Prostředí pro programování aplikací

Při vytváření databázových aplikací ve WinBase602 **lze využit jejího vnitřního programovacího jazyka.**

Tento jazyk vychází z jazyka **Pascal** a uchovává si jeho výhody: jednoduchost, přehlednost programů a jistou pedagogičnost. Programování ve vnitřním jazyce bude snadné pro ty, kteří znají **Pascal**, nebude ovšem příliš obtížné pro nikoho, kdo je zvyklý na některý vyšší programovací jazyk.

Programování je činností, která vyžaduje rozsáhlejší průpravu, než návrh databázových tabulek, dotazů, formulářů a jejich spojování do aplikace. Tento manuál *není* totiž učebnicí programování, nýbrž pouze návodem k použití vnitřního programovacího jazyka, a nepředpokládáme, že by se z něj mohl naučit programovat člověk, který s programovacím jazykem dosud nikdy nepřišel do styku. Kdybychom **WinBase602** přirovnali k automobilu, pak manuál nenahrazuje autoškolu, pouze detailně popisuje určitý typ auta.

V této kapitole se dále seznámíte s nástroji **WinBase602** umožňujícími editování, překládání, ladění a provozování programů ve vnitřním jazyce. V jedné z následujících kapitol je pak obsažen plný popis vnitřního jazyka.

Základní pojmy

Textu programu v podobě, v jaké jej vytvoří programátor, říkáme ZDROJOVÝ PROGRAM**.**

Zdrojový program je objektem uloženým pod svým jménem ve **WinBase602** v rámci některé aplikace a lze s ním pracovat z řídicího panelu.

Překlad a chyby při překladu

Aby zdrojový program mohl být proveden počítačem, musí být napřed PŘELOŽEN překladačem. Překladač zpracuje zdrojový program a vyrobí PŘELOŽENÝ PROGRAM neboli KÓD. Po překladu lze spustit BĚH PROGRAMU. Během překladu může dojít k CHYBÁM PŘI PŘEKLADU (někdy se jim také ne zcela přesně říká SYNTAKTICKÉ CHYBY). Dojde-li k takové chybě ve **WinBase602**, je překlad ukončen, aniž by vznikl přeložený program, a tudíž nelze program spustit. Příčinou chyb při překladu je zpravidla to, že zdrojový program není napsán podle pravidel programovacího jazyka.

Běhové chyby

Při běhu programu může dojít k řadě situací, kdy není jasné, jak by měl výpočet smysluplně pokračovat. Příkladem je třeba dělení nulou, vyčerpání dostupné operační paměti nebo pokus pracovat s neexistujícím záznamem v databázi. Takovým situacím říkáme BĚHOVÉ CHYBY.

Ladící systém

Myšlenkové chyby v návrhu aplikace se mohou projevit jak běhovými chybami, tak i jejím nežádoucím chováním. Nalezení a odstranění takových chyb usnadňuje ladící systém (debugger). Umožňuje provozovat aplikaci v ladícím režimu a přitom například krokovat program nebo prohlížet si hodnoty proměnných.

Projekt

Programy, vytvořené ve **WinBase602**, mohou hrát dvě role:

- Program může být PROVEDEN, mohou být tedy vykonány akce popsané v programu.
- Program se může stát PROJEKTEM, tedy může poskytovat objekty, která jsou v něm deklarovány, formulářům, menu a SQL příkazům.

Běžící program je vždy současně projektem. Jako projekt lze *nastavit* i program, který právě neběží.

Správa programů

Programy obsažené v aplikaci jsou zobrazeny na řídicím panelu aplikace. Jsou rozlišeny těmito grafickými symboly:

- samostatný program;
- část programu (include) - viz dále;
- program, který je startovním objektem;
- uložený přenos dat (viz kapitola *Přenosy dat v* prvním manuálu).

Programy označené posledním symbolem nepatří do tématu této kapitoly, protože nejsou vytvořeny ve vnitřním jazyce.

Nový program

Chcete-li vytvořit nový program, pak na řídicím panelu vyberte kategorii Programy a proveďte akci **Vytvořit**. Otevře se textový editor, v němž můžete začít psát zdrojový text programu. Před prvním uložením zadáte jméno programu.

Spuštění programu

Chcete-li spustit napsaný program, pak jej označte na řídicím panelu a proveďte akci **Spustit** nebo na jméno programu v seznamu poklepejte. Pokud zdrojový program byl od posledního překladu změněn nebo pokud dosud nebyl úspěšně přeložen, před spuštěním se automaticky vyvolá překlad programu. Dojde-li při tomto překladu k chybě, program nebude vůbec spuštěn. Místo toho se otevře editor a ukáže místo, v němž je chyba.

Pokud jste od posledního překladu programu změnili definici tabulky nebo dotazu, které se v programu používají, nebo některý include, je nutno místo akce **Spustit** použít akci **Překlad a běh**. Tím vyvoláte bezpodmínečný překlad programu před jeho spuštěním.

Modifikace programu

Chcete-li editovat nebo přeložit některý program, pak jej označte na řídicím panelu a proveďte akci **Modifikovat**. Tím otevřete editor s textem programu. Pokud chcete program editovat v samostatném okně, pak při otevírání editoru z řídicího panelu podržte stisknutou klávesu Ctrl. Pro editaci lze také použít externí textový editor, který zvolíte v parametrech **WinBase602** v menu ***Nástroje***. Externí editor se volá pomocí kontextového popup menu v seznamu objektů na řídicím panelu. Nedoporučujeme používat takové externí editory, které by do textu mohly vložit vlastní formátovací znaky.

Chcete-li spustit ladící běh programu, pak použijte akci **Ladit**. Spustit překlad a běh nebo zapnout ladící režim lze také z prostředí textového editoru.

Automatické spuštění programu

Program, označený na řídicím panelu aplikace, může být vybrán jako STARTOVNÍ OBJEKT aplikace. Takto vybraný program se automaticky spustí, jakmile spustíte aplikaci, která jej obsahuje.

Děje se tak ve dvou případech: když aplikaci spustíte ve vývojovém nebo provozním prostředí **WinBase602** pomocí akce **Spustit** (případně použitím tlačítka s běžcem na ovládací liště) nebo pokud spustíte **WinBase602 Runtime** se jménem této aplikace jako parametrem.

Startovní objekt aplikace vyberete zatržením čtverce **Startovní objekt** vpravo u spodního okraje řídicího panelu.

Práva a zámky

Ke spuštění programu je nutno mít právo číst program. K editaci je nutno mít právo přepsat program. Pokud program zrovna edituje některý jiný uživatel v síti počítačů, nelze jej otevřít pro editaci, avšak bude nabídnuta možnost otevřít jej pouze pro čtení.

Pokud dva programy současně pracují se stejnými daty, pak by měly zajistit svůj přístup k nim pomocí *zámků* (popsáno v kapitole *Programování komunikace s databází*) nebo pomocí vhodné úrovně izolace transakcí (popsáno v kapitole o SQL).

Členění programu do částí

Editovat velmi dlouhé programy je nepohodlné. Proto je vhodné je rozdělit do několika samostatně uložených částí a ty editovat nezávisle na sobě. Jedna část pak tvoří HLAVNÍ PROGRAM, zatímco ostatní jsou do ní při překladu INCLUDOVÁNY (odpovídající český termín by zněl *vkládány, zahrnovány* nebo *vtahovány*, ale nepoužívá se).

Každou část založíte z řídicího panelu aplikace jako nový program. Do hlavního programu vložíte direktivy odkazující na ostatní části. Tyto direktivy zajistí, že při překladu hlavního programu se do něj vloží ostatní části - každá na to místo, kde je na ní odkaz. Zápis direktiv je popsán v kapitole o vnitřním programovacím jazyce.

