

---

User's Guide

# GpProfile

By Primoz Gabrijelcic

---



# Contents

<b>Introduction</b>	<b>1</b>
What is GpProfile? .....	1
Features .....	1
How GpProfile works? .....	1
Warning! .....	2
What's new in this version? .....	2
System requirements .....	4
License .....	4
For mere humans .....	4
Credits .....	4
What's cooking? .....	5
Support .....	5
Alternatives .....	6
<b>Installing GpProfile</b>	<b>8</b>
Small and simple .....	8
Delphi IDE integration .....	8
<b>Guided tour</b>	<b>8</b>
Hands-on .....	9
<b>Using GpProfile</b>	<b>15</b>
User interface .....	15
Common functions .....	15
Preferences .....	15
Layout Manager .....	19
Instrumentation .....	20
Elements .....	20
Actions .....	21
Options .....	22
API .....	25
Metacomments .....	26
Analysis .....	26
Elements .....	27
Actions .....	29
Export .....	30
Options .....	30
Shortcut keys .....	31
<b>GpProfile Source</b>	<b>32</b>
Full Source Code Included .....	32
<b>Glossary of Terms</b>	<b>33</b>
<b>Index</b>	<b>Error! Bookmark not defined.</b>



# Introduction

---

## What is GpProfile?

GpProfile is a utility that allows Delphi developers to check the performance of their applications, find bottlenecks and remove them. In short, it is a profiler.

GpProfile will help you speed up your Delphi programs. Delphi version 2, 3, and 4 are supported. As you will see, GpProfile is intuitive to use and in no time you programs will run faster.

Best of all, GpProfile is completely free and even comes with full source code.

---

## Features

- source instrumenting profiler for Delphi 2, 3, and 4
- written in Delphi 4, source included
- works with Windows 95, 98, NT 4, NT 5
- free of charge for any use
- can profile programs, DLLs, and packages
- conditional compilation (IFDEF) support
- multithreaded program support
- ability to instrument procedures, written in built-in assembler
- integrated source preview
- integrated result viewer
- profiling results can be exported to standard delimited format
- API for profiling control
- conditional API execution with metacomments
- layout manager

---

## How GpProfile works?

*Inserting code into target.*

GpProfile does it's magic by instrumenting the target (translation: changing your program's source code). Special calls to time measuring routines are inserted into the target, which you must then recompile and run. Instrumented target records time passed inside procedures. At the end of run GpProfile reads, analyses and displays recorded profile.

---

## Warning!

GpProfile is tampering with your source files! I'm trying to make things as safe as possible but there is always a possibility that GpProfile will destroy your most important file. I cannot take the responsibility for anything that will happen to your source files if you use GpProfile. Always make a backup copy!

---

## Known problems

- Instrumenting the program with one comment type and then uninstrumenting with another type is a sure way to trash the program!
- GpProfile may display error Unable to replace image on startup and then terminate. This is caused by outdated comctl32 library. Get a new one from Microsoft (<http://www.microsoft.com/msdownload/ieplatform/ie/comctl3286.asp>).
- Because of a bug in an installation script GpProfile may not get correctly registered with Delphi when installed on Windows 95/98. If you can't find GpProfile entry in Tools menu, exit Delphi, run *GPPROF /REGISTER* and restart Delphi.
- \$INCLUDE is only partially supported - included files are processed but procedures in included files cannot be instrumented.

---

## What's new in this version?

From version 1.2.1 to 1.3, following changes were made:

- Support for DLL profiling added. Profiling starts when DLL is loaded and stops when DLL is unloaded.
  - Support for package profiling added. Profiling starts when package is loaded and stops when package is unloaded.
  - Parser bug fixed. Code, instrumented with {\$IFDEF GpProfile} markers could not be uninstrumented.
  - Parser bug fixed. Sometimes GpProfile could not find all files, included in the parsed project.
  - Parser bug fixed. Named used units were not found if path started with '\' character (example: test in '\dev\test.pas').
  - Small command line parser bug fixed.
  - Fixed several bugs related to output directory processing.
  - Better handling of ComCtl32 version problems.
  - Better error handling in file access procedures.
  - Better error handling in client module (gpprof.pas).
  - Source preview window is refreshed after instrumentation.
  - Included are few fixes, contributed by Frédéric Libaud, [flibaud@club-internet.fr](mailto:flibaud@club-internet.fr).
  - Fixed incorrect behaviour when output directory was a relative path.
-

- Slight parser speedup.

From version 1.2 to 1.2.1, following changes were made:

- Parser bug fixed. \$IFOPT compiler directive was not recognised and could cause wrongly instrumented source that couldn't be automatically uninstrumented.
- Parser bug fixed. Sometimes units were excluded even if they were not listed in *Excluded units* list.
- *Park* button added to title bar.

From version 1.1 to 1.2, following changes were made:

- Conditional compilation is fully supported!
- GpProfile now ignores all text following final end.
- New menu entry in Delphi's Tools menu: GpProfile - Remove instrumentation
- Fixed bug in TGpArrowListView (arrows were not drawn correctly)
- Fixed bug in Instrument&Run and Run (sometimes GpProfile crashed instead of running Delphi)
- Fixed bug when fetching library search path from registry.
- Exporting slightly polished
- Color of background in source preview window is now shown correctly
- Small bugs in layout manager fixed
- Caller/Called views are resized proportionally when form is resized
- Fixed bug when GpProfile would select wrong version of Delphi for a new project

From version 1.0 to 1.1, following changes were made:

- Added **limited** support for \$INCLUDE directive; included files are parsed but procedures in them cannot be instrumented
- Ability to display and browse caller/called statistics
- Added ability to show/hide Source Preview window
- Source preview is now syntax highlighted
- Context sensitive help, accessible with F1 key
- Full source package includes help, generated with Time2HELP
- Program history included in About box
- Fixed two stupid bugs that made profiling almost impossible
- Executable renamed to GPPROF.EXE due to occasional installation problems

---

## System requirements

GpProfile works with Delphi 2, 3, and 4 in any flavour (Standard, Professional, and Client/Server).

GpProfile runs on most operating systems supported by Delphi (Windows 95, 98, NT 4, NT 5).

You will also need a lot of disk space (recorded data can easily consume couple of gigabytes). A fast computer is also recommended.

I do all the testing on NT 4 Workstation with Delphi 3 and Delphi 4. Other combinations are not thoroughly tested.

---

## License

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA, or check <http://www.gnu.ai.mit.edu/copyleft/gpl.html> .

### For mere humans

What all this legal gibberish is trying to say is:

GpProfile is free! You may use it any way you want, at home and at work, on as many computers as you want. You may download the source and write your own profiler based on the source. I won't mind. **But if you are a shareware writer and you like my profiler so much that you would like to donate a copy of your software for my personal use you can be sure I won't hold it against you!**

---

## Credits

Writing GpProfile would surely take more time without these free components:

- BrowseFolder by Todd Fast
- EZDSL by Julian M Bucknall
- FPURLLabel by Filippo Passeggeri
- mPasLex and mwCustomEdit by Martin Waldenburg
- TRotateLabel by Jörg Lingner
- Widget by Robert R. Marsh, SJ

I would also like to thank a Richard Fellner, who donated a registered version of his beautiful installation program INF-TOOL.

Also, GpProfile wouldn't be the same without Uros Mesojedec, who created the program icon.

---



Delphi 2 version wouldn't be here without Fulvio Guzzon, who converted the gpprof.pas file.

I would like to thank Marko Bohanec and Andy Jeffries for providing me with a mirror download site.

John Hamm enhanced component gpArrowListView.

Source documentation is generated with Time2HELP, donated by Digital Logikk AS.

Links to components and their authors are available on the page <http://www.eccentrica.org/gabr/gpprofile/credits.htm> and in About window (select **Help.About** in GpProfile).

---

## What's cooking?

A lot. I won't run out of ideas soon. I'll just give you a few samples of features waiting to be implemented.

### ***Instrumentation***

- full \$INCLUDE support
- tight integration with Delphi IDE

### ***Execution***

- more precise time measurement (at least under Windows NT)
- faster and less disk-intensive operation during profiling phase
- idle time measurement
- better API

### ***Analysis***

- snapshot comparison
- filters
- call graph analysis
- coverage analysis
- printing

### ***General***

- shortcut reassignment

---

## Support

If you have problems running GpProfile, first check the web (<http://www.eccentrica.org/gabr/gpprofile/gpprofile.htm>) to see if you're running a newest version. Be sure to add yourself to GpProfile Announcement List (subscription form available on above-mentioned page).

If you need more information, email me at [primoz.gabrijelcic@altavista.net](mailto:primoz.gabrijelcic@altavista.net).

