

Irie Pascal User's Manual (Windows Edition)

Author: Stuart King

Version: Irie Pascal Version 2.6

Page: 1

TABLE OF CONTENTS

[1 What's New In Version 2.6](#)

[1.1 New IDE features](#)

[2 Getting Started](#)

[2.1 Getting help](#)

[2.2 Installing and uninstalling](#)

[2.3 Getting started creating programs](#)

[2.4 Sample Programs](#)

[2.4.1 Getting started with the sample programs](#)

[2.4.2 The hello world sample program](#)

[2.4.3 The hello world sample program \(CGI version\)](#)

[2.4.4 The hello world sample program \(Windows API version\)](#)

[3 How To...](#)

[3.1 How to get help](#)

[3.2 How to buy](#)

[3.3 How to contact Irie Tools](#)

[3.4 How to create programs](#)

[3.4.1 Creating new programs](#)

[3.4.2 Opening existing programs](#)

[3.4.3 Compiling programs](#)

[3.4.4 Running programs](#)

[3.5 How To Print](#)

[3.5.1 Page setup](#)

[3.5.2 Printing files](#)

[3.6 How To Choose Fonts](#)

[3.6.1 Choosing fonts](#)

[3.7 How To Choose Colors](#)

[3.7.1 Choosing colors](#)

[3.8 How To Work With Sections Of Text](#)

[3.8.1 Selecting a section of text](#)

[3.8.2 Replacing a section of text](#)

[3.8.3 Moving a section of text](#)

[3.8.4 Copying a section of text](#)

[3.8.5 Deleting a section of text](#)

[3.8.6 Printing a section of text](#)

[3.8.7 Indenting a section of text](#)

[3.8.8 Unindenting a section of text](#)

[3.8.9 Unselecting a section of text](#)

[3.9 How To Create Irie Pascal Executables](#)

[3.9.1 Creating IVM executables](#)

[3.9.2 Creating EXE executables](#)

[3.9.3 WinNT/2000 Services](#)

[3.9.3.1 Creating WinNT/2000 services](#)

[3.9.3.2 Writing well behaved services](#)

[3.10 How To Use COM/ActiveX Objects](#)

[3.10.1 What are COM/ActiveX objects?](#)

[3.10.2 Instantiating ActiveX objects](#)

[3.10.3 Calling ActiveX object methods](#)

[3.10.4 Accessing ActiveX object properties](#)

[3.10.5 Disposing of ActiveX object](#)

[3.11 How To Use Databases](#)

[3.11.1 Connecting To Databases](#)

[3.11.1.1 Connecting to ODBC databases](#)

[3.11.1.2 Connecting to MySQL databases](#)

[3.11.2 Executing Database Commands](#)

[3.11.2.1 Using the execute method](#)

[3.11.3 Querying Databases](#)

[3.11.3.1 Using the recordset object](#)

[3.12 How To Program Sockets](#)

[3.12.1 Programing Sockets](#)

[3.13 How To Distribute Irie Pascal Programs](#)

[3.13.1 Distributing Irie Pascal programs](#)

[4 The Irie Pascal IDE](#)

[4.1 Introducing the Irie Pascal IDE](#)

[4.2 Irie Pascal projects](#)

[4.3 The IDE Menus](#)

[4.3.1 The IDE File Menu](#)

[4.3.1.1 File menu](#)

[4.3.2 The IDE Edit Menu](#)

[4.3.2.1 Edit menu](#)

[4.3.2.2 Preferences...](#)

[4.3.2.2.1 Changing user preferences](#)

[4.3.2.2.1.1 The User Preferences dialog box](#)

[4.3.2.2.1.2 The Text Preferences page](#)

[4.3.2.2.1.3 The Executable Preferences page](#)

[4.3.3 The IDE Search Menu](#)

[4.3.3.1 Search menu](#)

[4.3.3.2 Find...](#)

[4.3.3.2.1 The Find Dialog](#)

[4.3.3.3 Replace...](#)

[4.3.3.3.1 The Replace Dialog](#)

[4.3.3.4 Goto Line...](#)

[4.3.3.4.1 The Goto Line dialog box](#)

[4.3.4 The IDE Project Menu](#)

[4.3.4.1 Project menu](#)

[4.3.4.2 New...](#)

[4.3.4.2.1 Creating new projects](#)

[4.3.4.3 Open...](#)

[4.3.4.3.1 Opening existing projects](#)

[4.3.4.4 Options...](#)

[4.3.4.4.1 Changing Project Options](#)

[4.3.4.4.1.1 The Project Options dialog box](#)

[4.3.4.4.1.2 Warnings](#)

[4.3.4.4.1.2.1 Warning options](#)

[4.3.4.4.1.3 Extensions](#)

- [4.3.4.4.1.3.1 Extension options](#)
 - [4.3.4.4.1.4 Run-Time](#)
 - [4.3.4.4.1.4.1 Run-Time options](#)
 - [4.3.4.4.1.5 Code Generation](#)
 - [4.3.4.4.1.5.1 Code generation options](#)
 - [4.3.4.4.1.6 Environment](#)
 - [4.3.4.4.1.6.1 Environment options](#)
 - [4.3.4.4.1.7 Miscellaneous](#)
 - [4.3.4.4.1.7.1 Miscellaneous options](#)
 - [4.3.4.5 Compile](#)
 - [4.3.4.5.1 The Compile menu entry](#)
 - [4.3.4.6 Run](#)
 - [4.3.4.6.1 Running programs](#)
 - [4.3.4.7 Select Program...](#)
 - [4.3.4.7.1 Selecting the program](#)
 - [4.3.4.8 Recent Project List](#)
 - [4.3.4.8.1 Opening recent projects](#)
 - [4.3.5 The Window Menu](#)
 - [4.3.5.1 Window menu](#)
 - [4.3.6 The IDE Help Menu](#)
 - [4.3.6.1 Help menu](#)
 - [4.3.6.2 View Irie Pascal prices...](#)
 - [4.3.6.3 Buy Now!...](#)
 - [4.3.6.4 Go to IrieTools.com](#)
 - [4.3.7 The IDE Context Menu](#)
 - [4.3.7.1 Context menu](#)
 - [4.4 The IDE Toolbar](#)
 - [4.4.1 IDE Toolbar](#)
 - [4.5 Editor Bookmarks](#)
 - [4.5.1 Bookmarks](#)
 - [4.6 Mouse wheels](#)
 - [4.6.1 Using mouse wheels](#)
 - [4.7 The Keyboard](#)
 - [4.7.1 Using the keyboard](#)
 - [4.7.2 Editing text](#)
 - [4.7.3 Using menus](#)
 - [4.7.4 Using dialog boxes](#)
 - [4.7.5 Keyboard short-cuts](#)

[5 The Command-Line Tools](#)

 - [5.1 The Command-Line Compiler](#)
 - [5.1.1 Using the command-line compiler](#)
 - [5.1.2 Compiler Options](#)
 - [5.1.2.1 Compiler options overview](#)
 - [5.1.2.2 Options List](#)
 - [5.1.2.2.1 -aN Align on N bytes](#)
 - [5.1.2.2.2 -ao* Trap assignment overflow errors](#)
 - [5.1.2.2.3 -A* Enable Asserts](#)
 - [5.1.2.2.4 -b Use brief messages](#)
 - [5.1.2.2.5 -C Identifiers are case-sensitive](#)
 - [5.1.2.2.6 -cm20 Compatibility mode](#)
 - [5.1.2.2.7 -ead* Auto-declare input & output](#)
 - [5.1.2.2.8 -ebh* Allow binary & hex integers](#)
 - [5.1.2.2.9 -eco* Enable non-standard constants](#)
 - [5.1.2.2.10 -ecr* Allow constant ranges](#)
 - [5.1.2.2.11 -edq* Allow double-quoted literals](#)
 - [5.1.2.2.12 -efn* Enable non-standard functions](#)