Označení části programu

Na úplný začátek každé části, která se vkládá do jiného programu, napište klíčové slovo

```
INCLUDE
```

a pak pokračujte normálním textem programu. Toto slovo je nezbytné ke správnému označení objektu na řídicím panelu, ke správnému otevírání projektu, a také zamezuje překladu částí při exportu aplikace.

Pokud na řídicím panelu poklepete na jméno části programu, pak se rozdíl od samostatných programů nespustí, ale otevře se na ní editor. Pokoušet se překládat nebo spouštět části, které nejsou hlavním programem, nemá smysl. Při ladění a při lokalizování chyb v programu **WinBase602** otevírá okna s textem jednotlivých částí podle potřeby.

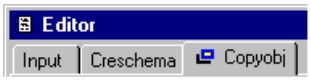
Pokud změníte některou z includovaných částí programu a spustíte běh hlavního programu, pak **WinBase602** tuto situaci rozpozná a hlavní program znovu přeloží.

Prostředí editoru programů

Pro editaci programů se používá stejný editor, jako pro texty uložené v databázi. Jeho ovládání je popsáno v manuálu uživatele **WinBase602**. Zde si pouze povšimneme specifických rysů editace programu.

Při editaci programu na rozdíl od editace textů obsažených v databázových tabulkách nedochází k automatickému rozlamování textu do řádek. Jsou vypnuty také funkce zarovnávání pravého okraje a formátování odstavce. Konce odstavců nejsou označeny žádným zvláštním symbolem.

Různé části programu uložené v samostatných objektech lze otevřít na stránkách editoru a přecházet mezi nimi stiskem kombinace kláves Ctrl+Tab. Před jménem objektu na záložkách přepínajících stránky se může objevit značka indikující, že text je oproti obsahu databáze změněn.



Nástroje pro editaci programu

Kromě běžných editačních kláves (Delete, Home, Page Down, Ctrl+šipky) editor umí (mimo jiné) i tyto klávesové zkratky a kombinace:

Klávesa	*Alternativa*	*Funkce*
Insert		Přechod mezi vkládacím a přepisovacím režimem
F12		Přechod na další záložku

Shift F12		Přechod na předchozí záložku
Ctrl F12		Vložit / zrušit záložku na běžné řádce
Ctrl Shift F12		Vymazat všechny záložky
Ctrl F		Vyhledat
Ctrl H		Nahradit
Ctrl S		Uložit text
Ctrl Delete		Smazat do konce slova
Ctrl Shift L	**Ctrl Y**	Smazat řádek
Alt Backspace	**Ctrl Z**	Odvolat poslední změnu
Alt Shift Backspace	**Ctrl W**	Zopakovat naposled odvolanou změnu
F3	**Ctrl L**	Zopakovat poslední hledání / nahrazování
Ctrl F3		Vyhledat označený text nebo slovo pod kurzorem
F2	**F9**	Vložit / zrušit bod zastavení
F7		Přeložit program
F5		Spustit běh programu

Odsazování

Označený blok textu lze odsazovat a předsazovat pomocí kláves **Tab** a **Shift**+**Tab**.

Označení slova

Slovo v textu lze pohodlně označit poklepáním na ně. Tento rys je však vypnut během ladění programu, kdy se poklepáním vkládají a odstraňují body zastavení.

Záložky

ZÁLOŽKY (angl. *bookmarks*) jsou trvalé značky v textu usnadňující orientaci v rozsáhlejším programu. Stiskem kombinace kláves lze je vkládat na určená místa a přecházet mezi nimi. Řádka, na níž je umístěna záložka, se obarví šedě.

Kontextové menu

Pokud kliknete do textu programu pravým tlačítkem myši, vyvoláte kontextové popup menu, které nabízí řadu užitečných akcí. Z nich si povšimněte možnosti vyhledat slovo ve všech objektech v aplikaci, vyvolat nápovědu nebo otevřít textový editor na includovaný modul.

Návrh faktoru

Z kontextového popup menu lze také vyvolat interaktivní *návrhář faktorů*, dovolující snadno vložit do programu volání funkce nebo metody, případně odkaz na proměnnou nebo vlastnost některého formuláře či jeho složky.

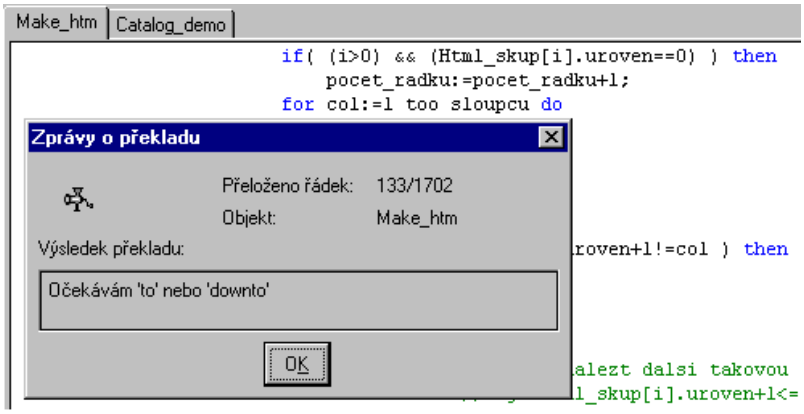
Překlad programu

Překlad programu spustíte příkazem **Překlad** z menu ***Překlad/Běh*** v editoru nebo stiskem ovládacího tlačítka na liště nebo klávesou F7.

Během překladu se na obrazovce objevují průběžné zprávy o počtu přeložených řádek (v rámci překládaného modulu a celkem) a jméno právě překládaného modulu. Pokud při překladu dojde k chybě, pak:

1. překlad se ukončí;
2. v okně na obrazovce se objeví informace o druhu chyby;
3. po stisku tlačítka **OK** se řádka, na níž byla chyba rozpoznána, přemístí do horní části okna;
4. kurzor se umístí na pozici, kde byla rozpoznána chyba.

Zpráva o výsledku překladu



Při hledání místa, kde je v programu chyba, berte v úvahu tato pravidla: Jde-li o syntaktickou chybu, pak kurzor je na prvním znaku nepřípustného symbolu. Je-li chyba sémantické povahy, kurzor může být až na následujícím symbolu, pokud pro rozpoznání chyby překladač musel přečíst další symbol. Pokud je chybný poslední symbol na řádce, může být kurzor až na začátku další řádky.

Interpretace chyb

Vysvětlení chyby může být poněkud zavádějící. Je totiž formulováno z pohledu překladače, který v jisté situaci zjistil, že program už nemůže považovat za správný. Proto při opravování chyby vycházejte v první řadě ze své znalosti programovacího jazyka a jeho popisu v tomto manuálu.

Pokud se zdrojový program skládá z více než jedné části, pak překlad lze spustit pouze na hlavní program. Ostatní části budou do překladu automaticky zahrnuty. Pokud se při překladu objeví chyba v některé z těchto částí, otevře se na ní okno editoru. Pokus o samostatný překlad includované části programu povede k ohlášení syntaktické chyby.

Překlad hlavního programu z includu

Překlad hlavního programu lze spustit i z prostředí editované includované části. Slouží k tomu tlačítko na liště s mlýnkem a hvězdičkou. Ekvivalentní akci provede příkaz **Překlad startovního programu** z menu ***Překlad/Běh***.

Takto kompilovaný hlavní program musí být buď označen jako startovní objekt nebo musí být jediným hlavním programem v aplikaci.

Běh programu

Běh programu se z editoru spouští pomocí položky **Běh** ze submenu ***Překlad/Běh*** nebo ovládacím tlačítkem na liště nebo klávesou F5. Chování programu po spuštění závisí na tom, zda je zapnut ladící režim - viz níže.

