, zadaného parametrem *Arr*.





Knihovna funkcí - prohlížení

 *Kategorie: Obecné*

 *Téma: GetVersionEx*

Deklarace:

procedure GetVersionEx(var Major, Minor, Release: LongInt)

Parametry:

var Major: LongInt

var Minor: LongInt

var Release: LongInt

Do těchto parametrů jsou vráceny informace o verzi programu Programmer's Wizard 2.

Popis:

Funkce *GetVersionEx* se používá ke zjištění informací o verzi programu Programmer's Wizard, na kterém je právě skript spuštěn. Parametr *Major* je v této verzi nastaven na 2, *Minor* na 0 a *Release* na 1. Příklad: Programmer's Wizard 3 (v3.1.2) bude vracet hodnoty 3, 1 a 2.

Návratový výraz:


Funkce vždy kopíruje informace o verzi do parametrů *Major*, *Minor* a *Release*.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Halt*

Deklarace:

procedure Halt;

Popis:

Funkce *Halt* provede okamžité ukončení skriptu a předeá řízení zpět programu Programmer's Wizard 2.

Návratový výraz:


Nevrací žádný výraz.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Chr*

Deklarace:

function Chr(B: Byte): Char

Parametry:

B: Byte

Byte, který se má převést na znak

Popis:

Funkce *Chr* převádí byte na znak (typ Char). Konverze je užitečná zvláště pro vkládání některých speciálních znaků do řetězce, konec řádky je například možné vytvořit kombinací *Chr(13)+Chr(10)*. Opačná konverze je možná obdobnou funkcí Ord.

Návratový výraz:


Funkce vždy vrátí výsledek převedení byte *X* na ASCII znak.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Inc*

Deklarace:

```
procedure Inc(var X: LongInt);
```

Parametry:

var X: LongInt

Proměnná, jejíž obsah má být inkrementován.

Popis:

Funkce *Dec* zvyšuje hodnotu proměnné *X* o jednu a je ekvivalentní zápisu " $X := X + 1$ ".

Návratový výraz:


Nevrací žádný výraz přímo, mění hodnotu parametru *X*.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Insert*

Deklarace:

procedure Insert(SubStr: string; var S: String; Index: LongInt)

Parametry:

SubStr S: String

Øetizec, který má být vložen

var S: String

Cílový øetizec

Index: LongInt

Pozice, na kterou se má øetizec *SubStr* vložit

Popis:

Funkce *Insert* vkládá do øetizce *S* øetizec *SubStr* a to na pozici udanou parametrem *Index*. Výsledek operace je zkopírován zpít do parametru *S*. Nová délka øetizce po této operaci bude "*Length(S) + Length(SubStr)*".

Návratový výraz:

Nevrací žádný výraz pøímo, miní hodnotu parametru *S*.





Knihovna funkcí - prohlížení

Kategorie: Utility

Téma: IntToStr

Deklarace:

function IntToStr(X: LongInt): String

Parametry:

X: LongInt

Ordinální výraz, který má být převeden na řetězec.

Popis:

Funkce *IntToStr* převádí parametr *X*, který je ordinální, na řetězec znaků. Příklad: výsledkem volání *IntToStr(123)* bude řetězec "123". Opačná konverze je možná voláním funkce StrToInt.

Návratový výraz:

Funkce vždy vrátí řetězec, který vyjadřuje hodnotu parametru *X*.





Knihovna funkcí - prohlížení

Kategorie: Utility

Téma: Length

Deklarace:

function Length(S: String): LongInt

Parametry:

S: *String*

Øetizec, jehož délka se má zjistit

Popis:

Funkce *Length* zjišØuje aktuální délku dynamického Øetizce. Øetizec pak lze indexovat podle klíèe "*1 <= index <= Length(S)*".

Návratový výraz:


Funkce vrací vždy délku Øetizce S.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: LowerCase*

Deklarace:

function LowerCase(S: String): String

Parametry:

S: *String*

Øetizec, jehož obsah má být pøeveden na malá písmena

Popis:

Funkce *LowerCase* pøevádí øetizec udaný parametrem S tak, aby obsahoval jen malá písmena (pø. øetizec "AbCd" bude pøeveden na "abcd").

Návratový výraz:


Funkce vrací vždy øetizec S pøevedený tak, aby obsahoval jen malé znaky.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Ord*

Deklarace:

function Ord(Ch: Char): Byte

Parametry:

Ch: Char

Znak, který se má převést na byte

Popis:

Funkce *Ord* převádí ASCII znak (typ Char) na typ byte. Konverze je užitečná zvláště pro zjišťování některých speciálních znaků v řetězci. Opačná konverze je možná obdobnou funkcí *Chr*.

Návratový výraz:


Funkce vždy vrátí výsledek převedení ASCII znaku *Ch* na byte.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Pi*

Deklarace:

function Pi: Extended

Návratový výraz:


Vždy vrací 3.1415926535897932385.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Pos*

Deklarace:

function Pos(SubStr, S: String): LongInt

Parametry:

SubStr: String

Hledaný řetězec

S: String

Řetězec, který se má prohledat.

Popis:

Funkce *Pos* hledá v řetězci *S* první výskyt řetězce *SubStr* a po jeho nalezení vrátí index prvního znaku *SubStr* v rámci řetězce *S* (př. "*Pos('BCD', 'ABCDE')*" vrátí hodnotu "2").

Návratový výraz:


Funkce vrátí index prvního znaku řetězce *SubStr* nalezeného v řetězci *S*. Pokud řetězec *S* řetězec *SubStr* neobsahuje, funkce vrátí hodnotu nula (0).





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Randomize*

Deklarace:

procedure Randomize

Popis:

Funkce *Randomize* inicializuje generátor pseudonáhodných čísel. Tato funkce by měla být zavolána na začátku skriptu, který používá funkci Random ale nesmí být používána mezi jednotlivými voláními funkce Random ve smyčce.

Návratový výraz:


Nevrací žádný výraz.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Random*

Deklarace:

function Random(Range: LongInt): LongInt;

Parametry:

Range: LongInt

Rozsah výsledku

Popis:

Funkce *Random* vrací pseudonáhodné číslo v rozsahu $0 \leq \text{výsledek} < \text{Range}$. Před použitím této funkce by měl být generátor pseudonáhodných čísel nastaven voláním funkce *Randomize*.