[5.1.2.2.13 -enn* Allow non-numeric labels](#)
[5.1.2.2.14 -eop* Enable non-standard operators](#)
[5.1.2.2.15 -eow* Allow otherwise](#)
[5.1.2.2.16 -ep* Enable non-standard procedures](#)
[5.1.2.2.17 -erd* Allow relaxed declarations](#)
[5.1.2.2.18 -ety* Enable non-standard types](#)
[5.1.2.2.19 -eui* Allow underscores in identifiers](#)
[5.1.2.2.20 -eva* Enable non-standard variables](#)
[5.1.2.2.21 -E* Enable all extensions](#)
[5.1.2.2.22 -gs Generate a WinNT/2000 service](#)
[5.1.2.2.23 -hTEXT Add #!TEXT header](#)
[5.1.2.2.24 -i* Trap I/O errors](#)
[5.1.2.2.25 -I Control inforatory messages](#)
[5.1.2.2.26 -ln* Insert line-number debug info](#)
[5.1.2.2.27 -mb Generate Borland compatible messages](#)
[5.1.2.2.28 -mc* Display message context](#)
[5.1.2.2.29 -meN Set maximum errors](#)
[5.1.2.2.30 -mm Generate Microsoft compatible messages](#)
[5.1.2.2.31 -mwN Set maximum warnings](#)
[5.1.2.2.32 -nc Allow nested comments](#)
[5.1.2.2.33 -nu Non-standard unary operators](#)
[5.1.2.2.34 -oNAME Set output filename](#)
[5.1.2.2.35 -p Require parentheses](#)
[5.1.2.2.36 -r* Trap range errors](#)
[5.1.2.2.37 -rtlf Send run-time errors to log file](#)
[5.1.2.2.38 -rtmb* Send run-time errors to message box](#)
[5.1.2.2.39 -rtsc* Send run-time errors to screen](#)
[5.1.2.2.40 -s* Strict var strings](#)
[5.1.2.2.41 -sc* Use short-circuit evaluation](#)
[5.1.2.2.42 -so Maximum stack overflow checking](#)
[5.1.2.2.43 -Snn Set stack size in K](#)
[5.1.2.2.44 -u* Trap use of undefined values](#)
[5.1.2.2.45 -v* Trap use of inactive variants](#)
[5.1.2.2.46 -W Control warning messages](#)

[5.2 The Command-Line Interpreter](#)

[5.2.1 Using the interpreter](#)

[5.3 The Header Utility](#)

[5.3.1 Using the header utility](#)

[6 Extensions To Standard Pascal](#)

[6.1 Overview of extensions to Standard Pascal](#)

[6.2 Auto declare input and output](#)

[6.3 Allow binary/hexadecimal constants](#)

[6.4 Enable non-standard constants](#)

[6.5 Allow constant ranges](#)

[6.6 Allow double-quoted literals](#)

[6.7 Enable non-standard functions](#)

[6.8 Allow non-numeric labels](#)

[6.9 Enable non-standard operators](#)

[6.10 Allow 'otherwise'](#)

[6.11 Enable non-standard procedures](#)

[6.12 Allow relaxed declaratons](#)

[6.13 Enable non-standard types](#)

[6.14 Allow underscores \(_ \) in identifiers](#)

[6.15 Enable non-standard variables](#)

[7 Read Me](#)

[7.1 What is Irie Pascal?](#)

[7.2 Compliance](#)

[7.3 License and distribution rights](#)

[7.4 Disclaimer-Agreement](#)

[7.5 How To Get Help](#)

[7.5.1 Getting help from the IDE](#)

[7.5.2 Getting help from the website](#)

[7.5.3 Contacting customer support](#)

[7.6 Irie Pascal Prices](#)

[7.6.1 Checking prices](#)

[7.6.2 Irie Pascal Windows Edition Prices](#)

[7.6.2.1 Irie Pascal Windows Edition Prices \(in US\\$\)](#)

[7.6.2.2 Irie Pascal Windows Edition Prices \(in CA\\$\)](#)

[7.6.2.3 Irie Pascal Windows Edition Prices \(in UK\)](#)

[7.6.2.4 Irie Pascal Windows Edition Prices \(in Euros\)](#)

[7.6.3 Irie Pascal Linux Edition Prices](#)

[7.6.3.1 Irie Pascal Linux Edition Prices \(in US\\$\)](#)

[7.6.3.2 Irie Pascal Linux Edition Prices \(in CA\\$\)](#)

[7.6.3.3 Irie Pascal Linux Edition Prices \(in UK\)](#)

[7.6.3.4 Irie Pascal Linux Edition Prices \(in Euros\)](#)

[7.6.4 Irie Pascal FreeBSD Edition Prices](#)

[7.6.4.1 Irie Pascal FreeBSD Edition Prices \(in US\\$\)](#)

[7.6.4.2 Irie Pascal FreeBSD Edition Prices \(in CA\\$\)](#)

[7.6.4.3 Irie Pascal FreeBSD Edition Prices \(in UK\)](#)

[7.6.4.4 Irie Pascal FreeBSD Edition Prices \(in Euros\)](#)

[7.6.5 Irie Pascal Solaris/x86 Edition Prices](#)

[7.6.5.1 Irie Pascal Solaris/x86 Edition Prices \(in US\\$\)](#)

[7.6.5.2 Irie Pascal Solaris/x86 Edition Prices \(in CA\\$\)](#)

[7.6.5.3 Irie Pascal Solaris/x86 Edition Prices \(in UK\)](#)

[7.6.5.4 Irie Pascal Solaris/x86 Edition Prices \(in Euros\)](#)

[7.6.6 Irie Pascal Solaris/Sparc Edition Prices](#)

[7.6.6.1 Irie Pascal Solaris/Sparc Edition Prices \(in US\\$\)](#)

[7.6.6.2 Irie Pascal Solaris/Sparc Edition Prices \(in CA\\$\)](#)

[7.6.6.3 Irie Pascal Solaris/Sparc Edition Prices \(in UK\)](#)

[7.6.6.4 Irie Pascal Solaris/Sparc Edition Prices \(in Euros\)](#)

[7.6.7 Irie Pascal Universal Edition Prices](#)

[7.6.7.1 Irie Pascal Universal Edition Prices \(in US\\$\)](#)

[7.6.7.2 Irie Pascal Universal Edition Prices \(in CA\\$\)](#)

[7.6.7.3 Irie Pascal Universal Edition Prices \(in UK\)](#)

[7.6.7.4 Irie Pascal Universal Edition Prices \(in Euros\)](#)

[7.7 Buying Irie Pascal licenses](#)

[7.7.1 Why you should buy a license](#)

[7.7.2 How do you buy a license](#)

[7.7.3 Buying using the IDE](#)

[7.7.4 Buying through the web](#)

[7.7.5 Buying by telephone](#)

[7.7.6 Buying by Fax](#)

[7.7.7 Buying through the mail](#)

[7.7.8 Buying by wire transfer](#)

[7.7.9 Purchase orders](#)

[7.7.10 Irie Pascal Order Forms](#)

[7.7.10.1 Irie Pascal \(Windows Edition\) Order Forms](#)

[7.7.10.1.1 Irie Pascal \(Windows edition\) US\\$ Order Form](#)

[7.7.10.1.2 Irie Pascal \(Windows Edition\) CA\\$ Order Form](#)

[7.7.10.1.3 Irie Pascal \(Windows Edition\) UK Order Form](#)

[7.7.10.1.4 Irie Pascal \(Windows Edition\) Euro Order Form](#)

[7.7.10.2 Irie Pascal Order Forms \(Linux Edition\)](#)