Pokud byl zdrojový program od posledního uložení změněn, je před spuštěním programu zapsán do databáze. Je to pojistka pro případ, že by se chybou v programu podařilo přivodit pád aplikace.

Pokusíte-li se spustit dosud nepřeložený program, systém se ho napřed pokusí přeložit. Proto mohou nastat chyby při překladu popsané výše. Pokud překlad neproběhne bez chyby, program se vůbec nespustí.

Pokud se program skládá z více než jedné části, pak lze úspěšně spustit pouze hlavní program. Pokus o samostatné spuštění některé jiné části povede k pokusu o její překlad a ohlášení chyby.

Spuštění hlavního programu z includu

Hlavní program lze spustit i z okna editoru includované části. Slouží k tomu tlačítko na liště s běžcem a hvězdičkou. Ekvivalentní akci provede příkaz **Běh startovního programu** z menu ***Překlad/Běh***.

Takto spouštěný hlavní program musí být buď označen jako startovní objekt nebo musí být jediným hlavním programem v aplikaci. Před spuštěním se program vždy přeloží, aby bylo zaručeno, že se do něj promítnou změny provedené v includované části.

Zahájení běhu programu

Před spuštěním běhu kteréhokoliv programu se nejprve odstraní případné pozůstatky po dřívějších bězích programů. Pokud nějaký program je projektem, pak jim přestane být, jeho globální proměnné zaniknou a zruší se jeho vazba na dynamické knihovny.

Při spuštění se vytvoří nové proměnné programu. Program proto nenalezne v proměnných hodnoty, které v nich zanechal předchozí běh programu. Pak se odstraní menu a tlačítka z ovládací lišty a začnou se provádět příkazy programu.

Spustíte-li běh z prostředí editoru, nebude z obrazovky odstraněno okno editoru. Za běhu programu si pak můžete prohlížet jeho text. Změny, které byste při běhu provedli v textu programu, však nijak neovlivní jeho chování - projeví se až po zastavení běhu, překladu a jeho novém spuštění.

Ukončení běhu programu

Běh programu může skončit jedním ze čtyř způsobů:

- provedením všech příkazů (dosažením konce programu);
- provedením příkazu `Halt;`
- přerušením běhu uživatelem;
- běhovou chybou způsobující ukončení programu.

V posledním případě se na obrazovku vypíše informace o druhu chyby.

Přerušení běhu

Běžící program lze kdykoli **přerušit** klávesovým povelem Ctrl+Break nebo příkazem **Přerušit** ze systémového menu editoru nebo hlavního okna klienta. Je-li na obrazovce modální okno, je systémové menu nepřístupné a lze použít pouze kombinaci kláves.

Je-li běžící program uživatelem přerušen, je jeho běh definitivně ukončen, nelze v něm tedy pokračovat. Jinak je tomu při ladícím běhu - přerušením programu se dostanete do debuggeru a pokračovat v běhu můžete.

Úklid po skončení běhu

Jakmile skončí běh programu (ať už se tak stane kterýmkoli ze čtyř výše uvedených způsobů), systém provede tyto úklidové akce:

- uzavře všechny formuláře na obrazovce;
- odstraní menu hlavního okna **WinBase602** dosazené programem a vrátí místo něj standardní menu;
- vrátí původní tlačítka na ovládací lištu;
- uzavře všechny kurzory neuzavřené programem (včetně ODBC kurzorů) a vydá o tom upozornění;
- uzavře všechny soubory neuzavřené programem a vydá o tom upozornění;
- odemkne všechny záznamy zamčené touto aplikací (kromě textu editovaného programu, je-li editor otevřen).

Důvod úklidu

Úklid se provádí zejména kvůli programům, které skončí běhovou chybou. Správně napsané aplikace nespoléhají na systémový úklid, ale samy odstraňují zámky a uzavírají soubory a kurzory, jakmile je nepotřebují.

Po ukončení běhu program zůstane projektem (až do spuštění jiného programu nebo nastavení jiného projektu). Díky tomu nezaniknou jeho proměnné ani se neodinstalují knihovny DLL, které program použil. Tyto prostředky pak mohou používat ostatní objekty aplikace.

Běhové chyby

V závislosti na druhu běhové chyby dojde při jejím výskytu k jedné ze dvou reakcí:

- pokračuje se v provádění programu, avšak některý mezivýsledek nebo stav programu může být jiný než očekávaný;
- ukončí se běh programu a oznámí se druh chyby.

Aritmetické chyby

První reakce nastává u většiny aritmetických chyb (výjimky jsou uvedeny níže). Výsledek chybné aritmetické operace pak není definován. Typickým případem takové chyby je aritmetická operace, jejíž výsledek nepatří do určeného typu (např. násobení 123456789 * 123456789 v typu **Integer**).

Nelze provést akce prezentační vrstvy

V programu se pokračuje např. i tehdy, pokud nelze provést některou z akcí uživatelského rozhraní, např. otevřít menu nebo formulář. Program se o takové situaci dozví z výsledku funkce, která měla akci provést. Je tedy na programátorovi, aby adekvátně reagoval na vzniklou situaci.

Pracuje-li uživatel aplikace za běhu programu s formulářem a provede-li nedovolenou akci (např. překročí svá přístupová práva), je tato skutečnost ošetřena již ve formuláři (například chybovým oknem) a na běh programu nemá vliv.

Databázové operace

Databázové operace vyvolávané z programu lze rozdělit do dvou skupin.

- První skupinu tvoří ty operace, které se provádějí voláním standardní funkce a které vrací příznak, zda došlo k chybě či nikoli. Běh programu se při chybě neukončuje, program však musí na chybu adekvátně reagovat - nemůže například pracovat s obsahem kurzoru, který se nepodařilo otevřít.
- Druhou skupinu tvoří ty databázové operace, které jsou vyvolány provedením přiřazovacího příkazu nebo vyčíslením výrazu. Ukončují program po chybě právě tehdy, pokud není nastaveno tzv. MASKOVÁNÍ CHYB pomocí funkce **`Err_mask`**. Jsou-li chyby maskovány, program může zjistit jejich výskyt voláním funkce **`Sz_error`** nebo **`Signalize`**.

Další důvody ukončení programu

K ukončení programu dojde dále při výskytu těchto chyb:

- dělení nulou (v celočíselné i reálné aritmetice);
- přetečení v typu **Real**;
- počítání odmocniny ze záporného čísla;
- počítání logaritmu z nekladného čísla;
- vyčerpání operační paměti (při nekonečném alokování dynamické paměti nebo při nekonečné rekurzi);
- chyba **Příliš složitý výraz** - vzniká buď při vyhodnocování extrémně složitého výrazu (vyskytuje zcela výjimečně a je to chyba systematická tj. nemůže se objevit náhodně v programu, který již jednou fungoval), anebo při příliš hluboké rekurzi.

Panel se jmény sloupců

Pomocí příkazu **Sloupce** v menu editoru programů lze otevřít plovoucí okno nabízející seznamy jmen sloupců. V horní části okna lze zvolit zobrazení tabulek nebo dotazů a vybrat určitou tabulku nebo dotaz uložený v databázi. Ve spodní části se pak objeví seznam sloupců zvolené tabulky nebo dotazu obsahující vedle jména také případnou délku řetězce nebo počet hodnot multiatributu (znak + indikuje možnost překročit tento počet). Označíte-li některý sloupec, vypíše se pod seznamem jeho typ.

Jméno vybraného sloupce lze pomocí kombinace kláves Ctrl+C nebo Ctrl+Ins přenést do schránky a pak vložit do textu programu.

Projekty

Slovem PROJEKT se označuje mechanismus, jak umožnit formulářům, menu a sestavám využívat vše, co je deklarováno v programu. Projekt umožňuje také dotazům a příkazům v SQL využívat proměnné z programu jako tzv. hostitelské proměnné.

