INTEGRATED DEVELOPMENT ENVIRONMENT PLUK IDE

Version 4.0

User Guide

Moscow, 2001

1 CONTENTS

2 INTRODUCTION

2.1 Application

Pluk IDE (Integrated Development Environment) is developed for:

- Editing source files written in different programming languages (mainly in Pluk Language).
- Creating and managing projects comprised of source files.
- Running and debugging projects, as well as stand-alone files.
- Browsing constants, variables, functions, classes, and methods.
- Graphic presentation of class hierarchy.
- Acquiring information on a process.
- Searching for information in a debugged application and file system.
- Working with the class library.
- Acquiring information on programming languages and system libraries used.

2.2 Requirements to User Skills

The user should have basic computer skills and study this manual. She also needs to be familiar with Pluk Language, Object To Database Mapping Language (ODML), Object To Window Mapping Language (OWML), Object Report Schema (ORS) Language, Platform Independent Resource (PIR) Language, Standard Library, and GUI Events Reference Books, as well as Pluk Builder User Guide.

3 DESCRIPTION OF OPERATIONS

3.1 Main Window Menu

The main window contains the menu only. All higher-level operations are conducted through the menu (Fig. 3.1-1).

Fig. 3.1-1 Main Window Menu

3.2 Source Files Editing

The Pluk IDE can work with text files that are source files in terms of containing binary code for the program (even if an interpreter is used). Source files could be written in various languages supported by the IDE and its system libraries. Following are the extensions of valid source files and the languages they are written in:

Extension	Language
plk	Pluk
h	Pluk, Microsoft Resource
rc	Microsoft Resource
scw	Object To Window Mapping
scm	Object To Database Mapping
cls	Database Class Schema
ors	Object Report Schema
lex	BNF-scheme of LRParser
pir	Platform Independent Resource

3.2.1 *File* Menu of the Main Window

Contains tools for working with source files (Fig. 3.2-1). A source file is edited in the Edit window (see 3.2.2).

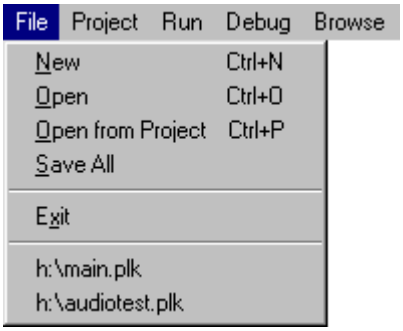


Fig. 3.2-1 *File* Menu of the Main Window

The last files opened by the user (no more than five) are shown at the menu's bottom. Selecting an item opens the file.

3.2.1.1 New

Opens the Edit window for editing a new source file.
Assigned keys — ***Ctrl+N***.

3.2.1.2 Open

Opens a source file for editing through the Open window.
Assigned keys — ***Ctrl+O***.

3.2.1.3 Open from Project

Opens the Edit window for a source file from a project through the *Open from Project* window presenting the list of the project's files.
Assigned keys — ***Ctrl+P***.

3.2.1.4 Save All

Saves all source files edited but not yet saved. For saving a particular file see 3.2.3.1.3, 3.2.3.1.4.

3.2.1.5 Exit

Terminates the IDE session.

3.2.2 Edit Window

Fig. 3.2-2 shows the Edit window for editing a source file. Information about the cursor position (line and column number) is given in the left bottom corner. The window supports a conventional edit mode, without syntax coloring.

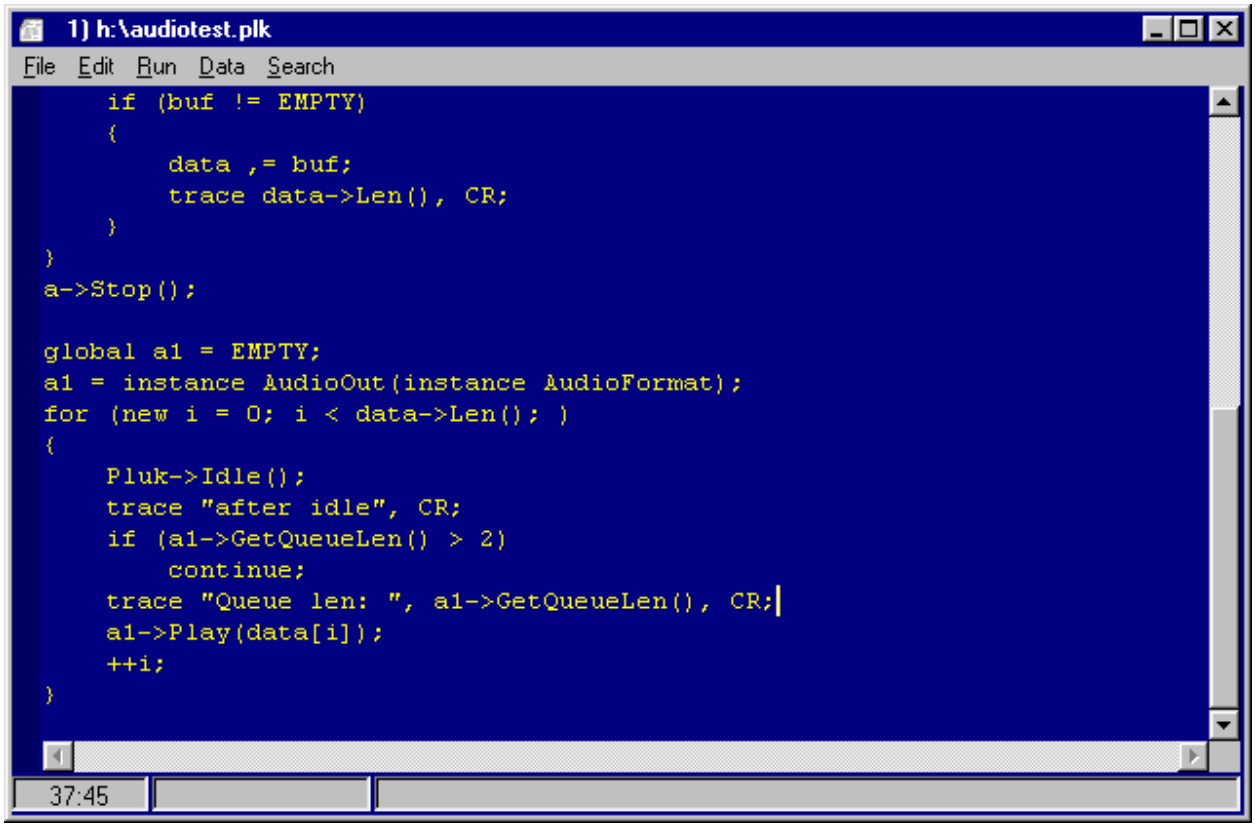


Fig. 3.2-2 Edit Window

3.2.3 Edit Window Menu

Fig. 3.2-3 shows the Edit window menu.

Fig. 3.2-3 Edit Window Menu

3.2.3.1 File

Contains tools for working with a source file (Fig. 3.2-4).

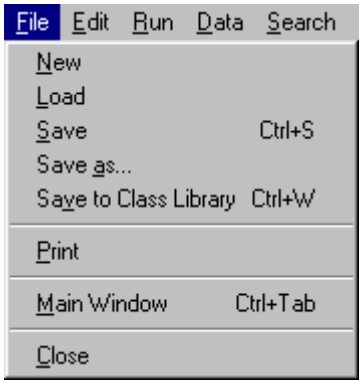


Fig. 3.2-4 *File* Menu

3.2.3.1.1 New

Opens a new window.

3.2.3.1.2 Load

Loads the source file through the Open window.

3.2.3.1.3 Save

Saves the window contents into the source file associated with the window. If not yet associated with any, the Save window will open.

Assigned keys — ***Ctrl+S***.

3.2.3.1.4 Save as

Saves the window contents into another file through the Save window.

3.2.3.1.5 Save to Class Library

Saves the window contents into the class library (see 3.6). If the window contains a **class** statement and methods of a class, they will be placed in the module of the class library pointed at by the class system comment (if there is no system comment, the class will be placed in the root module of the class library).

Assigned keys — ***Ctrl+W***.

3.2.3.1.6 Print

Prints out the window contents.

3.2.3.1.7 Main Window

Switches to the main window (see 3.1).

Assigned keys — ***Ctrl+Tab***.

3.2.3.1.8 Close

Closes the window.

3.2.3.2 Edit

Contains tools for text editing (Fig. 3.2-5).

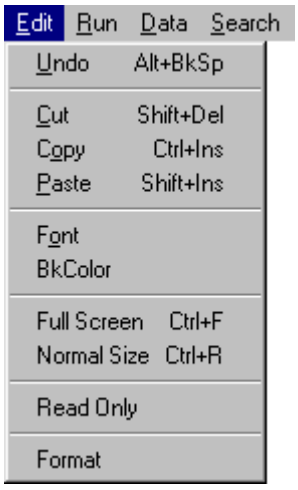


Fig. 3.2-5 *Edit* Menu

3.2.3.2.1 Undo

Cancels / restores the last change in the text being edited.
Assigned keys — ***Alt+BkSp***.

3.2.3.2.2 Cut

Copies a selected text into the clipboard, deleting it from the window.
Assigned keys — ***Shift+Del***.

3.2.3.2.3 Copy

Copies a selected text into the clipboard.
Assigned keys — ***Ctrl+Ins***.

3.2.3.2.4 Paste

Inserts the clipboard contents into the cursor position.
Assigned keys — ***Shift+Ins***.

3.2.3.2.5 Font

Opens the window for setting font parameters and color.

3.2.3.2.6 BkColor

Opens the window for setting background color.

3.2.3.2.7 Full Screen

Maximizes the window.
Assigned keys — ***Ctrl+F***.

3.2.3.2.8 Normal Size

Restores the window's size and position before maximization.
Assigned keys — ***Ctrl+R***.

3.2.3.2.9 Read Only

If checked, the user cannot edit text.

3.2.3.2.10 Format

Formats text written in Pluk Language in accordance with certain rules for setting brackets, operators, statements, etc.

3.2.3.3 Run

The menu (Fig. 3.2-6) contains tools for running text as code written in the following languages:

- Pluk;
- OWML;
- ODML;
- ORS;
- BNF;
- PIR.

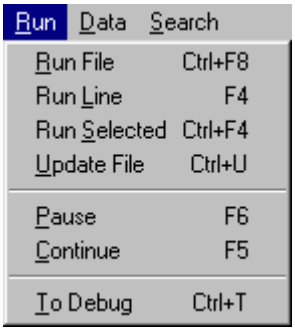


Fig. 3.2-6 *Run* Menu

3.2.3.3.1 Run File

Runs all the code in the window.
Assigned keys — ***Ctrl+F8***.

3.2.3.3.2 Run Line

Runs the line pointed at by the cursor.
Assigned key — ***F4***.

3.2.3.3.3 Run Selected

Runs a selected text fragment.
Assigned keys — ***Ctrl+F4***.

3.2.3.3.4 Update File

Saves the window contents into the associated source file. If not yet associated with any, the Save window will open.

After saving the file, all the code in the window will be executed (see *Run File*).
Assigned keys — ***Ctrl+U***.

3.2.3.4 Search

Contains tools for searching text (Fig. 3.2-7).

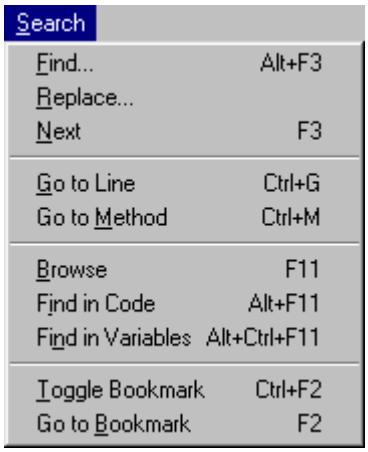


Fig. 3.2-7 *Search* Menu

3.2.3.4.1 Find

Opens the Search window.
Assigned keys — ***Alt+F3***.

3.2.3.4.2 Replace

Opens the Replace window.

3.2.3.4.3 Next

Continues to search for the next occurrence of the last entry in the text box.
Assigned key — ***F3***.

3.2.3.4.4 Go to Line

Moves the cursor to the line whose number the user has entered into the text box of the window.
Assigned keys — ***Ctrl+G***.

3.2.3.4.5 Go to Method

Moves the cursor to the beginning line of the:

- selected method written in Pluk Language through the *Go to Method* window;
- selected form specification written in OWML (see the details in OWML Reference Book) through the *Go to Form Specification* window;
- selected node written in ORS Language (see the details in ORS Language Reference Book) through the *Go to Node* window;
- selected resource written in PIR Language (see the details in PIR Language Reference Book) through the *Go to Resource* window.

Assigned keys — ***Ctrl+M***.

3.2.3.4.6 Toggle Bookmark

Sets / deletes a bookmark on the line pointed at by the cursor. The bookmark is a cyan triangle located at the left of the first symbol on the line.
Assigned keys — ***Ctrl+F2***.

3.2.3.4.7 Go to Bookmark

Moves the cursor to the next bookmarked line. The bookmark is a cyan triangle located at the left of the first symbol on the line.
Assigned key — ***F2***.

3.2.4 Window Menu of the Main Window

The menu contains tools for working with windows (Fig. 3.2-8).

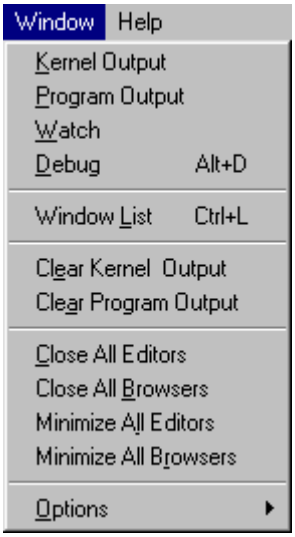


Fig. 3.2-8 *Window* Menu of the Main Window

3.2.4.1 Window List

Opens the list of Edit windows. Through the *Window List* window, the user selects a window and places it on top of other windows.

Assigned keys — ***Ctrl+L***.

3.2.4.2 Close All Editors

Closes all Edit windows.

3.2.4.3 Minimize All Editors

Minimizes all Edit windows.

3.3 Project Management

3.3.1 *Project* Menu of the Main Window

The menu contains tools for project management (Fig. 3.3-1).

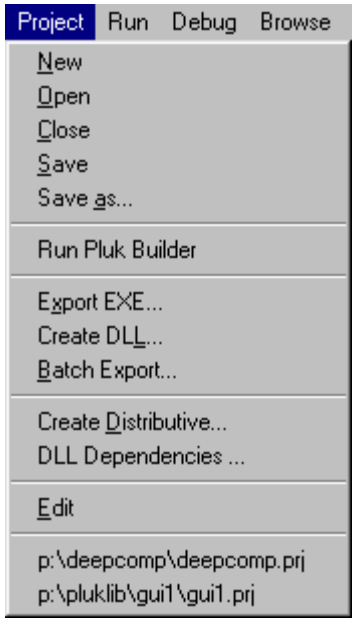


Fig. 3.3-1 *Project* Menu of the Main Window

A project is a file (*.prj) that contains a list (in binary format) of libraries (*.dll) and source files (see 3.2) to be executed in a selected order when running the project. It can also contain the name of the project icon (to be inserted into the exported executable file) or the name of the interface description file written in Microsoft Resource Language (the latter occurs quite rarely).

A file in Pluk Language is executed by assigning its contents to a temporary function with subsequent execution of the latter. Source files in the other languages are executed by passing their contents to certain system functions.

There are some rules that provide for the correct order of files in a project.

File Ext.	Project's Correct File Order Rules
plk	If language entities (variables, functions, classes, etc.) are used in a file at the file level, not inside function, this file must be put after the files where those entities are created.
scw	A file in OWML must be put after the owml2.dll, the latter after the gui.dll.
scm	A file in ODML must be put after the odml2.dll, the latter after podbc.dll.
ors	A file in ORS Language must be put after the orslib.dll, the latter after the replib.dll, the latter after the gui.dll.
lex	A file with BNF-scheme must be put after the lrparser.dll.
pir	A file in PIR Language must be put after the gui.dll.

Only one project at a time could be opened in the IDE.

The last files opened by the user (no more than five) are shown at the menu's bottom. Selecting an item opens the project.

3.3.1.1 New

Creates a new project (not yet associated with file) and opens the *Project* window for it (see 3.3.2).

3.3.1.2 Open

Opens a project, via the Open window, and opens the *Project* window for it (see 3.3.2).

3.3.1.3 Close

Closes the project.

In the configuration file **Plide.ini**, section **Default**, key **StartDefaultPlapp**, an **on** value could be set. It means that upon closing the project (as well as running the IDE without a project) an 'empty' application will always start up accompanied with the loading of the system libraries listed in this section, key **DefaultLib**.

3.3.1.4 Save

Saves the project to the associated file. If not yet associated with any, the Save window will open.

3.3.1.5 Save as

Saves the project to another file. The Save window opens.

3.3.1.6 Run Pluk Builder

Starts up the Pluk Builder interface designer. The designer allows graphic editing of project's files related to the interface. See Pluk Builder User Guide.

3.3.1.7 Export EXE

Exports the project into an executable file (*.exe) that comprises the contents (could be already in binary format) of all the source files. The Save window opens for the user to enter the name for the exported file. The icon of the executable file could be set in the *Project* window (see 3.3.2).

3.3.1.8 Create DLL

Exports the project into a library file (*.dll) that comprises the contents (could be already in binary format) of all the source files. The Save window opens for the user to enter the name for the exported file.

A library file differs from an executable one in that it will not be run, but will be included in projects and executable files as common file for code storage.

3.3.1.9 Batch Export

Exports a set of projects into executable and library files using a batch file (*.lst). The Open window opens for the user to enter the name for the batch file.

A batch file is a text file. Each line of a batch file describes the export of a particular project. A line consists of two or three words separated with commas, spaces or tabs. Fist word — the project filename, second — the name of the executable file (*.exe) or library file (*.dll), third — an optional word that makes sense only in case of a library file: the base address of the library while loading into the memory of the executable file (in the form **0xhhhhhhhh**, where **h** is a hexadecimal number). By setting different base addresses to different libraries (provided there is enough space between addresses) one could save valuable memory space while loading a number of libraries.