### ***Syntax errors***

If your source code contains syntax errors, GpProfile may not be able to properly instrument the target. If this happens, you may not receive an error message

indicating a syntax problem until GpProfile invokes the Delphi IDE, and you attempt to execute your application. GpProfile is not designed to detect syntax errors. The best recommendation is to make sure that your project will compile under the Delphi IDE prior to running it through GpProfile.

### ***GPPROF and GPPROFH***

GpProfile adds two units to every project it instruments. Those units are called GPPROF and GPPROFH. If your project uses modules with the same names, the project won't compile. This is a limitation of Delphi. Workaround: rename your units.

### ***Backup copies***

Two backup files are always kept: .BK2 is older, .BK1 is newer. Therefore, it is possible to restore the source even after two operations (for example failed instrumentation and failed deinstrumentation).

### ***Make a backup!***

You should always create backup before trying to run GpProfile just to be on the safe side.

---

## **Alternatives**

If GpProfile doesn't cover your needs, you might want to check other Delphi profilers:

### ***Delphi Profiler (Axiomatic Software)***

- <http://www.axiomati.demon.co.uk/Frogbit/examples-prof.htm>
- simple source code profiler for Delphi 2 and 3
- free with source

### ***Lightning (Techmarc)***

- <http://www.techmarc.ndirect.co.uk>
- source code profiler
- Delphi 3
- free
- last update March 17, 1998

### ***ProDelphi (Helmuth J. H. Adolph)***

- <http://www.hp.europe.de/prodelphi>
- source instrumenting profiler for Delphi 2, 3, and 4
- \$ 50

### ***SourceCoder (Peter Tiemann)***

- <http://www.preview.org/e/scoder.shtml>
  - profiler, coverage analysis, and much more
  - Delphi 1, 2, 3, and 4
  - \$89
-

- recommended!

### ***Speed Daemon (IntegrationWare)***

- <http://www.speeddaemon.integrationware.com/>
- source code profiler for Delphi 1, 2, and 3
- \$149
- not updated for a veeery long time

### ***SpeedItUp (Robert Lechner)***

- <http://members.xoom.com/SpeedItUp/index.htm>
- source code profiler
- Delphi 3, 4
- free

### ***VTune (Intel)***

- <http://developer.intel.com/vtune/analyzer/index.htm>
- sampling profiler and much more
- not working very well with Delphi
- \$429
- recommended!

# Installing GpProfile

---

## Small and simple

I want to keep distribution as small as possible. Therefore, GpProfile is distributed without a specialised installation program. All the work is done by Windows itself with a help of a very special program - INF-TOOL.

Although installation is very simple, it is complete. If you are reading this, everything went well and GpProfile is installed and integrated with Delphi IDE.

Removal is also very simple - just select Add/Remove Program in Control Panel or Uninstall icon in GpProfile folder. Removal process will automatically remove Delphi IDE integration, GpProfile folder and all files.

## Delphi IDE integration

Currently, GpProfile is not very much integrated into Delphi IDE. Installation program extends Delphi Tools menu with a two items. First opens current project in GpProfile and second removes all instrumentation from current project. Those two menu items work only if

1. Project was compiled at least once after the Delphi was started.
2. Project executable is stored in the same directory as main project file (.dpr).

Future GpProfile versions will remove this limitations and offer tighter Delphi integration.

## Guided tour

---

---

## Hands-on

An easiest way to understand GpProfile's approach is to see it in action. I have prepared a guided tour through simple profiling session, which will help you understand the GpProfile's design and operation.

To take this tour you will need to have Delphi's demos installed, as we will be using one of the demos as example.

### Starting up

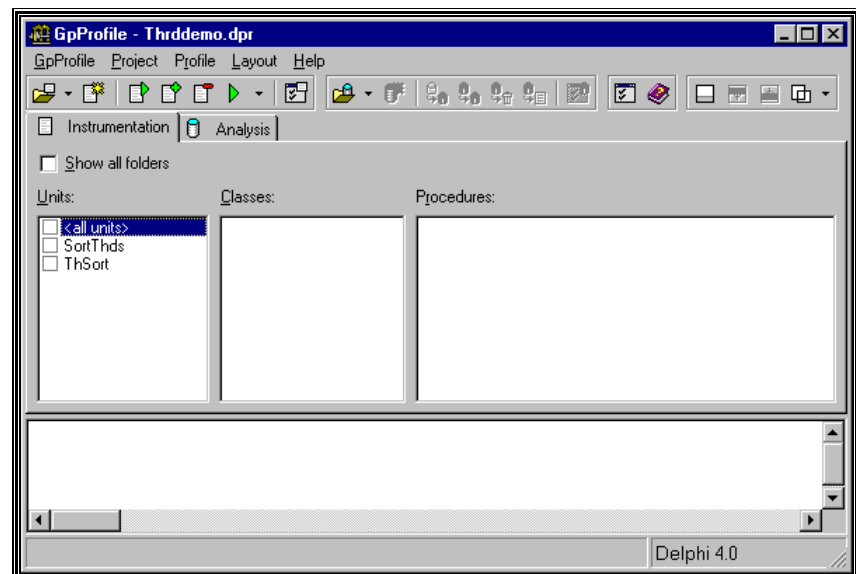
There are two ways to start the GpProfile.

From the **Tools** menu.

1. Start the Delphi.
2. Load ThrdDemo project (located in Demos\Threads under the main Delphi folder).
3. Compile it (Ctrl+F9). This will a) create \$EXENAME macro that is used when running GpProfile from the Tools menu and b) check the syntax of the program.
4. Run GpProfile from the **Tools** menu.

Standalone.

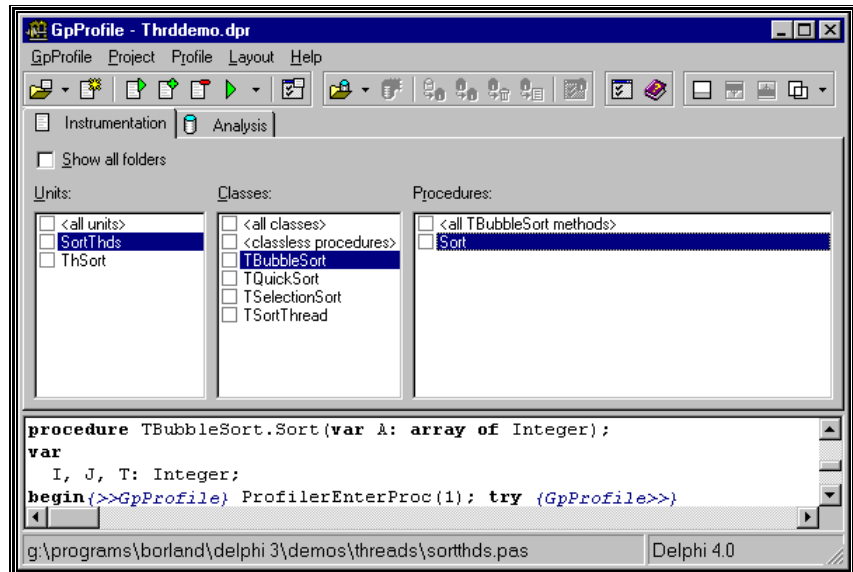
1. Start the GpProfile.
2. Select **Project.Open** and open ThrdDemo.dpr (located in Demos\Threads under the main Delphi folder).



### Browsing the target

As you can see in the *Units* list, ThrdDemo uses two units from the same folder, SortThds and ThSort. Click on the SortThds (on the unit name, not into the checkbox). GpProfile will fill the *Classes* and *Procedures* lists with names of classes and procedures declared in unit SortThds.

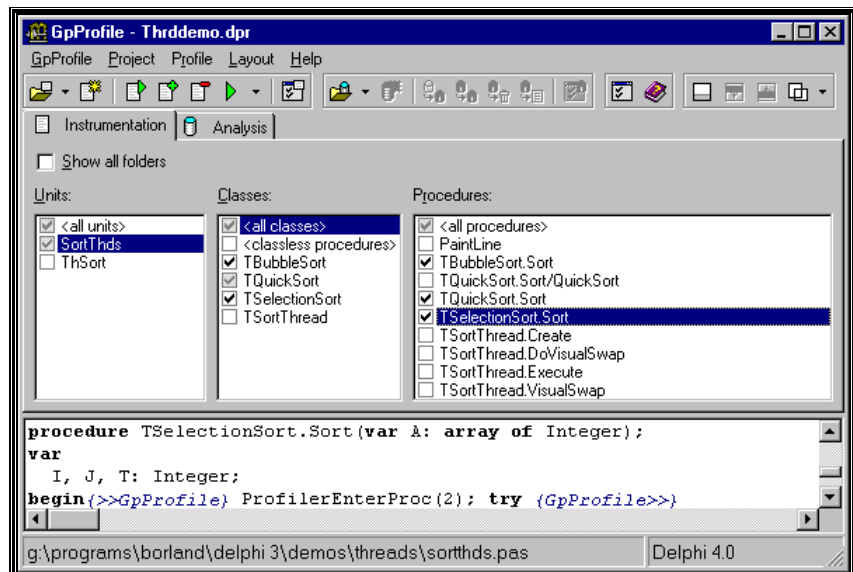
Now click on class TBubbleSort in the *Classes* list. GpProfile will update *Procedures* list to contain only the methods from the TBubbleSort class. If you now click on the Sort procedure (the only one), GpProfile will load file SortThds.pas into the bottom window and position it at the beginning of the Sort procedure.