- [7.7.10.2.1 Irie Pascal \(Linux Edition\) US\\$ Order Form](#)
- [7.7.10.2.2 Irie Pascal \(Linux Edition\) CA\\$ Order Form](#)
- [7.7.10.2.3 Irie Pascal \(Linux Edition\) UK Order Form](#)
- [7.7.10.2.4 Irie Pascal \(Linux Edition\) Euro Order Form](#)
- [7.7.10.3 Irie Pascal Order Forms \(FreeBSD Edition\)](#)
 - [7.7.10.3.1 Irie Pascal \(FreeBSD Edition\) US\\$ Order Form](#)
 - [7.7.10.3.2 Irie Pascal \(FreeBSD Edition\) CA\\$ Order Form](#)
 - [7.7.10.3.3 Irie Pascal \(FreeBSD Edition\) UK Order Form](#)
 - [7.7.10.3.4 Irie Pascal \(FreeBSD Edition\) Euro Order Form](#)
- [7.7.10.4 Irie Pascal Order Forms \(Solaris/x86 Edition\)](#)
 - [7.7.10.4.1 Irie Pascal \(Solaris/x86 Edition\) US\\$ Order Form](#)
 - [7.7.10.4.2 Irie Pascal \(Solaris/x86 Edition\) CA\\$ Order Form](#)
 - [7.7.10.4.3 Irie Pascal \(Solaris/x86 Edition\) UK Order Form](#)
 - [7.7.10.4.4 Irie Pascal \(Solaris/x86 Edition\) Euro Order Form](#)
- [7.7.10.5 Irie Pascal Order Forms \(Solaris/Sparc Edition\)](#)
 - [7.7.10.5.1 Irie Pascal \(Solaris/Sparc Edition\) US\\$ Order Form](#)
 - [7.7.10.5.2 Irie Pascal \(Solaris/Sparc Edition\) CA\\$ Order Form](#)
 - [7.7.10.5.3 Irie Pascal \(Solaris/Sparc Edition\) UK Order Form](#)
 - [7.7.10.5.4 Irie Pascal \(Solaris/Sparc Edition\) Euro Order Form](#)
- [7.7.10.6 Irie Pascal Order Forms \(Universal Edition\)](#)
 - [7.7.10.6.1 Irie Pascal \(Universal Edition\) US\\$ Order Form](#)
 - [7.7.10.6.2 Irie Pascal \(Universal Edition\) CA\\$ Order Form](#)
 - [7.7.10.6.3 Irie Pascal \(Universal Edition\) UK Order Form](#)
 - [7.7.10.6.4 Irie Pascal \(Universal Edition\) Euro Order Form](#)

1.1 New IDE features

This release (i.e. version 2.6) of Irie Pascal is mainly a maintenance release, which fixes a number of bugs, and includes a few improvements to the Integrated Development Environment (IDE).

New improvements to the IDE

- Additional keyboard short-cuts (see the complete list [here](#)).
- The use of mouse wheels are now supported (see [using mouse wheels](#) for more information).
- You can now select all of the available text display preferences (font and color) from a single dialog box (see the [Text Preferences page](#) for more information).
- The IDE can now operate in *file mode* (if you prefer not to have to start a new project for each new program that you create), and well as in *project mode* (which is the default mode and recommended for most people). See the [Executable Preferences page](#) for more information about file and project modes (including how to switch between them), and to select executable options for file mode.

2.1 Getting help

A variety of resources are available to help you get the most out of Irie Pascal. For more information see the list below:

- [Getting help from the IDE](#)
- [Getting help from the website](#)
- [Contacting customer support](#)

2.2 Installing and uninstalling

Minimum system requirements for Irie Pascal (Windows Edition)

- Win95 or later. **NOTE:** Installation on Windows NT 4.0 requires Windows NT 4.0 service pack 3 or later.
- Pentium processor.
- 5 MB disk space.

Installing Irie Pascal (Windows Edition)

To install Irie Pascal (Windows Edition) run the setup executable (i.e. either **ipw-eval.exe** or **ipw-260.exe**, and follow the directions given.

Uninstalling Irie Pascal

To uninstall Irie Pascal use the Add/Remove Program applet in the Windows Control Panel.

2.3 Getting started creating programs

See [creating new programs](#) for information on how to create programs.

2.4.1 Getting started with the sample programs

In order to help get you started, a number of sample [projects](#) have been included with Irie Pascal. These samples are silently copied to the your *Samples* folder the first time you run the Irie Pascal. Your *Samples* folder is located at

[My Documents]\Irie Tools\Pascal Projects\Samples

where [My Documents] is your *My Documents* folder in Windows.

Some of these sample projects are listed below:

- [The hello world sample program](#)
- [The hello world sample program \(CGI version\)](#)
- [The hello world sample program \(Windows API version\)](#)

2.4.2 The hello world sample program

One of the sample [projects](#) included with Irie Pascal is named **hello**. When this project is run, it displays the message "Hello world!" in a console window. [Compiling](#) and running this [project](#) is a useful way to check whether Irie Pascal is installed correctly.

2.4.3 The hello world sample program (CGI version)

One of the sample [projects](#) included with Irie Pascal is named **hellocgi**, and is a Common Gateway Interface (CGI) version of the [hello sample project](#). When the **hellocgi** project is run, it generates a web page with the message "Hello world!". [Compiling](#) this project and installing the executable on your web server is a useful way to check whether you have correctly configured the web server to execute Irie Pascal CGI programs.

NOTE: CGI is a very simple and popular way of integrating programs with web servers. You can find further information about CGI on the web (including on the Irie Tools website at the following URL: www.iriertools.com/cgi). There are also numerous books available that cover CGI.

2.4.4 The hello world sample program (Windows API version)

One of the sample [projects](#) included with Irie Pascal is named **helloworld**, and is a Windows API version of the **hello** project. When the **helloworld** project is run, it uses the Windows API to display the message "Hello world!" in a Windows message box.

3.1 How to get help

See [getting help](#) for information on the available Irie Pascal help resources and how to access them.

3.2 How to buy

See [buying licenses](#) for information on how to buy Irie Pascal licenses.

3.3 How to contact Irie Tools

Contact Information

E-Mail:

- sales@iriertools.com (for sales related enquiries)
- support@iriertools.com (for customer support enquiries)
- sking@iriertools.com (for all other enquiries)

Fax:

1-876-946-2703

Postal Mail:

Attn: Stuart King

3.4.1 Creating new programs

The best way to create a new program, with the Irie Pascal Integrated Development Environment (i.e. the IDE), depends on whether the IDE is in *project mode* or in *file mode*. **NOTE:** The [Executable Preferences page](#) of the [User Preferences dialog box](#) is used to switch between file and project modes, and to select executable options for file mode.

Creating programs in project mode

If the IDE is in *project mode*, you should create a new program by first creating a new [project](#) for the program. When you create the new [project](#) the IDE will automatically create the new program for you, and associate this program with the project (see [creating new projects](#) for more information). Whenever you compile or run a project, the IDE will compile or run the program associated with the project.

Creating programs in file mode

If the IDE is in *file mode*, you create a new program by simply creating a new file (see the [File menu](#) for more information on creating a new file). In this mode the executable options on the [Executable Preferences page](#) will affect how the program is compiled and run.

3.4.2 Opening existing programs

The best way to open an existing program, with the Irie Pascal Integrated Development Environment (i.e. the IDE), depends on whether the IDE is in *project mode* or in *file mode*. **NOTE:** The [Executable Preferences page](#) of the [User Preferences dialog box](#) is used to switch between file and project modes, and to select executable options for file mode.

Opening programs in project mode

If the IDE is in *project mode*, you should open an existing program by opening the [project](#) with which it is associated. When you open an existing [project](#) the IDE will automatically open the program associated with the project (see [opening existing projects](#) for more information). Whenever you compile or run a project, the IDE will compile or run the program associated with the project.