3.3.1.10 Create Distributive

Not implemented.

3.3.1.11 DLL Dependencies

Retrieves the libraries used by the project explicitly or implicitly. The result is given in the *DLL Dependencies* window in the list of the libraries (Fig. 3.3-2).

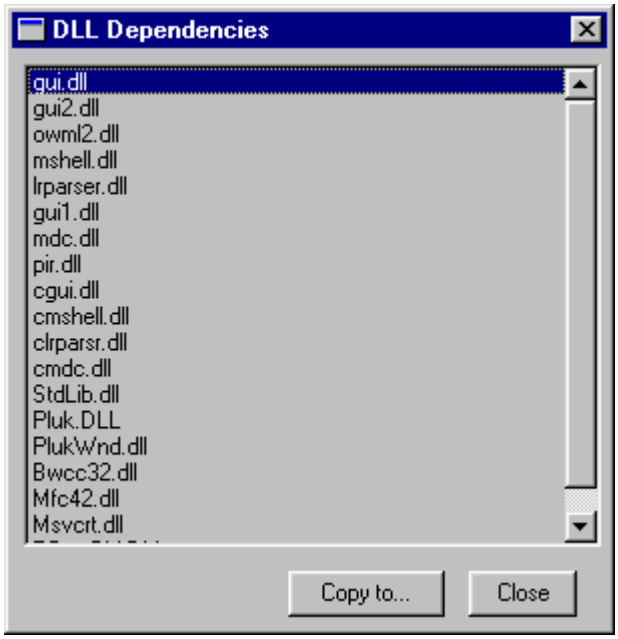


Fig. 3.3-2 *DLL Dependencies* Window

Copy to button allows copying the libraries used into a directory. The name for the directory is set in the Directory window.

3.3.1.12 Edit

Opens the *Project* window for the project (see 3.3.2).

3.3.2 Project Window

Fig. 3.3-3 shows the *Project* window for adding files to and removing files from the project.

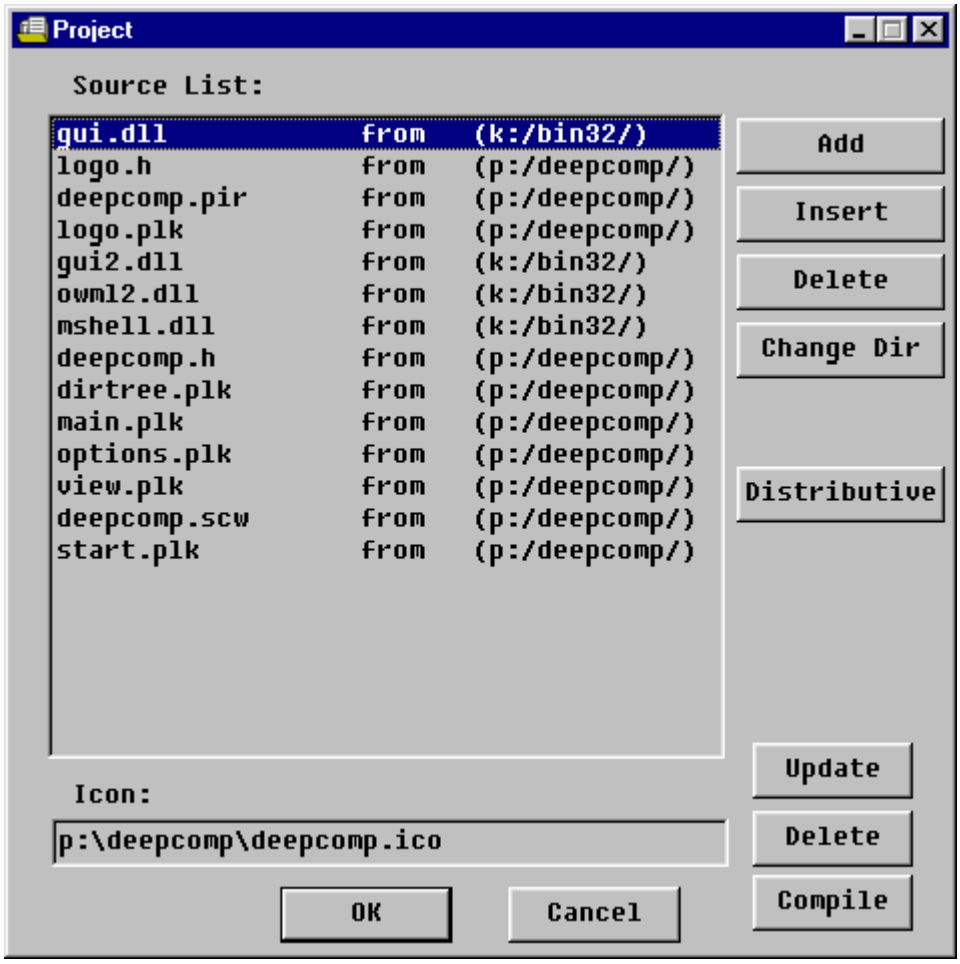


Fig. 3.3-3 *Project* Window

3.3.2.1 Source List

The list of source and library files loaded at the startup of the project in the order presented.

Following are the extensions of the source files and corresponding languages:

File ext.	Language
plk	Pluk
h	Pluk, Microsoft Resource
scw	Object To Window Mapping
scm	Object To Database Mapping
ors	Object Report Schema
lex	BNF-scheme of LRParser
pir	Platform Independent Resource

3.3.2.2 Add

Adds a library or source file at the end of the *Source List*.

3.3.2.3 Insert

Inserts a library or source file before the selected item.

3.3.2.4 Delete

Deletes the selected library or source file from the *Source List*.

3.3.2.5 Change Dir

Change the directory for all files of the project located in a certain directory and its subdirectories. First, a window opens to enter the name of the old directory (the directory prompted to user will contain the file selected from the *Source List*). The user can change the name by shortening, for instance, the number of nested subdirectories. Then the Directory window opens for selecting a new directory. As the result, all the filenames from the old directory and its subdirectories are converted to the filenames in the new directory (the subdirectory names do not change).

3.3.2.6 Distributive

Not implemented.

3.3.2.7 Icon

The name of the project icon that is inserted in an export executable file.

For outdated projects, with the interface written in Microsoft Resource Language, or for system projects that are supposed to use this language, this field is recognized as the name of a resource file in Microsoft Resource Language.

This field cannot be changed (for changing it, see below).

3.3.2.8 Update

Sets an icon (or a Microsoft Resource file) to the project through the Open window

3.3.2.9 Delete

Deletes an icon (or a Microsoft Resource file) from the project.

3.3.2.10 Compile

Rests if the *Icon box* is filled in.

Compiles a resource file into a library, with the same name and a *.dll extension, if the *Icon box* is filled in with the name of a Microsoft Resource file.

3.4 Running and Debugging

Running and debugging can be conducted by the user for the current project, as well as for particular files. Execution of a particular file is conducted by the startup of an 'empty' application and the file execution.

3.4.1 Kernel Output Window

The *Kernel Output* window is used for presenting:

- information on the loaded libraries (Fig. 3.4-1);
- information about errors (Fig. 3.4-1);
- stack contents at the moment of error throwing (Fig. 3.4-2);
- events following function calls (Fig. 3.4-3).

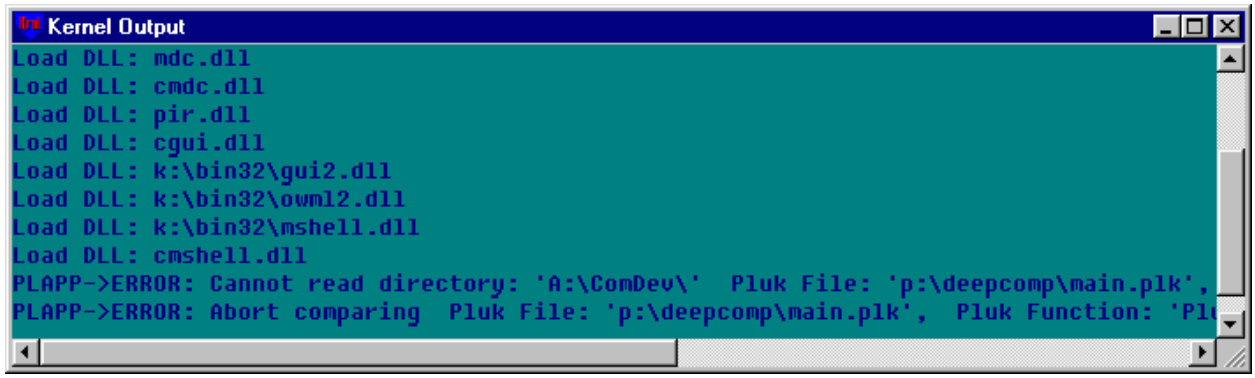


Fig. 3.4-1 *Kernel Output* Window with Information about Loaded Libraries and Errors

Left double-click on the line with the error information opens the source file, with the cursor set on the error line. It happens, however, only if an error occurs in a function loaded from a source file.

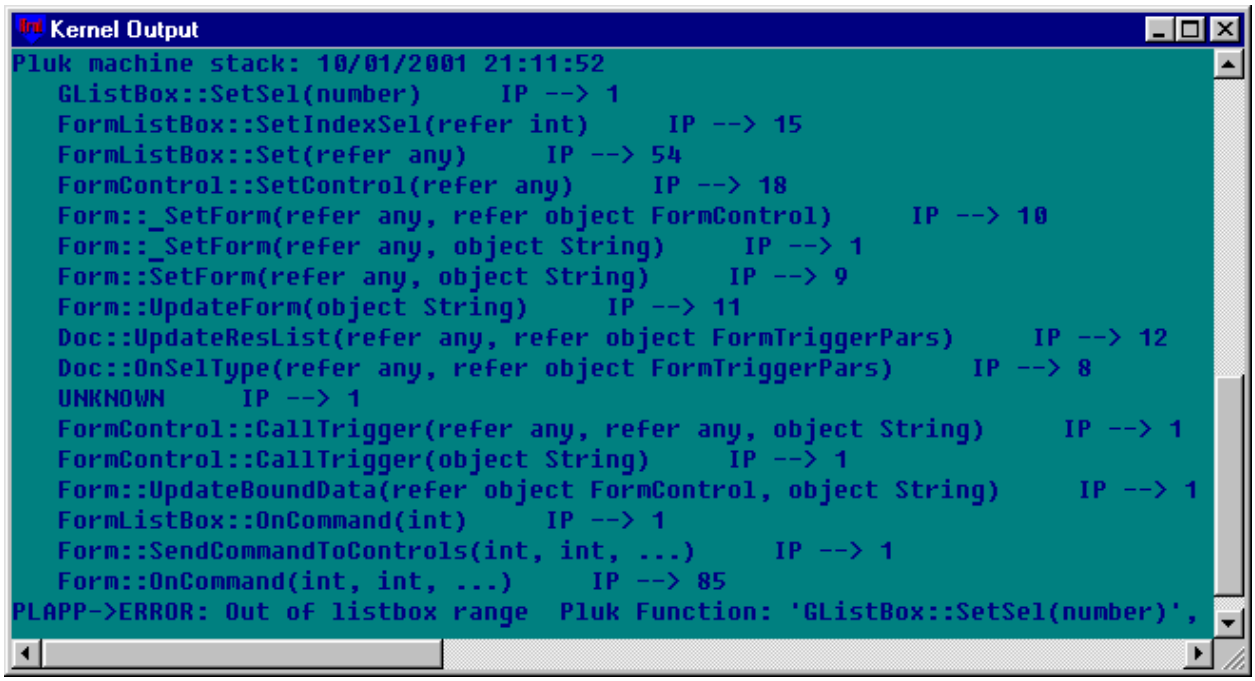


Fig. 3.4-2 *Kernel Output* Window with Stack Contents at the Moment of Error Throwing

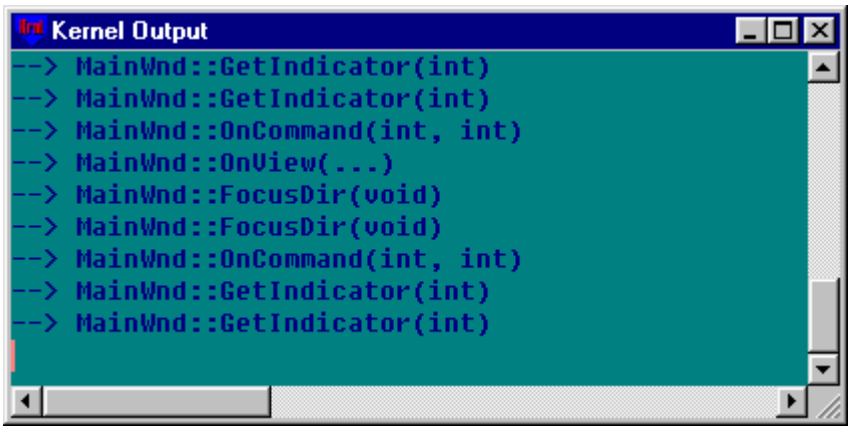


Fig. 3.4-3 *Kernel Output* Window with Events Ensued Function Calls

3.4.2 Program Output Window

The *Program Output* window (Fig. 3.4-4) is used for presenting various data of an application by means of a **trace** statement (see Pluk Language Reference Book).

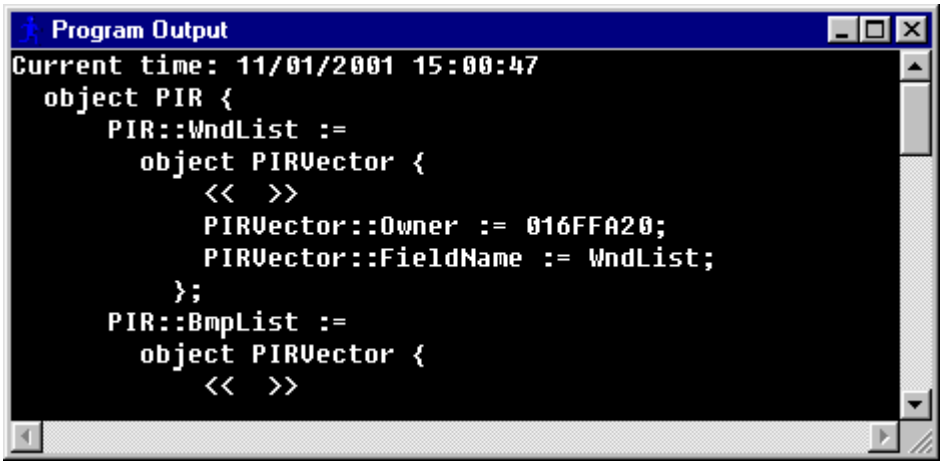


Fig. 3.4-4 *Program Output* Window

3.4.3 Debug Window

The *Debug* window is used for debugging applications (Fig. 3.4-5). The body of a function in Pluk Language could be loaded in the window. If an application has been halted on a line in the loaded body, this line will be pointed at in the window by a yellow triangle set at the left of the first symbol. The user can halt an application at will by setting breakpoints (see 3.4.4.2.9) in the body shown with red triangles set at the left of the first symbol on the line.

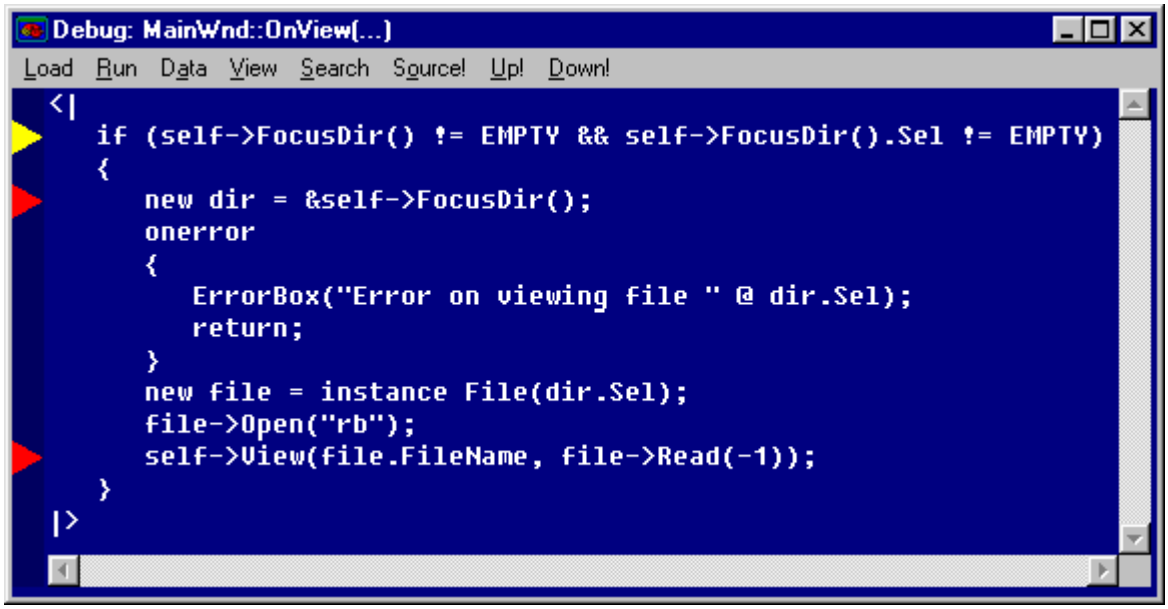


Fig. 3.4-5 *Debug* Window

Left double-click on the line opens the Edit window with the function, with the cursor positioned on the first line. If the function is loaded from a source file, this file will be opened. If the function is loaded from a library file, a temporary source file will be opened containing this function only. The filename begins with the string **noname**.

3.4.4 ***Debug* Window Menu**

Fig. 3.4-6 shows the *Debug* window menu.

Fig. 3.4-6 *Debug* Window Menu

3.4.4.1 Load

Contains tools for loading the body of a function (Fig. 3.4-7).

Fig. 3.4-7 *Load* Menu

3.4.4.1.1 Load Function

Opens the *Open to Debug Window* for loading the body of a global function, method, or global class field of type **function** (see 3.4.5).

3.4.4.1.2 Load from Stack

Opens the *Open to Debug Window* for loading the body of a function from the stack (see 3.4.5).