### Instrumenting the target

Next, you will select the procedures you want to instrument. Core of the ThrdDemo program is contained in three threads: TBubbleSort, TQuickSort, and TSelectionSort. Each does the important part of the job in its Sort method. If you want to measure their respective times you must instrument these methods.

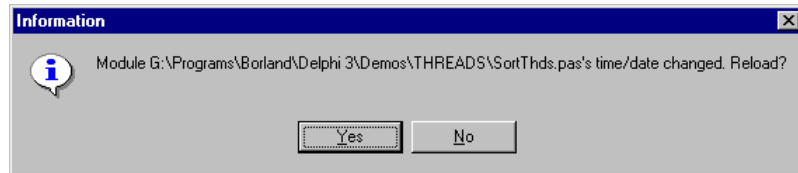
To do so, click *<all classes>* in the *Classes* list and then check the procedures *TBubbleSort.Sort*, *TQuickSort.Sort*, and *TSelectionSort.Sort*.



Those checks have no immediate effect on the program. If you quit GpProfile at that step, your program will not be instrumented. You will finish the instrumentation in the next step.

### Running Threads demo

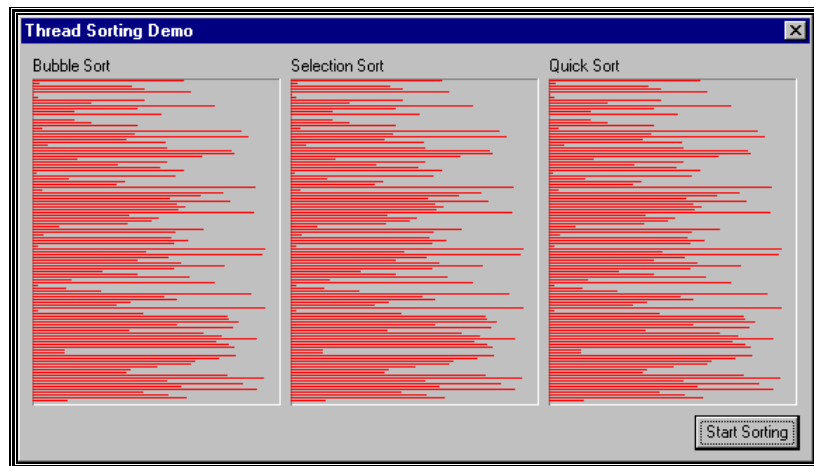
To instrument the Threads demo, select **Project.Instrument and Run**. This will instrument the program and open it in Delphi. You'll have to confirm that you want to reload changed files.



If you have started GpProfile from Delphi IDE through the Tools menu, GpProfile will just switch back to Delphi. If you have started GpProfile without starting Delphi, a fresh Delphi will be started for you.

If you have more than one Delphi installed, you should select the version you want to use with **Project.Options** before selecting **Instrument and Run**.

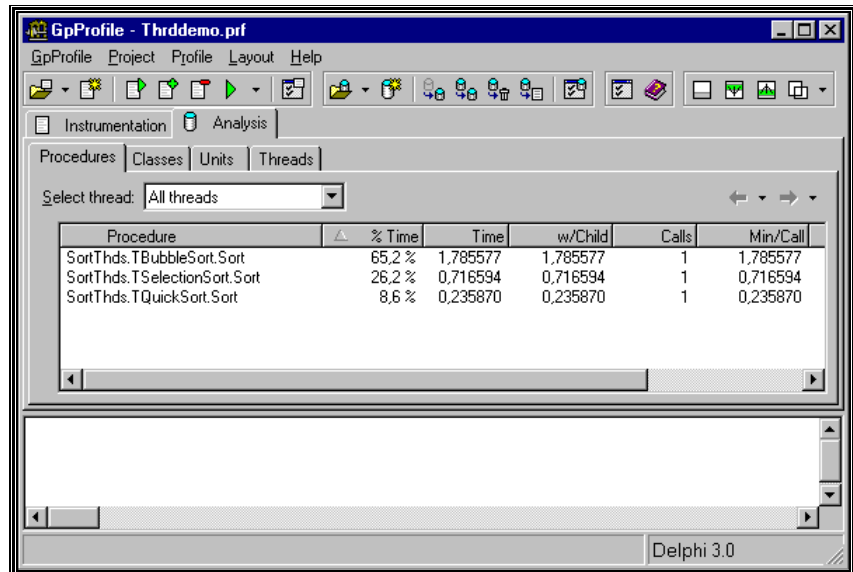
When you are back in Delphi IDE, compile the program (Ctrl+F9) and run it. Don't just press F9 (Run) as Delphi usually just runs the old uninstrumented EXE even if it knows that some files have changed. Go figure.



In the ThrdDemo program click *Start Sorting*. After the program finished its job, close it by clicking the X button in upper right corner. GpProfile will automatically pop up and open the recorded data.

### ***Analysing the results***

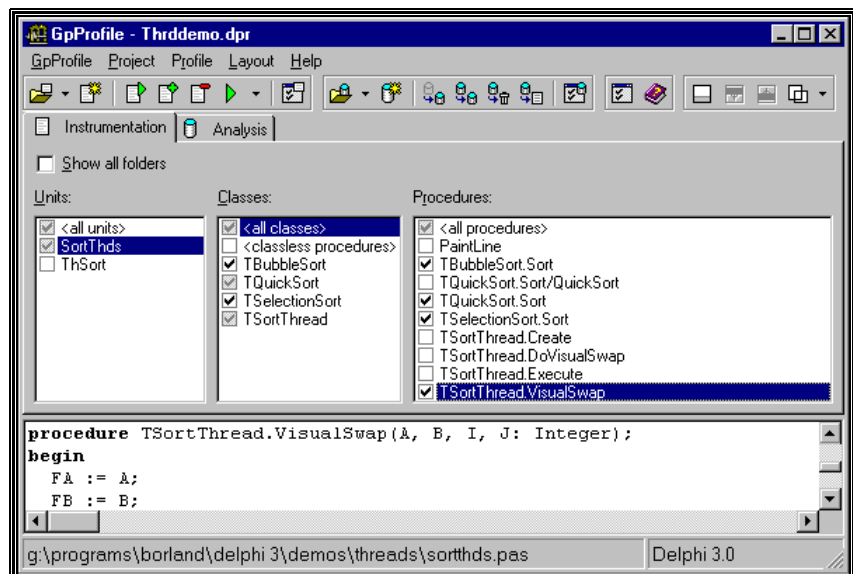
From the recorded data, you can see that Bubble sort is much slower than Selection sort, which is slower than Quick sort. You can now browse through other pages on Analysis tab and see how much time was used for different classes, units and threads.



### Improving the measurement

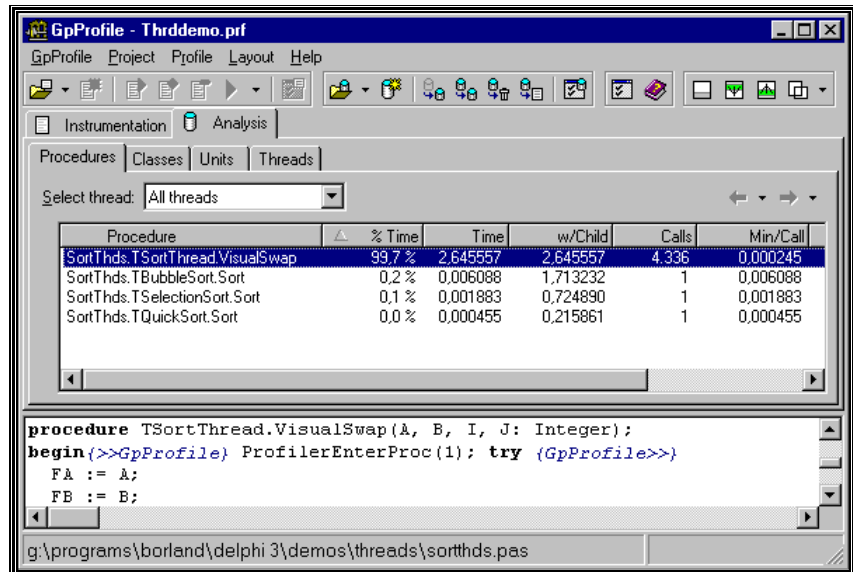
The important part of profiling is knowledge of what you want to profile. In this case you have measured how long it takes for threads to sort the data AND show this process to the user graphically. And it is obvious that graphical part of the program is taking much more time than the sorting itself. Exactly how much?