Opening programs in file mode

If the IDE is in *file mode*, you open an existing program by simply opening the file containing the program (see the [File menu](#) for more information on opening files). In this mode the executable options on the [Executable Preferences page](#) will affect how the program is compiled and run.

3.4.3 Compiling programs

The best way to compile a program, with the Irie Pascal Integrated Development Environment (i.e. the

IDE), depends on whether the IDE is in *project mode* or in *file mode*. **NOTE:** The [Executable Preferences page](#) of the [User Preferences dialog box](#) is used to switch between file and project modes, and to select executable options for file mode.

Compiling programs in project mode

If the IDE is in *project mode*, you should compile an existing program by compiling the [project](#) with which it is associated. When you compile an existing [project](#) the IDE will automatically compile the program associated with the project (see [The Compile menu entry](#) for more information). In this mode the executable options from the [Miscellaneous options page](#) will be used when compiling the program.

Compiling programs in file mode

If the IDE is in *file mode*, you compile an existing program by simply opening the file containing the program (see the [File menu](#) for more information on opening files), and then compiling it. In this mode the executable options on the [Executable Preferences page](#) will affect how the program is compiled (see [The Compile menu entry](#) for more information).

3.4.4 Running programs

The best way to run a program, with the Irie Pascal Integrated Development Environment (i.e. the IDE), depends on whether the IDE is in *project mode* or in *file mode*. **NOTE:** The [Executable Preferences page](#) of the [User Preferences dialog box](#) is used to switch between file and project modes, and to select executable options for file mode.

Running programs in project mode

If the IDE is in *project mode*, you should run an existing program by running the [project](#) with which it is associated. When you run an existing [project](#) the IDE will automatically run the program associated with the project (see [The Compile menu entry](#) for more information). In this mode the executable options from the [Miscellaneous options page](#) will be used when running the program.

Running programs in file mode

If the IDE is in *file mode*, you run an existing program by simply opening the file containing the program (see the [File menu](#) for more information on opening files), and then running it. In this mode the executable options on the [Executable Preferences page](#) will affect how the program is run. See [running programs](#) for more information.

3.5.1 Page setup


The Page Setup Dialog Box

The Page Setup dialog box is activated by choosing **Page Setup...** from the [File menu](#), and allows you to set common printer options that affect how pages are printed.

3.5.2 Printing files

Printing Files

The Irie Pascal IDE allows you to print files by doing any of the following:

1. Select **Print...** from the [File menu](#).
2. Press Ctrl+P.
3. Click on the **Print File** button  in the [toolbar](#).

All three of the above methods bring up the same print dialog box that allows you to specify various print options and then start printing the file you are currently editing. **NOTE:** If the file you are currently editing has some selected text then the print dialog box will automatically default to printing only the selected text. In this case, if you want to print the entire file you will need to choose the appropriate option (usually you choose an option called **All**).

3.6.1 Choosing fonts

The Irie Pascal IDE allows you to select your preferred font from among the monospace fonts available through Windows. To select your preferred font choose **Preferences...** from the [Edit menu](#). This will take you to the [Text Preferences page](#) of the [User Preferences dialog box](#).

The IDE will remember your preferred font and automatically use that font when it displays text. If you are not the only person using Irie Pascal on a particular computer, then in order for the IDE to remember your preferred font, you must log on to Windows using your own user id. If you log on to Windows with someone else's user id or a shared user id then the IDE will use the font choice of the last person to log on the Windows with that user id.

NOTE: The IDE stores your font preferences in the registry under the following keys:

`\HKEY_CURRENT_USER\Software\Irie Tools\Irie Pascal\Settings\font name` and
`\HKEY_CURRENT_USER\Software\Irie Tools\Irie Pascal\Settings\font size`.

3.7.1 Choosing colors

The Irie Pascal IDE allows you to select the background and text colors used when displaying text in the editor and message windows. To select colors choose **Preferences...** from the [Edit menu](#). This will take you to the [Text Preferences page](#) of the [User Preferences dialog box](#).

The IDE will remember your color selections and automatically use those colors whenever it displays text. If you are not the only person using Irie Pascal on a particular computer, then in order for the IDE to remember your color selections, you must log on to Windows using your own user id. If you log on to Windows with someone else's user id or a shared user id then the IDE will use the color selections of the last person to log on the Windows with that user id.

NOTE: The IDE stores color selections in the registry under the following key:

`\HKEY_CURRENT_USER\Software\Irie Tools\Irie Pascal\Settings\colors`.

3.8.1 Selecting a section of text

NOTE: The term *current file* is used below to mean the file in the active text editor window (or in other words the file you are currently editing).

Most of the time when editing text, you are working with individual characters. For example, when entering text with the keyboard you type each individual character. However sometimes you want to work with a section of text as a whole.

The first step in working with a section of text is to select it. You can use the mouse, keyboard, or [Edit menu](#) to select a section of text. After the text is selected, it can be [replaced](#), [moved](#), [copied](#), [deleted](#), [printed](#), [indented](#), [unindented](#), and [unselected](#)

Selecting text using the mouse

Selecting text by *dragging* the mouse pointer

To select text by *dragging* the mouse pointer, do the following:

1. Move the mouse pointer to the beginning of the text
2. Hold down the left mouse button
3. Move the mouse pointer to the end of the text
4. Release the left mouse button

You can also select text by *dragging* the mouse pointer from the end of the text to the beginning.

Selecting text by double-clicking with the mouse

If the **Double-click word search** project option is not selected, (see [environment options](#) for details) then you can select a word by double-clicking on it. If you double-click on the word again then the line containing the word will be selected. If you double-click on the selected line then the IDE will select a block of text. The first line of the block of text will be the line that you double-clicked on. The IDE will use the indentation of the first line in the block to determine where the block should end. If the IDE ends the block too soon and you would like the block to be extended to include more lines then you can double-click on the block again. NOTE: If the block reaches a line that is indented less than the first line in the block then the block will not be extended any further.

Selecting text using the keyboard

Selecting text by *dragging* the text caret

To select text by *dragging* the text caret, do the following:

1. Use the keyboard keys to move the caret to the beginning of the text
2. Hold down the **Shift** key
3. Use the keyboard keys to move the caret to the end of the text
4. Release the **Shift** key

You can also select text by *dragging* the caret from the end of the text to the beginning.

Selecting text using the Edit menu

To select text using the [Edit menu](#), choose one of the following four menu items.

1. **Select Word** - Selects the word under the text caret
2. **Select Line** - Selects the line under the text caret
3. **Select Block** - Selects a text block or extends an existing text block. The first line of the block of text will be the line under the text caret. The IDE will use the indentation of the first line in the block to determine where the block should end. If the IDE ends the block too soon and you would like the block to be extended to include more lines then you can choose **Select Block** from the [Edit menu](#) again. NOTE: If the block reaches a line that is indented less than the first line in the block then the block will not be extended any further.
4. **Select All** - Selects all of the text in the current file

3.8.2 Replacing a section of text

Sometimes you will want to replace a section of text with some other text. The Irie Pascal Integrated Development Environment (the IDE) allows you to replace sections of text.

The first step in replacing a section of text is to select it (see [selecting text](#) for more information). Then you can either type in the new text, or you can paste the new text from the Windows clipboard. In either case the selected text is deleted and replaced by the new text.

NOTE: Since it is possible to replace text by accident (all you have to do is forget that a section of text has been selected, and type some more text or paste some text from the clipboard), it is a good idea to unselect sections of text after you are finished working with them.

3.8.3 Moving a section of text

NOTE: The term *current file* is used below to mean the file in the active text editor window (or in other words the file you are currently editing).