3.4.4.2 Run

Contains tools for debugging a function (Fig. 3.4-8).

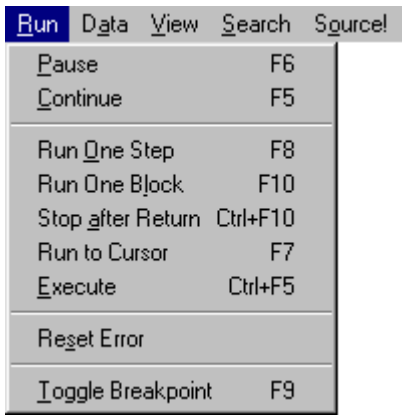


Fig. 3.4-8 *Run* Menu

3.4.4.2.1 Pause

Halts the application. It can occur only at the point where the application passes the program flow on to the system using the **Pluk::Idle** method, or at any point after invoking the **Pluk::AllowIdle** method (set **TRUE**), but before invoking the **Pluk::AllowIdle** method (set **FALSE**).

Assigned key — ***F6***.

3.4.4.2.2 Continue

Continues running the application halted earlier.
Assigned key — ***F5***.

3.4.4.2.3 Run One Step

Runs one line from the body. If a function is called in the line, the application will halt at the beginning of the function.
Assigned key — ***F8***.

3.4.4.2.4 Run One Block

Runs one line from the body. If a function is called in the line, the application will execute the function and halt right after the call.
Assigned key — ***F10***.

3.4.4.2.5 Stop after Return

Runs the function located at the bottom of the stack to the end of the body and breaks right after the call.
Assigned keys — ***Ctrl+F10***.

3.4.4.2.6 Run to Cursor

Runs the function to the line pointed at by the cursor and breaks on this line.
Assigned key — ***F7***.

3.4.4.2.7 Execute

Opens the *Run on Level* window (see 3.4.12) for running an arbitrary expression in the context of a function located at any stack level.
Assigned keys — ***Ctrl+F5***.

3.4.4.2.8 Reset Error

Resets the error thrown in the application. This will revoke the rollback from the rest of the functions in the stack.

3.4.4.2.9 Toggle Breakpoint

Switches on / off a breakpoint on a line. The running application halts on the line.
Assigned key — ***F9***.

3.4.4.3 Data

Contains tools for viewing expression values (Fig. 3.4-9). The *Watch* (see 3.4.6) and *Run on Level* (see 3.4.12) windows are used for viewing.

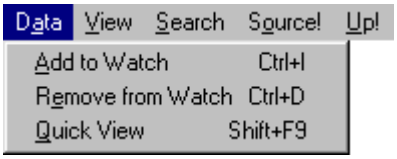


Fig. 3.4-9 *Data* Menu

3.4.4.3.1 Add to Watch

Adds an expression to the *Watch* window. The expression is the selected text from the *Debug* window or the word pointed at by the cursor in the *Debug* window. The expression

appears in the text box of the *Add to Watch Window* and can be modified by the user. Then it is added to the *Watch* window.

Assigned keys — ***Ctrl+I***.

3.4.4.3.2 Remove from Watch

Removes expressions from the *Watch* window. The *Remove* window opens that contains the list of expressions viewed in the *Watch* window to select the expressions to be removed.

Assigned keys — ***Ctrl+D***.

3.4.4.3.3 Quick View

Adds an expression to the *Run on Level* window to view its value. The expression is a selected text from the *Debug* window or the word pointed at by the cursor in the *Debug* window. The expression appears in the text box of the *Add to Watch Window* and can be modified by the user.

Assigned keys — ***Shift+F9***.

3.4.4.4 View

Contains tools for setting viewing parameters (Fig. 3.4-10).

Fig. 3.4-10 *View* Menu

3.4.4.4.1 BCode

If unchecked, the *Debug* window shows the source code of a function. If checked, the B-code of a function (Pluk-processor commands).

3.4.4.4.2 Font

Opens the window for setting font parameters.

3.4.4.4.3 BkColor

Opens the window for setting background color.

3.4.4.5 Search

Contains tools for searching text (Fig. 3.4-11).

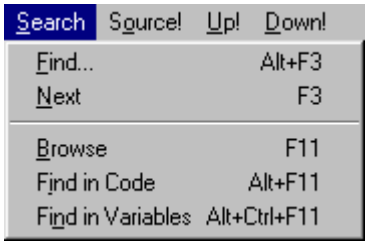


Fig. 3.4-11 *Search* Menu

3.4.4.5.1 Find

Opens the Search window.
Assigned keys — ***Alt+F3***.

3.4.4.5.2 Next

Continues to search for the next occurrence of the last entry in the text box.
Assigned key — ***F3***.

3.4.4.6 Source

Opens the Edit window with the function with the cursor positioned on the first line. If the function is loaded from a source file, the latter will be opened. If the function is loaded from a library file, a temporary source file will be opened containing this function only. The filename begins with the string **noname**.

3.4.4.7 Up

Moves up the stack, i.e. loads the function that calls a given function into the *Debug* window.

3.4.4.8 Down

Moves down the stack, i.e. loads the function that is called by a given function into the *Debug* window.

3.4.5 Open to Debug Window

The *Open to Debug Window* (Fig. 3.4-12) is used for loading the body of a function into the *Debug* window (see 3.4.3).

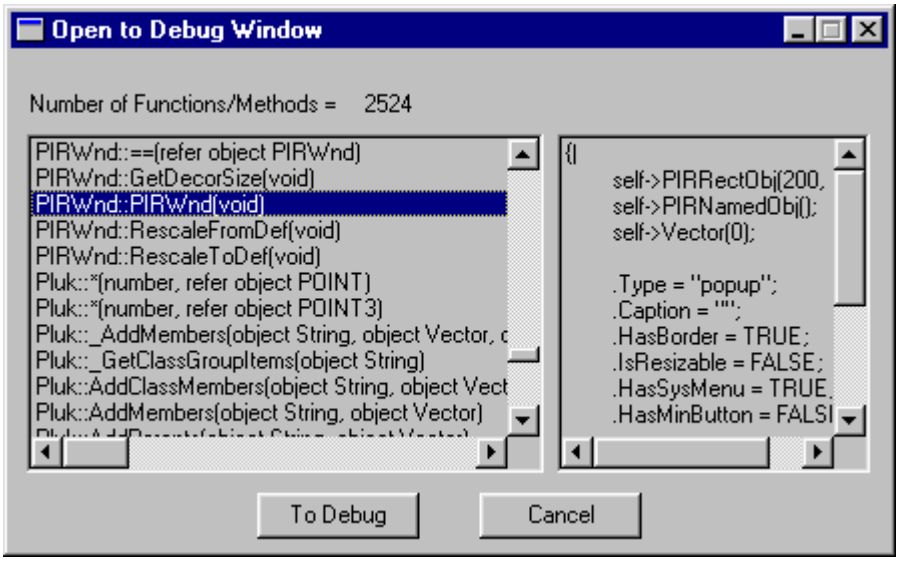


Fig. 3.4-12 ***Open to Debug Window***

3.4.5.1 Number of Functions / Methods

Total number of functions on the list.

3.4.5.2 Functions List

Contains the list of functions. At the right, the body of a selected function is presented.

Left double-click on an item loads the selected function into the *Debug* window (see 3.4.3).

3.4.5.3 To Debug

Loads a selected function into the *Debug* window (see 3.4.3).

3.4.6 Watch Window

The *Watch* window (Fig. 3.4-13) is used for the continuous viewing of expression values.

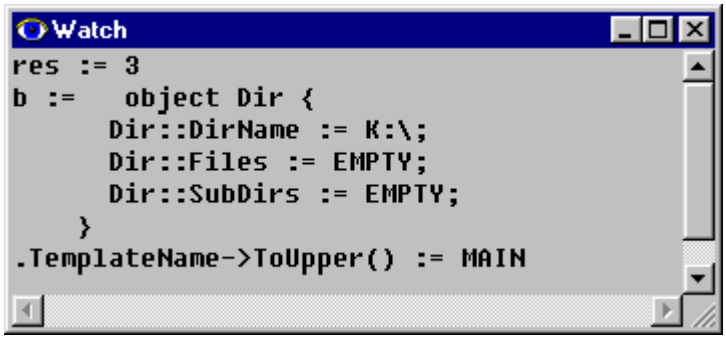


Fig. 3.4-13 *Watch* Window

3.4.7 Edit Window Menu

Fig. 3.2-3 shows the Edit window menu.

3.4.7.1 Run

The menu (Fig. 3.2-6) contains tools for running text as code written in the following languages:

- Pluk;
- OWML;
- ODML;
- ORS;
- BNF;
- PIR.

3.4.7.1.1 Pause

Halts the application. The break can occur only at the point where the application passes the program flow on to the system using the **Pluk::Idle** method, or at any point after invoking the **Pluk::AllowIdle** method (set **TRUE**), but before invoking the **Pluk::AllowIdle** (set **FALSE**).

Assigned key — ***F6***.

3.4.7.1.2 Continue

Continues running the application halted earlier.

Assigned key — ***F5***.

3.4.7.1.3 To Debug

Loads a function in the *Debug* window (see 3.4.3). The selected text or the word pointed at by the cursor in the *Debug* window are used as a name for a global function. The selected text

in the *Debug* window is used as a name for a method. The name of the method comprises the name of the class, the name of the method itself, and the types of the arguments. For example, **A::F(refer object String, int)** (see Pluk Language Reference Book).

Assigned keys — ***Ctrl+T***.

3.4.7.2 Data

Contains tools for viewing expression values (Fig. 3.4-14). The *Watch* (see 3.4.6) and *Run on Level* (see 3.4.12) windows are used for viewing.

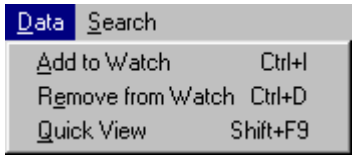


Fig. 3.4-14 *Data* Menu

3.4.7.2.1 Add to Watch

Adds an expression to the *Watch* window. The expression is the selected text or the word pointed at by the cursor in the Edit window. The expression appears in the text box of the *Add to Watch Window* and can be modified by the user. Then it is added to the *Watch* window.

Assigned keys — ***Ctrl+I***.

3.4.7.2.2 Remove from Watch

Removes expressions from *Watch* window. The *Remove* window opens that contains the list of expressions viewed in the *Watch* window to select the expressions to be removed.

Assigned keys — ***Ctrl+D***.

3.4.7.2.3 Quick View

Adds an expression to the *Run on Level* window. The expression is the selected text or the word pointed at by the cursor in the Edit window. The expression appears in the text box of the *Add to Watch Window* and can be modified by the user. Then it is added to the *Run on Level* window.

Assigned keys — ***Shift+F9***.

3.4.8 ***Run* Menu of the Main Window**

The menu contains tools for running and debugging a project (Fig. 3.4-15).

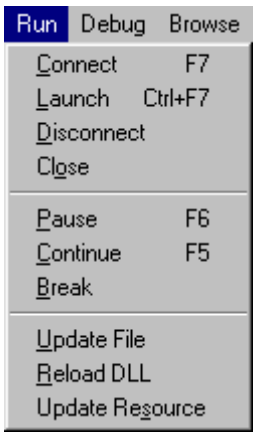


Fig. 3.4-15 *Run* Menu of the Main Window

3.4.8.1 Connect

Opens the *Connect* window for connecting to the already running application (see 3.4.9).
Assigned key — ***F7***.

3.4.8.2 Launch

Opens the *Launch* window for running an application (see 3.4.10).
Assigned keys — ***Ctrl+F7***.

3.4.8.3 Disconnect

Breaks the application connection.

3.4.8.4 Close

Terminates the application.

3.4.8.5 Pause

Halts the application. The break can occur only at the point where the application passes the program flow on to the system using the **Pluk::Idle** method, or at any point after invoking the **Pluk::AllowIdle** method (set **TRUE**), but before invoking the **Pluk::AllowIdle** method (set **FALSE**).

Assigned key — ***F6***.

3.4.8.6 Continue

Continues running the application halted earlier.
Assigned key — ***F5***.

3.4.8.7 Break

Sets up a break mode for the application.

If the application is running in the break mode, it will halt at the point where the application passes the program flow on to the system using **Pluk::Idle** method, or at any point after invoking the **Pluk::AllowIdle** method (set **TRUE**), but before invoking the **Pluk::AllowIdle** method (set **FALSE**).

If the application is halted in the break mode, the next executed statement in Pluk Language will bring about the application break. This will result in the interruption of all functions in the application and switching into a standby mode.

3.4.8.8 Update File

Runs the contents of a source file. The file is selected through the Open window. The extensions of valid source files are presented in 3.3.2.1.

3.4.8.9 Reload DLL

Loads a new library replacing the already loaded one. First, the destination library, i.e. the to-be-replaced library, must be selected in the Open window. Second, in another window, the source library, i.e. the replacing one, must be selected.

Replacement is impossible if the destination is locked by another application (except the IDE).

3.4.8.10 Update Resource

Compiles a resource file into a library with the same name and a *.dll extension, and updates it in memory, if the *Icon* box in the *Project* window (see 3.3.2) contains the name of a Microsoft Resource file.

This feature is important only for outdated projects with the interface written in Microsoft Resource Language and for system projects that are supposed to use this language.

3.4.9 Connect Window

The *Connect* window is used for connecting to a running application (Fig. 3.4-16).

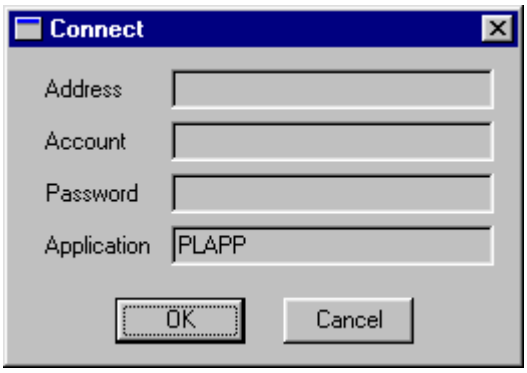


Fig. 3.4-16 *Connect* Window

The user usually does not change this window because most of the time:

- she connects to the application located on the same computer;
- the application's name is PLAPP (another name will appear when the project is exported into another file with another name, see 3.3.1.7).

3.4.9.1 Address

The address of a remote computer (to connect to a remote application).

3.4.9.2 Account

The user name registered on the remote computer (to connect to a remote application).

3.4.9.3 Password

The user password registered on the remote computer (to connect to a remote application).

3.4.9.4 Application

The application's name (usually PLAPP).

3.4.10 Launch Window

The *Launch* window is used for running an application (Fig. 3.4-17).

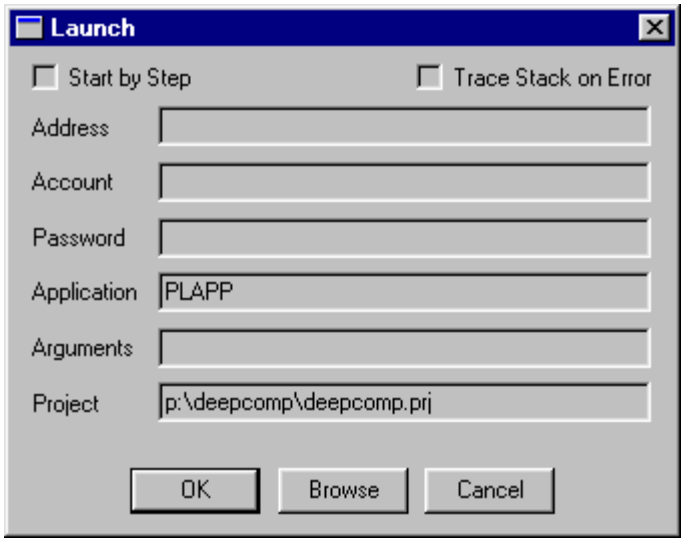


Fig. 3.4-17 *Launch* Window

The user usually does not change the settings because most of the time:

- she connects to the application located on the same computer;
- the application's name is PLAPP (another name will appear when the project is exported into another file with another name, see 3.3.1.7);
- command line arguments are not needed;
- the current project is to be run.

3.4.10.1 Address

The address of a remote computer (to connect to a remote application).

3.4.10.2 Account

The user name registered on the remote computer (to connect to a remote application).

3.4.10.3 Password

The user password registered on the remote computer (to connect to a remote application).

3.4.10.4 Application

The application's name (usually PLAPP).

3.4.10.5 Arguments

Arguments of the command line. The arguments are accessible in the application as a **CmdLine** constant — a vector of strings.

3.4.10.6 Project

The name of the project to be run. The current project is prompted, but the user can change the name.

3.4.10.7 Browse

Opens the Open window for entering the name of the project to be run.

3.4.10.8 Start by Step

Runs the application in a step-by-step mode. The IDE halts on the first statement, which gives the user an option to run step-by-step, run continuously, or do something else.

3.4.10.9 Trace Stack on Error

If checked, any error throwing results in putting the stack contents into the *Kernel Output* window (Fig. 3.4-2).

3.4.11 Debug Menu of the Main Window

The menu contains tools for debugging a project (Fig. 3.4-18).

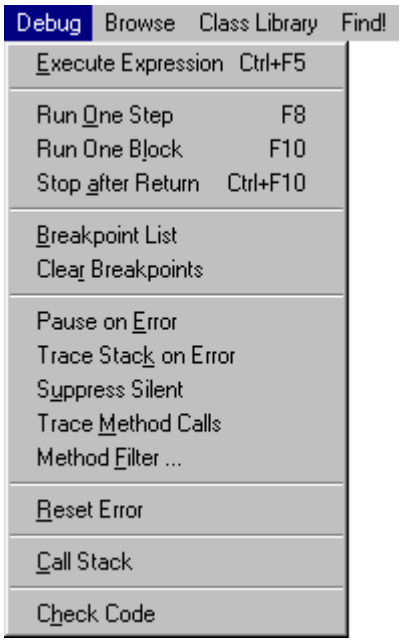


Fig. 3.4-18 *Debug* Menu of the Main Window

3.4.11.1 Execute Expression

Opens the *Run on Level* window (see 3.4.12) for executing an arbitrary expression in the context of a function located at any stack level.

Assigned keys — ***Ctrl+F5***.