To find this out, look at the Sort methods. A short glance at the code shows that all sorting routines are using method VisualSwap to display a swapping operation. Therefore, you should instrument this method too.



You must now again choose **Project.Instrument and Run**, accept the changes in Delphi, compile and run the program, click *Start Sorting* and close the program. This time the displayed results are completely different.

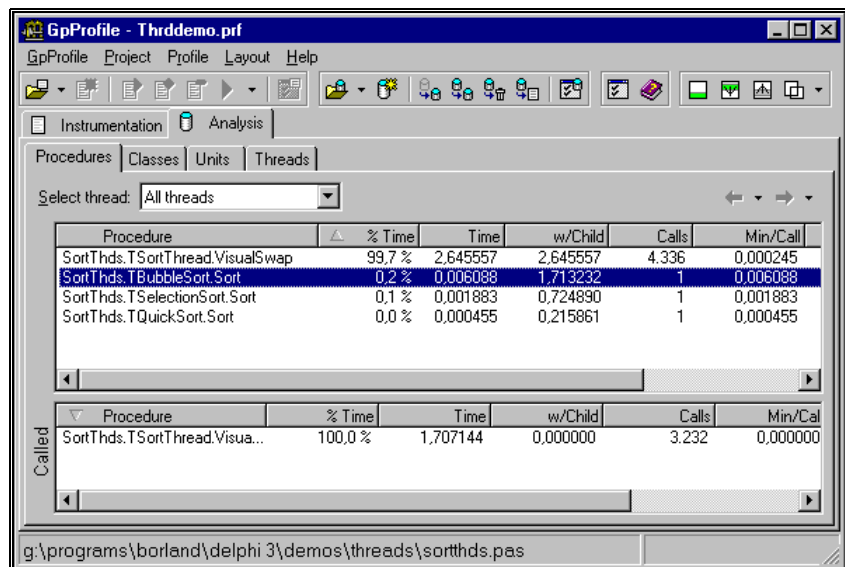




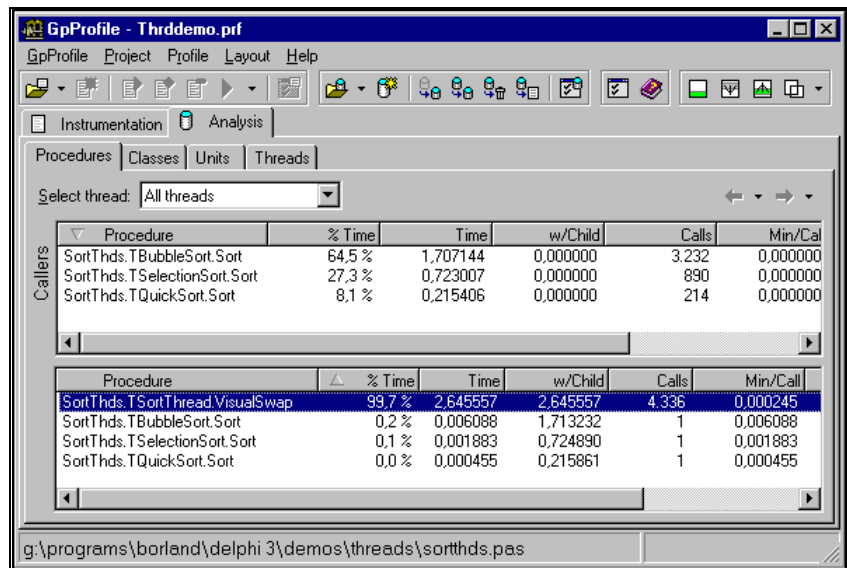
This more exact picture clearly shows that most of the time is spent in the VisualSwap method. Bubble sort is still the slowest but it is now only two times slower than Selection sort (as opposed to nearly three times shown by previous measurement).

### More Details

To get more details, you should enable caller/called analysis. Select *Procedures* tab, then select **Layout.Show Called**. A new list with *Called procedures* will appear. When you click a procedure in *Procedures* list, *Called procedures* list will display all procedures that were called from the selected procedure. In our simple example, only one procedure (VisualSwap) was called.



It is also possible to all procedures that called selected procedure. When Procedures tab is active, select **Layout.Show Callers**. To see it in action, select VisualSwap. Three procedures will appear in *Callers* list, showing that VisualSwap was called from *TBubbleSort.Sort*, *TSelectionSort.Sort*, and *TQuickSort.Sort*. Although *Callers* list displays caller names, all times in this list are times of selected procedure (*VisualSwap* in our example). Therefore, numbers in the following picture are telling us that *VisualSwap* took 1,707 seconds when called from *TBubbleSort.Sort*, 0,723 seconds when called from *TSelectionSort.Sort*, and 0,215 seconds when called from *TQuickSort.Sort*.



Although there is no much data displayed in this example, you can try browsing through the procedures. Just double click any procedure in *Callers* or *Called* list and it will become active procedure. You can return to the previous selection with browser buttons, positioned in upper right corner of *Procedures* tab.

### **Removing instrumentation**

To remove all instrumentation code from the ThrdDemo program, select **Project.Remove Instrumentation**, switch to Delphi, and recompile.

Alternatively, you can select **Tools.GpProfile - remove instrumentation** in Delphi.

# Using GpProfile

---

## User interface

Functionally, GpProfile is divided into two parts - Instrumentation and Analysis. The user interface follows this division. The main part of GpProfile's form can be switched into instrumentation or analysis mode; there are two main toolbars, and two significant menus - Project (instrumentation part) and Profile (analysis part).

There is not much of the program dedicated for the housekeeping tasks. Beside the help system there are only two common functions - Preferences and Layout Manager, both accessible through the GpProfile menu or main (rightmost) toolbar.

---

## Common functions

Common functions are accessible through the menus **GpProfile** and **Help** and through the main (second from the right) toolbar.

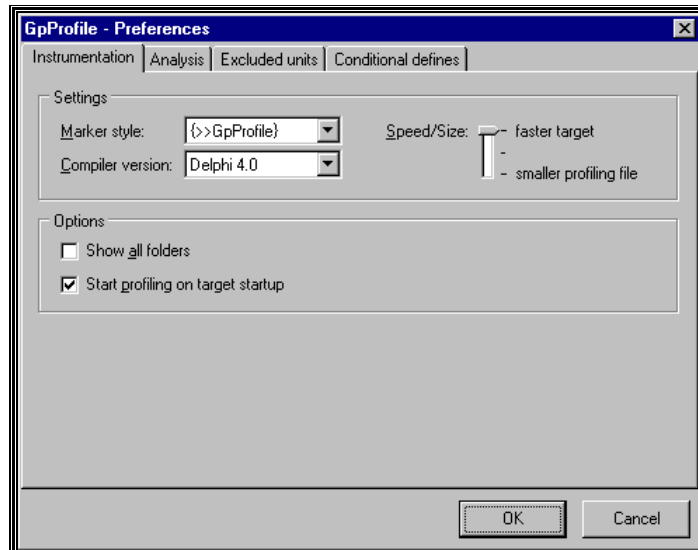


### Preferences

Use *Preferences* dialog to set global options. These options will be used for all new projects. *Preferences* dialog has four pages - Instrumentation, Analysis, Excluded units, and Conditional defines.

### *Instrumentation*

On *Instrumentation* page you set defaults for various items relevant to the instrumentation process.



### Marker style

GpProfile uses markers to mark instrumented code in the program so it can be safely uninstrumented later. Default marker style is `{>>GpProfile}`, which you can change to `{$IFDEF GpProfile}`. If you use the second option don't forget to manually `$DEFINE GpProfile` in all units you want to profile.

### Compiler version

Here you can select default Delphi environment if you have more than one Delphi installed.

### Speed/Size

Profile files get pretty big (you'll reach gigabyte sizes without a problem) and sometimes you'll want to keep them smaller at the expense of target program speed. Beside uncompressed data, GpProfile supports two levels of compression.