Návratový výraz:


Vždy vrací pseudonáhodné číslo v rozsahu $0 \leq \text{výsledek} < \text{Range}$.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Round*

Deklarace:

function Round(X: Extended): LongInt

Parametry:

X: Extended

Výraz, který má být zaokrouhlen

Popis:

Funkce *Round* zaokrouhluje hodnotu udanou výrazem *X* na neblíží hodnotu bez desetinných míst a vrací výsledek jako *LongInt*. Pro případy kde je zaokrouhlování nežádoucí je vhodnější funkce Trunc.

Návratový výraz:


Funkce vždy vrací hodnotu výrazu *X* zaokrouhlenou na hodnotu bez desetinných míst.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: SetArrayLength*

Deklarace:

procedure SetArrayLength(var Arr: Array; NewLength: LongInt)

Parametry:

var Arr: Array

Pole, jehož velikost se má zjistit.

Popis:

Funkce *SetArrayLength* nastavuje délku dynamického pole udaného parametru *Arr* na hodnotu parametru *NewLength*. Po provedení této funkce lze pole indexovat podle tohoto klíče: " $0 \leq index < \text{GetArrayLength}(Arr)$ ". Zjistit velikost dynamického pole lze obdobnou funkcí [GetArrayLength](#).

Návratový výraz:


Nevrací žádný výraz.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Sin*

Deklarace:

function Sin(X: Extended): Extended

Parametry:

X: Extended

Parametr funkce Sin

Popis:

Funkce provádí standardní operaci *Sin*, parametrem je *X*. Parametr i výsledek jsou typu *Extended* a jsou chápány v radiánech.

Návratový výraz:

Funkce vždy vrací výsledek operace *Sin X*.





Knihovna funkcí - prohlížení

Kategorie: Utility

Téma: StrGet

Deklarace:

function StrGet(S: String; Index: LongInt): Char;

Parametry:

S: String

Cílový řetězec

Index: LongInt

Index požadovaného znaku

Popis:

Funkce *StrGet* vrací *Index*-tý znak z řetězce *S*. *Index* musí být v rozsahu " $1 \leq S \leq Length(S)$ ". Více informací lze nalézt u popisu specifik datového typu String v Pascal Scriptu. Nastavit individuální znak řetězce lze obdobnou funkcí *StrSet*.

Návratový výraz:


Vrací *Index*-tý znak z řetězce *S*.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: StrSet*

Deklarace:

procedure StrSet(Ch: Char; Index: LongInt; var S: String)

Parametry:

Ch: Char

Znak, na kterou má být prvek řetězce S nastaven

Index: LongInt

Index požadovaného znaku

var S: String

Cílový řetězec

Popis:

Funkce *StrSet* nastavuje *Index*-tý znak z řetězce *S* na znak *Ch*. *Index* musí být v rozsahu " $1 \leq S \leq Length(S)$ ". Více informací lze nalézt u popisu specifik datového typu String v Pascal Scriptu. Opačnou funkcí je StrGet.

Návratový výraz:


Nevrací žádnou hodnotu přímo, miní řetězec udaný parametrem *S*.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: StrToInt*

Deklarace:

function StrToInt(S: String; Default: LongInt): LongInt

Parametry:

S: *String*

Øetizec, který má být pøeveden na ordinální výraz

Default: *LongInt*

Ordinální výraz, který funkce vrátí, když nelze øetizec pøevést

Popis:

Funkce *StrToInt* pøevádí øetizec vyjadøující ordinální hodnotu (jako je øetizec "123") na ordinální datový typ. Opaèná konverze je možná voláním funkce *IntToStr*.

Návratový výraz:


Funkce vrací ordinální hodnotu, vyjádøenou øetizcem S. Pokud øetizec nelze pøevést (tj. pokud obsahuje jiné znaky než èíslice), vrací funkce hodnotu udanou parametrem *Default*.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: Round*

Deklarace:

function Trunc(X: Extended): LongInt

Parametry:

X: Extended

Výraz, který má být zbaven desetinných míst

Popis:

Funkce *Trunc* převádí výraz *X* na typ *LongInt* bez zaokrouhlování, prostým odstraněním desetinné části výrazu. Pro situace, kde je zapotřebí zaokrouhlování před převodem je vhodnější funkce Round.

Návratový výraz:


Funkce vždy vrátí hodnotu výrazu *X* bez desetinných míst.





Knihovna funkcí - prohlížení

 *Kategorie: Utility*

 *Téma: UpperCase*

Deklarace:

function UpperCase(S: String): String

Parametry:

S: String

Øetizec, jehož obsah má být pøeveden na velká písmena

Popis:

Funkce *UpperCase* pøevádí øetizec udaný parametrem *S* tak, aby obsahoval jen velká písmena (pø. øetizec "AbCd" bude pøeveden na "ABCD").

Návratový výraz:

Funkce vrací vždy øetizec *S* pøevedený tak, aby obsahoval jen velké znaky.





Knihovna funkcí - prohlížení

Kategorie: Registr

Téma: RegReadInt

Deklarace:

```
function RegReadInt(Name: String; DefaultValue: LongInt): LongInt
```

Parametry:

Name: String

Øetizec udávající název hodnoty, která se má pøeèíst

DefaultValue: LongInt

Hodnota, která má být vrácena v pøípadì, že hodnota s názvem *Name* neexistuje.

Popis:

Funkce *RegReadInt* ète z registru døíve uloženou pojmenovanou hodnotu. Takovou hodnotu lze zapsat voláním funkce *RegWriteInt*, hodnota pak zůstává v registru uložena pro pozdijší použití pøi pøíštím spuštíní skriptu. Všechny skripty sdílejí stejnou složku v registru.

Návratový výraz:

Funkce ète hodnotu pojmenovanou *Name* a vrací ji jako typ *LongInt*. Pokud hodnota neexistuje, vrací hodnotu *DefaultValue*.

Poznámky:

Výchozí umístíní složky registru, která se používá pro tyto operace:


HKEY_CURRENT_USER\Software\Programmers Wizard\Scripting





Knihovna funkcí - prohlížení

 *Kategorie: Registr*

 *Téma: RegReadString*

Deklarace:

```
function RegReadString(Name: String; DefaultValue: String): String
```

Parametry:

Name: String

Øetizec udávající název hodnoty, která se má pøeèíst

DefaultValue: String

Hodnota, která má být vrácena v pøípadi, že hodnota s názvem *Name* neexistuje.