Sometimes you will want to move a section of text (i.e. remove the text from one place and put it somewhere else). The Irie Pascal Integrated Development Environment (the IDE) allows you to move sections of text. The first step in moving sections of text is to select the section of text you want to move (see [selecting text](#) for more information). The next step is to *cut* the selected text (i.e. move the text into the Windows clipboard). The next step is to *paste* the text where you want to put it (i.e. copy the text from the Windows clipboard to the place you want to put it). NOTE: A copy of the text remains in the Windows clipboard even after you *paste* it. **NOTE:** Remember to unselect the section of text (See [unselecting text](#) for more information) when you are done.

Cutting Text

You can *cut* text by using the keyboard or by using the [Edit menu](#). To *cut* text using the keyboard, press **Ctrl-X**, or **Shift-Delete** (i.e. hold down the **Ctrl** key and press the **X** key, or hold down the **Shift** key and press the **Delete** key). You can also *cut* text by choosing **Cut** from the [Edit menu](#). No matter what method you prefer the result is the same, any selected text in the current file is removed and stored in the Windows clipboard. If no text has been selected, in the current file, then attempting to *cut* text has no effect on the file or on the Window clipboard. NOTE: Many Windows applications can retrieve text from the Windows clipboard, so you can use this method to move text into another application.

Pasting Text

You can *paste* text by using the the keyboard or by using the [Edit menu](#). To *paste* text using the keyboard, press **Ctrl-V**, or **Shift-Insert** (i.e. hold down the **Ctrl** key and press the **V** key, or hold down the **Shift** key and press the **Insert** key). You can also *paste* text by choosing **Paste** from the [Edit menu](#). No matter what method you prefer the result is the same, any text in the Windows clipboard is copied into the current file. If there is selected text in the current file then the text from the clipboard will replace the selected text in the file. If there is no selected text in the current file then the text from the clipboard will be inserted into the current file at the text caret position. If there is no text in the clipboard then attempting to *paste* text has no effect. NOTE: Many Windows applications allow you to put text into the Windows clipbaord, so you can use this method to retrieve text from another application.

3.8.4 Copying a section of text

NOTE: The term *current file* is used below to mean the file in the active text editor window (or in other words the file you are currently editing).

Sometimes you will want to copy a section of text (i.e. create a duplicate of the text and store that duplicate in a file or in another application). The Irie Pascal Integrated Development Environment (the IDE) allows you to copy sections of text. The first step in copying a section of text is to select the section of text you want to copy (see [selecting text](#) for more information). The next step is to *copy* the selected text (i.e. copy the text into the Windows clipboard). The next step is to *paste* the text where you want to put it (i.e. copy the text from the Windows clipboard to the place you want to put it). A copy of the text remains in the Windows clipboard even after you *paste* it. **NOTE:** Remember to unselect the section of text (See [unselecting text](#) for more information) when you are done.

Copying Text

You can *copy* text by using the the keyboard or by using the [Edit menu](#). To *copy* text using the keyboard, press **Ctrl-C**, or **Ctrl-Insert** (i.e. hold down the **Ctrl** key and press **X**, or hold down the **Ctrl** key and press **Insert**). You can also *copy* text by choosing **Copy** from the [Edit menu](#). No matter what method you prefer the result is the same, any selected text in the current file is copied and stored in the Windows clipboard. If no text has been selected, in the current file, then attempting to *copy* text has no effect. NOTE: Many Windows applications can retrieve text from the Windows clipbaord so you can use this method to copy text into another application.

Pasting Text

You can *paste* text by using the the keyboard or by using the [Edit menu](#). To *paste* text using the keyboard, press **Ctrl-V**, or **Shift-Insert** (i.e. hold down the **Ctrl** key and press the **V** key, or hold down the **Shift** key and press the **Insert** key). You can also *paste* text by choosing **Paste** from the [Edit menu](#). No matter what method you prefer the result is the same, any text in the Windows clipboard is copied into the current file. If there is selected text in the current file then the text from the clipboard will replace the selected text in the file. If there is no selected text in the current file then the text from the clipboard will be inserted into the current file at the text caret position. If there is no text in the clipboard then attempting to *paste* text has no effect. NOTE: Many Windows applications allow you to put text into the Windows clipbaord so you can use this method to retrieve text from another application.

3.8.5 Deleting a section of text

Sometimes you will want to delete a section of text. The Irie Pascal Integrated Development Environment (the IDE) allows you to delete sections of text. First you need to select the section of text you want to delete (see [selecting text](#) for more information). The next step is to delete the selected text.

Deleting Selected Text

You can *delete* selected text by using the keyboard or by using the [Edit menu](#). To *delete* selected text using the keyboard, press the **Delete** key. To *delete* selected text using the [Edit menu](#), select **Clear** from the [Edit menu](#).

3.8.6 Printing a section of text

Sometimes you don't want to print an entire file, instead you just want to print a section of the file. The Irie Pascal Integrated Development Environment (the IDE) allows you to print sections of files. In order to print a section of a file, you should do the following:

- Select the section of text you want to print (see [selecting text](#) for more information)
- Bring up the **Print** dialog box (see [printing files](#) for more information).

NOTE: Remember to unselect the section of text (see [unselecting text](#) for more information) when you are done.

3.8.7 Indenting a section of text

The Irie Pascal Integrated Development Environment (the IDE) allows you to indent sections of text. The first step in indenting a section of text is to select the section of text you want to indent (see [selecting text](#) for more information). The next step is to press the **Tab** key until the section of text is indented to the level you want. **NOTE:** Remember to unselect the section of text (See [unselecting text](#) for more information) when you are done.

3.8.8 Unindenting a section of text

The Irie Pascal Integrated Development Environment (the IDE) allows you to unindent sections of text (i.e. reduce the indentation level of a section of text). The first step in unindenting a section of text is to select the section of text you want to indent (see [selecting text](#) for more information). The next step is to press Shift-Tab (i.e. hold down the **Shift** key and press the **Tab** key) until the section of text is unindented to the level you want. **NOTE:** Remember to unselect the section of text (See [unselecting text](#) for more information) when you are done.

3.8.9 Unselecting a section of text

After you have finished working with a selected section of text, it is a good idea to remove the selection so that you do not accidentally replace or delete the selected section of text. See [replacing a section of text](#) for more information.

You can remove a selection using the mouse or the keyboard. To remove a selection using the mouse simply click, with the left mouse button, anywhere in the edit window containing the selection. To remove a selection using the keyboard simply use the arrow keys to move the text caret.

3.9.1 Creating IVM executables

What Are IVM Executables?

First let me briefly describe what executables are in general, before going on to tell you what IVM executables are. As you probably already know, a computer has no will of its own, and will not do anything unless it is told exactly what to do. A set of instructions telling a computer what to do is called a program, and these programs are usually stored in computer files. The computer is able to read the files, containing programs, and execute the instructions. The files containing instructions to be executed are sometimes called *executable files* or just *executables* for short.

IVM executables are files containing instructions for an imaginary computer called the Irie Virtual Machine (IVM). So you ask "what good is it to have files with instructions telling an imaginary computers what to do"? Well, it is possible for real computers to be programmed to execute instructions for other (real and imaginary) computers. Programs that tell one computer how to execute the instructions of other computers are called interpreters. In order to make use of an IVM executable, a real computer needs an interpreter for the IVM. An interpreter for the IVM is included with Irie Pascal and consists of two files:

- [ivm.exe](#) - The interpreter's user interface
- [iriert26.dll](#) - The interpreter's run-time engine

Creating IVM Executables

By default Irie Pascal creates IVM executables so usually you don't have to do anything to make it create an IVM executable. However if you want to make certain that you create an IVM executable, then all you have to do is make sure that the name of the executable you generate ends with `.ivm` (actually it might be more accurate to say that all you have to do is make sure that the name of the executable does **not** end with `.exe`). See [miscellaneous options](#) for information on how to specify the name of the executable.