3.4.11.1.1 Run One Step

Runs one line from the body. If a function is called in the line, the application will break at the beginning of the function.

Assigned key — ***F8***.

3.4.11.1.2 Run One Block

Runs one line from the body. If a function is called in the line, the application will execute the function and breaks right after the call.

Assigned key — ***F10***.

3.4.11.1.3 Stop after Return

Runs the function located at the bottom of the stack to the end of the body and breaks right after the call.

Assigned keys — ***Ctrl+F10***.

3.4.11.2 Breakpoints List

Opens the *List of Breakpoints* window with the breakpoints list. A breakpoint is presented as the name of the function and the line number in its body. Left double-click on an item on the list loads the function into the *Debug* window (see 3.4.3) with the cursor pointing at the breakpoint line.

3.4.11.3 Clear Breakpoints

Removes all breakpoints.

3.4.11.4 Pause on Error

If checked, the application halts if an error occurs.

3.4.11.5 Trace Stack on Error

If checked, any error throwing results in putting the stack contents into the *Kernel Output* window (Fig. 3.4-2).

3.4.11.6 Suppress Silent

If checked, the user can view in the *Kernel Output* window the information about all errors suppressed by the class **Silent**. For more information about the class **Silent,** see Standard Library Reference Book.

3.4.11.7 Trace Method Calls

If checked, the events that follow function calls are presented in the *Kernel Output* window (Fig. 3.4-3). It is almost impossible to trace the calls of all functions due to a dramatic decrease in the IDE's performance. So the user should select appropriate methods in the *Method Filter* window. Invoking these methods should result in event generation.

3.4.11.8 Method Filter

Opens the *Method Filter* window to select the methods whose invocation brings about the events presented in the *Kernel Output* window (Fig. 3.4-3).

3.4.11.9 Reset Error

Resets an error generated in the application. This halts the rollback from the remaining functions in the stack.

3.4.11.10 Call Stack

Opens the *Call Stack* window for viewing stack functions (see 3.4.14).

3.4.11.11 Check Code

Checks all the loaded code written in Pluk Language for errors that could be spotted. Since the language does not support variables' types, type-mismatch errors (between calling code and called code) could not be spotted.

3.4.12 Run on Level Window

Fig. 3.4-19 shows the *Run on Level* window for running an arbitrary expression in the context of a function located on any stack level.

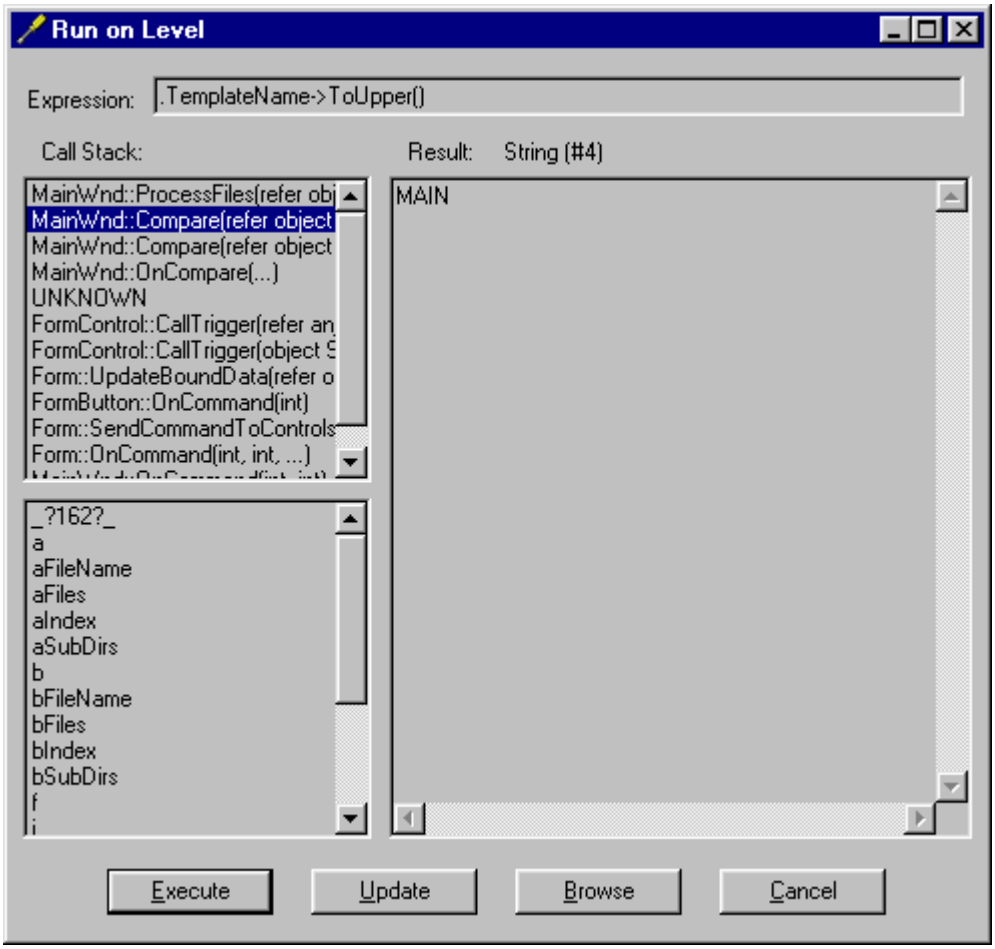


Fig. 3.4-19 *Run on Level* Window

3.4.12.1 Expression

The expression that uses global names, as well as local names of the function selected from the list *Call Stack*. After pressing ***Enter,*** the *Expression* is calculated and the result is placed in the *Result* box.

3.4.12.2 Call Stack

The list of functions in the stack.

Below it, there is the list of local variables of the function selected from the list *Call Stack*.

Left double-click on a local variable inserts its name in the *Expression* box at the end of the string.

Variables with the identical names, but located at different scopes (see Pluk Language Reference Book), could be present in one function. On the local variables list such variables will differ in suffix (**@** + three-digit number). There could also be variables on the list surrounded with **_?** and **?_**. These are the temporary variables: the conditions of a **switch** statement (see Pluk Language Reference Book).

3.4.12.3 Result

The result of the *Expression* calculation.

3.4.12.4 Execute

Calculates the expression in the *Expression* box and places the result in the *Result* box.

3.4.12.5 Update

Updates the *Call Stack* list and, accordingly, the local variables list.

3.4.12.6 Browse

If the *Expression* box is not empty, the value browsing window will open for browsing the expression value (see 3.5.16). If the *Expression* box is empty, the value browsing window will open for browsing the value of the selected local variable (see 3.5.16).

3.4.13 Method Filter Window

Fig. 3.4-20 shows the *Method Filter* window for selecting methods whose invocation brings about the events presented in the *Kernel Output* window (Fig. 3.4-3).

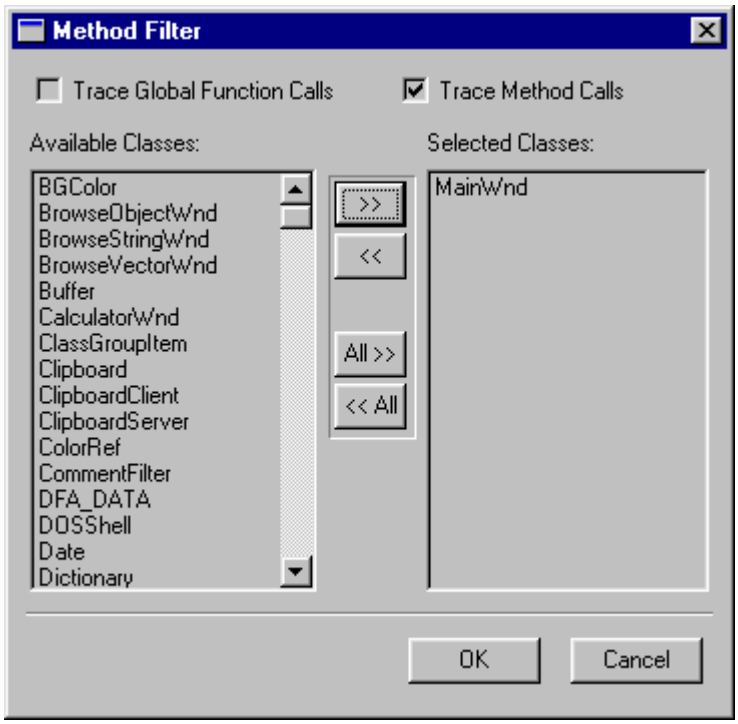


Fig. 3.4-20 *Method Filter* Window

3.4.13.1 Trace Global Function Calls

If checked, calling any global function brings about an event.

3.4.13.2 Trace Method Calls

If checked, invoking a method of any class on the *Selected Classes* list brings about an event.

3.4.13.3 Available Classes

The list of all classes, excluding the classes on the *Selected Classes* list.

3.4.13.4 Selected Classes

The list of selected classes.

3.4.13.5 >>

Moves the selected classes from the *Available Classes* list to the *Selected Classes* list.

3.4.13.6 <<

Moves the selected classes from the *Selected Classes* list to the *Available Classes* list.

3.4.13.7 All >>

Moves all the classes from the *Available Classes* list to the *Selected Classes* list.

3.4.13.8 << All

Moves all the classes from the *Selected Classes* list to the *Available Classes* list.

3.4.14 Call Stack Window

Fig. 3.4-21 shows the *Call Stack* window for viewing source code and local variables of a function located at any stack level.

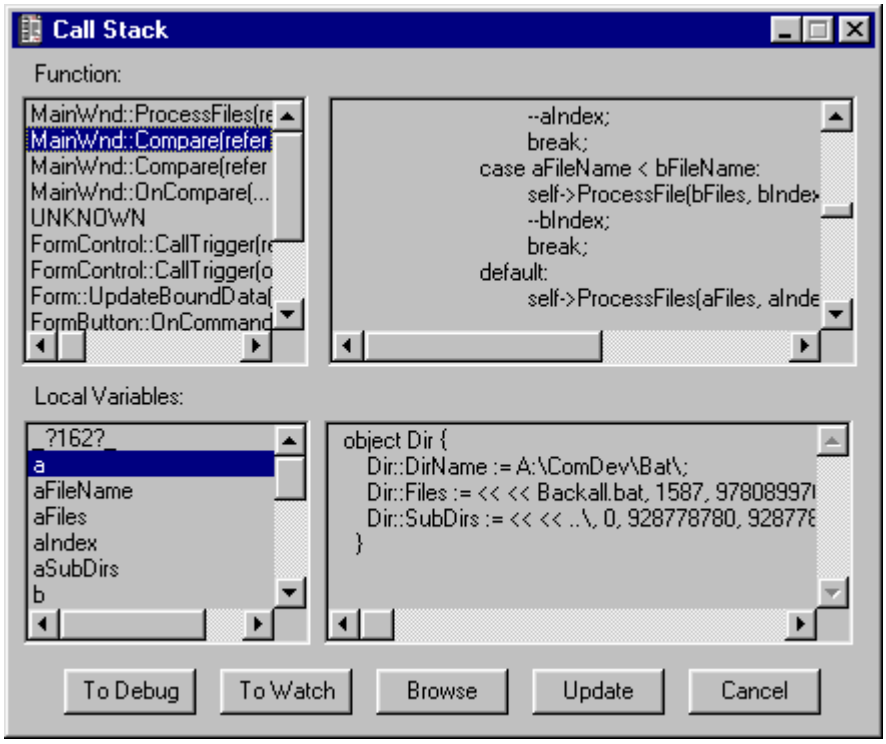


Fig. 3.4-21 *Call Stack* Window

3.4.14.1 Function

The list of functions in the stack.

At the right, there is the text box that contains the source code of the selected function. The cursor is positioned on the line where the function has been halted.

Left double-click on a list item opens the Edit window with the function, with the cursor positioning on the first line.

3.4.14.2 Local Variables

Local variables of the selected function on the *Function* list.

At the right, there is the text box that contains the value of the selected variable.

Left double-click on a list item adds the local variable selected from the *Local Variables* list into the *Watch* window (see 3.4.6). The *Add to Watch Window* opens with the name of the variable that could be changed by the user. Then it is added to the *Watch* window.

Variables with the identical names, but located at different scopes (see Pluk Language Reference Book), could be present in one function. On the local variables list such variables will differ in suffix (**@** + three-digit number). There could also be variables on the list surrounded with **_?** and **?_**. These are the temporary variables: the conditions of a **switch** statement (see Pluk Language Reference Book).

3.4.14.3 To Debug

Loads the function selected from the *Function* list into the *Debug* window (see 3.4.3).

3.4.14.4 To Watch

Adds the local variable selected from the *Local Variables* list into the *Watch* window (see 3.4.6). The *Add to Watch Window* opens with the name of the variable that could be changed by the user. Then it is added to the *Watch* window.

3.4.14.5 Browse

Opens the value browsing window for browsing the value of the selected variable on the *Local Variables* list (see 3.5.16).

3.4.14.6 Update

Updates the *Function* list and, accordingly, the *Local Variables* list.

3.4.15 Watch Menu of the Main Window

The menu contains tools for viewing expression values (Fig. 3.4-22). The *Watch* window is used for viewing (see 3.4.6).

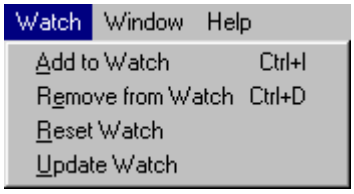


Fig. 3.4-22 *Watch* Menu of the Main Window

3.4.15.1 Add to Watch

Adds an expression to the *Watch* window. The *Add to Watch Window* opens for the user to enter the expression. Then it is placed into the *Watch* window.

Assigned keys — ***Ctrl+I***.

3.4.15.2 Remove from Watch

Removes expressions from the *Watch* window. The *Remove* window opens that contains the list of expressions viewed in the *Watch* window to select the expressions to be removed.

Assigned keys — ***Ctrl+D***.

3.4.15.3 Reset Watch

Removes all expressions from the *Watch* window.

3.4.15.4 Update Watch

Updates all expression values in the *Watch* window.

3.4.16 Window Menu of the Main Window

The menu contains tools for managing windows (Fig. 3.2-8).

3.4.16.1 Kernel Output

Places the *Kernel Output* window (see 3.4.1) on top of the other windows.

3.4.16.2 Program Output

Places the *Program Output* window (see 3.4.2) on top of the other windows.

3.4.16.3 Watch

Places the *Watch* window (see 3.4.63.4.5) on top of the other windows.

3.4.16.4 Debug

Places the *Debug* window (see 3.4.3) on top of the other windows.
Assigned keys — ***Ctrl+D***.

3.4.16.5 Clear Kernel Output

Erases the contents of the *Kernel Output* window (see 3.4.1).

3.4.16.6 Clear Program Output

Erases the contents of the *Program Output* window (see 3.4.2).

3.4.16.7 Options

The menu (Fig. 3.4-23) contains tools for setting parameters for the *Kernel Output* (see 3.4.1), *Program Output* (see 3.4.2), and *Watch* (see 3.4.6, 3.4.5) windows.

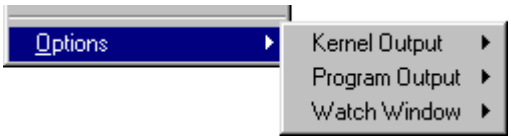


Fig. 3.4-23 *Options* Menu

3.4.16.7.1 Kernel Output

The menu (Fig. 3.4-24) contains tools for setting parameters for the *Kernel Output* window (see 3.4.1).

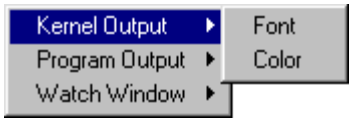


Fig. 3.4-24 *Kernel Output* Menu

3.4.16.7.1.1Font

Opens the window for setting font parameters and color.

3.4.16.7.1.2Color

Opens the window for setting background color.

3.4.16.7.2 Program Output

The menu contains tools for setting parameters for the *Program Output* window (see 3.4.2). It is identical with the *Kernel Output* menu.

3.4.16.7.3 Watch Window

The menu contains tools for setting parameters for the *Watch* window (see 3.4.6). It is identical with the *Kernel Output* menu.

3.5 Browsing

3.5.1 *Browse* Menu of the Main Window

The menu contains tools for browsing information (Fig. 3.5-1).

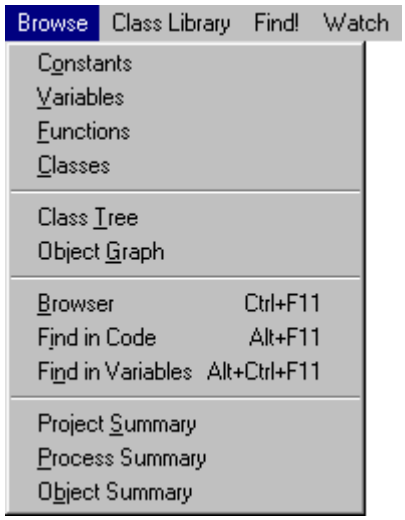


Fig. 3.5-1 *Browse* Menu of the Main Window

3.5.1.1 Constants

Opens the *Constants* window for browsing constants of an application (see 3.5.2).

3.5.1.2 Variables

Opens the *Global Variables* window for browsing global variables of an application (see 3.5.3).

3.5.1.3 Functions

Opens the *Global Functions* window for browsing global functions of an application (see 3.5.4).

3.5.1.4 Classes

Opens the *Class* window for browsing classes of an application (see 3.5.5).

3.5.1.5 Class Tree

Opens the *Class Tree* window for browsing the class inheritance hierarchy of an application (see 3.5.8).

3.5.1.6 Object Graph

Opens the *Object Browser* window for browsing class inheritance / inclusion hierarchy (see 3.5.11).

3.5.1.7 Browser

Opens the *Browser* window for searching for the names of application entities, e.g. variables (see 3.5.15).

Assigned keys — ***Ctrl+F11***.

3.5.1.8 Find in Code

Opens the *Find in Code* window for searching for a text fragment in global functions and global class fields of type **function** (see 3.5.17).

Assigned keys — ***Alt+F11***.

3.5.1.9 Find in Variables

Opens the *Find in Variables* window for searching for a text fragment in global variables of type **string** and their fields (see 3.5.18).