Popis:

Funkce *RegReadString* ète z registru døíve uloženou pojmenovanou hodnotu. Takovou hodnotu lze zapsat voláním funkce *RegWriteString*, hodnota pak zůstává v registru uložena pro pozdijší použití pøi pøíštím spuštíní skriptu. Všechny skripty sdílejí stejnou složku v registru.

Návratový výraz:

Funkce ète hodnotu pojmenovanou *Name* a vrací ji jako typ *String*. Pokud hodnota neexistuje, vrací hodnotu *DefaultValue*.

Poznámky:

Výchozí umístíní složky registru, která se používá pro tyto operace:


HKEY_CURRENT_USER\Software\Programmers Wizard\Scripting





Knihovna funkcí - prohlížení

 *Kategorie: Registr*

 *Téma: RegWriteInt*

Deklarace:

function RegWriteInt(Name: String; Value: LongInt): Boolean

Parametry:

Name: String

 Øetizec udávající název hodnoty, která se má zapsat

Value: LongInt

 Vlastní hodnota

Popis:

Funkce *RegWriteInt* se používá k uložení libovolných informací do registru. V registru jsou uložené hodnoty organizovány jako dvojice *NÁZEV = HODNOTA*. Této dvojici odpovídají parametry *Name* a *Value*. Skript může použít tuto funkci pro zapsání informací, které bude potøebovat pøi dalším spuštíní, protože tyto informace v registru zůstávají i po ukonèení skriptu. Pøi dalším spuštíní může skript použít funkci *RegReadInt* a svou hodnotu si pøeèíst.

Návratový výraz:

Funkce vrací *TRUE* pokud vše probíhlo bez problémù a hodnota byla zapsána, jinak vrací *FALSE*.

Poznámky:


Výchozí umístíní složky registru, která se používá pro tyto operace:
HKEY_CURRENT_USER\Software\Programmers Wizard\Scripting





Knihovna funkcí - prohlížení

 *Kategorie: Registr*

 *Téma: RegWriteString*

Deklarace:

function RegWriteString(Name: String; Value: LongInt): Boolean

Parametry:

Name: String

Øetizec udávající název hodnoty, která se má zapsat

Value: LongInt

Vlastní hodnota

Popis:

Funkce *RegWriteString* se používá k uložení libovolných informací do registru. V registru jsou uložené hodnoty organizovány jako dvojice *NÁZEV = HODNOTA*. Této dvojici odpovídají parametry *Name* a *Value*. Skript může použít tuto funkci pro zapsání informací, které bude potøebovat při dalším spuštění, protože tyto informace v registru zůstávají i po ukonèení skriptu. Při dalším spuštění může skript použít funkci RegReadString a svou hodnotu si přeèíst.

Návratový výraz:

Funkce vrací *TRUE* pokud vše probíhlo bez problémù a hodnota byla zapsána, jinak vrací *FALSE*.

Poznámky:

Výchozí umístění složky registru, která se používá pro tyto operace:
HKEY_CURRENT_USER\Software\Programmers Wizard\Scripting





Registrace

Po úspěšné instalaci programu je potřeba Vaši kopii zaregistrovat. Registrace je bezplatná, zabere Vám jen pár minut a odesílá se pomocí e-mailu na adresu pwiz@volny.cz nebo petr.esner@atlas.cz. Vinujte prosím těch několik málo minut vyplnění registračního průvodce, pomůže mi to v udržení statistik o využitelnosti programu a počtu uživatelů. Pokud nevidím ohlas uživatelů, jaký budu mít důvod k pokračování vývoje tohoto programu?

Program samotný nebude nijak urgovat registraci, nikdy nezačne zobrazovat zdržující dialogy s upozorněním na nutnou registraci, nebude zobrazovat žádnou reklamu a nikdy neomezí některé své funkce, aby tak donutil uživatele k registraci. Přesto (nebo právě proto, oceníte takový přístup?) je potřeba program zaregistrovat.

Registraci můžete spustit ze stránek pwiz.hyperlink.cz.

Registrací nezískáte jen dobrý pocit :-), získáte hlavní přednostní přístup k technické podpoře a jistotu, že Vaše e-maily nikdy nebudou odbyty automatickou odpovědí (tímto se omlouvám neregistrovaným uživatelům; je velmi náročné odpovídat velkému množství uživatelů na stejné dotazy a reagovat na podobné připomínky).

Poznámka pro vlastníky verze Programmer's Wizard 2000:

I když je Programmer's Wizard 2000 zaregistrován, je potřeba zaregistrovat verzi Programmer's Wizard 2 zvlášť. Registrace pak samozřejmě není nutná u dalších verzí programu Programmer's Wizard 2 případně při stažení aktualizovaných zásuvných modulů.


Autor





Knihovna funkcí - prohlížení

 *Kategorie:* Obecné

 *Téma:* SendMail

Deklarace:

function SendMail(To, Subject, Text: String): Boolean

Parametry:

To: String

E-mailová adresa příjemce

Subject: String;

Titulek zprávy

Text: String

Vlastní text zprávy

Popis:

Funkce *SendMail* odesílá emailovou zprávu. Parametr *To* udává adresu nebo adresy příjemce(ů), jednotlivé adresy musí být odděleny středníkem. *Subject* obsahuje titulek (subjekt) zprávy a v parametru *Text* by měl být vlastní text zprávy. Funkce při svém zavolání také zobrazí dialog, ve kterém může uživatel pozmínit text zprávy, případně přidat přílohy.

Tato funkce neumožňuje nastavení příjemců kopie zprávy ani přikládání příloh, pro tyto dodatečné funkce lze použít funkci *SendMailEx*.