### Show all folders

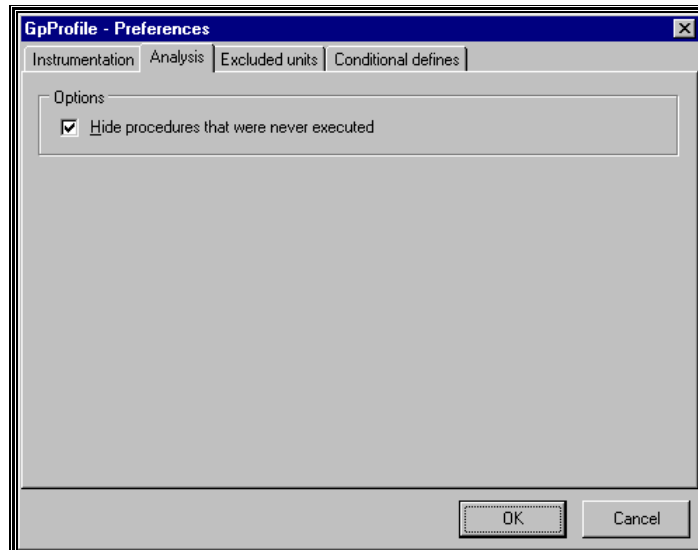
Check if you want every new project to show all used units, not only units in the project folder.

### Start profiling on target startup

When you want to measure just some part of the program, you must disable profiling autostart with this option and start/stop profiling manually by using API calls.

### Analysis

On *Analysis* page you set the parameters affecting the profile analysis process. Currently, there is only one option.

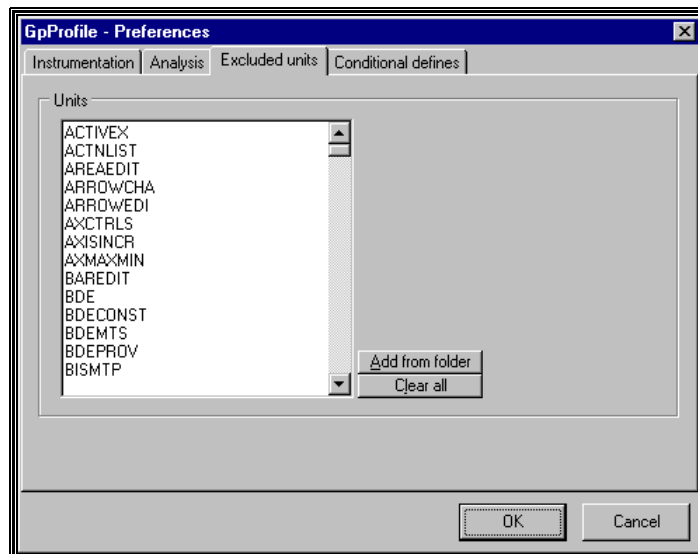


### Hide procedures that were never executed

If checked (default), GpProfile will hide procedures, which were instrumented but never called, so the profile display will be more readable.

### Excluded units

On *Excluded units* page you set units that won't be parsed. Typically you put here system units, which you don't want to instrument anyway. This list contains all the units from Delphi 3 and Delphi 4 by default.



### Units

Simple Memo field with unit names. One per line, alphabetical order is not important. Case insensitive. If you would like to add another unit, just type it in. Likewise, if you would like to remove a unit, just delete it from this field.

### Clear all

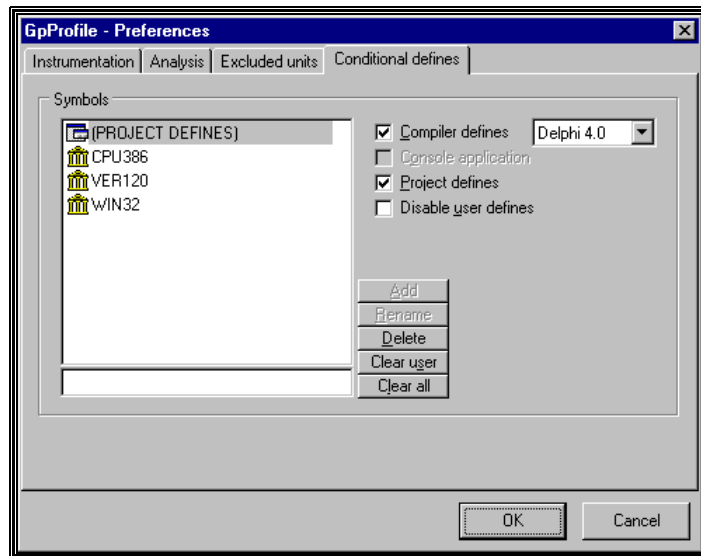
Clear the *Units* field. No units will be excluded.

### Add from folder

Click this button to display folder selection dialog. All units from the selected folder will be added to the *Units* field.

## Conditional defines

On *Conditional defines* page you set symbols that should be pre-set for each parsed unit. Typically, you'll want to set *Compiler defines* and *Project defines*.



### Symbols

List of pre-set symbols

### Compiler defines

If checked (default), symbols CPU386, WIN32, and VER90/VER100/VER120 will be pre-set.

### Console application

If checked (default for applications, which have *Generate console application* option set), symbol CONSOLE will be pre-set. You cannot set default value for this option.

### Project defines

If checked (default), project symbols (specified in Delphi IDE in **Project.Options**) will be pre-set.

### Disable user defines

If checked (default unchecked), additional user-defined symbols (added in this dialog) will be disabled.

### Add

Adds user-defined symbol, specified in entry field.

### Rename

Renames selected symbol to a name, specified in entry field.

### Delete

Deletes selected symbol.

### Clear user

Removes all user-defined symbols.

### Clear all

Removes all symbols.

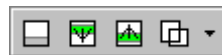
---

---

# Layout Management

While working with GpProfile you'll notice that it has persistent interface - everything you change (window position & size, list sizes, source preview height...) is remembered for the next run. Therefore it is very easy to use GpProfile in two sizes - one small (to fit beside then Delphi IDE) and other maximised.

Layout Manager is a feature of GpProfile that allows you to use more than one UI arrangement. It is accessible through the menu **Layout** and through the layout (rightmost) toolbar.



## Show/Hide Source Preview

Use *Show/Hide Source Preview* switch to show or hide source preview window.

## Show/Hide Callers

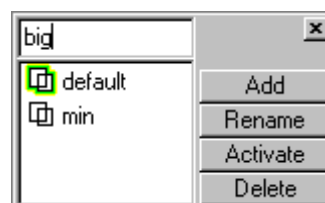
Use *Show/Hide Callers* switch to show or hide callers window.

## Show/Hide Called

Use *Show/Hide Callers* switch to show or hide called window.

## Layout Manager

Layout Manager is implemented as a non-modal window meaning that you can continue to work with the GpProfile while it is displayed (you can for example fine-tune some UI settings).



### Adding a layout

Type layout name into entry field (top left) and click *Add* button. New layout will be created and activated. Everything you now do to GpProfile will be stored under this layout name. Layouts are saved automatically.

### Renaming a layout

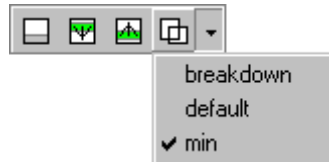
Select layout in the layout list (bottom left) with single click or arrow keys, then type new name in the entry field and click *Rename* button.

## Activating a layout

Select layout in the layout list, and then click the *Activate* button. Layout settings will be restored.

Alternatively, you can double click on the layout name. This will activate the layout and close Layout Manager.

You can activate a layout without opening the Layout Manager. Just click the little arrow beside the Layout Manager button and select layout from the list.



## Deleting a layout

Select layout in the layout list, then click the *Delete* button.

## Undeleting a layout

Until you close the GpProfile, you can undelete any deleted layout. Select the layout and click the *Undelete* button, which will appear instead of *Delete* button.

## Restoring default settings

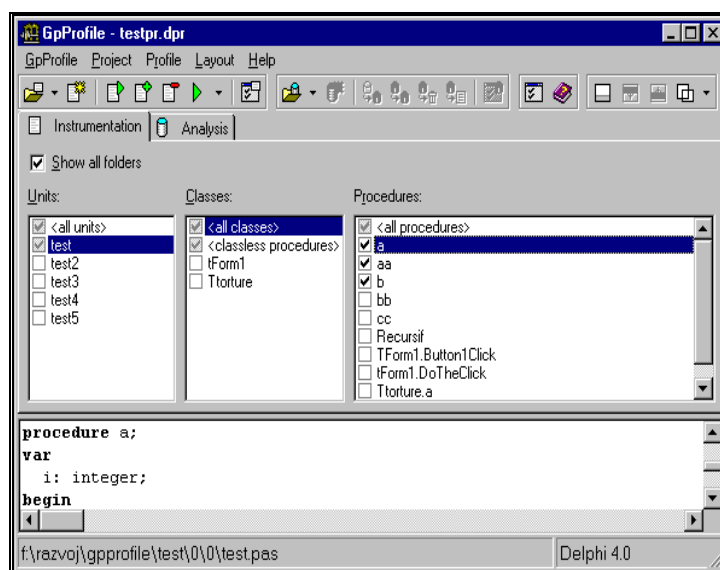
If you delete all the layouts and restart GpProfile, it will use default settings (as when it was started for the first time). Use this if you really mixed everything up.