Assigned keys — ***Alt+Ctrl+F11***.

3.5.1.10 Project Summary

Opens the *Pluk System Summary* window that is showing the information on the current state of the Pluk-processor (see 3.5.19).

3.5.1.11 Process Summary

Opens the window for selecting a process from the list of all current processes in the computer (including those that do not use the Pluk-processor). After the selection, the *Process Summary* window will open with information about the process (see 3.5.20).

3.5.1.12 Object Summary

Opens the *Static Object List* window with the list of static objects of the Pluk-processor: global variables and global class fields (see 3.5.21).

3.5.2 Constants Window

The *Constants* window (Fig. 3.5-2) is used for browsing constants.

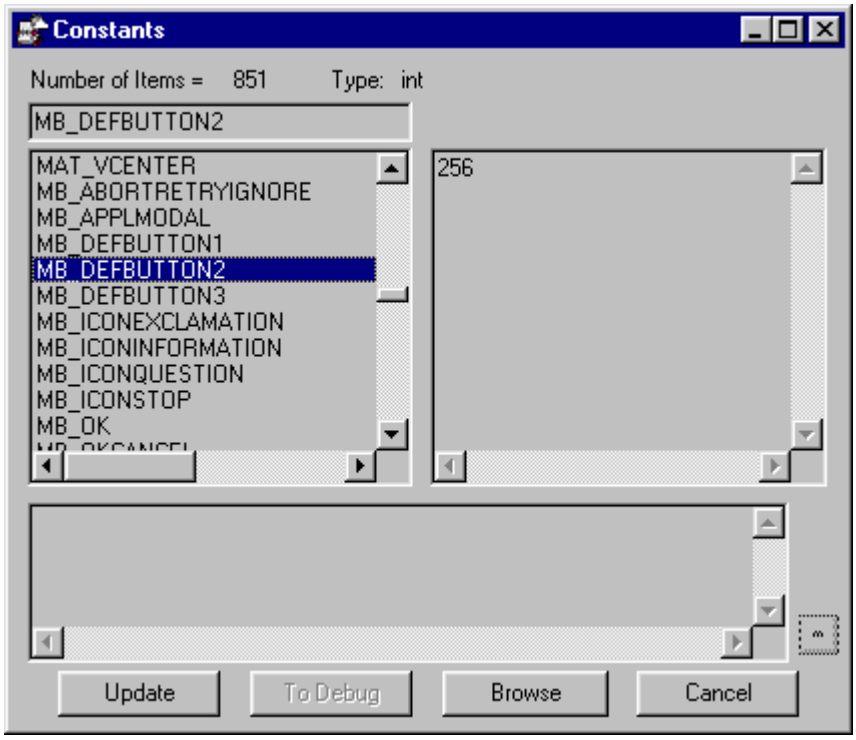


Fig. 3.5-2 *Constants* Window

3.5.2.1 Number of Items

Number of constants.

3.5.2.2 List of Constants

This box contains the list of constants. In the text box at the right, the selected constant value is shown.

Left double-click on a list item opens the value browsing window with the constant value (see 3.5.16).

3.5.2.3 Type

The type of the selected constant.

3.5.2.4 Update

Updates the list of constants.

3.5.2.5 Browse

Opens the value browsing window for the selected constant (see 3.5.16).

3.5.3 Global Variables Window

The *Global Variables* window (Fig. 3.5-3) is used for browsing global variables.

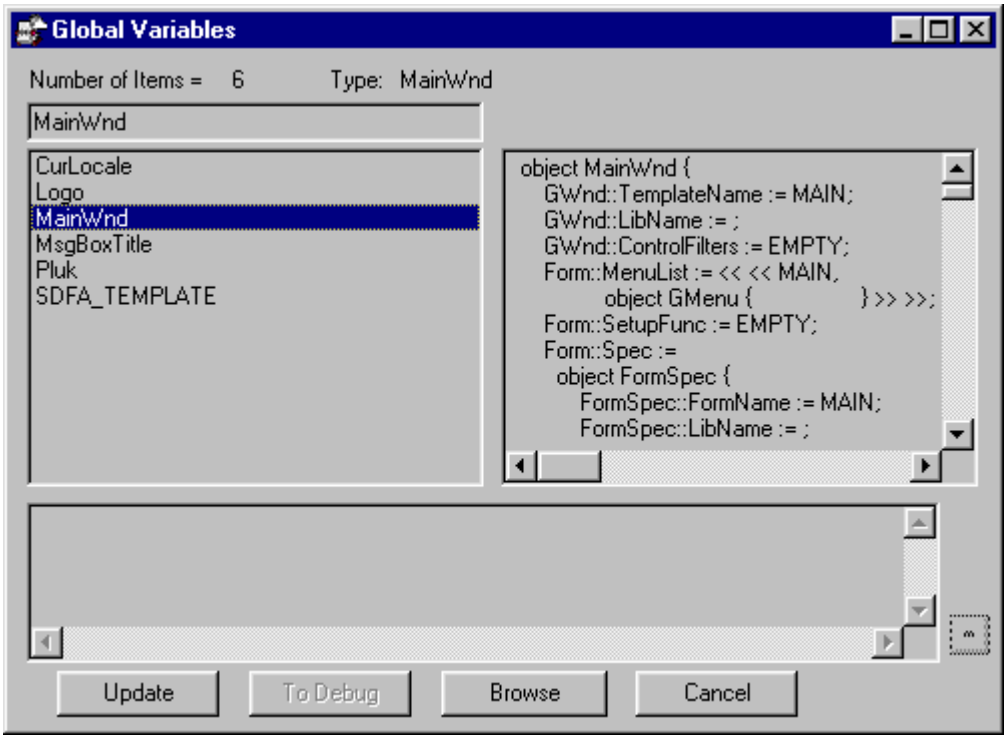


Fig. 3.5-3 *Global Variables* Window

3.5.3.1 Number of Items

Total number of global variables.

3.5.3.2 List of Global Variables

This box contains the list of global variables. At the right, there is the box that shows the value of the selected variable.

Left double-click on an item opens the value browsing window for the variable (see 3.5.16).

3.5.3.3 Type

The type of the selected global variable.

3.5.3.4 Update

Updates the list of global variables.

3.5.3.5 Browse

Opens the value browsing window for the selected global variable (see 3.5.16).

3.5.4 Global Functions Window

The *Global Functions* window (Fig. 3.5-4) is used for browsing global functions.

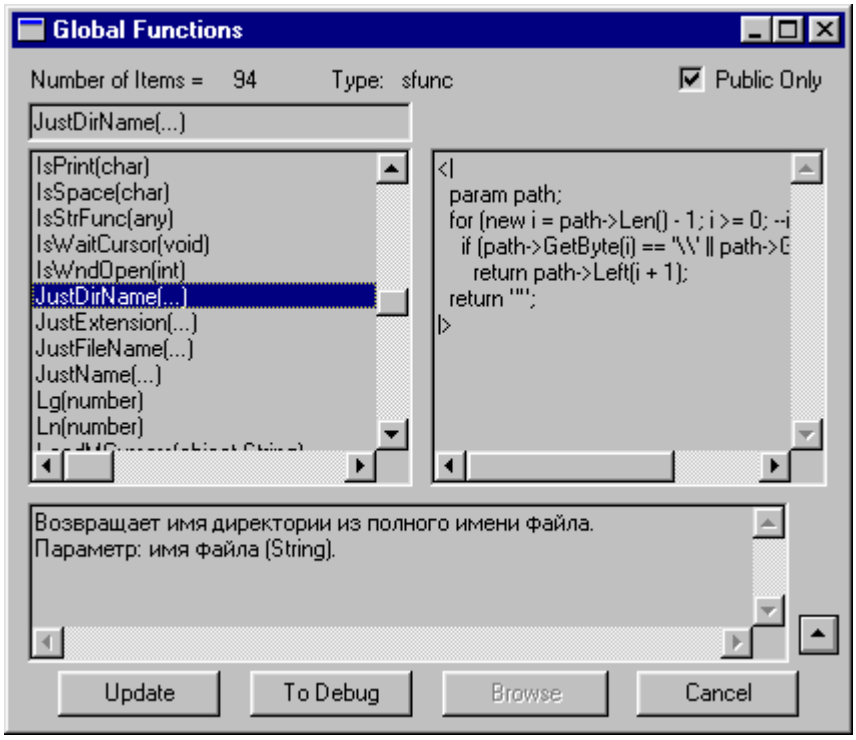


Fig. 3.5-4 *Global Functions* Window

3.5.4.1 Number of Items

Total number of global functions.

3.5.4.2 List of Global Functions

This box contains the list of global functions. In the box at the right, the body of the selected function is shown.

Left double-click on a list item opens the Edit window with the function; the cursor is positioned on the first line. If the function is loaded from a source file, the latter will be opened. If the function is loaded from a library file, a temporary source file will be opened that contains this function only. The filename begins with the string **noname**.

3.5.4.3 

Opens the *Comment* window showing system comment on the selected global function (see 3.5.6). The same comment could be viewed in the text box at the left from the button.

3.5.4.4 Type

The type of the selected function.

3.5.4.5 Public Only

If checked, only public global functions will be listed (see Pluk Language Reference Book).

3.5.4.6 Update

Updates the list of global functions.

3.5.4.7 To Debug

Loads the selected global function into the *Debug* window (see 3.4.3).

3.5.5 Class Window

The *Class* window is used for browsing application classes. Classes can be presented on one consolidated list (Fig. 3.5-5), as well as on the list partitioned into modules that form a tree structure (Fig. 3.5-6).

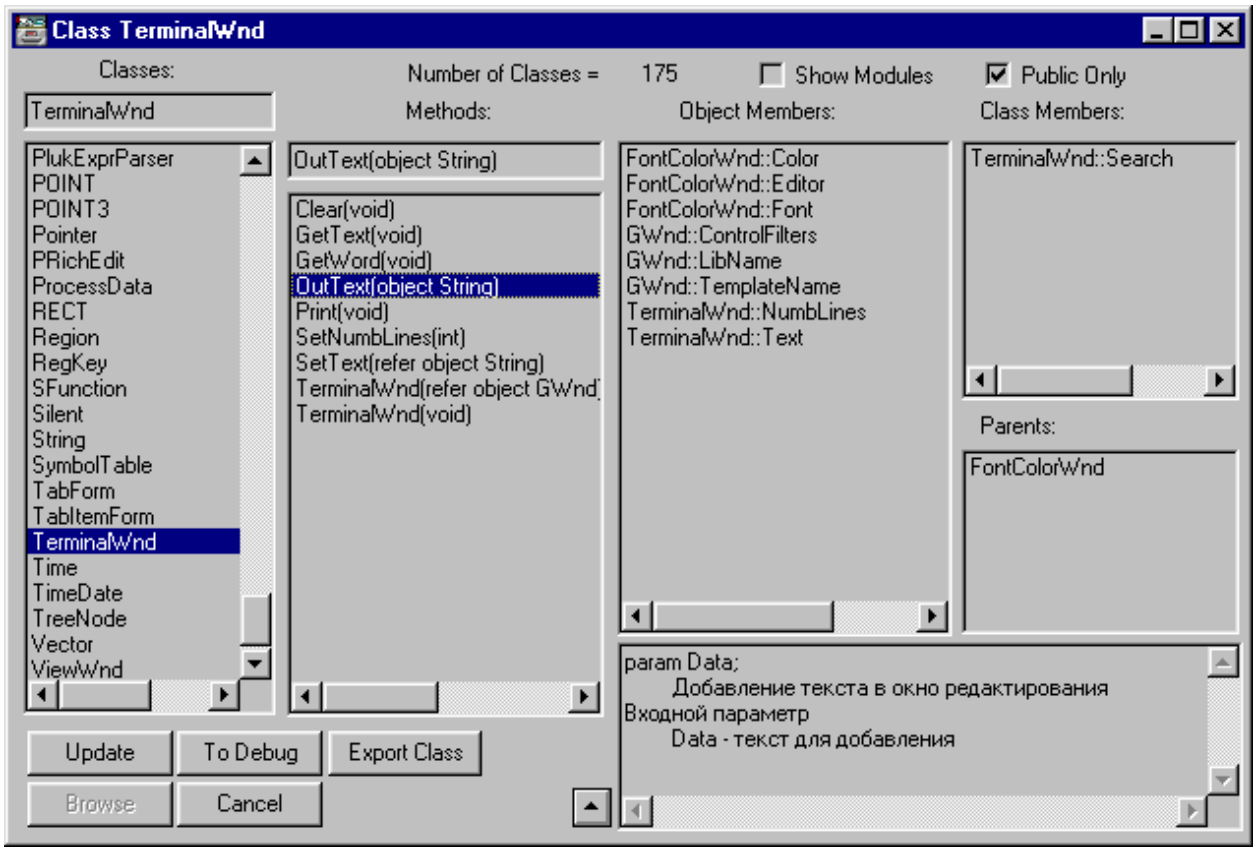


Fig. 3.5-5 *Class* Window without Module Partition

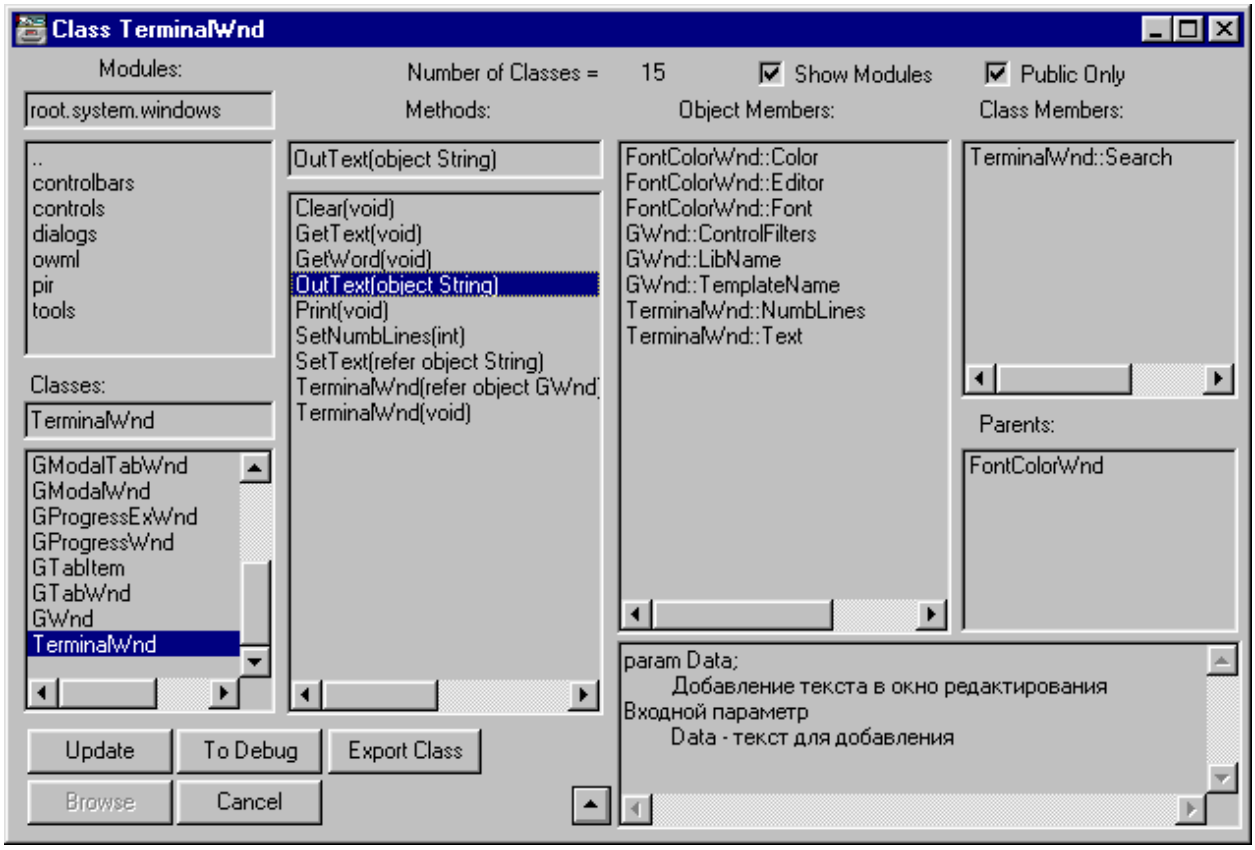


Fig. 3.5-6 *Class* Window with Module Partition

3.5.5.1 Modules

The list of modules in the current module. Left double-click on a module makes one step down the module hierarchy, this module becomes current. Left double-click on **..** makes one step up the module hierarchy, the higher-level module becomes current.

3.5.5.2 Number of Classes

If *Show Modules* is unchecked, the field shows total number of classes. If *Show Modules* is checked, the field shows total number of classes in the current module (excluding classes in sub-modules).

3.5.5.3 Classes

The list of all classes, if *Show Modules* unchecked. If *Show Modules* checked, the list of classes in the current module (excluding classes in sub-modules).

Left double-click on a list item opens the Edit window with a **class** statement; the cursor is positioned on the first line. If the class is loaded from a source file, the latter will be opened. If the class is loaded from a library file, a temporary source file will be opened that contains the **class** statement only. The filename begins with the string **noname**.

3.5.5.4 Methods

The list of methods of the selected class from the list *Classes*.

Left double-click on a list item opens the Edit window with the method; the cursor is positioned on the first line. If the method is loaded from a source file, the latter will be opened. If

the method is loaded from a library file, a temporary source file will be opened that contains this method only. The filename begins with the string **noname**.

3.5.5.5 Object Members

The list of fields in the selected class.

3.5.5.6 Class Members

The list of global fields in the selected class.
Left double-click on an item opens the value browsing window for the field (see 3.5.16).

3.5.5.7 Parents

The list of base classes of the selected class.
Left double-click on an item opens the *Classes* window for browsing the base class.

3.5.5.8 

Opens the *Comment* window (see 3.5.6) that shows one of the following:

- system comments on the class selected from the *Classes* list;
- system comments on the method selected from the *Methods* list;
- value of the global field selected from the *Class Members* list.

The comments or value could also be seen in the text box at the right of the button.