Návratový výraz:


Funkce vrací *TRUE* pokud byla zpráva úspěšně odeslána, jinak vrací *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: Obecné*

 *Téma: SendMail*

Deklarace:

```
function SendMailEx(var ToList, CcList, AttachmentList: Array of String; Subject, Text: String): Boolean
```

Parametry:

var ToList: Array of String

Pole se seznamem příjemců zprávy

var CcList: Array of String

Pole se seznamem příjemců kopie zprávy

var AttachmentList: Array of String

Pole se seznamem souborů, které mají být ke zprávě přiloženy

Subject: String;

Titulek zprávy

Text: String

Vlastní text zprávy

Popis:

Funkce *SendMailEx* odesílá emailovou zprávu. Parametr *ToList* je dynamické pole a jeho prvky udávají adresy příjemce(ů). *CcList* obsahuje stejným způsobem uložené adresy příjemců kopií zprávy a pole *AttachmentList* může obsahovat seznam souborů, které mají být přiloženy. *Subject* obsahuje titulek (subjekt) zprávy a v parametru *Text* by měl být vlastní text zprávy. Funkce při svém zavolání také zobrazí dialog, ve kterém může uživatel pozměnit text zprávy, případně přidat přílohy.

Pro případy kde není potřeba parametrů *CcList* a *AttachmentList* je vhodnější funkce SendMail .

Návratový výraz:


Funkce vrací *TRUE* pokud byla zpráva úspěšně odeslána, jinak vrací *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie:* Obecné

 *Téma:* EnumFiles

Deklarace:

procedure EnumFiles(var Files: Array of String)

Parametry:

var Files: Array of String

Do tohoto parametru je vrácen seznam otevřených dokumentů

Popis:

Funkce *EnumFiles* se používá k získání informací o právi otevřených dokumentech. Funkce vrátí seznam dokumentů do parametru *Files*, seznamem je myšlen výpis všech pojmenovaných dokumentů, které jsou právi na otevřené ploše; nezáleží na tom, zda je otevřen projekt. Prvky pole *Files* budou obsahovat plné cesty k otevřeným dokumentům, skript pak může zjistit velikost pole a tím i počet dokumentů voláním funkce GetArrayLength.

Návratový výraz:

Funkce nevrací žádnou hodnotu přímo, miní parametr *Files*.

Poznámky:


Tato funkce vytvoří seznam otevřených dokumentů, ignoruje nové, ještě nepojmenované dokumenty. Pokud potřebuje skript i tuto informaci, měl by použít funkci *EnumWindows*.





Knihovna funkcí - prohlížení

 *Kategorie:* [Okna](#)

 *Téma:* EnumWindows

Deklarace:

procedure EnumWindows(var Handles: Array of LongInt)

Parametry:

var Handles: Array of LongInt

Do tohoto parametru je vrácen seznam otevřených oken programu

Popis:

Funkci *EnumWindows* lze použít pro zjištění stavu plochy programu. Funkce vytváří seznam všech otevřených oken a vkládá jej do parametru *Handles*. Seznamem oken jsou zde myšlena všechna okna nástrojů včetně oken nástrojů pro editaci dokumentů (jako je např. [SyntaxEditor](#)) i oken jiných nástrojů, např. nástroje [Process Navigator](#). Vrácený seznam obsahuje identifikátory oken, které lze použít v dalších funkcích a tím zjistit více podrobností.

Návratový výraz:

Funkce nevrací žádnou hodnotu přímo, miní parametr *Handles*.

Poznámky:

Tato funkce vytvoří seznam všech otevřených oken včetně nových, ještě nepojmenovaných dokumentů. Pokud skript nepotřebuje informace o nepojmenovaných dokumentech, měl by použít funkci [EnumFiles](#).





Knihovna funkcí a procedur (Pascal Script)

(zpit do obsahu knihovny [tudy](#))

[Termíny použité v této kategorii](#)

Kategorie Okna

[EnumWindows](#)

[GetCurrentWindow](#)

[IsWindow](#)

[WindowToFileName](#)

[WindowToPluginID](#)

[WindowToPluginEntryID](#)

Odkazy:

[Skriptování](#)

[Datové typy](#)

[Cykly a příkazy](#)

[Matematické operace](#)


[Příklady](#)





Knihovna funkcí - prohlížení

 *Kategorie:* Okna

 *Téma:* WindowToFileName

Deklarace:

function WindowToFileName(Handle: LongInt; var FileName: String): Boolean

Parametry:

Handle: LongInt

Identifikátor okna

var FileName: String

Do tohoto parametru je vrácen název dokumentu, upravovaném v okně *Handle*.

Popis:

Funkce *WindowToFileName* převádí identifikátor (*Handle*) okna na název dokumentu, který se v daném okně právě upravuje. Název dokumentu funkce zkopíruje do parametru *FileName*.

Návratový výraz:

Funkce vrací *TRUE* pokud vše proběhlo bez chyb a název dokumentu byl zkopírován do parametru *FileName*. Pokud okno neexistuje, vrací funkce *FALSE*.

Poznámky:


Pro získání seznamu všech existujících oken by skript měl použít funkci [EnumWindows](#).





Knihovna funkcí - prohlížení

 *Kategorie: Okna*

 *Téma: IsWindow*

Deklarace:

function IsWindow(Handle: LongInt): Boolean

Parametry:

Handle: LongInt

Identifikátor okna

Popis:

Funkce *IsWindow* se používá ke zjištění, zda okno s identifikátorem *Handle* existuje. Oknem je zde myšleno okno na ploše programu Programmer's Wizard 2, většinou okno některého nástroje.

Návratový výraz:


Funkce vrací *TRUE* pokud okno existuje, jinak vrací *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie:* Okna

 *Téma:* WindowToPluginID

Deklarace:

function WindowToPluginID(Handle: LongInt; var PluginID: String): Boolean

Parametry:

Handle: LongInt

Identifikátor okna

var PluginID: String

Do tohoto parametru je vrácen název dokumentu, upravovaném v okně *Handle*.

Popis:

Funkce *WindowToFileName* převádí identifikátor (*Handle*) okna na název dokumentu, který se v daném okně právě upravuje. Název dokumentu funkce zkopíruje do parametru *FileName*.

Návratový výraz:

Funkce vrací *TRUE* pokud vše proběhlo bez chyb a název dokumentu byl zkopírován do parametru *FileName*. Pokud okno neexistuje, vrací funkce *FALSE*.

Poznámky:


Pro získání seznamu všech existujících oken by skript měl použít funkci [EnumWindows](#).