3.9.2 Creating EXE executables

By default Irie Pascal does not generate EXE executables, but instead generates IVM executables (see [creating IVM executables](#) for more information). However all you have to do to make Irie Pascal generate EXE executables is make sure that the name of the executable you generate ends with `.exe` (see [miscellaneous options](#) for information on how to specify the name of the executable).

3.9.3.1 Creating WinNT/2000 services

What are services

Irie Pascal can now generate Windows NT/2000 services. A service is a special kind of application, that

can be started automatically at system boot time, or can be started manually by using the Services Control Panel applet. Services are intended to be non-interactive programs that run in the background, and can even run when no user is logged on to the system. Services generated by Irie Pascal can be run on Windows NT and Windows 2000. The authoritative source of information about services is the Microsoft Developers Network (MSDN). You can access the MSDN on the Microsoft website at msdn.microsoft.com.

Creating WinNT/2000 services

Services must be EXE executables, so in order to create a services you must generate an EXE executable. See [creating EXE executables](#) for more information. In order to generate a service you also need to tell Irie Pascal to generate a service, this is done by selecting **Generate service application** in the Code Generation Project Options. See [code generation options](#) for more information.

3.9.3.2 Writing well behaved services

Irie Pascal makes it very easy to create services, see [creating services](#) for more information. However there is one thing you will need to take care. It is up to you to make sure that the services you create stop promptly when requested to do so. Services are requested to stop when Windows is shutting down, and when you stop them using the Services Control Applet in the Control Panel. In either case the system will not wait forever for your service to stop. Well behaved services should not take too long (certainly not longer than a few seconds) to stop. Irie Pascal supports two built-in functions that you can use to detect when your service has been request to stop. These two functions are the *StopServiceEvent* function and the *wait* function.

The StopServiceEvent Function

The *StopServiceEvent* function takes no arguments and returns one of two values depending on whether your application is a service or not. If your application is **not** a service then the *StopServiceEvent* function will always return zero. If you application is a service then the *StopServiceEvent* function will return a handle to the event object that will signal when your service has been requested to stop. See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more information.

The Wait Function

The *wait* function takes at least two arguments, the first argument is the maximum length of time the function should wait (in milliseconds). The other arguments are handles to objects that the *wait* function should wait on. The *wait* function will wait until one of the objects it is waiting on signals it or until the maximum wait interval expires. The *wait* function returns the handle to the first object that signals it, or zero if no object signals it before the maximum wait interval expires. **NOTE:** If the value of the first argument to the *wait* function is zero then the *wait* function doesn't actually wait instead it just checks whether any of the objects it is waiting on has signaled it. If the value of the first argument to the *wait* function is equal to the built-in constant *maxword* then the *wait* function will wait forever for one the objects it is waiting on to signal it. See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more information.

Well Behaved Services

Well behaved services will frequently check to see if they have been requested to stop by calling the

wait function and passing the handle returned by the *StopServiceEvent* function as the second argument. The first argument will usually be zero so that no time is wasted if the services has not been requested to stop.

The authoritative source of information about services is the Microsoft Developers Network (MSDN). You can access the MSDN on the Microsoft website at msdn.microsoft.com.

3.10.1 What are COM/ActiveX objects?

The Component Object Model

The Microsoft Developer Network (MSDN) is an important source of information for developers about Microsoft technologies. The MSDN Library (October 1999 edition) defines COM as follows:

The Component Object Model (COM) is a platform-independent, distributed, object-oriented, system for creating binary software components that can interact. COM is the foundation technology for Microsoft's OLE (compound documents), ActiveX (internet enabled components), as well as others.

Overview Of Automation

The MSDN Library (October 1999 edition), and defines Automation follows:

Automation (formerly called OLE Automation) is a technology that allows software packages to expose their unique features to scripting tools and other applications. Automation uses the Component Object Model (COM), but may be implemented independently from other OLE features, such as in-place activation. Using Automation, you can:

- 1. Create applications and programming tools that expose objects.*
- 2. Create and manipulate objects exposed in one application from another application.*
- 3. Create tools that access and manipulate objects. These tools can include embedded macro languages, external programming tools, object browsers, and [compilers](#).*

The objects an application or programming tool exposes are called ActiveX objects. Applications and programming tools that access those objects are called ActiveX clients. ActiveX objects and clients interact as follows:

Applications and other software packages that support ActiveX technology define and expose objects which can be acted on by ActiveX components. ActiveX components are physical files (for example .exe and .dll files) that contain classes, which are definitions of objects. Type information describes the exposed objects, and can be used by ActiveX components at either compile time or at run time.

Irie Pascal and ActiveX objects

Irie Pascal accesses the type information about exposed ActiveX objects at run time only. At compile time Irie Pascal does not know anything about the names or types of the objects' methods or properties.

3.10.2 Instantiating ActiveX objects

Instantiating ActiveX Objects

To instantiate (i.e. create an instance of) an ActiveX object, you first need to declare a variable of type *object*, or of type *class*. Then you just call the built-in function *CreateObject*. The built-in function *CreateObject* takes as an argument a string expression that evaluates to the name of the ActiveX object type you want to instantiate. See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more information.

NOTE: After you have finished using the ActiveX object make sure you get rid of it (see [disposing of ActiveX objects](#)).

3.10.3 Calling ActiveX object methods

Invoking the Methods of ActiveX Objects

The methods exposed by ActiveX objects define the *actions* that the objects can perform. Before invoking the methods of an ActiveX object you need to instantiate the object (see [instantiating ActiveX objects](#)). Invoking a method of an instance of an ActiveX object is similar to calling a function or procedure. The difference is that where the name of the function or procedure would normally go you need to put the following:

object-variable '.' name

where **object-variable** is an object variable that contains an instance of the ActiveX object

and **name** is the name of the method being invoked.

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more information.

NOTE: After you have finished using the ActiveX object make sure you get rid of it (see [disposing of ActiveX objects](#)).

3.10.4 Accessing ActiveX object properties

Accessing the properties of ActiveX objects

The properties of ActiveX objects are functions that allow you to access the state of the objects. Before accessing the properties of an ActiveX object, you need to instantiate the object (see [instantiating ActiveX objects](#)). Although properties are really functions, the syntax you use to access a property is similar to the syntax you would use to access the fields of a record. The difference is that you use

object-variable '.' property-name

instead of

record-variable '! field-name

where **object-variable** is an object variable that contains an instance of the ActiveX object and **property-name** is the name of the property being accessed.

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more information.

NOTE: After you have finished using the ActiveX object make sure you get rid of it (see [disposing of ActiveX objects](#)).

3.10.5 Disposing of ActiveX object

After you have finished using an ActiveX object you should get rid of it using the built-in procedure *dispose*.

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more information.

3.11.1.1 Connecting to ODBC databases

What is ODBC?

The Open Database Connectivity (ODBC) interface is widely used to access Database Management Systems (DBMSs). ODBC was originally created by Microsoft but has now become an industry standard, and is supported by all major DBMSs for Windows.

Connecting to ODBC databases

In order to connect to an ODBC database, you need to:

1. Declare a variable of the built-in type *connection*
2. Use *new* on the variable to create a *connection* object
3. Invoke the *open* method of the *connection* object

The *open* method of the *connection* object takes one argument, the connection string. ODBC connection strings can take one of two forms:

odbc-connection-string = odbc-connection-string-1 | odbc-connection-string-2

odbc-connection-string-1 = 'ODBC' ';' dsn-parm ';' uid-parm ';' pwd-parm [';' [driver-specific-text]]

odbc-connection-string-2 = 'ODBC' ';' name ';' [uid] ';' [password]

dsn-parm = 'DSN' '=' name

uid-parm = 'UID' '=' uid

pwd-parm = 'PWD' '=' [password]

where [] indicates optional parameters

and **name** is a valid data source name (DSN)

and **uid** is an identifier for the user accessing the database

and **password** is the password of the user accessing the database

For example DSN=test;UID=sa;PWD=

When the connection string is in the first form then it is passed, without further processing, to *SQLDriverConnect* to open the connection. When the connection string is in the second form then the **name**, **id**, and **password** parameters are extracted from the connection string, if present, and passed to *SQLConnect* to open the connection. NOTE: The first form of the connection string is the recommended form, support for the second form is provided for completeness only.