Typické příklady takového poskytnutí vnitřních deklarací programu jiným objektům jsou:

- ve formuláři se edituje hodnota proměnné deklarované v programu (tzv. PROMĚNNÉ PROJEKTU);
- formulář při stisku tlačítka volá proceduru deklarovanou v programu;
- formulář volá funkcí programu k vyhodnocení podmínky viditelnosti své složky;
- menu volá procedury deklarované v programu;
- menu otevírá formuláře přesměrované na kurzory deklarované a otevřené v programu;
- otevíraný dotaz využívá hodnotu jedné proměnné v podmínce WHERE a zapisuje nalezená data do jiné proměnné uvedené v klauzuli INTO.

Program, jehož jediným účelem je být projektem, má zpravidla za deklaracemi *tělo* skládající se pouze ze dvou řádek:

```
begin
end.
```

Takový program nebývá nikdy spuštěn, pouze poskytuje své deklarace.

Pokud program běží, pak hraje roli projektu, takže ostatní objekty mohou využívat jeho deklarace.

Zpřístupnění deklarací z programu, který právě neběží, pro ostatní objekty aplikace, nazýváme NASTAVENÍM PROJEKTU.

Nastavení projektu

Nastavení projektu musíte provést vždy, když chcete, aby určitý program poskytl své deklarace ostatním objektům aplikace v době, kdy neběží.

Pokud například některý dotaz nebo formulář používá cokoliv, co je deklarováno v programu, pak bez nastavení projektu jej nebude možno mimo program otevřít.

Nastavení projektu provedete takto:

1. na řídicím panelu aplikace zobrazíte programy;
2. označíte ten program, který má poskytovat své deklarace;
3. stisknete tlačítko **Nastavit projekt**.

Pokud označený program byl od posledního překladu změněn, pak se jej **WinBase602** pokusí přeložit. Po úspěšném překladu bude projekt nastaven, při neúspěchu nebude nastaven žádný projekt.

Hodnoty proměnných

Nastavením projektu vzniknou všechny proměnné deklarované v programu. Na začátku mají nulové hodnoty. Tyto proměnné budou existovat a podrží si své hodnoty po celou dobu, po kterou bude projekt nastaven, jejich existence není tedy vázaná na dobu, po níž běží určitý program. Proměnné zaniknou až tehdy, když bude nastaven jiný projekt, spuštěn jiný program nebo uzavřena aplikace.

Automatické nastavení projektu

Po otevření aplikace, která obsahuje jediný program, se tento program automaticky nastaví jako projekt - kromě případu, kdy program není dosud přeložen a dojde k chybě při jeho překladu.

V návrhu formuláře, sestavy nebo menu může být uveden program, jehož deklarace tento objekt využívá. V tom případě se projekt automaticky nastaví při otevření objektu. Objekt nelze používat při běhu jiného programu.

Monitorování chyb a událostí klienta

Při vývoji aplikací je užitečné průběžně získávat informace o dění a o chybách v klientovi i na serveru. Tato informace se vypisují do plovoucího okna, které lze otevřít pomocí příkazu **Monitor** v menu ***Nástroje***. Pokud se vyvíjená aplikace nechová podle očekávání, doporučujeme sledovat zejména první stránku s chybami klienta.

Na stránce **Klient** je jméno, pod nimž je uživatel přihlášen na server, a množství paměti využívané klientem. Pokud množství paměti během práce aplikace trvale roste, svědčí to o chybě v aplikaci. Tyto údaje se neaktualizují průběžně, ale až po stisku tlačítka **Aktualizovat**. Po nimi je seznam posledních událostí a chyb klienta v pořadí výskytu. Nevypisují se sem chyby, které klient vyvolal na serveru.

Chyby a událostí, k nimž dochází na serveru, lze sledovat na stránce **Server**. Nové informace se do okna přenášení vždy po stisku tlačítka **Aktualizovat**.

Na stránce **Stav serveru** lze mezi interními informacemi najít i počet kurzorů otevřených na serveru. Pokud v průběhu práce aplikace tento počet neustále roste, svědčí to o chybě v aplikaci.

Ladění programů ve vnitřním jazyce

Součástí vývojového prostředí WinBase602 **je ladící systém (krátce** *debugger***) usnadňující nalezení a odstranění chyb v aplikacích využívajících programy ve vnitřním jazyce.**

Předtím, než popíšeme ovládání debuggeru, zamysleme se nad postupy používanými při vývoji programů a nad tím, k čemu používat a k čemu nepoužívat ladění.

Malá reflexe:

Když se autor těchto řádek učil programovat, nosil krabici s děrnými štítky do výpočetního střediska a na provedení programu čekal řádově dny. Mohl si přibližně spočítat, kolikrát stihne program spustit před termínem stanoveným pro jeho dokončení. Snažil se proto vkládat do počítače pouze takový program, o němž na základě svých schopností usoudil, že je napsán správně a bude tedy dělat přesně to, co má.

Jakmile však zasedl k terminálu a později k osobnímu počítači, byl vystaven svodům okamžité odezvy: “Proč dlouze přemýšlet nad programem, když stačí prostě zkusit, zda funguje? Pokud nebude fungovat, snad mi jeho chování napoví, co je třeba změnit, a pak uskutečním další pokus.”

Tato změna stylu práce činí z počítače partnera při vývoji programu. Jejím přínosem je, že ušetří programátorovi čas věnovaný hledání chybějících středníků ve zdrojovém programu a podobných bezvýznamných syntaktických chyb. Katastrofálním vedlejším efektem je však to, že programátor může ztratit myšlenkový nadhled nad programem a program se z přehledného soustrojí stane konglomerátem záplat nad dosud objevenými chybami.

Tato cesta vede ke dvěma možným koncům. Buď se program, který se ukázal složitější než ty předchozí, nepodaří odladit vůbec. Anebo se z programu po jistém čase podaří odstranit všechny projevy chyb na všech testovacích úlohách, ale nedokážeme zaručit, jak se bude chovat na úlohách jiných.

Spolehlivý program vznikne pouze tehdy, pokud bude jeho autor přesně vědět, proč spolehlivý být musí. Autorovi musí být jasné, že program nemá jinou možnost než udělat to, co se od něj očekává. Ověřit to, znamená přesvědčit se o správnosti použitého algoritmu a o jeho bezchybném převedení do podoby programu.

Algoritmy se dají matematickými prostředky dokázat, netriviální programy zpravidla nikoli. Proto přesvědčení autora o správnosti programu spočívá v méně formální rovině, než je matematický důkaz.

Za nejosvědčenější praktickou metodu odhalování chyb v algoritmech považuji nikoli ladění a testování, ale detailní vyložení algoritmu některému kolegovi. Budete-li se pokoušet přesvědčit pochybujícího kolegu o správnosti vadného algoritmu, obvykle shledáte, že nedokážete přesvědčit ani sami sebe.

Databázové aplikace obvykle neoplývají zvlášť složitými algoritmy, zato často zahrnují nepřehledné množství kódu obsluhujícího interakce s databází a s uživatelem. Klíčem k zvládnutí takovýchto programů je jejich rozčlenění do přirozených částí s jasně definovaným rozhraním. Podaří-li se vám vytvořit úplný popis toho, co očekáváte od každé části programu, pak se ověření správnosti programu podstatně zjednoduší: postačí ověřit správnost jednotlivých částí a pak správnost jejich interakcí.

Než začnete ladit program, uvědomte si: provoz na počítači může prokázat, že váš program nefunguje, ale neprokáže, že je správný. Vysokou míru jistoty o správnosti programu můžete získat pouze jasným chápáním jeho vnitřní logiky.