Knihovna funkcí - prohlížení

 *Kategorie: Okna*

 *Téma: WindowToPluginEntryID*

Deklarace:

function WindowToPluginEntryID(Handle: LongInt; var PluginEntryID: String): Boolean

Parametry:

Handle: LongInt

Identifikátor okna

var PluginEntryID: String

Do tohoto parametru je vrácen název dokumentu, upravovaném v okně *Handle*.

Popis:

Funkce *WindowToFileName* převádí identifikátor (*Handle*) okna na název dokumentu, který se v daném okně právě upravuje. Název dokumentu funkce zkopíruje do parametru *FileName*.

Návratový výraz:

Funkce vrací *TRUE* pokud vše proběhlo bez chyb a název dokumentu byl zkopírován do parametru *FileName*. Pokud okno neexistuje, vrací funkce *FALSE*.

Poznámky:

Pro získání seznamu všech existujících oken by skript měl použít funkci [EnumWindows](#).





Knihovna funkcí - prohlížení



Kategorie: Okna

Téma: Použité termíny



"Identifikátor okna Handle"

V této kategorii se často používá a mluví o identifikátoru okna *Handle*. Identifikátorem *Handle* je zde myšleno skutečné *Handle* systému Windows (více informací o tomto datovém typu v nápovědi pro WinAPI). Některé funkce této kategorie mají shodná jména jako funkce Windows, jde o funkce jako *EnumWindows* a *IsWindow*. Tyto funkce pracují obdobně jako jejich WinAPI ekvivalenty s tím rozdílem, že pracují pouze v kontextu plochy programu Programmer's Wizard 2.



"Okna na ploše programu"


Pro aplikace systému Windows, kontextem pro funkce typu *EnumWindows* je samotný OS Windows a funkcí lze zjistit výpis všech dceřinných oken k danému rodičovskému. Pro skript programu Programmer's Wizard 2 je kontextem a jediným rodičovským oknem MDI klient aplikace, tedy plocha, která obsahuje všechny okna nástrojů.






Knihovna funkcí - prohlížení

 *Kategorie: Projekt*

 *Téma: Použité termíny*

 "identifikátor projektu **ProjectID**"


V programu Programmer's Wizard 2 jsou všechny projekty uspořádány do pole prvků, které lze adresovat jako pole Pascal Scriptu. Prvním prvkem pole (a tedy i prvním projektem) je projekt s indexem 0, další jsou 1, 2 atd. Index posledního prvku v poli lze zjistit voláním funkce GetProjectCount, platné hodnoty ProjectID jsou pak v rozsahu " $0 \leq index < GetProjectCount$ ". Skript nemůže předpokládat, že toto pole prvků je neměnné; pole se může měnit za běhu programu tak, jak uživatel vytváří nové projekty; může být také změněno samotným skriptem za jeho běhu funkcemi z této kategorie.





Knihovna funkcí - prohlížení

 *Kategorie Plochy*

 *Téma: Použité termíny*

 "identifikátor projektu **WorkspaceID**"

V programu Programmer's Wizard 2 jsou všechny plochy projektu uspořádány do pole prvků, které lze adresovat jako pole Pascal Scriptu. Prvním plochou pole je plocha s indexem 0, další jsou 1, 2 atd. Index posledního prvku v poli lze zjistit voláním funkce GetWorkspaceCount, resp. GetWorkspaceCountEx. Platné hodnoty WorkspaceID jsou pak v rozsahu " $0 \leq index <$

GetWorkspaceCount/GetWorkspaceCountEx". Skript nemůže předpokládat, že toto pole prvků je pro každý projekt nemenné; pole se může měnit za běhu programu tak, jak uživatel vytváří nové plochy a odebírá existující; může být také zminino samotným skriptem za jeho běhu funkcemi z této kategorie.





Knihovna funkcí a procedur (Pascal Script)

(zpět do obsahu knihovny [tady](#))

[Termíny použité v této kategorii](#)

Kategorie Zásuvné moduly

[EnumPluginIDs](#)

[EnumPluginEntryIDs](#)

[PluginEntryIDToName](#)

[PluginIDToFullName](#)

[PluginIDToShortName](#)

Odkazy:

[Skriptování](#)

[Datové typy](#)

[Cykly a příkazy](#)

[Matematické operace](#)

[Příklady](#)





Knihovna funkcí - prohlížení

Kategorie Zásuvné moduly

Téma: Použité termíny

"identifikátor **PluginID**"

V programu Programmer's Wizard 2 má každý zásuvný modul svůj identifikátor, zvaný *PluginID*. Toto ID je řetězec, který se nemění při přidávání a odebrání modulů. Programmer's Wizard 2 po svém načtení vždy vytvoří seznam svých modulů a přiřadí jim jejich *PluginID*. Skript může za všech okolností předpokládat, že žádné dva zásuvné moduly nedostanou přiřazeno stejné *PluginID*. *PluginID* je řetězec ve formátu: **_pluginXXXXXXXX**, kde **XXXXXXXX** je osmimístný hexadecimální výraz, vypočítaný programem a přiřazený modulu.


"identifikátor **PluginEntryID**"

V programu Programmer's Wizard 2 může každý zásuvný modul obsahovat jeden nebo více nástrojů. Každý z těchto nástrojů má svůj identifikátor, zvaný *PluginEntryID*. Identifikátor obsahuje dvě části: *PluginID* a index nástroje v rámci modulu. *PluginEntryID* je unikátním řetězcem pro každý nástroj, který je načten. Stejně jako u identifikátoru *PluginID*, i zde může skript počítat s neměnností výrazu. *PluginEntryID* je řetězec ve formátu: **PluginID_entryX**, kde *PluginID* je identifikátor modulu a **X** je index nástroje.





Knihovna funkcí - prohlížení

 *Kategorie: Zásuvné moduly*

 *Téma: PluginIDToFullName*

Deklarace:

function PluginIDToFullName(PluginID: String; var FullName: String): Boolean

Parametry:

PluginID: String

Identifikátor modulu

var FullName: String

Do tohoto parametru je vrácen plný název modulu, definovaného parametrem *PluginEntryID*

Popis:

Funkce *PluginIDToFullName* převádí identifikátor (*PluginID*) modulu na jeho plný název. Tento název funkce zkopíruje do parametru *FullName*. Pro získání krátkého jména modulu může skript použít funkci [PluginIDToShortName](#).