---

# Instrumentation

Instrumentation is a process of inserting special code into target. When instrumenting, you choose procedures that will be measured. After that, you apply the changes, switch to Delphi, recompile the program and run it.

## Elements





## **Units**

*Units* window shows all program units or all units in project folder, depend on the state of Show all folders flag.

Click on a unit name to select it and display its classes in Classes window. Click a checkbox next to the unit name to mark/unmark all procedures in the selected unit for instrumentation.

Click checkbox next to the *<all units>* item to mark/unmark all units in program (or in project folder).

## **Classes**

*Classes* window shows all classes from the selected unit, listed alphabetically.

Click on a class name to select it and display its method in Procedures window. Click a checkbox next to the class name to mark/unmark all methods in the selected class for instrumentation.

Click checkbox next to the *<all classes>* item to mark/unmark all procedures in selected unit.

All procedures that are not part of any class are collected under the *<classless procedures>* item.

## **Procedures**

*Procedures* window shows all methods in the selected class, all procedures, which are not part of any class, or all procedures from the selected unit.

Click on a procedure name to select it and display its source in the Source preview window. Click a checkbox next to the procedure name to mark/unmark it.

Click checkbox next to the *<all procedures>/<all classless procedures>/<all TXXX methods>* item to mark/unmark all procedures, all procedures that are not part of any class, or all procedures in class TXXX.

## **Source preview**

*Source preview* window shows selected unit, positioned at the beginning of selected procedure.

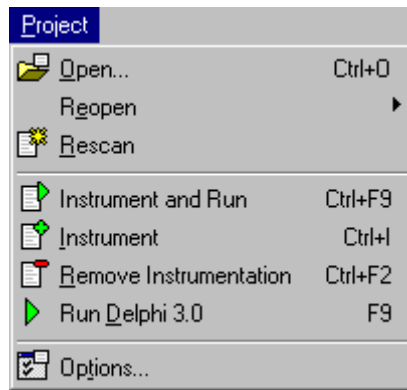
## **Show all folders**

Toggles the *units* window display. In unchecked state *units* window displays only the units that are positioned in the project directory. In checked state *units* window displays all units referenced in project (minus units excluded in Options).

You can set the *Show all folders* default state in Preferences.

## **Actions**

Instrumentation actions are divided into three groups: opening the project, instrumenting the project, and setting the project options. You can access them through the Project menu or through the Project toolbar.



### ***Open***

Displays File Open dialog with which you can select the project (.dpr file) to instrument.

### ***Reopen***

Shows the list of recently opened project. Select a project to reopen it.

### ***Rescan***

Rescans the current project. As GpProfile checks files every time it becomes active this will rarely be necessary.

### ***Instrument and Run***

Instruments all selected procedures and starts the Delphi IDE.

### ***Instrument***

Instruments all selected procedures and removes instrumentation from all unselected procedures.

### ***Remove Instrumentation***

Removes instrumentation from all procedures in project. Useful for cleaning up after profiling session.

### ***Run Delphi***

Runs Delphi environment. If you have more than one Delphi installed, select the proper version in Options, or with an arrow button right to the *Run Delphi* toolbar button.

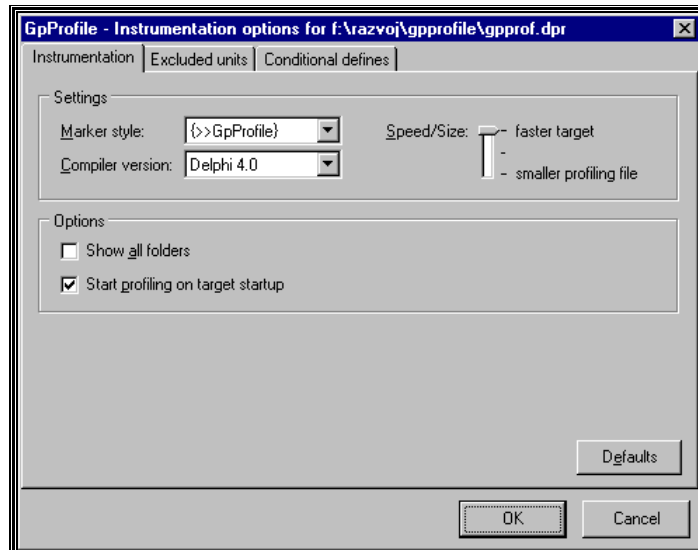
## **Options**

Use *Options* dialog to set local options for the current project. These options override options, set in Preferences dialog. *Options* dialog has two pages - Instrumentation, and Excluded units.

### ***Instrumentation***

On *Instrumentation* page, you set defaults for various items relevant to the instrumentation process

---



### Marker style

GpProfile uses markers to mark instrumented code in the program so it can be safely uninstrumented later. Default marker style is `{>>GpProfile}`, which you can change to `{$IFDEF GpProfile}`. If you use the second option don't forget to manually `$DEFINE GpProfile` in all units you want to profile.

### Compiler version

Here you can select Delphi environment if you have more than one Delphi installed.

### Speed/Size

Profile files get pretty big (you'll reach gigabyte sizes without a problem) and sometimes you'll want to keep them smaller at the expense of target program speed. Beside uncompressed data, GpProfile supports two levels of compression.

### Show all folders

Check if you want to see all used units, not only units in the project folder.

### Start profiling on target startup

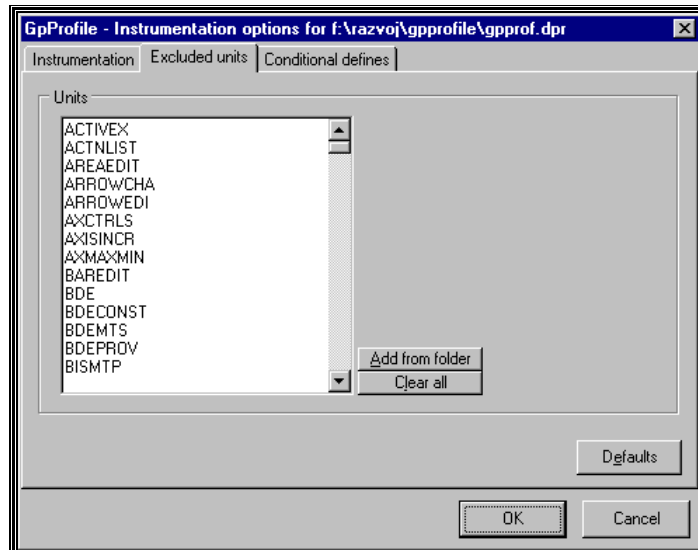
When you want to measure just some part of the program, you must disable profiling autostart with this option and start/stop profiling manually by using API calls.

### Defaults

Resets all settings to default values (as set in *Preferences* dialog).

### Excluded units

On *Excluded units* page you set units that won't be parsed. Typically, you put here system units, which you don't want to instrument anyway.



### Units

Simple Memo field with unit names. One per line, alphabetical order is not important. Case insensitive. If you would like to add another unit, just type it in. Likewise, if you would like to remove a unit, just delete it from this field.

### Clear

Clear the *Units* field. No units will be excluded.

### Add from folder

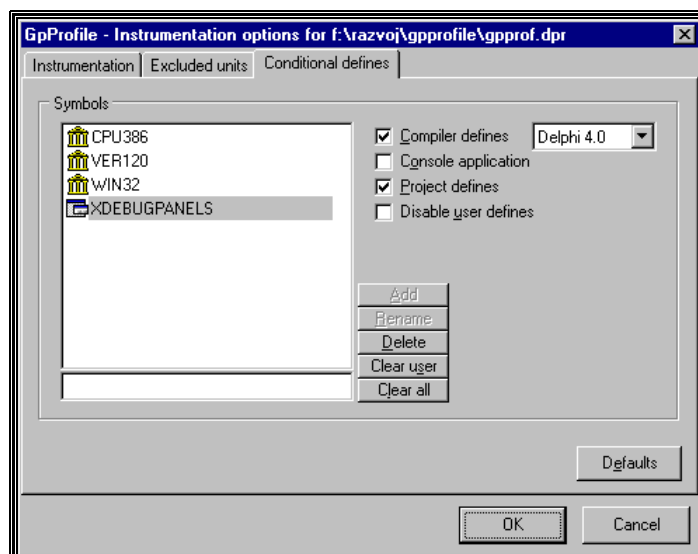
Click this button to display folder selection dialog. All units from the selected folder will be added to the *Units* field.