Provoz aplikace v ladícím režimu

Před spuštění laděné aplikace je nutno zapnout tzv. LADÍCÍ REŽIM. K tomu slouží příkaz **Ladící režim** v menu ***Parametry*** nebo tlačítko na ovládací liště. Aplikace zůstane v ladícím režimu, dokud není explicitně vypnut. Ladící režim lze zapnout i tlačítkem **Ladit** na řídicím panelu nebo z menu **Překlad/běh** v editoru nebo tlačítkem na liště editoru během editace programu.

Pokud v ladícím režimu spustíte běh programu, zastaví se před provedením první řádky programu a vstoupíte do prostředí debuggeru. V něm můžete program krokovat, prohlížet si a měnit hodnoty proměnných a provádět další, níže popsané akce.

Během ladění nelze program editovat - editace bude dovolena, až ladící program doběhne.

Procedury a funkce volané z formulářů a menu lze ladit, aniž by byl spuštěn běh programu - stačí do nich umístit bod zastavení.

Jak pracuje debugger

Debugger umožňuje provádět program po částech a průběžně kontrolovat jeho stav. Tak lze odhalit, že se program v určitém okamžiku vydá jinou než očekávanou cestou nebo že v některé proměnné je jiná než očekávaná hodnota.

Během ladění programu budete přecházet mezi dvěma stavy:

- stavem, kdy je aktivní laděná aplikace
- stavem, kdy je aplikace pozastavena a aktivní je debugger.

Rozlišení těchto dvou stavů je klíčem ke správnému ovládání debuggeru.

Stav, kdy je aktivní aplikace

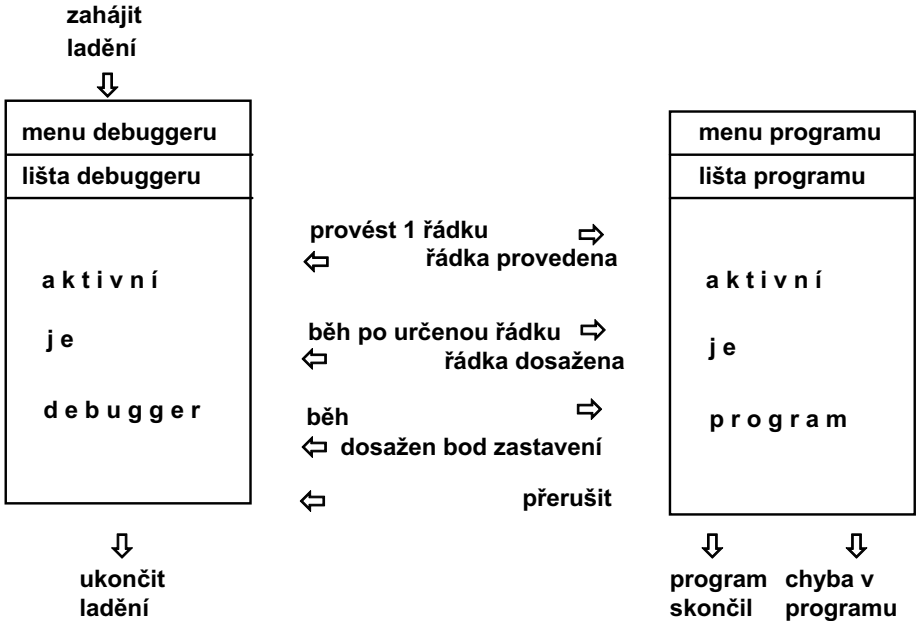
Je-li aktivní vaše aplikace, pak je na obrazovce její menu (pokud nějaké má) a případně i ovládací lišta některého formuláře. S aplikací můžete pracovat tak, jako při normálním běhu, až do té doby, dokud se nezmocní aktivity debugger.

Stav, kdy je aktivní debugger

Je-li aktivní debugger, je na obrazovce menu a ovládací lišta debuggeru a do popředí se dostalo okno editoru, v němž vidíte barevně vyznačenou řádku programu, která se právě má provést. Laděný program je pozastaven, můžete analyzovat jeho stav.

V obou popsaných stavech lze přecházet mezi okny na obrazovce, listovat v nich atd. V **každém okamžiku je však nutno si uvědomovat, ve kterém stavu se nacházíte. Nejsnáze to vždy zjistíte pohledem na menu.**

Přepínání mezi aplikací a debuggerem



Barevné odlišení významných řádků

Je-li debugger aktivní, program stojí a v okně editoru je označen řádek s příkazem, který se vykoná v dalším kroku. Tento řádek je pro názornost barevně odlišen.

Umístíte-li do textu programu na nějaké řádky body zastavení (viz níže), budou zvýrazněny jinou barvou, aby se snadno odlišily od ostatních řádků programu.

Třetí odlišnou barvou bude označený ten řádek, který je kombinací dvou předchozích případů - program stojí na řádku, na němž je umístěn bod zastavení.

Nastavení barev

Barvy textu a pozadí řádků je možno nastavit tak, aby vyhovovaly typu monitoru, uživatelově vkusu a možnostem grafické karty. Nastavení se provádí v globálních parametrech **WinBase602**.

Příklad laděného programu

V následujících sekcích popíšeme základní obraty, které vám debugger dovolí využívat při ladění. Budeme je zároveň demonstrovat na příkladu tohoto programu:

```
VAR i     : Integer;
    vstup : string[10];
    konec : Boolean;

PROCEDURE výpis(číslo : Integer);
VAR odmocnina : Real; zpráva : String[50];
BEGIN
  odmocnina := Sqrt(číslo);
  zpráva := 'Druhá odmocnina ' + Int2str(číslo) +
            ' je ' + Real2str(odmocnina, -4);
  Info_box('Výsledek', zpráva);
END;

BEGIN
  vstup := '';
  Konec := FALSE;
  REPEAT
    IF Input_box('Zadejte celé číslo:', vstup, 10) THEN
    BEGIN
      i := Str2int(vstup);
      IF i <> NONEINTEGER
        THEN výpis(i);
    END ELSE
      konec := TRUE;
  UNTIL konec;
END.
```

Program cyklicky žádá o zadání celého čísla a vypisuje na obrazovku jeho odmocninu. Program skončí, když ve vstupním okně stisknete tlačítko **Zrušit**. Pokud je vám tento

program nesrozumitelný, prostudujte napřed kapitolu popisující syntaxi a sémantiku vnitřního programovacího jazyka.

Řízení běhu programu

Program spuštěný v ladícím režimu se zastaví na své první řádce. Nyní (a také po každém dalším zastavení) můžete volit, jak pokračovat.

Při spuštění programu v ladícím režimu se ozve jedno pípnutí za každý bod zastavení, který byl vyznačen v editoru, ale nelze jej použít, protože na jeho řádce není žádný proveditelný kód.

Řádka programu, která má být provedena jako další, je vždy barevně odlišena.

Krokování programu

Krokováním se rozumí provádění programu po jedné zdrojové řádce.

Pokud na řádce, kterou se zrovna chystáte provést, je volání podprogramu, pak můžete buď vstoupit *dovnitř* podprogramu (angl. *trace into*), anebo podprogram provést jako jeden příkaz a zastavit se až po návratu z něj (angl. *step over*).

Krok dovnitř podprogramu provedete příkazem **Krok dovnitř** z menu ***Běh*** debuggeru nebo stisknutím tlačítka na ovládací liště, případně stisknutím klávesy F11 resp. F7.

Vstoupit můžete jen do těch podprogramů, které jsou součástí programu. Debuggerem **nelze** vstupovat dovnitř standardních podprogramů, jako jsou například `Input_box`, `Str2int`, `Sqrt` nebo `Info_box`.

Krok přes celý podprogram provedete příkazem **Krok přes** z menu ***Běh*** debuggeru nebo stisknutím tlačítka na ovládací liště, případně stisknutím klávesy F10 resp. F8.