Návratový výraz:

Funkce vrací *TRUE* pokud vše proběhlo bez chyb a název modulu byl zkopírován do parametru *FullName*. Pokud modul definovaný parametrem *PluginID* neexistuje, vrací funkce *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: Zásuvné moduly*

 *Téma: PluginIDToShortName*

Deklarace:

function PluginIDToShortName(PluginID: String; var ShortName: String): Boolean

Parametry:

PluginID: String

Identifikátor modulu

var ShortName: String

Do tohoto parametru je vrácen krátký název modulu, definovaného parametrem *PluginEntryID*

Popis:

Funkce *PluginIDToShortName* převádí identifikátor (*PluginID*) modulu na jeho krátký název. Tento název funkce zkopíruje do parametru *ShortName*. Pro získání plného jména modulu může skript použít funkci [PluginIDToFullName](#).


Návratový výraz:

Funkce vrací *TRUE* pokud vše proběhlo bez chyb a název modulu byl zkopírován do parametru *ShortName*. Pokud modul definovaný parametrem *PluginID* neexistuje, vrací funkce *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: Zásuvné moduly*

 *Téma: PluginEntryIDToName*

Deklarace:

function PluginEntryIDToName(PluginEntryID: String; var Name: String): Boolean

Parametry:

PluginEntryID: String

Identifikátor modulu

var Name: String

Do tohoto parametru je vrácen název nástroje, definovaného parametrem *PluginEntryID*

Popis:

Funkce *PluginEntryIDToName* převádí identifikátor (*PluginEntryID*) nástroje modulu na jeho název. Tento název funkce zkopíruje do parametru *Name*.


Návratový výraz:


Funkce vrací *TRUE* pokud vše proběhlo bez chyb a název nástroje modulu byl zkopírován do parametru *Name*. Pokud nástroj definovaný parametrem *PluginEntryID* neexistuje, vrací funkce *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: Zásuvné moduly*

 *Téma: EnumPluginIDs*

Deklarace:

procedure EnumPluginIDs(var PluginIDs: Array of String)

Parametry:

var PluginIDs: Array of String

Pole, do kterého funkce zkopíruje seznam nainstalovaných modulů

Popis:

Funkce *EnumPluginIDs* vytváří seznam všech nainstalovaných zásuvných modulů a kopíruje jej do parametru *PluginIDs*. Více informací o modulech lze získat použitím dalších funkcí této kategorie. Skript může procházet nástroje nalezených modulů funkcí EnumPluginEntryIDs.

Návratový výraz:

Nevrací žádnou hodnotu přímo, miní parametr *PluginIDs*.





Knihovna funkcí - prohlížení

 *Kategorie: Zásuvné moduly*

 *Téma: EnumPluginEntryIDs*

Deklarace:

function EnumPluginEntryIDs(PluginID: String; var PluginEntryIDs: Array of String): Boolean

Parametry:

PluginID: String

Identifikátor modulu, jehož nástroje má funkce zjistit

var PluginEntryIDs: Array of String

Pole, do kterého funkce zkopíruje seznam nástrojů daného modulu

Popis:

Funkce *EnumPluginEntryIDs* vytváří seznam všech nástrojů, které jsou dostupné v modulu definovaném parametrem *PluginID*. Funkce kopíruje vytvořený seznam do parametru *PluginEntryIDs*. Pro zjištění nainstalovaných modulů a jejich identifikátorů *PluginID* může skript použít funkce EnumPluginIDs.

Návratový výraz:


Funkce vrací *TRUE* pokud vše proběhlo v pořádku a seznam byl úspěšně vytvořen a zkopírován. Pokud modul *PluginID* neexistuje, vrací funkce *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: Obecné*

 *Téma: GetCurrentScript*

Deklarace:

function GetCurrentScript: String

Popis:

Funkce *GetCurrentScript* vrací název a plnou cestu k souboru, ze kterého byl právě běžící skript načten.

Návratový výraz:

Funkce vrací vždy název a plnou cestu k souboru, ze kterého byl skript načten.

Poznámky:

Skript by měl tuto funkci používat v souvislosti s funkcemi registru *RegReadInt*, *RegReadString*, *RegWriteInt* a *RegWriteString*. Před voláním těchto funkcí by skript vždy měl použít název sebe sama jako prefix k názvům hodnot, které chce zapisovat a číst, tak lze zabránit případným konfliktům.





Knihovna funkcí - prohlížení

 *Kategorie:* Okna

 *Téma:* GetCurrentWindow

Deklarace:

function GetCurrentWindow: LongInt

Popis:

Funkce *GetCurrentWindow* vrací *Handle* okna, které je právě aktivní. Pokud je otevřeno jen jedno okno, funkce použije jeho handle, při více oknech vrací funkce handle toho okna, které je právě aktivní.

Návratový výraz:


Funkce vrací *Handle* právě aktivního okna. Pokud není otevřeno žádné okno, vrací funkce hodnotu nula (0).





Knihovna funkcí - prohlížení

 *Kategorie: Obecné*

 *Téma: Sleep*

Deklarace:

procedure Sleep(Milliseconds: LongInt);

Parametry:

Milliseconds: LongInt

Interval, na který má být skript zmrazen

Popis:

Funkce *Sleep* pozastaví provádění skriptu na dobu uvedenou v parametru *Milliseconds*. I když je skript zmrazen, uživatel může stále použít funkci *Pøerušit skript* a tím běžící skript ukončit.

Návratový výraz:


Nevrací žádný výraz.





Knihovna funkcí - prohlížení

 *Kategorie: Obecné*

 *Téma: OpenFileEx*

Deklarace:

function OpenFileEx(FileName, PluginEntryID: String): Boolean

Parametry:

FileName: String

Název dokumentu k otevření

PluginEntryID: String

Identifikátor nástroje, kterým se má modul otevřít

Popis:

Funkce *OpenFileEx* otevírá dokument udaný parametrem *FileName* v programu Programmer's Wizard 2. Dokument se otevře nástrojem, který je identifikován parametrem *PluginEntryID* (více informací [zde](#)). Tato funkce vždy otevře soubor v programu Programmer's Wizard 2, pro spuštění programu nebo otevření dokumentu jeho asociovaným program je vhodnější funkce [ExecuteFile](#).