3.5.5.9 Show Modules

If unchecked, the window shows the classes on one consolidated list (Fig. 3.5-5). If checked, the classes are partitioned into modules that form a tree structure (Fig. 3.5-6).

3.5.5.10 Public Only

If checked, only public classes and public methods (see Pluk Language Reference Book) are visible on respectively the *Classes* list and the *Methods* list.

3.5.5.11 Update

Updates all lists.

3.5.5.12 To Debug

Loads the method selected from the *Methods* list into the *Debug* window (see 3.4.3).

3.5.5.13 Export Class

Saves the **class** statement and methods to file through the Save window.

3.5.5.14 Browse

Opens the value browsing window (see 3.5.16) for the global field selected from the *Class Members* list.

3.5.6 Comment Window

The *Comment* window is used for viewing system comments (Fig. 3.5-7).

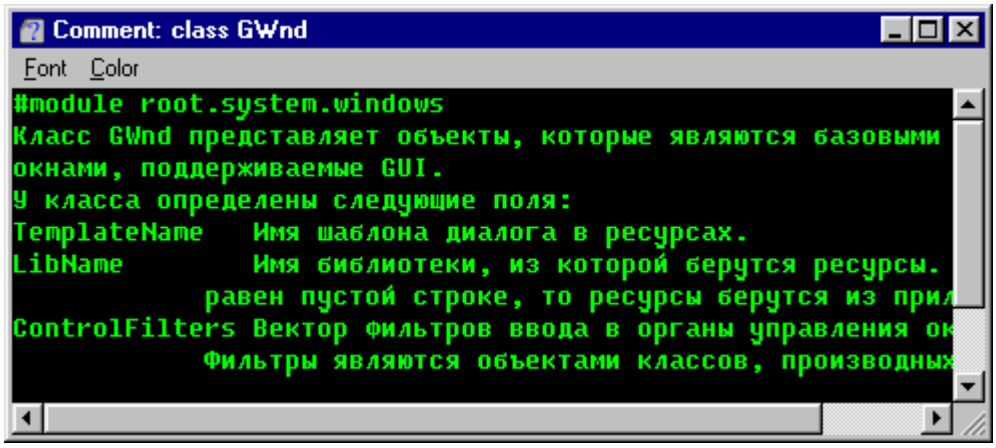


Fig. 3.5-7 *Comment* Window

3.5.7 Comment Window Menu

Fig. 3.5-8 shows the *Comment* window menu.

Fig. 3.5-8 *Comment* Window Menu

3.5.7.1 Font

Opens the window for setting font parameters and color.

3.5.7.2 Color

Opens the window for setting background color.

3.5.8 Class Tree Window

The *Class Tree* window is used for browsing the class inheritance hierarchy in an application (Fig. 3.5-9). Each class is presented in a white rectangle containing its name. The relations of inheritance are denoted by lines with two small circles: the black circle tags a base class, the white one a derived class. The user can select a class (the rectangle will turn gray), the relations of inheritance from it will be denoted by red lines.

Left double-click on a class opens the *Class* window with this class selected (see 3.5.5). Right-click opens the popup menu (see 3.5.10).

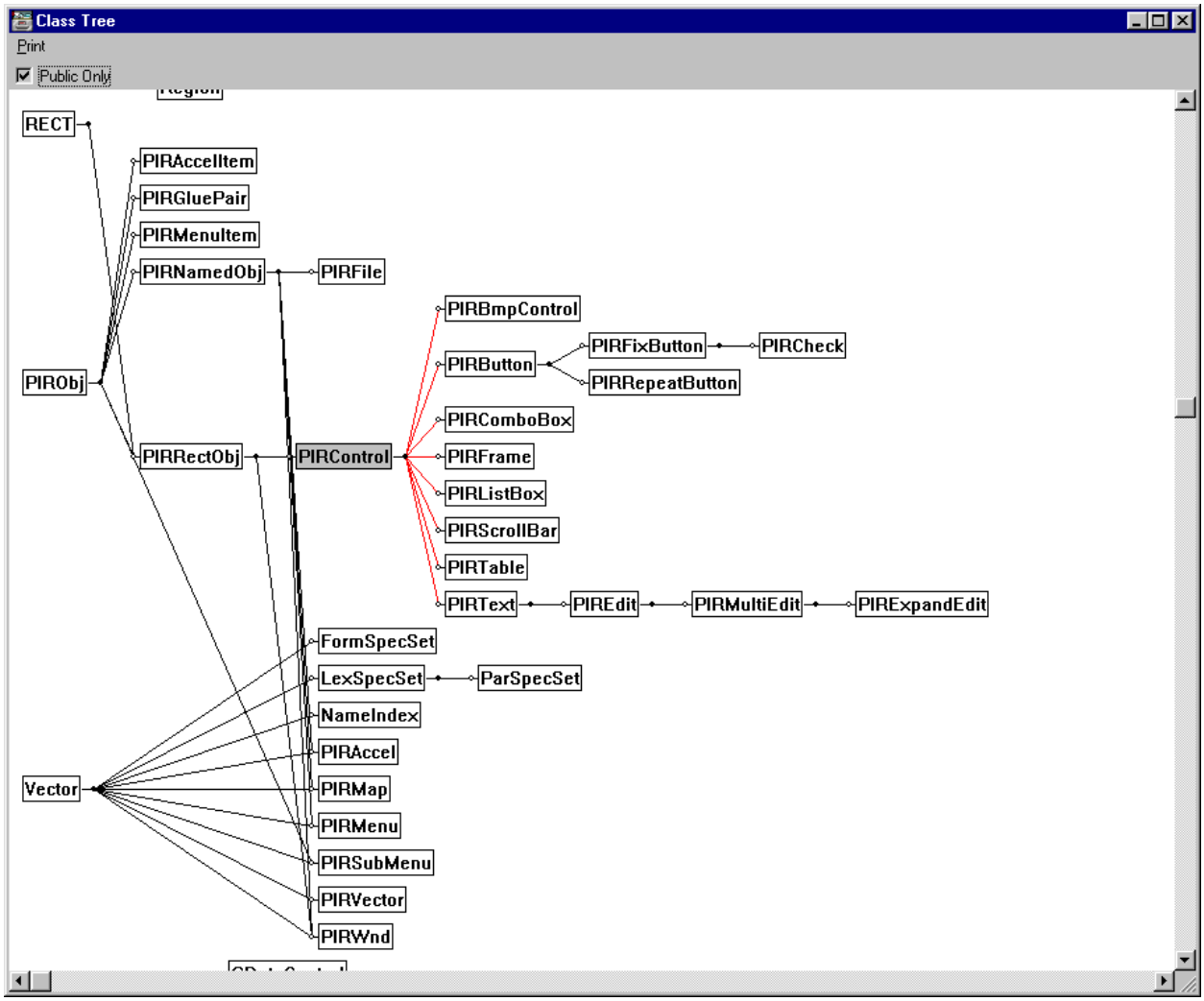


Fig. 3.5-9 *Class Tree* Window

3.5.8.1 Public Only

If checked, only public classes are visible (see Pluk Language Reference Book).

3.5.9 *Class Tree* Window Menu

Fig. 3.5-10 shows the *Class Tree* window menu.

Fig. 3.5-10 *Class Tree* Window Menu

3.5.9.1 Print

Prints out the inheritance hierarchy.

3.5.10 Class Menu in *Class Tree* Window

The menu that pops up after right-click on a class in the *Class Tree* window is shown in Fig. 3.5-11.

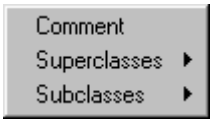


Fig. 3.5-11 Class Menu in *Class Tree* Window

3.5.10.1 Comment

Opens the *Comment* window for the class (see 3.5.6).

3.5.10.2 Superclasses

Contains the list of base classes of the class. The user can select a base class.

3.5.10.3 Subclasses

Contains the list of derived classes of the class. The user can select a derived class.

3.5.11 Object Browser Window

The *Object Browser* window is used for browsing class inheritance / inclusion hierarchy (Fig. 3.5-12). Since Pluk is a non-typed language, a class field or vector element can contain any type of value, including an empty one. At any given point in time, however, they contain a value of a certain type. The window allows browsing all global variables and static class fields and registering the class inheritance / inclusion hierarchy at given points in time.

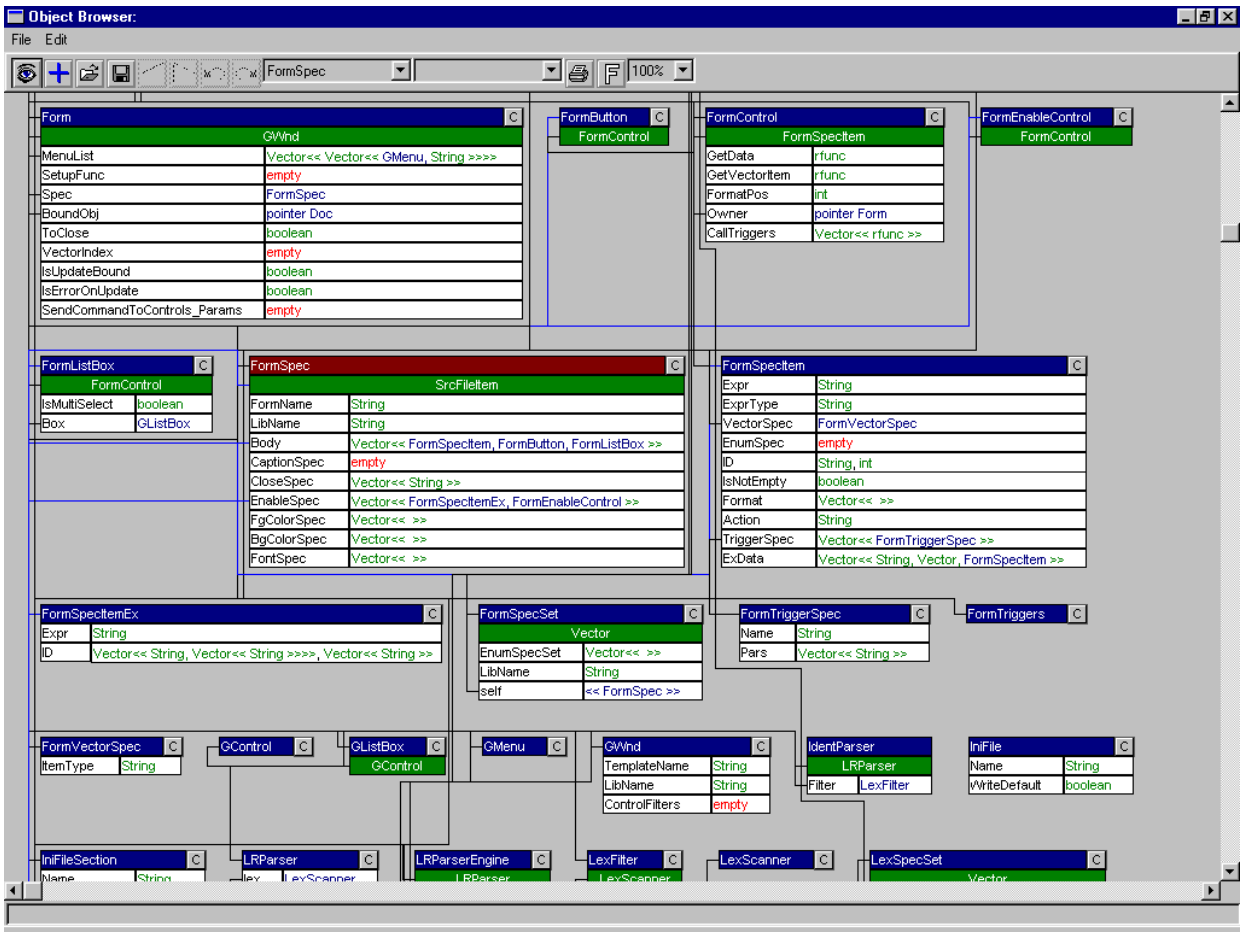


Fig. 3.5-12 *Object Browser* Window

Each class is presented in table form (Fig. 3.5-13). The class name is displayed in the table title bar (for example, **FormSpec**). Names of base classes (for example, **SrcFileItem**) are presented on the next row on green background. The left column contains the names of class fields. The same field in different objects can contain different types of values. All of the field types found are listed (separated with commas) in the right column (in our example, the field **FormName** contains an object of the type **String**, the field **Body** contains a type **Vector** object comprised of objects of the classes **FormSpecItem**, **FormButton**, **FormListBox**, the field **CaptionSpec** is empty). Intrinsic types are shown in green, user-defined classes are blue, the empty value is red. Vector is shown as follows**: Vector<<type$_1$, type$_2$, …>>**, where **type$_i$** is one of possible types of a vector element.

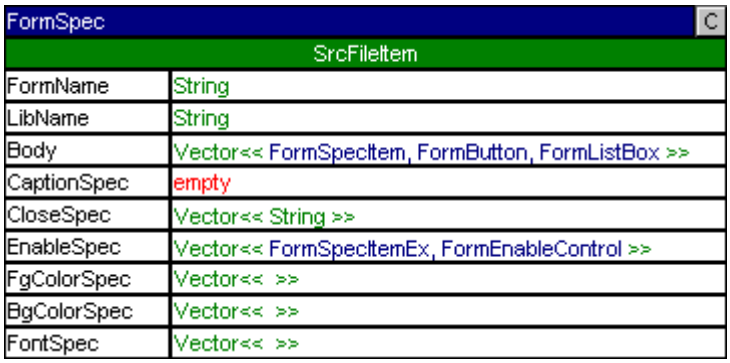


Fig. 3.5-13 Class Presentation in the *Object Browser* Window

C-button on the title bar toggles on / off system comments on the class. Fig. 3.5-14 shows the class with the system comment.

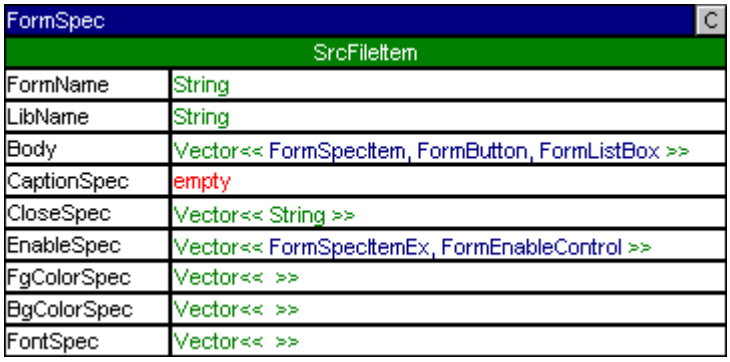


Fig. 3.5-14 Class Presentation with the System Comment in the *Object Browser* Window

The relations of inheritance / inclusion are denoted by lines. The user can select a class. The color of the selected class title bar changes from blue to purple, the color of the lines denoting the relations of inheritance / inclusion into the class from black to blue (Fig. 3.5-12). Thus the user can find out which classes are contained in the selected class and from which classes it was inherited. But there is no change in the color of the lines denoting the relations of inheritance from / inclusion of the selected class. To find out which classes contain the selected class and which ones were inherited from it, see 3.5.12.9.

The user can drag-and-drop classes using left mouse button.

3.5.12 Object Browser Window Toolbar

3.5.12.1 

Same as *New* in the main window menu (see 3.5.13.1.1).

3.5.12.2 

Same as *Add* in the main window menu (see 3.5.13.1.2).

3.5.12.3 

Same as *Open* in the main window menu (see 3.5.13.1.3).

3.5.12.4 

Same as *Save* in the main window menu (see 3.5.13.1.4).

3.5.12.5 

Same as *Back* in the main window menu (see 3.5.13.2.4).

3.5.12.6 

Same as *Forward* in the main window menu (see 3.5.13.2.3).

3.5.12.7 

Same as *Undo* in the main window menu (see 3.5.13.2.1).

3.5.12.8 

Same as *Redo* in the main window menu (see 3.5.13.2.2).

3.5.12.9 

The combo box contains the list of classes in the hierarchy. The class selected in the box will also be selected in the window.

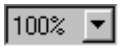
If the selected class is contained in another or another class is inherited from it, the list of such classes will be given in the combo box at the right. The class selected in this box will also be selected in the window.

3.5.12.10 

Same as *Print* in the main window menu (see 3.5.13.1.6).

3.5.12.11 

Same as *Font* in the main window menu (see 3.5.13.2.5).

3.5.12.12 

A scaling tool for zooming the hierarchy being browsed.

3.5.13 Object Browser Window Menu

Fig. 3.5-15 shows the *Object Browser* window menu.

Fig. 3.5-15 *Object Browser* Window Menu

3.5.13.1 File

Contains tools for working with class inheritance / inclusion hierarchy and hierarchy storage files (*.clj) (Fig. 3.5-16).

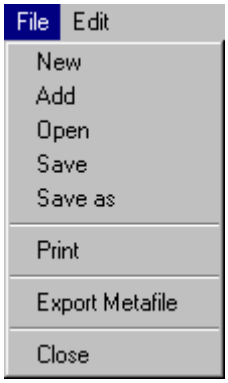


Fig. 3.5-16 *File* Menu

3.5.13.1.1 New

Creates the hierarchy for the current state of the application. The hierarchy is not yet associated with a file.

3.5.13.1.2 Add

Adds the classes, fields and base classes that have appeared in the current state of the application to the existing hierarchy. The disappeared classes, fields and base classes will still remain in the hierarchy.

3.5.13.1.3 Open

Opens a hierarchy storage file through the Open window.

3.5.13.1.4 Save

Saves a hierarchy storage file. If the hierarchy is not yet associated with a file, the Save window will open.

3.5.13.1.5 Save as

Saves the hierarchy into another file through the Save window.

3.5.13.1.6 Print

Prints out the hierarchy. First, the hierarchy appears in the *Preview* window (Fig. 3.5-17) in reduced form on one or several pages. The user can choose between *Portrait* and *Landscape* page layouts. She can move the hierarchy in a drag-and-drop mode by left-clicking on the rectangle border. The user can proportionally resize the hierarchy in the same mode by left-clicking on one of the four small rectangles in the corners of the border. Number of pages shown always adjusts to resizing to provide for better fit: additional pages appear when the image becomes too big, and extra pages disappear when it gets too small.