### Defaults

Resets all settings to default values (as set in *Preferences* dialog).

### Conditional defines

On *Conditional defines* page you set symbols that should be pre-set for each parsed unit. Typically, you'll want to set *Compiler defines* and *Project defines*.



### Symbols

List of pre-set symbols

### **Compiler defines**

If checked (default), symbols CPU386, WIN32, and VER90/VER100/VER120 will be pre-set.

### **Console application**

If checked (default for applications, which have *Generate console application* option set), symbol CONSOLE will be pre-set.

### **Project defines**

If checked (default), project symbols (specified in Delphi IDE in **Project.Options**) will be pre-set.

### **Disable user defines**

If checked (default unchecked), additional user-defined symbols (added in this dialog) will be disabled.

### **Add**

Adds user-defined symbol, specified in entry field.

### **Rename**

Renames selected symbol to a name, specified in entry field.

### **Delete**

Deletes selected symbol.

### **Clear user**

Removes all user-defined symbols.

### **Clear all**

Removes all symbols.

### **Defaults**

Resets all settings (except *Console application*, which is set according to application type) to default values (as set in *Preferences* dialog).

## **API**

You can use GpProfile API to start/stop profiling from the profiled program.

### ***ProfilerStart***

Activates profiling process. Can be used more than once (in pair with *ProfilerStop*).

```
procedure ProfilerStart;
```

### ***ProfilerStartThread***

Activates profiling process in single-threaded mode, measuring only the thread from which *ProfilerStartThread* was called. Can be used more than once (in pair with *ProfilerStop*).

```
procedure ProfilerStartThread;
```

### ***ProfilerStop***

Deactivates profiling process. Can be used more than once (in pair with *ProfilerStart* or *ProfilerStartThread*).

```
procedure ProfilerStop;
```

## **ProfilerTerminate**

Terminates profiling process and opens current profile in GpProfile. Target continues running. After this call, profiling process cannot be restarted without rerunning the target.

```
procedure ProfilerTerminate;
```

## **Metacomments**

Metacomments are special comments that will be expanded if at least one procedure in current unit is instrumented.

You can use them to conditionally execute commands only when unit is profiled.

Typical example is to start/stop profiling. Instead of writing

```
procedure Test;
begin
    ProfilerStart;
    { some code }
    ProfilerStop;
end; { Test }
```

you should write

```
procedure Test;
begin
    {GPP:ProfilerStart;}
    { some code }
    {GPP:ProfilerStop;}
end; { Test }
```

When at least one procedure from the unit is instrumented, instrumentation process will change this to

```
procedure Test;
begin
    {>>GpProfile API}ProfilerStart;{GpProfile API>>}
    { some code }
    {>>GpProfile API}ProfilerStop;{GpProfile API>>}
end; { Test }
```

When no procedure from the unit are instrumented, instrumentation process will change this to original (commented out) form.

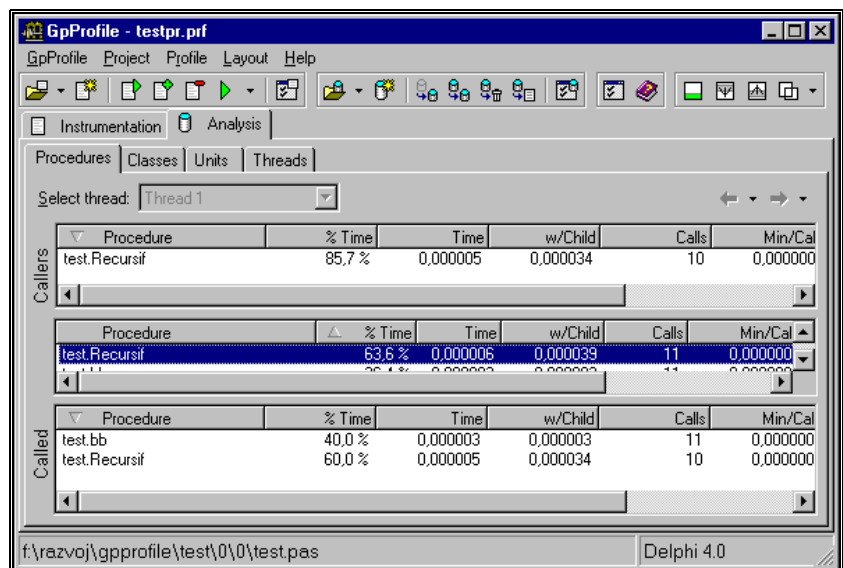
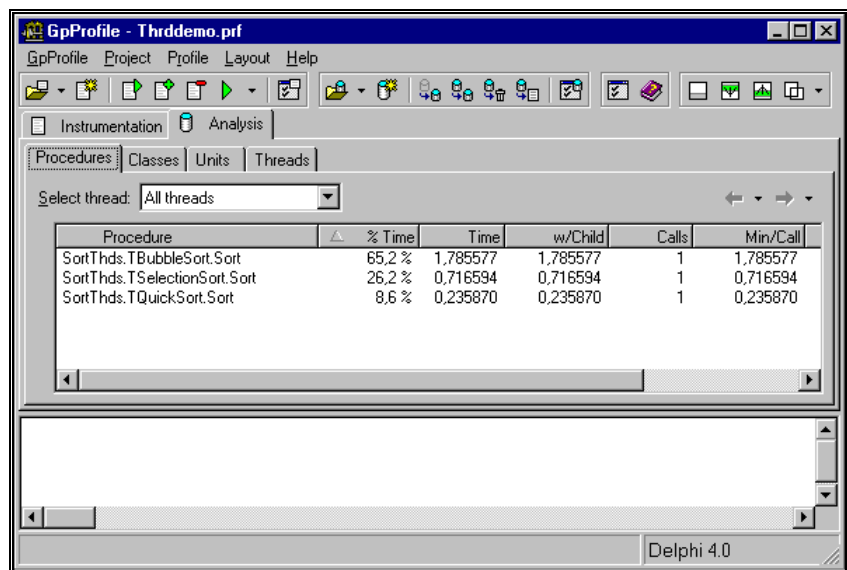
---

## **Analysis**

Analysis is a process of interpreting the profile.

---

## Elements



### Select thread

*Select thread* list allows you to select the thread for which you want to analyse the data. You can also select *All threads* and analyse combined data from all threads. If measured program had only one thread, this selector is disabled.

### Procedures

*Procedures* pane shows profiling data grouped by procedures. You can sort the view by procedure name, % time, time, total time (procedure + all children), number of calls, shortest execution time, longest execution time, and average execution time. You can also resize columns.

### Classes

*Classes* pane shows profiling data grouped by classes. You can sort the view by class name, % time, time, total time (procedure + all children), and number of calls. You can also resize columns.

## Units

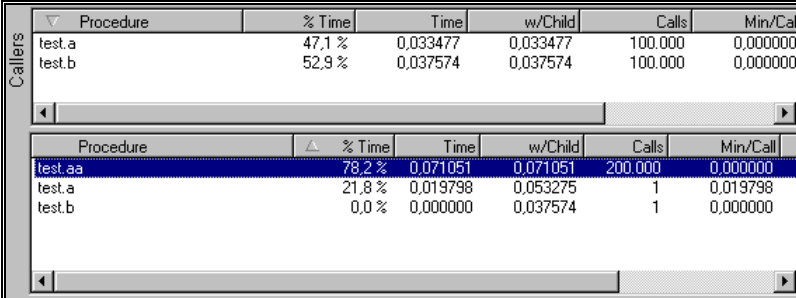
*Units* pane shows profiling data grouped by units. You can sort the view by unit name, % time, time, total time (procedure + all children), and number of calls. You can also resize columns.

## Threads

*Threads* pane shows profiling data grouped by threads. You can sort the view by thread name, % time, time, total time (procedure + all children), and number of calls. You can also resize columns.