Návratový výraz:


Funkce vrací *TRUE* pokud byl soubor úspěšně otevřen, jinak vrací *FALSE*.





Knihovna funkcí - prohlížení

 *Kategorie: Přímá komunikace s uživatelem*

 *Téma: ShowCheckListDialog*

Deklarace:

function ShowCheckListDialog(Caption, Prompt: String; var Lines: Array of String; var CheckBoxes: Array of boolean): LongInt

Parametry:

Caption: String

Titulek okna

Prompt: String

Text uvnitř okna

var Lines: Array of String

Seznam řetězců, které se mají zobrazit jako jednotlivé položky

var CheckBoxes: Array of boolean

Pole popisující stav zaškrtačacích tlačítek u jednotlivých položek

Popis:

Funkce *ShowCheckListDialog* pracuje obdobně jako funkce *ShowListDialog*, navíc dokáže vedle každé položky v seznamu zobrazit zaškrtačací tlačítko. Uživatel může zaškrtnout nebo odškrtnout jakékoliv tlačítko. Při otevírání okna jsou tlačítka nastavena na hodnoty, udané parametrem *CheckBoxes*; každému prvku z pole *Lines* odpovídá příslušný prvek z *CheckBoxes*. Pokud je prvek pole nastaven na hodnotu *TRUE*, bude tlačítko na příslušném řádku zaškrtnuto, při hodnotě *FALSE* bude existovat. Uživatel může zaškrtačacím a odškrtačacím tlačítkem měnit stav pole *CheckBoxes*.


Návratový výraz:


Funkce vrací index vybraného řetězce. Tento index lze přímo použít do pole *Lines* takto: "Lines[result]" kde *result* je hodnota vrácená funkcí *ShowListDialog*. Když uživatel zruší výběr prvku tlačítkem *Storno*, funkce vrací hodnotu -1. Skript musí vždy ověřit, zda není výsledná hodnota -1 protože bez tohoto testu by mohla být provedena tato instrukce: "Lines[-1]". Instrukce je neplatná protože první prvek z pole má index 0 - skript by byl okamžitě ukončen s chybovým hlášením. Stav zaškrtačacích tlačítek je vrácen do pole *CheckBoxes* a přistupovat k nim lze stejně, jako k poli *Lines*.





Knihovna funkcí - prohlížení

 *Kategorie: FS (soubory a složky)*

 *Téma: CopyFile*

Deklarace:

function CopyFile(ExistingName, NewName: String; FailIfExists: Boolean): Boolean

Parametry:

ExistingName: String

Plná cesta k existujícímu souboru

NewName: String

Nový název souboru

Popis:

Funkce *CopyFile* přejmenuje fyzický soubor udaný parametrem *ExistingName* na název *NewName*.

Návratový výraz:

Funkce vrací *TRUE* pokud byl soubor smazán, jinak vrací *FALSE*.

Poznámky:

Tato funkce je jednou z chráněných funkcí a může být v nastavení skriptování zakázána. Skript by si měl vždy ověřit zda jsou tyto funkce povoleny voláním *IsSafeScriptingEnabled*.





Knihovna funkcí - prohlížení



Kategorie: Obecné

Téma: GetProjectsDir

Deklarace:

function GetProjectsDir: String

Popis:

Funkce *GetProjectsDir* se používá ke zjištění cesty ke složce, ve které jsou uloženy všechny projekty.

Návratový výraz:


Funkce vrací vždy plnou cestu ke složce, ve které jsou uloženy všechny projekty.





Knihovna funkcí - prohlížení

 *Kategorie:* Obecné

 *Téma:* GetScriptsDir

Deklarace:

function GetScriptsDir: String

Popis:

Funkce *GetScriptsDir* se používá ke zjištění cesty ke složce, ve které jsou uloženy všechny skripty. Uživatel ale může spustit skript z jakékoliv složky a běžící skript proto nemůže předpokládat, že je spuštěn ze složky vrácené touto funkcí. Pro zjištění aktuální složky, ze které je skript spuštěn, je vhodnější funkce GetCurrentScript.

Návratový výraz:

Funkce vrací vždy plnou cestu ke složce, ve které jsou uloženy všechny skripty.





Knihovna funkcí - prohlížení

 *Kategorie: Přímá komunikace s uživatelem*

 *Téma: SelectDirectory*

Deklarace:

function SelectDirectory(Caption: String; var Directory: String): Boolean

Parametry:

Caption: String

Titulek okna

var Directory: String

Do tohoto parametru je při uzavření okna zkopírována složka, kterou uživatel vybral

Popis:

Funkce *SelectDirectory* zobrazuje dialogové okno, ve kterém může uživatel vybrat libovolný fyzický adresář. Funkci lze použít všude tam, kde má mít uživatel možnost vybrat složku.

Návratový výraz:

Funkce vrací *TRUE* pokud bylo stisknuto tlačítko OK a parametr *Directory* nyní obsahuje plnou cestu k vybrané složce, jinak vrací *FALSE*.





Knihovna funkcí - prohlížení

Kategorie: Přímá komunikace s uživatelem

Téma: SelectFiles

Deklarace:

function SelectFiles(InitialDir, Filter: String; var Files: Array of String): Boolean

Parametry:

InitialDir: String

Titulek okna

Filter: String

Tento parametr udává filtr, který se má zobrazit v dialogu Otevřít. Filtr musí mít tento formát:

'Název typu souboru 1|*.příklad souboru 1|' +

'Název typu souboru 2|*.příklad souboru 2|' +

'Název typu souboru 3|*.příklad souboru 3'

var Files: String

Do tohoto parametru je při uzavření okna zkopírován seznam vybraných souborů

Popis:

Funkce *SelectFiles* dovoluje zobrazit dialog, ve kterém může uživatel vybrat libovolný existující soubor nebo soubory. Parametr *InitialDir* určuje výchozí adresář; ten bude vybrán po otevření okna. Parametr *Filter* udává seznam položek v seznamu *Typ souboru*, více informací dále.

Návratový výraz:

Funkce vrací *TRUE* pokud bylo stisknuto tlačítko OK a parametr *Files* nyní obsahuje seznam všech vybraných souborů, jinak vrací *FALSE*.