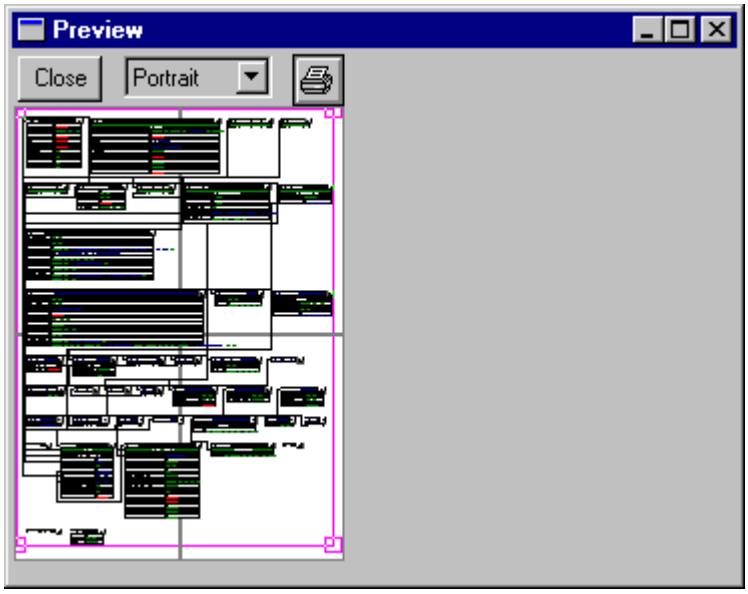


Fig. 3.5-17 *Preview* Window for Printing out the Class Inheritance / Inclusion Hierarchy

To start printing, the user should press the "printer" button.

3.5.13.1.7 Export Metafile

Saves the hierarchy to a metafile (*.wmf). First, the hierarchy appears in the *Preview* window (Fig. 3.5-18) in reduced form on one or several pages. The user can choose between *Portrait* and *Landscape* page layouts. She can move the hierarchy in a drag-and-drop mode by left-clicking on the rectangle border. The user can proportionally resize the hierarchy in the same mode by left-clicking on one of the four small rectangles in the corners of the rectangle border. Number of pages shown always adjusts to resizing to provide for better fit: additional pages appear when the image becomes too big, and extra pages disappear when it gets too small.

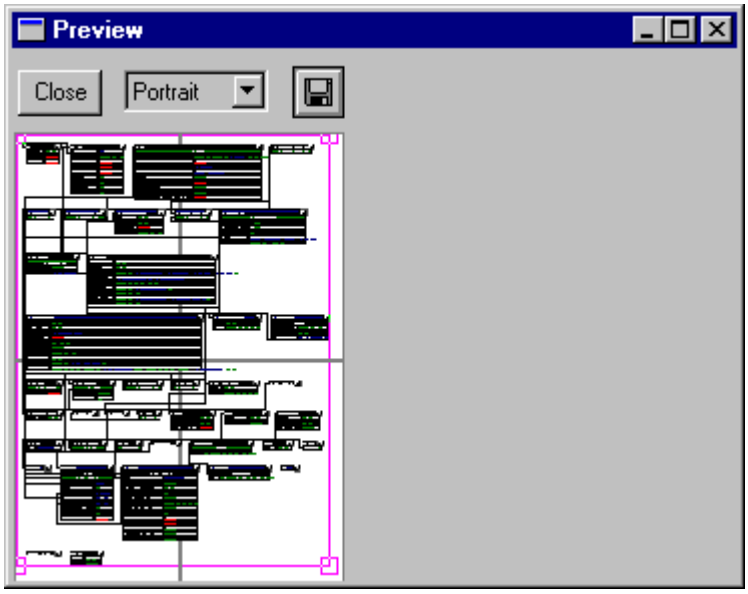


Fig. 3.5-18 *Preview* Window for Saving the Class Hierarchy to Metafile

To save the hierarchy, the user should press the "floppy" button. The Save window opens for the user to enter a basic name (should not coincide with any filename). Each page is saved

into a metafile with the name combined of the basic name and the page number. Then there will be an option either to save in color or black-and-white form.

3.5.13.1.8 Close

Closes the window.

3.5.13.2 Edit

Contains tools for editing class inheritance / inclusion hierarchy (Fig. 3.5-19).

Fig. 3.5-19 *Edit* Menu

3.5.13.2.1 Undo

Cancels the last change made in the hierarchy.

3.5.13.2.2 Redo

Restores the change canceled by *Undo*.

3.5.13.2.3 Forward

Restores the class selection canceled by *Back*.

3.5.13.2.4 Back

Selects the class selected one step before last.

3.5.13.2.5 Font

Opens the window for setting font parameters.

3.5.13.2.6 Open Comments

Opens system comments on all classes.

3.5.13.2.7 Close Comments

Closes system comments on all classes.

3.5.13.2.8 Arrange Tables

Arranges the classes in the hierarchy in alphabetical order across the screen.

3.5.13.2.9 Refresh

Refreshes the hierarchy.

3.5.14 Right-click Menu in the *Object Browser* Window

If the user right-clicks off the classes area in the *Object Browser* window, the popup menu will open (Fig. 3.5-20) whose items are identical with the ones in the Edit window menu (see 3.5.13.2).

Fig. 3.5-20 Off-Class Right-click Menu

If the user right-clicks on a class name, the popup menu will contain *Open Browser* on top (Fig. 3.5-21). Selecting this item results in the opening of the *Class* window (see 3.5.5) that contains the selected class.

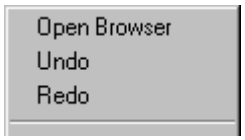


Fig. 3.5-21 Top of the On-Class Right-click Menu

If the user right-clicks on a base class name, *Go to* + the base class name will be added on top of the menu (Fig. 3.5-22). Selecting this item selects the base class.

If the user right-clicks on the field that refers to a class, the popup menu will contain *Go to* + the contained class name (Fig. 3.5-22). Selecting this item selects the referred-to class.

Fig. 3.5-22 Top of the On-Base-Class or On-Refer-to-Class Right-click Menu

If the user right-clicks on the field that refers to several classes, the popup menu will contain *Go to* menu listing the contained classes (Fig. 3.5-23). Selecting one of the items will select the referred-to class.

Fig. 3.5-23 Top of the On-Refer-to-Classes Right-click Menu

3.5.15 Browser Window

The *Browser* window is used for searching names in an application (Fig. 3.5-24). The user can search for:

- constant name;
- global variable name;
- global function name;
- class name;
- method name without reference to class name;
- method name with reference to class name;
- class field (including global) name without reference to class name;
- class field (including global) name with reference to class name.

Fig. 3.5-24 *Browser* Window

The user should enter the name to be found into the top text box and click *OK* or press ***Enter***. Alternatively, the entry can be made from the list of the items found earlier by selecting an item.

The user can set a regular expression as a search name. Following are the most frequent constructions:

[xyz]	character belongs to the set {x, y, z}
[~xyz]	character does not belong to the set {x, y, z}
[x-y]	character in the range between x and y
.	any character
?	zero or one occurrences of the preceding pattern
+	one or more occurrences of the preceding pattern
*	zero or more occurrences of the preceding pattern

For example, **.*** matches any character pattern, including an empty one. For the details, refer to Standard Library Reference Book (lrparser library).

3.5.15.1 Update

Updates information on the names in the application.

3.5.15.2 OK

Searches for the name. If several entities under the same name have been found, each entity will be presented in the corresponding window:

- the *Constants* window (see 3.5.2) will show the constant;
- the *Global Variables* window (see 3.5.3) will show the global variable;
- the *Global Functions* window (see 3.5.4) will show the global function;
- the *Class* window (see 3.5.5) will show the class;
- the *Class* window (see 3.5.5) will show the method;
- the *Class* window (see 3.5.5) will show the field.

3.5.16 Value Browsing Window

The value browsing window is used for browsing the value of a constant, variable, function, and arbitrary expression. The window works in one of three modes depending on the type of the value being browsed:

- intrinsic type (Fig. 3.5-25);
- type **String** (Fig. 3.5-26);
- class (Fig. 3.5-28).

3.5.16.1.1 Update

Updates the expression value.

3.5.16.1.2 Back

Returns to the previous expression.

3.5.16.1.3 Single Window

If unchecked, the new expression will be browsed in another window. If checked, the window will be replaced in accordance with the new expression.

3.5.16.2 Intrinsic Type Value Browsing Window

Fig. 3.5-25 Intrinsic Type Value Browsing Window

The type of the value appears in the window title bar. The expression being browsed appears in the top text box. The user can change the expression and press ***Enter***. The bottom text area contains the value of the expression.

3.5.16.3 Type **String** Value Browsing Window

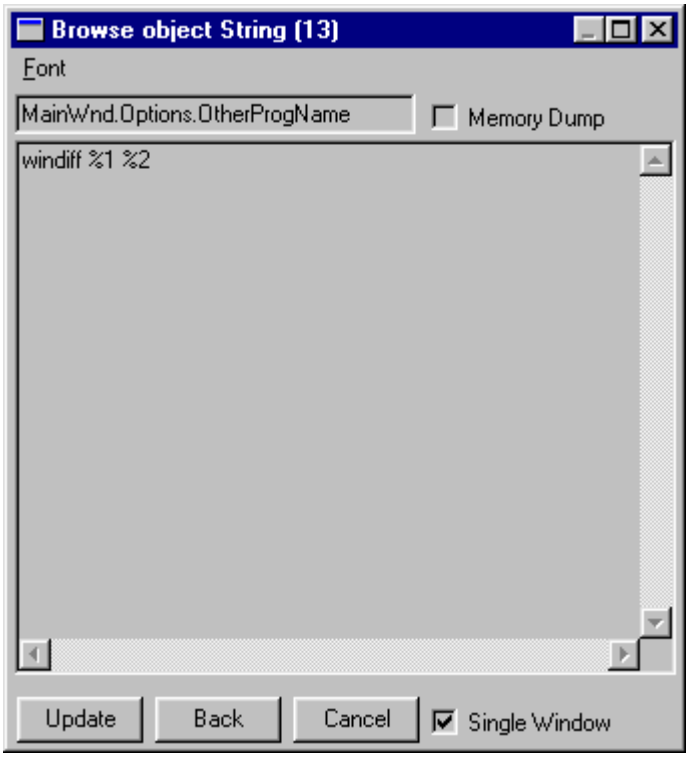


Fig. 3.5-26 Type String Value Browsing Window

The value type and string length appear in the title bar. The expression being browsed appears in the top text box. The user can change the expression and press ***Enter***. The bottom text area contains the value of the expression.

3.5.16.3.1 Memory Dump

If unchecked, the value is shown as a string of characters (Fig. 3.5-26). If checked, the value is shown as binary code, with the string shown at the right. (Fig. 3.5-27).

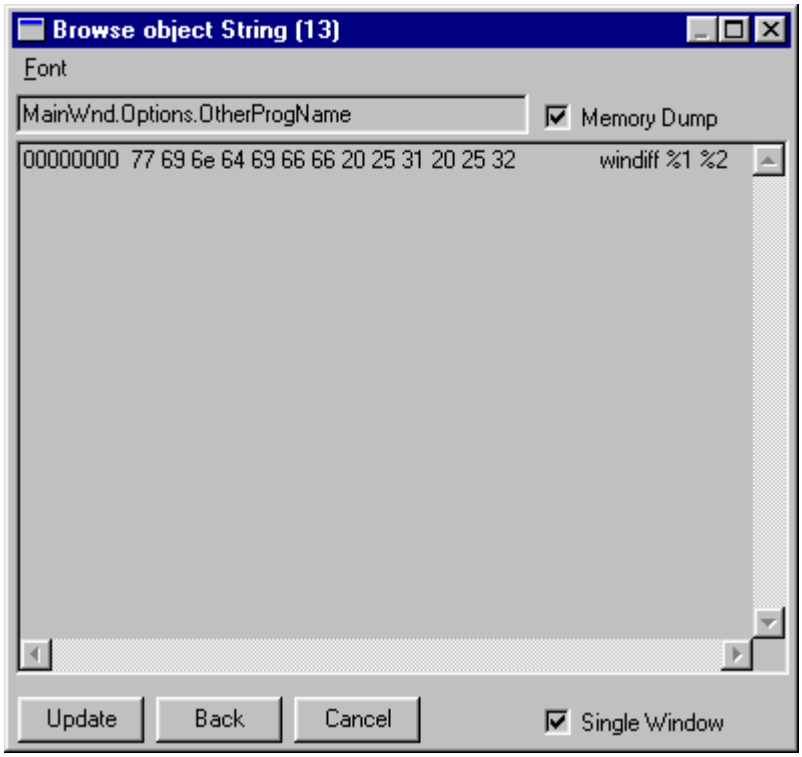


Fig. 3.5-27 Type String Value Browsing Window with Memory Dump Checked

3.5.16.3.2 Font

Opens the window for setting font parameters.

3.5.16.4 Object Value Browsing Window

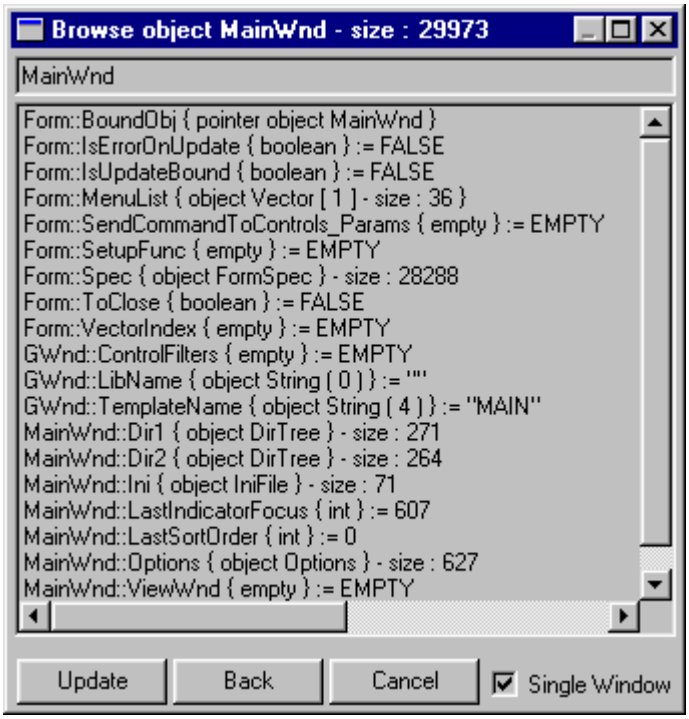


Fig. 3.5-28 Object Value Browsing Window

The value type and size (in bytes) appear in the title bar. The expression being browsed appears in the top text box. The user can change the expression and press ***Enter***. The bottom list contains the object's fields. Left double-click on a field makes one step down the inclusion hierarchy.

3.5.17 Find in Code Window

The *Find in Code* window is used for searching text in global functions and global class fields of type **function** (Fig. 3.5-29).

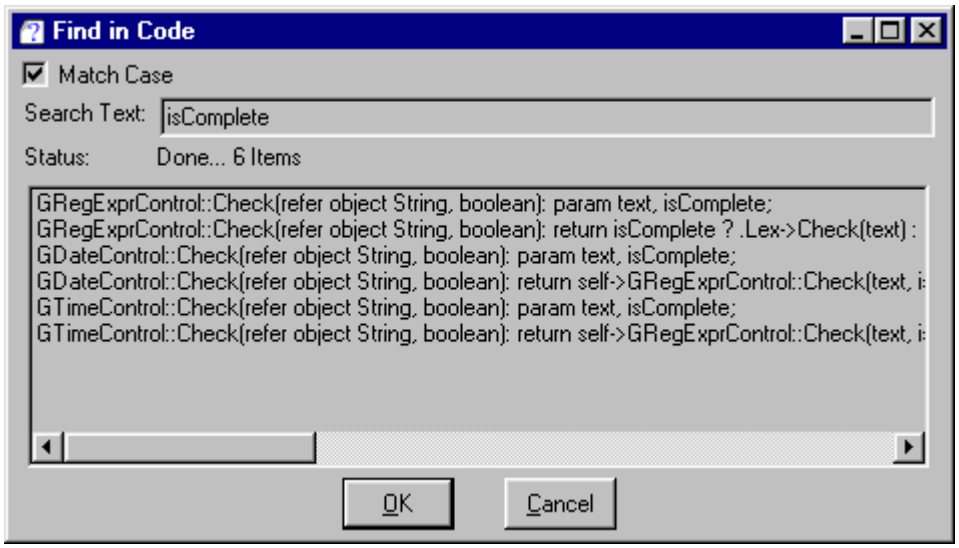


Fig. 3.5-29 *Find in Code* Window

3.5.17.1 Search Text

Text to be found in functions. The search process starts after clicking *OK* or pressing ***Enter***.

3.5.17.2 Match Case

If checked, the search is case-sensitive.

3.5.17.3 Status

Shows *Working…* during the search process. When the search has finished, it will show *Done…* and number of functions found.

3.5.17.4 List of the Functions Found

Left double-click on a line opens the Edit window with the function with the cursor positioned on the first line. If the function is loaded from a source file, this file will be opened. If the function is loaded from a library file, a temporary source file will be opened containing this function only. The filename begins with the string **noname**.

3.5.18 Find in Variables Window

The *Find in Variables* window is used for searching text in global variables of type **string** and their fields (Fig. 3.5-30).