Indikace prováděných kroků

Během krokování můžete sledovat, jak postupuje vpřed barevně odlišená řádka ukazující na místo, které se bude v dalším kroku provádět. Stručně budete říkat, že program "je" na řádce s odlišnou barvou.

Pokud je program na řádce, na níž není volání podprogramu, pak oba druhy kroků provedou tutéž akci.

Krokování na příkladu

Vyzkoušejte si krokování na výše uvedeném programu. Poté, co zahájíte ladění programu, je program na řádce obsahující příkaz `vstup:=''`. Provedením jednoho kroku se vykoná toto přiřazení a program se dostane na následující řádku. Pod dalším kroku se dostane až na řádku začínající `Input_box...`, protože na řádce se slovem `REPEAT` není co provádět.

Provedete-li další krok, pak se na obrazovce objeví okno pro vstup údaje a do debuggeru se zatím nevrátíte. Program je totiž uvnitř provádění funkce `Input_box`. Zadejte nějaké

celé číslo a stiskněte tlačítko **OK**. Tím ukončíte funkci `Input_box` a program se vzápětí zastaví na následující řádce.

Krokujte dále až do řádky, na níž je volání procedury `výpis.` Na ní proveďte krok dovnitř. Tak se program dostane na první příkazovou řádku této procedury.

Během krokování uvnitř procedury `výpis` se při zavolání procedury `Info_box` vrátíte do debuggeru opět až poté, co stisknete tlačítko **OK** v informačním okně.

Po provedení všech řádek podprogramu se krokování vrátí do hlavního programu. V dalším oběhu cyklu již nevstupujte dovnitř podprogramu `výpis`, ale proveďte jej jako celek. Program se nezastaví na žádné jeho řádce, ale až za místem jeho volání.

Program ukončete tak, že při dalším vstupu čísla stiskněte tlačítko **Zrušit**. V dalších krocích opustíte cyklus a po provedení příkazu `END` na poslední řádce ladění skončí.

Běh po kurzor

Chcete-li nechat program běžet bez zastavení až po určitou řádku a tam jej zastavit, pak postupujte podle následujícího návodu.

1. Umístěte v laděném programu kurzor na tu řádku, před jejímž provedením se běh programu má zastavit a řízení se má odevzdat debuggeru.

2. Proveďte z menu ***Běh*** debuggeru příkaz **Běh po kurzor** nebo stiskněte klávesu F4 nebo použijte tlačítko na liště.

V příkladu...

Spusťte výše uvedený program v ladícím režimu, přemístěte kurzor na druhou řádku v proceduře `výpis` a proveďte z menu ***Běh*** příkaz **Běh po kurzor.** Program poběží a zastaví se až uvnitř procedury výpis, před provedením druhého příkazu.

Běh do opuštění podprogramu

Chcete-li nechat program běžet až do konce podprogramu a pak jej zastavit, můžete pro to využít zvláštní funkci debuggeru. Buď z menu ***Běh*** proveďte příkaz **Návrat z podprogramu**, nebo použijte klávesový povel Ctrl+F8, anebo stiskněte tlačítko na ovládací liště. Tato funkce debuggeru a položka menu je aktivní pouze tehdy, pokud program je uvnitř nějakého podprogramu.

V příkladu...

Ve výše uvedeném programu vstupte do procedury `výpis` a pak proveďte výše popsanou akci. Zbytek podprogramu bude vykonán bez zastavení a běh se zastaví na příkazu následujícím za voláním této procedury.

Kontinuální pokračování běhu

Chcete-li nechat program dále volně běžet, pak z menu ***Běh*** debuggeru proveďte příkaz **Běh** nebo stiskněte klávesu F5 nebo stiskněte tlačítko na liště. Takto běžící program se zastaví buď tehdy, až skončí, nebo pokud narazí na bod zastavení (viz dále).

Chcete-li ukončit ladění programu a nenechat program doběhnout do konce, pak z menu ***Běh*** debuggeru spusťte příkaz **Ukončit ladění**.

Ukončení ladění

K ukončení režimu ladění před koncem vlastního programu slouží příkaz menu **Ukončit ladění** nebo klávesy Shift+F5. Tímto se zároveň ukončí běh programu.

Situace je komplikovanější při vnořování běhu programů. Pokud program otevře formulář a akce provedená uživatelem ve formuláři zavolá podprogram, který bude přerušen debuggerem, pak lze pouze ukončit vnořený běh podprogramu, ale nikoli běh hlavního programu. Chcete-li v takovém případě ukončit hlavní program, vložte bod zastavení do hlavního programu na místo, kam program dospěje po dokončení vnořeného běhu, a až pak proveďte popsanou akci

Přerušení při ladění programu

Pokud laděný program běží, pak klávesovým povelem Ctrl+Break nebo provedením příkazu Přerušit **ze systémového menu jej neukončíte, nýbrž pouze pozastavíte a přejdete do debuggeru. V běhu pak můžete libovolně pokračovat.**

Pokud přerušíte program uvnitř nějaké standardní procedury nebo funkce, k přechodu do debuggeru dojde až po jejich opuštění, na řádce následující za jich voláním. Většina standardních podprogramů končí velmi záhy po zavolání, takže chování debuggeru odpovídá očekávání.

Existují však standardní podprogramy, které trvají tak dlouho, dokud uživatel neprovede určitou akci. Většina z nich je při přerušení ukončena předčasně. Platí to o proceduře **`Info_box`** a funkcích **`Yesno_box`**, **`Input_box`** a **`Get_ext_message`**. Přerušená funkce **`Yesno_box`** a **`Input_box`** vrátí FALSE, funkce **`Get_ext_message`** vrátí TRUE a v jejím výstupním parametru bude hodnota -2.

Body zastavení v laděném programu

Na řádku laděného programu lze umístit tzv. BOD ZASTAVENÍ (angl. *Breakpoint*). Kdykoli program při svém běhu na tuto řádku vstoupí, běh se přeruší a aktivuje se debugger. Poté si může např. prohlédnout hodnoty proměnných v programu a buď pokračovat v běhu programu nebo jej například krokovat.

Bod zastavení umístíte tak, že na řádku poklepete levým tlačítkem myši nebo z popup menu provedete příkaz **Vložit / zrušit bod zastavení**, anebo tak, že na řádku najedete kurzorem a buď z menu ***Běh*** provedete příkaz **Bod zastavení** nebo stisknete klávesu F9 resp. F2 nebo použijete tlačítko na liště. Řádka, na níž je bod zastavení, bude barevně

odlišena. Pokud stejnou akci zopakujete, bod zastavení z této řádky odstraníte. Umístění bodů zastavení zůstává zachováno při opakovaném spouštění laděného programu, body zastavení však zmizí při uzavření editoru s textem programu.

Využití bodů zastavení

Body zastavení využijete zejména tehdy, pokud hodláte ladit pouze určitou část programu a nechcete krokovat od začátku programu až k ní. Po vložení bodu zastavení spustíte program a necháte jej volně běžet. Debugger se opět aktivuje v okamžiku, kdy program vstoupí na bod zastavení.

Body zastavení jsou nezbytné, pokud ladíte podprogramy volané z formulářů a menu. Uvnitř těchto podprogramů se nelze zastavit jinak než pomocí bodů zastavení.

Bod zastavení lze vložit i tehdy, když debugger není zrovna aktivní. Protože není k dispozici menu ani lišta debuggeru, využijte popup menu na řádce zdrojového programu.

Pozor !

Nedávejte body zastavení do podprogramů, které se volají při vyhodnocování podmínek aktivity nebo zatržení položek menu. Přerušení těchto akcí uvede ***Windows*** ve zmatek a může vést až k pádu systému.

Během ladění nelze vložit bod zastavení na řádku, na níž není žádný proveditelný kód. Pokud program neběží, takovýto bod zastavení sice vložit lze, ale nikdy se neuplatní.