Příklad parametru *Filter*:

Filter := "Textové soubory (.txt;*.log)|*.txt;*.log|Wordpad (*.rtf)|*.rtf"*







Knihovna funkcí a procedur (Pascal Script)


(zpřít do obsahu knihovny tudy)


Dokumenty (soubory a složky)


 EnumDocuments


 SetDocuments


Odkazy:

 Skriptování

 Datové typy

 Cykly a příkazy

 Matematické operace

 Příklady





Knihovna funkcí - prohlížení



Kategorie: Dokumenty

Téma: EnumDocuments

Deklarace:

procedure EnumDocuments(var Documents: Array of String)

Parametry:

var Documents: Array of String

Do tohoto parametru je vrácen seznam souborů ve složce dokumentů

Popis:

Funkci *EnumDocuments* lze použít pro zjištění stavu virtuální složky Dokumenty. Funkce vrací seznam souborů ve složce Dokumenty do parametru *Documents*.


Návratový výraz:

Funkce nevrací žádnou hodnotu přímo, miní parametr *Documents*.





Knihovna funkcí - prohlížení

 *Kategorie: Dokumenty*

 *Téma: SetDocuments*

Deklarace:

procedure SetDocuments(var Documents: Array of String)

Parametry:

var Documents: Array of String

Obsah tohoto parametru je zkopírován do složky Dokumenty

Popis:

Funkce *SetDocuments* miní obsah složky Dokumenty tak, aby odpovídal seznamu souborů, udaného parametrem *Documents*.

Návratový výraz:


Nevrací žádný výraz.





Knihovna funkcí - prohlížení

 *Kategorie: Projekt*

 *Téma: EnumProjectFiles*

Deklarace:

function EnumProjectFiles(ProjectID: LongInt; var Files: Array of String): Boolean

Parametry:

ProjectID: LongInt

ID projektu

var Files: Array of String

Do tohoto parametru je zkopírován seznam souborů v projektu

Popis:

Funkce *EnumProjectFiles* získává seznam všech souborů, které jsou v projektu definovaném parametrem *ProjectID*. Tento seznam pak kopíruje do parametru *Files*.

Návratový výraz:


Funkce vrací *TRUE* pokud byly informace zkopírovány do parametrů, jinak vrací *FALSE*. Funkce většinou vrátí *FALSE* jako následek nesprávného parametru *ProjectID*.





Knihovna funkcí - prohlížení

 *Kategorie: Projekt*

 *Téma: SetProjectFiles*

Deklarace:

function SetProjectFiles(ProjectID: LongInt; var Files: Array of String): Boolean

Parametry:

ProjectID: LongInt

ID projektu

var Files: Array of String

Obsah tohoto parametru je zkopírován do seznamu souborů v daném projektu

Popis:

Funkce *SetProjectFiles* miní seznam souborů v projektu *ProjectID* tak, aby odpovídat seznamu souborů definovaným parametrem *Files*.

Návratový výraz:


Funkce vrací *TRUE* pokud byly informace zkopírovány, jinak vrací *FALSE*. Funkce většinou vrátí *FALSE* jako následek nesprávného parametru *ProjectID*.





Knihovna funkcí - prohlížení

 *Kategorie: Projekt*

 *Téma: CreateProject*

Deklarace:

function CreateProject(Name, Description: String): LongInt

Parametry:

Name: String

Název nového projektu

Description: String

Popis nového projektu

Popis:

Funkce *CreateProject* vytváří nový projekt a vrací jeho identifikátor *ProjectID*. Nový projekt bude mít název *Name* a popis *Description*, projektu bude také přiřazena jeho primární plocha. Skript by nikdy neměl vytvářet projekt bez jména, tedy příkazem "*CreateProject(,)*". Projekt není po vytvoření automaticky otevřen, pro otevření projektu může skript použít funkci *OpenProject*.

Návratový výraz:


Funkce vrací vždy *ProjectID* vytvořeného projektu.





Knihovna funkcí - prohlížení

 *Kategorie: Plochy*

 *Téma: CreateWorkspace*

Deklarace:

function CreateWorkspace(Name, Description: String): LongInt

Parametry:

Name: String

Název nové plochy

Description: String

Popis nové plochy

Popis:

Funkce *CreateWorkspace* vytváří novou plochu právi otevřeného projektu a vrací její identifikátor *WorkspaceID*. Nová plocha bude mít název *Name* a popis *Description*. Skript by nikdy neměl vytvářet plochu bez jména, tedy příkazem "*CreateWorkspace*(\", \")". Plocha není po vytvoření automaticky otevřena, pro otevření plochy může skript použít funkci *OpenWorkspace*. Tato funkce vytváří plochu právi otevřeného projektu, pokud skript potřebuje vytvořit plochu jiného projektu, měl by použít funkci *CreateWorkspaceEx*.

Návratový výraz:


Funkce vrací *WorkspaceID* vytvořené plochy. Pokud není otevřen žádný projekt, vrací funkce *WorkspaceID -1*, což je pro všechny další funkce neplatné ID.





Knihovna funkcí - prohlížení

 *Kategorie: Plochy*

 *Téma: CreateWorkspace*

Deklarace:

function CreateWorkspace(ProjectID: LongInt; Name, Description: String): LongInt

Parametry:

ProjectID: LongInt

ID projektu, kterému má být plocha vytvořena

Name: String

Název nové plochy

Description: String

Popis nové plochy

Popis:

Funkce *CreateWorkspaceEx* vytváří novou plochu projektu *ProjectID* a vrací její identifikátor *WorkspaceID*. Nová plocha bude mít název *Name* a popis *Description*. Skript by nikdy neměl vytvářet plochu bez jména, tedy příkazem "*CreateWorkspaceEx(ProjectID, "", "")*". Plocha není po vytvoření automaticky otevřena, pro otevření plochy může skript použít kombinaci funkcí *OpenProject* a *OpenWorkspace*.

Návratový výraz:

Funkce vrací *WorkspaceID* vytvořené plochy. Pokud není *ProjectID* platný, vrací funkce *WorkspaceID -1*, což je pro všechny další funkce neplatné ID.

Poznámky:

Pokud potřebuje skript vytvořit plochu právě otevřeného projektu, neměl by používat zápis "*CreateWorkspaceEx(GetCurrentWorkspace, Name, Desc)*". Místo toho by měl k vytvoření plochy využít funkci *CreateWorkspace*.