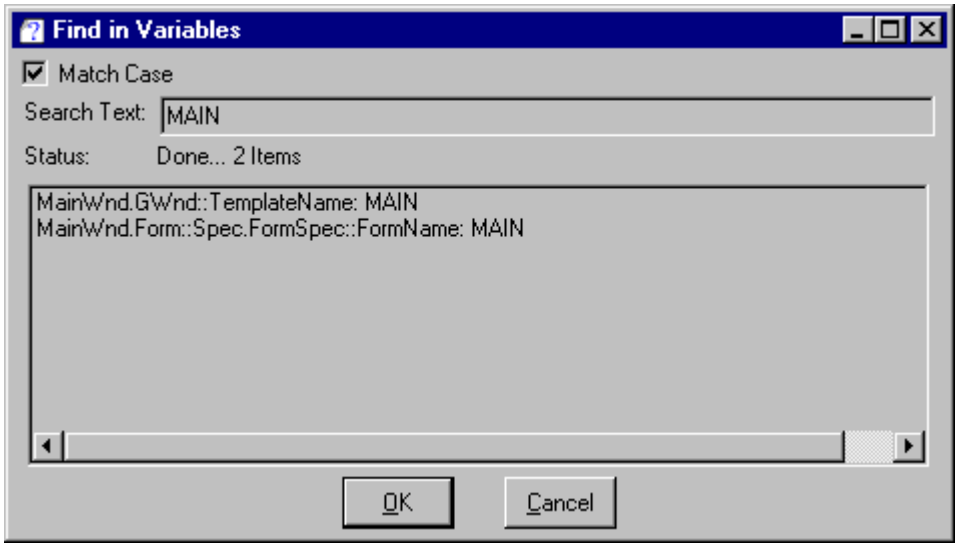


Fig. 3.5-30 *Find in Variables* Window

3.5.18.1 Search Text

Text to be found in global variables of type **string** and their fields. The search process starts after clicking *OK* or pressing ***Enter***.

3.5.18.2 Match Case

If checked, the search is case-sensitive.

3.5.18.3 Status

Shows *Working…* during the search process. When the search has finished, it will show *Done…* and number of variables and their fields found.

3.5.18.4 List of Found Variables and Fields

Left double-click on a line opens the *Add to Watch Window* with the global variable or its field (with the full path to it). The user can change the name. Then the expression is viewed in the *Run on Level* window (see 3.4.12).

3.5.19 Pluk System Summary Window

The *Pluk System Summary* window provides information on the current state of the Pluk-processor (Fig. 3.5-31).

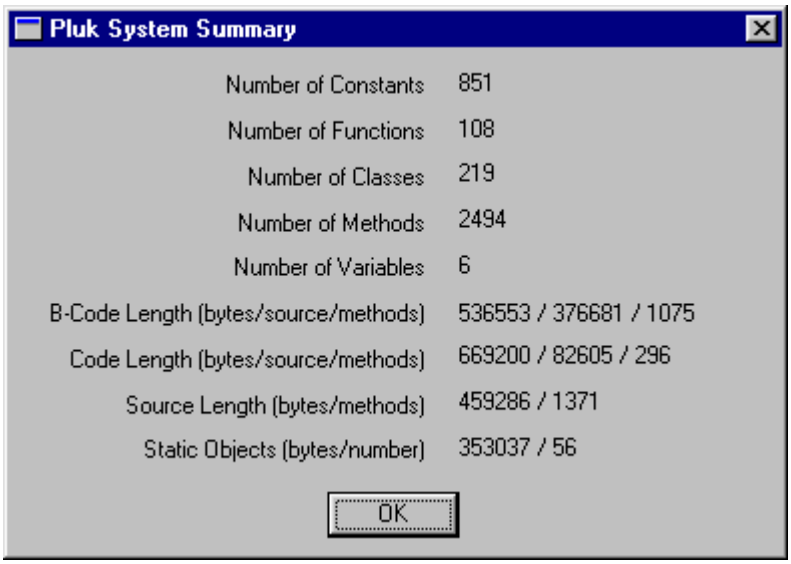


Fig. 3.5-31 *Pluk System Summary* Window

3.5.19.1 Number of Constants

Total number of constants.

3.5.19.2 Number of Functions

Total number of global functions.

3.5.19.3 Number of Classes

Total number of classes.

3.5.19.4 Number of Methods

Total number of methods.

3.5.19.5 Number of Variables

Total number of global variables.

3.5.19.6 B-Code Length (bytes/source/methods)

The volume of the functions that were interpreted to B-code by the Pluk-processor (see Pluk Language Reference Book). The characteristics measured:

- B-code size in bytes;
- source code size in bytes;
- number of functions.

3.5.19.7 Code Length (bytes/source/methods)

The volume of the functions that were interpreted to machine code by the Pluk-processor (see Pluk Language Reference Book). The characteristics measured:

- B-code size in bytes;
- source code size in bytes;
- number of functions.

3.5.19.8 Source Length (bytes/methods)

The volume of the functions that were interpreted by the Pluk-processor. The characteristics measured:

- source code size in bytes;
- number of functions.

Number of functions that are interpreted by the Pluk-processor is less than total number of functions and methods since part of the latter are written in C++.

3.5.19.9 Static Objects (bytes/number)

The volume of static objects, i.e. global variables and global class fields. The characteristics measured:

- size of static objects in bytes;
- number of static objects.

3.5.20 Process Summary Window

The *Process Summary* window provides the information about the current state of a process (Fig. 3.5-32).

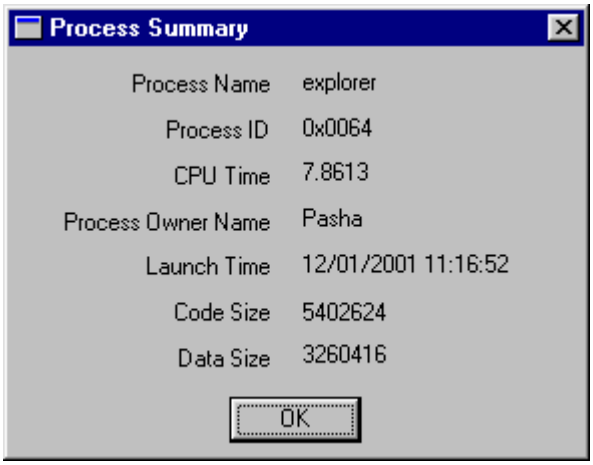


Fig. 3.5-32 *Process Summary* Window

3.5.20.1 Process Name

The name of the process.

3.5.20.2 Process ID

Process ID in the operating system.

3.5.20.3 CPU Time

Process performance period in seconds.

3.5.20.4 Process Owner Name

The user that owns the process.

3.5.20.5 Launch Time

Date and time of the process's startup.

3.5.20.6 Code Size

Size of the code segment.

3.5.20.7 Data Size

Size of the data segment.

3.5.21 Static Object List Window

The *Static Object List* window presents the list of static objects of the Pluk-processor: global variables and global class fields (Fig. 3.5-33). A list item contains the size and name of an object. The static objects are sorted by size.

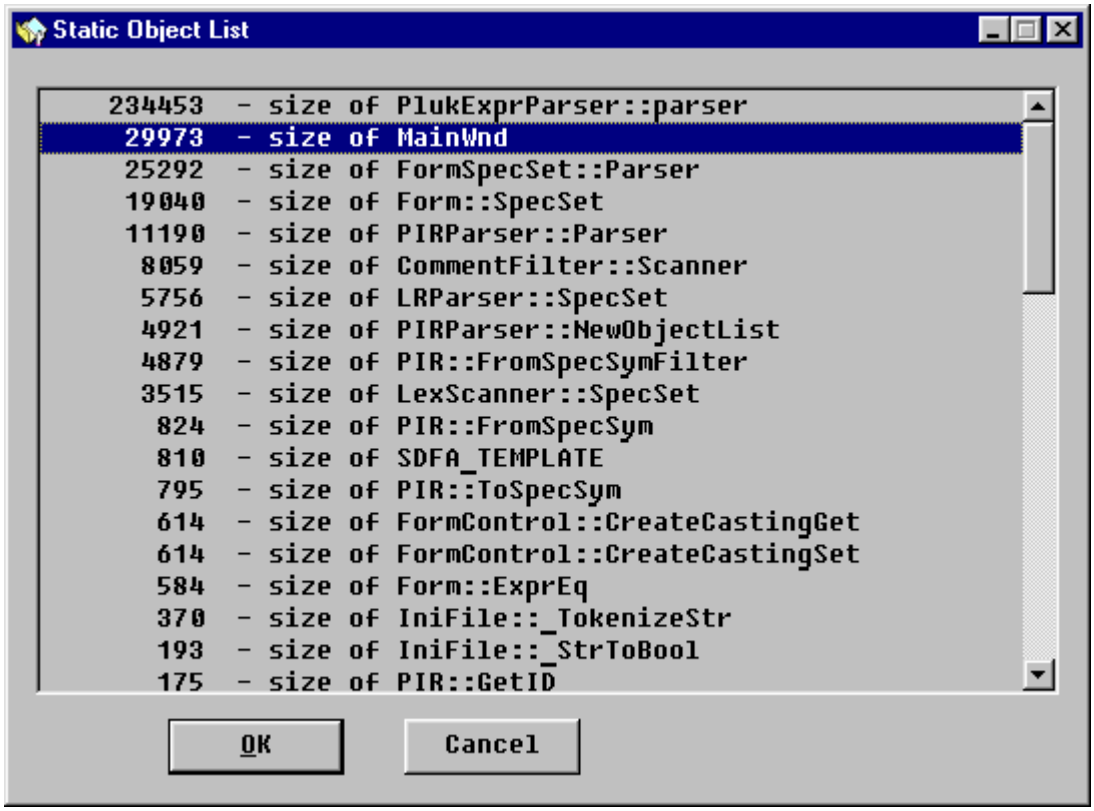


Fig. 3.5-33 *Static Object List* Window

3.5.22 Find Menu of the Main Window

Opens the *Find in Files* window to search for text in files (see 3.5.23).

3.5.23 Find in Files Window

The *Find in Files* window is used for finding text in files (Fig. 3.5-34).

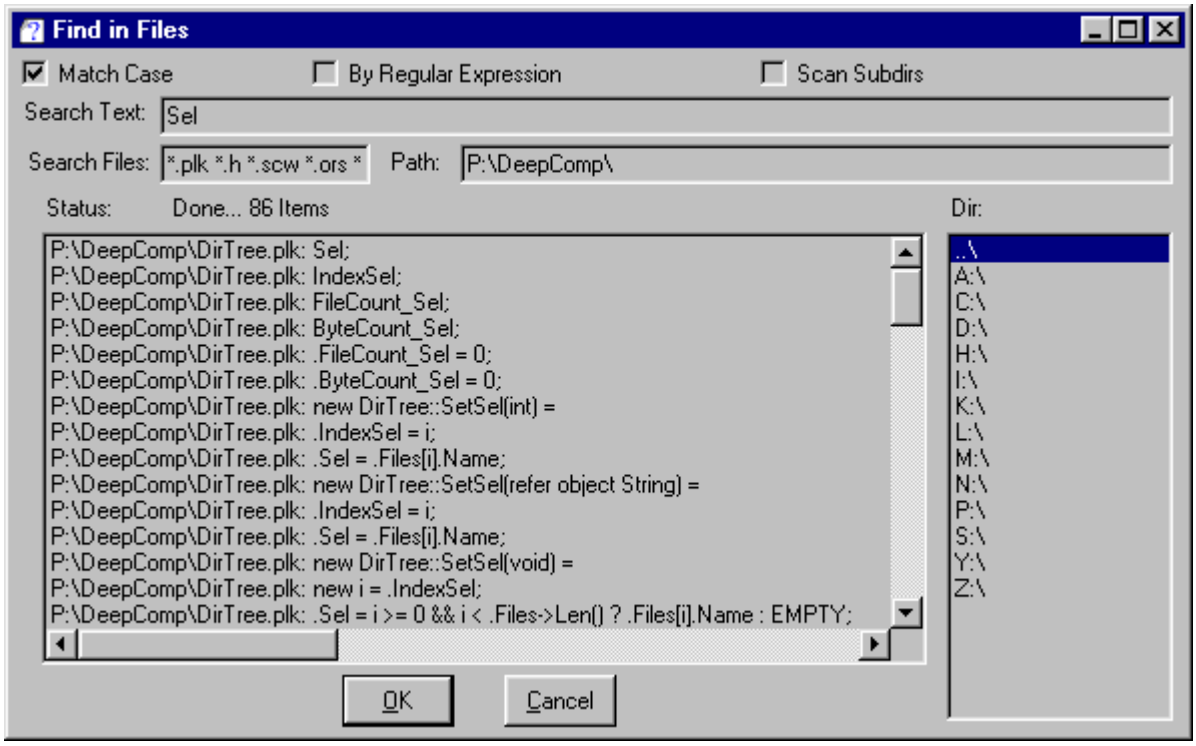


Fig. 3.5-34 *Find in Files* Window

3.5.23.1 Search Text

Text to be found in files. The search process starts after clicking *OK* or pressing ***Enter***.

3.5.23.2 Search Files

Extensions of the files to be searched.

3.5.23.3 Path

The directory to be searched. The user fills in this box by selecting the directory from the *Dir* list.

3.5.23.4 Dir

Directory hierarchy. Left double-click on a directory makes one step down the hierarchy, the directory becomes current. Left double-click on **..** makes one step up the directory hierarchy, the higher-level directory becomes current.

3.5.23.5 Match Case

If checked, the search is case-sensitive.

3.5.23.6 By Regular Expression

If checked, a regular expression should be entered into *Search Text*. Following are the most frequent constructions:

[xyz]	character belongs to the set {x, y, z}
[~xyz]	character does not belong to the set {x, y, z}
[x-y]	character in the range between x and y
.	any character
?	zero or one occurrences of the preceding pattern

+	one or more occurrences of the preceding pattern
*	zero or more occurrences of the preceding pattern

For example, **.*** matches any character pattern, including an empty one. For the details, refer to Standard Library Reference Book (lrparser library).

3.5.23.7 Scan Subdirs

If unchecked, only the *Path* directory will be searched. If checked, the *Path* directory with all subdirectories will be searched.

3.5.23.8 Status

Shows *Working…* during the search process. When the search has finished, it will show *Done…* and number of files found.

3.5.23.9 List of Files and Lines found

Left double-click on a line opens the Edit window with the file, with the cursor positioned on the line found.

3.5.24 Window Menu of the Main Window

Contains tools for working with windows (Fig. 3.2-8).

3.5.24.1 Window List

Opens the list of all browsing windows. Via the *Window List* window, the user selects a window and places it on top of other windows.

Assigned keys — ***Ctrl+L***.

3.5.24.2 Clear All Browsers

Closes all browsing windows.

3.5.24.3 Minimize All Browsers

Minimizes all browsing windows.

3.5.25 Edit Window Menu

Fig. 3.2-3 shows the Edit window menu.

3.5.25.1 Search

Contains tools for browsing information (Fig. 3.2-7).

3.5.25.1.1 Browser

Opens the *Browser* window for searching for names in an application, e.g. names of variables (see 3.5.15).

Assigned keys — ***Ctrl+F11***.

3.5.25.1.2 Find in Code

Opens the *Find in Code* window for searching for text in global functions and global class fields of type **function** (see 3.5.17).

Assigned keys — ***Alt+F11***.

3.5.25.1.3 Find in Variables

Opens the *Find in Variables* window to search for text in global variables of type **string** and their fields (see 3.5.18).

Assigned keys — ***Alt+Ctrl+F11***.

3.5.26 Debug Window Menu

Fig. 3.4-6 shows the *Debug* window menu.

3.5.26.1 Search

Contains tools for browsing information (Fig. 3.4-11).

3.5.26.1.1 Browser

Opens the *Browser* window for searching for names in an application, e.g. names of variables (see 3.5.15).

Assigned keys — ***Ctrl+F11***.

3.5.26.1.2 Find in Code

Opens the *Find in Code* window for searching for text in global functions and global class fields of type **function** (see 3.5.17).

Assigned keys — ***Alt+F11***.

3.5.26.1.3 Find in Variables

Opens the *Find in Variables* window to search for text in global variables of type **string** and their fields (see 3.5.18).

Assigned keys — ***Alt+Ctrl+F11***.

3.6 Class Library

The description of the class library is beyond the scope of this manual.

3.7 Help

The IDE provides the user with information about itself, Pluk Language and other languages based on the latter.

3.7.1 Help Menu of the Main Window

Contains tools for working with help information (Fig. 3.7-1).

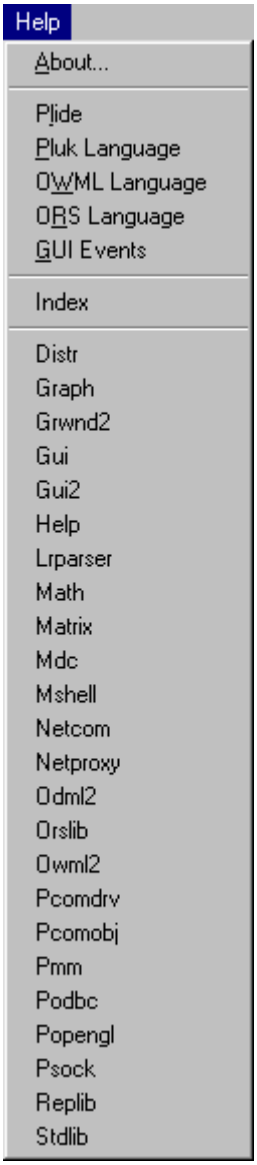


Fig. 3.7-1 *Help* Menu of the Main Window

3.7.1.1 About

Opens the window with the version number and copyright information.

3.7.1.2 Plide

Opens this User Guide.

3.7.1.3 Pluk Language

Opens Pluk Language Reference Book.

3.7.1.4 OWML Language

Opens OWML Reference Book.

3.7.1.5 ORS Language

Opens ORS Language Reference Book.

3.7.1.6 GUI Events

Opens GUI Events Reference Book.

3.7.1.7 Index

Opens the alphabetical index created for standard libraries.

3.7.1.8 Standard Libraries

Selection of an item below the *Index* opens the reference book on the selected standard library (see 4.1).

4 APPENDIX

4.1 System Libraries

Library	Field of Application
distr	Creation and installation of distributive
graph	Graphs
grwnd2	Graphs with two cursors
gui	Window interface
gui2	Additional window interface
help	Help system (for *.hlp file)
lrparser	Syntax and grammar parser
math	Mathematical classes
matrix	Mathematical vectors and matrixes
mdc	Graphic interface
mshell	Extra services of the operating system
netcom	Networking at object level
netproxy	Remote objects
odml2	Object To Database Mapping Language support
orslib	Object Report Schema Language support
owml2	Object To Window Mapping Language support
pcomdrv	Com-ports
pcomobj	COM-objects
pmm	Multimedia
popengl	OpenGL — a 3D graphic interface
podbc	Open Database Connectivity protocol support
psock	Networking at TCP/IP socket level
replib	Reports
stdlib	Standard library