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more information.

The authoritative source of information about ODBC is the Microsoft Developers Network (MSDN). You can access the MSDN on the Microsoft website at msdn.microsoft.com.

3.11.1.2 Connecting to MySQL databases

What is MySQL?

MySQL is a popular open source database management system (DBMS). MySQL is particularly popular for computers running Linux and FreeBSD. You can go to www.MySQL.com for more information about MySQL.

Connecting to MySQL databases

In order to connect to a MySQL database, you need to:

1. Declare a variable of the built-in type *connection*
2. Use *new* on the variable to create a *connection* object
3. Invoke the *open* method of the *connection* object

The *open* method of the *connection* object takes one argument, the connection string. MySQL connection strings take the following form:

```
mysql-connection-string = 'MYSQL' ';' mysql-parameter-list
```

```
mysql-parameter-list = mysql-parameter ';' mysql-parameter-list | empty
```

```
mysql-parameter = mysql-host-parameter | mysql-user-parameter | mysql-password-parameter | mysql-database-parameter | mysql-port-parameter | mysql-socket-parameter | mysql-compress-parameter
```

```
mysql-host-parameter = 'host' '=' "" host-name ""
```

```
mysql-user-parameter = 'user' '=' "" user-name ""
```

mysql-password-parameter = 'password' '=' ''' password '''

mysql-database-parameter = 'database' '=' ''' database-name '''

mysql-port-parameter = 'port' '=' port-number

mysql-socket-parameter = 'socket' '=' ''' socket '''

mysql-compress-parameter = 'compress' '=' boolean-value

boolean-value = 'yes' | 'no' | 'true' | 'false'

For example `MYSQL;user="testuser";database="testdb";socket="/tmp/mysql.sock";`

The connection parameters are extracted from the connection string and passed to *mysql_real_connect* to open the connection. NOTE: *mysql_real_connect* is the MySQL C API function that is used to open a connection to a MySQL database.

The effect of each of the parameters is described below:

The **mysql-host-parameter** specifies the hostname or IP address of the MySQL database server. If **mysql-host-parameter** is not specified or if **host-name** is an empty string or is equal to "local-host" then the connection is opened to the local MySQL server over a UNIX socket.

The **mysql-user-parameter** specifies the username used to connect to the MySQL database server. If the **mysql-user-parameter** is not specified or if **user-name** is an empty string then the login name of the person running the application is used.

The **mysql-password-parameter** specifies the password of the user who will be connected to the database server. If the **mysql-password-parameter** is not specified or if **password** is an empty string then the connection is rejected if the user actually has a password.

The **mysql-database-parameter** specifies the initial database selected when the connection is opened. If the **mysql-database-parameter** is not specified or if **database** is an empty string then no initial database is selected. In which case you must call the **selectdatabase** method later on to select a database.

The **mysql-port-parameter** specifies the port used to remotely connect to a MySQL database server over TCP. If the **mysql-port-parameter** is not specified or if **port-number** is 0 then the default port is used.

The **mysql-socket-parameter** specifies the filename of the UNIX socket used to connect to a MySQL database server on the local machine. If the **mysql-socket-parameter** is not specified or if **socket** is an empty string then the default socket is used.

The **mysql-compress-parameter** specifies the compression is to be used when communicating with the MySQL database server. If the **mysql-compress-parameter** is not specified then compression is not used.

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more information.

3.11.2.1 Using the execute method

Executing Database Commands

Most major Database Management Systems (DBMSs) use the Structured Query Language (SQL) as their query language. So executing a database command usually means sending an SQL statement to the DBMS for execution. You can call the *execute* method of a *connection* object to execute a database command. Before calling the *execute* method the *connection* object must represent an open connection to a database. See [connecting to ODBC databases](#) and [connecting to MySQL databases](#) for more information about connecting to databases.

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more information.

3.11.3.1 Using the recordset object

Most major Database Management Systems (DBMSs) use the Structured Query Language (SQL) as their query language. Some SQL statements return result sets. The *recordset* object is used to send SQL statements to DBMSs and access the result set(s) returned. The *open* method of a *recordset* object is used to send the SQL statement to the DBMS and prepare to access the result set returned. After opening the *recordset* object you can access the result set, one record at a time.

The *field* property of a *recordset* object is used to retrieve the value of a field of the current record in the result set. For example if you have a *recordset* object **rs** with an open recordset and the records in the recordset have a field named **last_name** then

```
rs.field('last_name')
```

contains the contents of the field **last_name** for the current record in the recordset.

field is a read-only property of type *variant*.

NOTE: The *field* property is read-only so you can not use it to change the contents of the record fields (i.e. you can't use this property to effect the data in the database). If you want to affect the data in the database you should call the *execute* method of an *connection* object and pass it a SQL UPDATE statement.

NOTE: Since the *field* property is used so often Irie Pascal allows you to leave out the name of the property. So for example

```
rs.('last_name')
```

is equivalent to

```
rs.field('last_name')
```

Other useful methods and properties of *recordset* objects are:

- The *movenext* method of a *recordset* object can be used to advance to the next record in the result set returned by a query.
- The *eof* property of a *recordset* object indicates whether or not the current record of the result set is the last record.
- The *close* method of a *recordset* object should be used when you have finished accessing a result

set.

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more information.

3.12.1 Programing Sockets

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more information on how to program sockets.

3.13.1 Distributing Irie Pascal programs

Distributing Irie Pascal Programs

So you have written your program, finished testing it, and now you are ready to distribute it. Well, most Windows programs nowadays are distributed as setup packages, and these setup packages provide an easy way for the programs to be installed to, and uninstalled from, the user's computer. If you decide to distribute your program as a setup package, you will need to use a third-party utility to create the setup package, because Irie Pascal does not include such a utility. Using a third-party utility should not be a problem because Irie Pascal programs are very easy to install.

Setup Folders

Most, if not all, setup packages will prompt the user to select the *installation folder* (i.e. the folder that the program files should be installed into). The utility that creates the setup package will allow you to specify which files you want to put into the *installation folder*. Most setup creation utilities also allow you to specify which files to put in special directories, such as the *Windows folder*, or the *Windows Systems folder*. In the case of Irie Pascal programs you basically have two choices for placing files:

1. You can put all the files into the *installation folder*
2. You can put all the files, except the Irie Run-Time Engine, into the *installation folder*. The Irie Run-Time Engine can be installed into the *Windows System folder* or the *Windows folder*.

See below for more information about the files you will need to distribute with your program.

Running Your Program

You probably can't depend on the users of your program having Irie Pascal installed on their computers, so you will probably want the setup package to provide an easy way to run your program. Most setup creation utilities can create setup packages that will add your program to the Windows Start Menu, or create a shortcut, on the Desktop, to your program, or both.

If your program is an IVM executable then there is a slight problem you will need to overcome. The problem is that Windows, right out of the box, does not know how to run Irie Virtual Machine (IVM) executables.