## Callers view

*Callers view* shows all callers of the selected procedure. Statistical data displayed in the callers view is data for **selected procedure**, not for the calling procedure! Take a look at the following example:



Procedure	% Time	Time	w/Child	Calls	Min/Cal
test.a	47.1 %	0,033477	0,033477	100.000	0,000000
test.b	52.9 %	0,037574	0,037574	100.000	0,000000

Procedure	% Time	Time	w/Child	Calls	Min/Cal
test.aa	78.2 %	0,071051	0,071051	200.000	0,000000
test.a	21.8 %	0,019798	0,053275	1	0,019798
test.b	0,0 %	0,000000	0,037574	1	0,000000

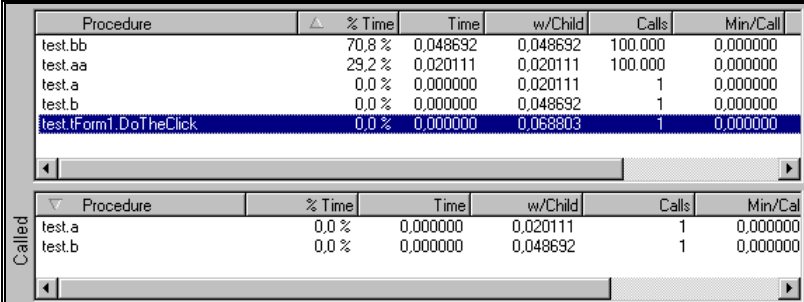
Procedure *aa* was called 200.000 times during the program execution. Its total execution time was 0.071051 second. It was called from procedure *a* and from procedure *b*. It was called called 100.000 times from both callers and was executing 0.033477 second when called from *a* and 0.037574 second when called from *b*.

*Callers view* is hidden by default. Use Show/Hide Callers switch to display it.

Double click on a procedure in *callers view* selects that procedure in the *procedure view*. You can return to then previous state with browser buttons.

## Called view

*Called view* shows timing data for all procedures, called from the selected procedure. Take a look at the following example:



Procedure	% Time	Time	w/Child	Calls	Min/Cal
test.bb	70.8 %	0,048692	0,048692	100.000	0,000000
test.aa	29.2 %	0,020111	0,020111	100.000	0,000000
test.a	0,0 %	0,000000	0,020111	1	0,000000
test.b	0,0 %	0,000000	0,048692	1	0,000000
test.tForm1.DoTheClick	0,0 %	0,000000	0,068803	1	0,000000

Procedure	% Time	Time	w/Child	Calls	Min/Cal
test.a	0,0 %	0,000000	0,020111	1	0,000000
test.b	0,0 %	0,000000	0,048692	1	0,000000

Total execution time for procedure *DoTheClick* (together with all procedures called from it) was 0.068803 second. The execution was split between procedure *a* (0.020111 second) and procedure *b* (0.048692 second).

*Called view* is hidden by default. Use Show/Hide Called switch to display it.

Double click on a procedure in *called view* selects that procedure in the *procedure view*. You can return to then previous state with browser buttons.



## Browser buttons

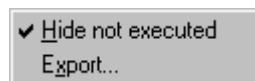
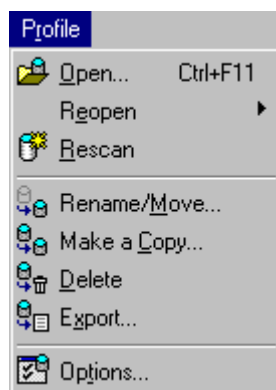
With *browser buttons* you can move to the previous/next selected procedure.

## Source preview

*Source preview* window shows selected unit, positioned at the beginning of selected procedure.

## Actions

Analysis actions are divided into three groups: opening the profile, manipulating the profile, and setting the profile options. You can access them through the Profile menu, through the Profile toolbar, or through the context menu, which appears when you right-click on the analysis results.



## Open

Displays File Open dialog with which you can select the profile (.prf file) to analyse. After the profile is opened, it is overwritten with much smaller *digest*, which contains all the data in compact form that will be opened much faster in the future.

## Reopen

Shows the list of recently opened profiles. Select a profile to reopen it.

## Rescan

Rescans the current profile. As GpProfile rescans profile every time the measured program terminates, this will rarely be necessary.

## Rename/Move

Renames the current profile.

## **Make a Copy**

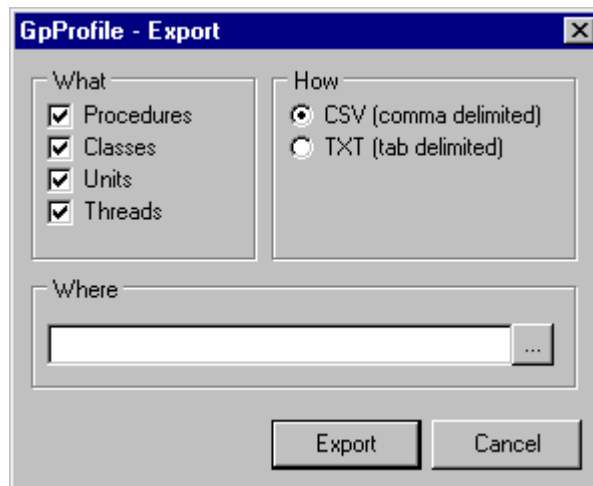
Makes a copy of current profile.

## **Delete**

Deletes a current profile. You can undelete last deleted profile until you open a new profile or close GpProfile.

## **Export**

Displays the Export dialog. Typically, you would want to export profile to create a graph from it or to make a hardcopy.

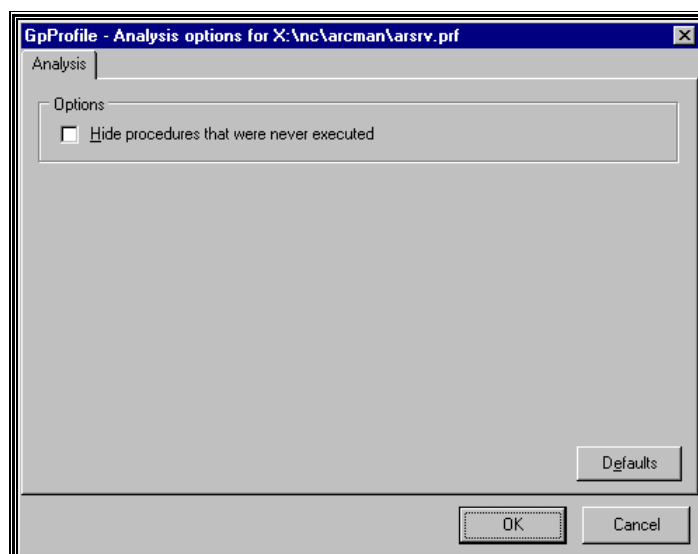


## **Options**

Use *Options* dialog to set local options for the current profile. These options override options, set in Preferences dialog. *Options* dialog has only one page - Analysis.

## **Analysis**

On *Analysis* page, you set the parameters affecting the Profile analysis process. Currently, there is only one option.



### Hide procedures that were never executed

If checked (default), GpProfile will hide procedures, which were instrumented but never called, so the profile display will be more readable.

### Defaults

Resets all settings to default values (as set in Preferences dialog).

---

## Remote Profiling

It is possible to instrument and compile program on one computer and run it on another. Together with <program>.exe you must copy files <program>.gpi and <program.gpd>. Then open copy of <program>.gpi and edit path, specified in TableName setting so it would point to the copied <program>.gpd. Run the program.

You can open results on the second computer or copy <program>.prf file to the first computer and open it there. I recommend that you open <program>.prf once on second computer so that GpProfile would make it smaller and then transfer this file to first computer and work on it there so you'll have working source preview.

---

## DLL and Package Profiling

Well, nothing fancy here. Just open a DLL/package source (.dpr/.dpk), instrument it and compile it. Profiling will start on DLL/package load and stop on DLL/package unload. If you don't like that behaviour, disable *Start profiling on target startup* setting and use GpProfile API to start/stop profiling.

---

## Shortcut keys

Keys	Action
Ctrl+O	Open Project
Ctrl+F9	Instrument and Run
Ctrl+I	Instrument
Ctrl+F2	Remove Instrumentation
F9	Run Delphi
Ctrl+F11	Open Profile
F1	Context sensitive help
Alt+X	Exit

# GpProfile Source

---

## Full Source Code Included

GpProfile is completely free, including the source code. It is available on GpProfile web (<http://www.eccentrica.org/gabr/gpprofile/files.htm>).

---

# Glossary of Terms

## **project folder**

Folder that contains DPR file.

## **profiling**

A process of measuring time spent in various parts of program.

## **profile**

A file with timing data. Result of profiling.

## **deinstrumentation**

Removing inserted code from a target. Inverse of instrumentation.

## **target**

A program you are trying to profile.

## **instrumentation**

Inserting code into target.

## **profiler**

A tool for profiling.