V příkladu...

Ve výše uvedeném programu zkuste umístit bod zastavení na řádku s příkazem **`konec := TRUE`**. Pak nechte program běžet. Pracujte s ním tak dlouho, jak chcete. Jakmile ve vstupním okně stisknete **Zrušit**, program vstoupí na řádku s bodem zastavení a přejdete do debuggeru. Poté můžete program krokovat a sledovat, jak opouští cyklus a končí.

Persistence bodů zastavení

Umístění bodů zastavení si editor pamatuje a body zastavení nastavené při jednom běhu programu zůstanou funkční i při dalším běhu. Body zastavení se vymažou až při uzavření editoru.

Během ladění lze vložit bod zastavení pouze na takovou řádku, na níž je akce, před jejímž provedením se lze zastavit. Pokud zrovna neladíte, pak ladící systém nemůže tuto podmínku zkontrolovat, a proto dovolí umístit bod zastavení na kteroukoli neprázdnou řádku. Při spuštění ladícího běhu však ověří, zda body zastavení jsou umístěny správně, a pro každý nepoužitelný bod zastavení jednou pípne.

Ladění programu řízeného událostmi

Pokud program ve WinBase602 **obsahuje cyklus volající funkci** `Get_ext_message` **a volící jednu z mnoha akcí na základě čísla obdržené zprávy, nedoporučujeme umísťovat bud zastavení do tohoto cyklu nebo jej krokovat.**

Přechody mezi aktivitou laděného programu a debuggerem totiž generují řadu bezvýznamných notifikačních zpráv, které byste takto odchytávali donekonečna. Proto umístěte bod zastavení do programu až za rozpoznání té zprávy nebo zpráv, které vás zajímají.

Dokud je program uvnitř funkce **`Get_ext_message`** (je tam pokaždé, když čeká na další akci uživatele, tedy "skoro stále"), je aktivní laděný program, nikoli debugger. Nemůžete proto například prohlížet si proměnné. Čekání na zprávu se dá přerušit stiskem **Ctrl** + **Break**.

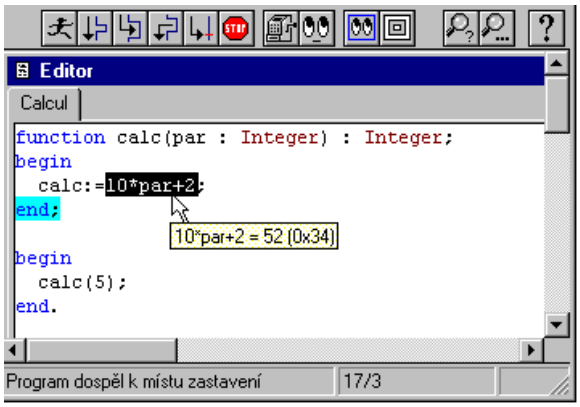
Práce debuggeru s daty

Debugger umožňuje prohlížet si a modifikovat hodnoty proměnných. Všechny níže popsané akce se týkají stavu, kdy je běh aplikace pozastaven a aktivní je debugger.

Rychlé zobrazení hodnoty proměnné nebo výrazu

Pokud umístíte kurzor myši nad proměnnou nestrukturovaného typu nebo typu řetězec, zhruba po sekundě se otevře popup okénko s hodnotou proměnné.

Chcete-li zjistit hodnotu výrazu nebo proměnné zpřístupněné složitěji, než pouze jménem (např. složky pole), pak označte úsek zdrojového programu, který se má vyhodnotit, a ponechte nad nim kurzor.



Pokud takto zvýrazníte volání funkce, funkce se zavolá a vyhodnotí. Jisté riziko skýtají funkce bez parametrů, které se vyhodnotí, kdykoli se kurzor ocitne nad jejich jménem. Pokud by tyto funkce měly vedlejší efekty, mohly by interferovat s funkcí laděného programu.

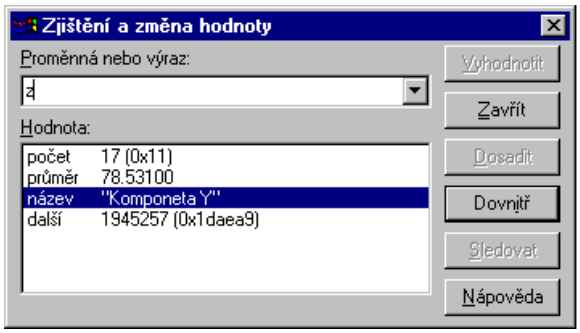
Vyhodnocení proměnné

Okno pro prohlížení hodnot proměnných a vyhodnocování libovolných výrazů se otevírá příkazem **Ukázat / změnit** z menu ***Data***, z kontextového popup menu editoru, tlačítkem na ovládací liště nebo stiskem kláves Shift+F9 resp. Ctrl+F5.

Pokud při otevírání tohoto okna je kurzor editoru uvnitř nějakého identifikátoru nebo pokud je označený blok, pak se tento text přenese do okna a vyhodnotí se. Proto je výhodné před otevřením okna kliknout na proměnnou, o kterou se zajímáte, nebo označit výraz, který chcete vyhodnotit.

V okně můžete do pole **Proměnná nebo výraz** zapsat libovolný výraz a vyhodnotit jej stiskem tlačítka **Vyhodnotit**. Způsob zobrazení hodnoty závisí na tom, zda je jednoduchého nebo strukturovaného typu.

Prohlížení hodnoty typu záznam



Strukturované hodnoty se vypisují po jednotlivých složkách (kromě krátkých znakových řetězců). Stiskem tlačítka **Dovnitř** lze přejít k zobrazení hodnoty konkrétní složky.

Pokud zjišťujete hodnotu databázového sloupce použitého uvnitř příkazu `WITH`, je nutno do okna napsat celý přístup ke sloupci, tedy jméno tabulky nebo kurzoru, v hranatých závorkách číslo záznamu a za tečkou jméno sloupce.

Pozice kurzoru při prohlížení proměnných

Při prohlížení si hodnot proměnných se kurzor musí nacházet v té části programu, v níž je proměnná definována. To je velmi důležité, protože ve dvou rozpracovaných podprogramech mohou být dvě stejně pojmenované proměnné a debugger pouze na základě polohy kurzoru pozná, kterou z nich míníte.

Prohlížení obsahu tabulek a kurzorů

Vyhodnocením jména tabulky deklarované v programu získáte její číslo. Vyhodnocením identifikátoru kurzoru (pevného nebo proměnného) získáte jeho číslo, je-li otevřen, nebo hodnotu 0xFFFF nebo 0, není-li otevřen.

Stiskem tlačítka **Dovnitř** můžete otevřít standardní formulář do obsahu této tabulky nebo kurzoru. Formulář se otevře v plovoucím okně, které lze na obrazovce ponechat i po zavření dialogu na vyhodnocení proměnné. Nedoporučujeme však ponechávat otevřené

okno s obsahem kurzoru po zavření tohoto kurzoru - jeho další obsah by byl nedefinovaný.

Změna hodnoty proměnné

Pokud chcete změnit hodnotu zobrazené proměnné, novou hodnotu zapište o pole **Hodnota** v dialogovém okně (případný obsah pole předtím smažte) a stiskněte tlačítko **Dosadit**. Ihned po osazení se na druhé řádce vypíše aktuální hodnota proměnné.

V příkladu...

Vyzkoušejte si na výše uvedeném příkladu programu, jaké jsou hodnoty jeho proměnných a jak se v průběhu krokování programu mění. Vyzkoušejte také, co se stane, pokud v průběhu práce s programem změníte hodnotu proměnné `konec` na TRUE.

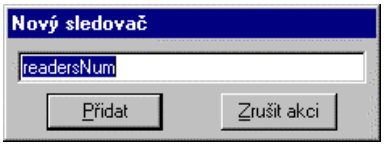
Sledovače

Potřebujete-li po určitou dobu průběžně sledovat hodnotu proměnné nebo výrazu, je nejpohodlnější vytvořit tzv. SLEDOVAČ, tj. umístit proměnnou či výraz do zvláštního okna, v němž se po každém kroku aktualizuje výpis hodnoty.

Vytvoření sledovače

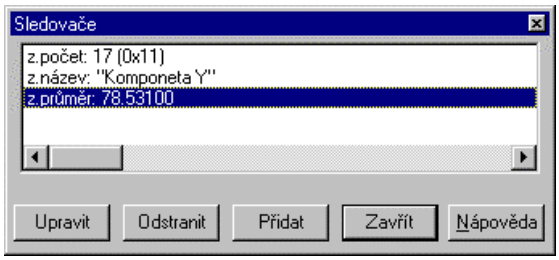
Sledovač vytvoříte příkazem **Přidat sledovač** z menu ***Data***, z kontextového popup menu editoru, tlačítkem na ovládací liště nebo klávesovým povelem Ctrl+F7. Na obrazovce se objeví dialogové okno, v němž vyplníte identifikátor proměnné nebo výraz, který se má objevit i se svou hodnotou ve sledovači. Zadávání proměnně či výrazu urychlit tak, že je předem označíte jako blok nebo na nich umístíte kurzor editoru.

Přidání sledovače



Sledovače jsou umístěny v plovoucím okně. Pokud jej chcete z obrazovky odstranit nebo na obrazovku vrátit zpět, stiskněte tlačítko na liště nebo proveďte z menu ***Data*** příkaz **Okno se sledovači**. Zavřením a opětovným otevřením okna se sledovači se jeho obsah neztratí.

Okno se sledovači



Tlačítka v okně se sledovači

Tlačítkem **Upravit** můžete změnit označený sledovač, např. místo jedné proměnné nechat sledovat jinou. Tlačítkem **Odstranit** odstraníte označený sledovač. Tlačítko **Přidat** umožňuje přidávat nový sledovač a tlačítko **Zavřít** zavírá okno se sledovači.

Sledovače neumožňují měnit hodnoty sledovaných proměnných.

V příkladu...

V příkladu programu můžete po zahájení ladění umístit všechny proměnné do okna se sledovači. Jak budete procházet programem budou se měnit jejich hodnoty. Pokud nebudete uvnitř procedury **`výpis`**, nebudete vidět hodnoty jejích lokálních proměnných, protože mimo tuto proceduru neexistují.

Prohlížení zásobníku vnořených podprogramů

Pokud ladíte program zahrnující volání podprogramů, pak se během práce s debuggerem můžete podívat, které podprogramy jsou rozpracovány, odkud jsou volány a jaké jsou hodnoty jejich parametrů.

Přehled rozpracovaných podprogramů otevřete tlačítkem na liště nebo příkazem **Podprogramy** v menu ***Běh***. Každá řádka v okně odpovídá jednomu rozpracovanému podprogramu. Hlavní program je na první řádce, podprogram, v němž byl přerušen běh, na poslední. Za jménem každého podprogramu jsou uvedeny hodnoty jeho parametrů oddělené čárkami. Parametry předávané referencí mají před hodnotou uvedeno slovo VAR. Pokud se podprogramu předává struktura, je její hodnota nahrazena hvězdičkou.

Místo volání podprogramů

Chcete-li vidět, na které řádce byl z hlavního programu nebo z podprogramu zavolán podprogram uvedený ve výčtu pod ním, poklepejte na něj nebo jej označte a stiskněte tlačítko **Ukázat.** Kurzor se ve zdrojovém textu programu nastaví na tu řádku, na níž je přerušeno provádění vybraného podprogramu.

Ladění programu složeného z více částí

Při ladění programu složených z více includovaných částí je každá část na zvláštní stránce editoru. Další části můžete otevírat pomocí menu ***Moduly***, v němž jsou uvedena jména všech částí laděného programu. V mnoha případech však není nutno toto submenu používat. Při krokování nebo při zastavení programu v kterékoli části se její text otevře v editoru (není-li již otevřen) a aktivuje se. Menu ***Moduly*** využijete zejména k tomu, abyste mohli otevřít ty části programu, do nichž chcete umístit body zastavení.

Ladění procedur uložených na serveru a triggerů

Procedury uložené na serveru a triggery se dají ladit podobným způsobem jako klientské programy ve vnitřní jazyce **WinBase602**, existuje však i několik rozdílů.

Hlavní princip:	**Při ladění spolupracují dva klienti stejného serveru - laděný klient, který volá uložené procedury nebo spouští triggery, a ladící klient, který může krokovat a sledovat aktivitu laděného klienta.**

Jak postupovat při ladění?

Ladění se zahájí tak, že ladící klient umístí do procedury nebo triggeru bod zastavení a laděný klient se na něm zastaví. Postupujte takto:

- Připojte na stejný server dva klienty a v obou otevřete stejnou aplikaci. Nehraje roli, zda oba klienti budou na stejném nebo na různých počítačích.
- V ladícím klientovi otevřete textový editor na proceduru nebo trigger, při jejichž provádění budete chtít laděného klienta zastavit a sledovat.
- Na ovládací liště stiskněte tlačítko zapínající ladící režim. Otevře se dialog se jmény klientů, které můžete ladit. Vyberte laděného klienta.
- Na vhodná místa do procedury nebo triggeru vložte body zastavení.
- Přejděte do laděného klienta a proveďte akci, kterou chcete ladit, například zavolejte uloženou proceduru. Pokud se při jejím provádění narazí na bod zastavení, klient "zamrzne"
- V ladícím klientovi se ukáže místo, kde se laděný proces zastavil. Nyní jej můžete nechat běžet dál, krokovat atd.

Na stavové řádce ladícího klienta se průběžně zobrazuje informace o stavu laděného procesu.

Jak ukončit ladění?

Ladění končí ve třech případech:

- Když se laděný klient odpojí od serveru, čímž laděný proces skončí.
- Když ladící klient provede příkaz menu **Ukončit ladění**. V tom případě může laděný klient pracovat dál, ale již bez intervencí ladícího klienta. Pokud laděný klient čekal na bodu zastavení, rozběhne se.
- Když ladící klient skončí. Také v tomto případě laděný klient pokračuje v práci.

Vlastnosti ladění

Veškeré ladění se týká výlučně laděného klienta. Pokud se například umístí do procedury bod zastavení, pak se na něm zastaví pouze laděný klient, zatímco všichni ostatní klienti mohou provádět stejnou proceduru zcela nerušeně.

Při krokování příkazů INSERT, UPDATE a DELETE, které spouštějí provedení triggerů, záleží na tom, zda provedete "krok dovnitř" nebo "krok přes". "Krok dovnitř" vstoupí do triggeru, zatímco "krok přes" provede triggery neviditelně.

Ladit lze i handlery výjimek. Lze do nich umístit body zastavení nebo do nich krokovat.

Ladění uložený a procedur a triggerů může probíhat současně s laděním aplikačních programů. V takovém případě také laděný klient pracuje v ladícím režimu. V obou klientech lze krokovat příkazy, prohlížet hodnoty proměnných atd.

Problémy při ladění

Umístíte-li v ladícím klientovi kurzor myši nad nějaké slovo v textu procedury, debugger se pokusí zjistit jeho hodnotu a zobrazit ji v plovoucím okénku. Pokud slovo nemá hodnotu, která by se dala vyčíslit, v logu serveru se objeví zpráva o chybě. Takovéto chyby je třeba ignorovat.

Hodnoty reálných čísel se vypisují v semilogaritmickém tvaru.