- One way to deal with this problem is to *teach* Windows how to run IVM executables. This is done by creating an association in Windows between the extension *.ivm* and the IVM Interpreter (i.e. tell Windows that the IVM Interpreter should be used to execute/open files with the *.ivm* extension). When Irie Pascal is installed on a computer, this association is automatically created,

however if Irie Pascal is not being installed then the setup package will have to create this association. If your setup creation utility allows you to create associations then this is a good choice. You can then have the Start Menu entry or Shortcut on the Desktop refer directly to your program.

- Another way to deal with this problem is to create a batch file, that will execute the IVM Interpreter and pass your program as an argument. For example if your program is named **hello.ivm**, then the batch file could contain the line **ivm hello.ivm**. You would then point the Start Menu entry or the shortcut on the Desktop at the batch file, instead of at your program. **NOTE:** If you choose this option remember to distribute the batch file along with your program.
- Yet another way to deal with this problem is to point the Start Menu entry or the Shortcut on the Desktop, at the IVM Interpreter with your program being passed as an argument.

Files You Need To Distribute With IVM Executables

If your program is an Irie Virtual Machine (IVM) executable then in addition to the .ivm file containing your program, you will also need to distribute the IVM Interpreter (i.e. [ivm.exe](#)), and the Irie Run-Time Engine (i.e. the file **iriert26.dll**), along with any other files required by your particular program. If you have purchased a license to use Irie Pascal, then you may have the right to distribute the files necessary to run your programs (see the [license](#) agreement for details).

Files You Need To Distribute With EXE Executables

If your program is an EXE executable then in addition to your program, you will also need to distribute the Irie Run-Time Engine (i.e. **iriert26.dll**), along with any other files required by your particular program. **NOTE:** In this case there is no need to distribute the IVM Interpreter (i.e. [ivm.exe](#)).

4.1 Introducing the Irie Pascal IDE

The Irie Pascal Integrated Development Environment (IDE) is an easy to use programming environment, and includes the following important components:

- A multi-file text editor - Can be used to edit your Pascal source program, or any other kind of text file.
- The Irie Pascal [Compiler](#) - Used to translate your Pascal source program into an executable program (either an IVM or an EXE executable program).
- The Message Window - Used to display the messages generated by the [Compiler](#) while it is translating your Pascal source program. The Message Window and the text editor work together, to allow you to easily move from a warning or error message, in the Message Window, to the position in your Pascal source program, referred to by the message. If you double-click, with the mouse, on a warning or error message, then you will be moved to the position in your Pascal source program, referred to by the message. You can also move from a warning or error message to the position referred to by the message, using the keyboard. To do this, you need to first select the message by single-clicking with the mouse on the message, or by changing the currently selecting message with the Up-Arrow and Down-Arrow keys. Once the message you are interested in is selected, then just press the Enter key to move from that message to the position in your Pascal source program it refers to.
- The Project Manager - Used to set project options and other project specific settings. The Project Manager remembers recently opened [projects](#), as well as the files that were most recently opened or closed in each [project](#).
- The IVM Interpreter - Used to run the executable programs generated by the [Compiler](#).
- The Online Help System - Includes this User's Manual and the Irie Pascal Programmer's Reference Manual (in "progref.html"). The IDE dialog boxes provide context sensitive help by opening the

relevant topic in the User' Manual. The text editor and the Online Help System work together, to allow you to search the Irie Pascal Programmer's Reference Manual (in "progref.html") for topics keyed to individual words in the editor. To search for the current word (i.e. the word under the text editor's caret), just press the F1 key. Depending on the setting of the **Double-click word search** check box in the [environment options page](#) of the [Project Options dialog box](#), you may also be able to search for words simply by double-clicking on them.

- Other Components - The other components that make up the IDE include the menus, dialog boxes, and toolbar.

4.2 Irie Pascal projects

Irie Pascal Projects

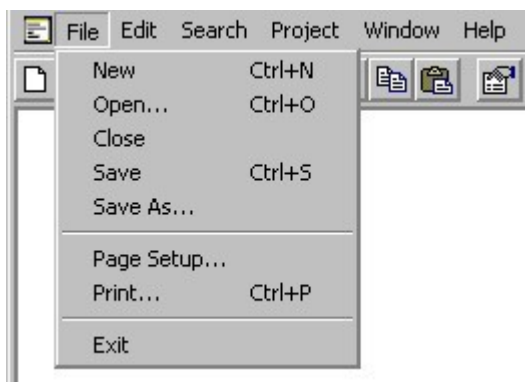
When the Irie Pascal IDE is in *project mode* (see [The Executable Preferences page](#) for more information) every program you work with must be associated with a project. Each Irie Pascal project is stored in a project file with the extension (.ipj), and only one program can be associated with an Irie Pascal project at any one time. When a project is compiled, it is the program associated with the project, that is actually compiled. You can change the program associated with a project by using the **Choose Program...** menu entry from the [Project menu](#). In addition to the program, Irie Pascal projects also contain a number of options that affect the editor, the [compiler](#), as well as the executable generated by the [compiler](#). It is recommended that you create a new project for each program that you write. This allows you to specify project options separately for each program that you write. If you do not specifically create or open a project, then the default project is used.

The Default Project

When you start the Irie Pascal IDE it automatically opens the default project. Whenever you create a new project it starts off with the options specified for the default project. So it's best to specify the options for the default project that you use most often.

4.3.1.1 File menu

The **File** menu, as the name suggests, is mainly used to perform actions on files, and is shown below:



File Menu Entries

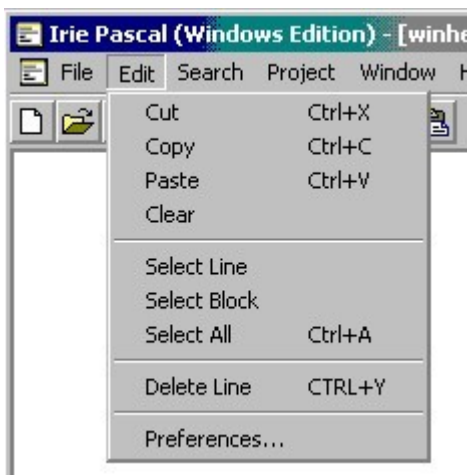
NOTE: The term *current file* is used below to mean the file in the active text editor window (or in other

words the file you are currently editing).

- **New** - This menu entry opens an empty text editor file window and gives it a temporary name (the temporary name will be similar to **new.pas**). The newly opened text becomes the current file, and has the keyboard focus so you can immediately begin typing text into the file. A new file is not actually created until you save the contents of the text editor file window, at which time you will be able to override the temporary name and choose a permanent one.
- **Open...** - This menu entry displays the Standard Open dialog box from the Windows Common Dialog Library, which gives you the chance to select a file to open. If you select a file that is not already open in the IDE, then the file is opened and loaded into a new text editor file window, and becomes the current file. If you select a file that is already open in the IDE, then that file becomes the current file (i.e. the text editor window containing the file becomes active).
- **Close** - This menu entry closes the current file. **NOTE:** You will be prompted to save any unsaved changes that have been made to the file.
- **Save** - This menu entry saves the current file, and depending on the setting of the **Create backup file** Environment Project Option a backup copy of the original contents of the file may be created. If the file being saved is new, and has never been saved before, then this menu entry works like the **Save As** menu entry described next.
- **Save As...** - This menu entry displays the Standard Save As dialog box from the Windows Common Dialog Library, which gives you a chance to choose a destination file. If you choose a destination file, then the current file is saved into the destination file, and depending on the setting of the **Create backup file** Environment Project Option a backup copy of the original contents of the destination file may be created.
- **Page Setup...** - This menu entry displays the Standard Page Setup dialog box from the Windows Common Dialog Library, which gives you a chance to set certain attributes of the printed page.
- **Print...** - This menu entry displays the Standard Print dialog box from the Windows Common Dialog Library, which gives you a chance to set printer options and print some or all of the current file.
- **Exit** - This menu entry exits the Irie Pascal IDE, and is the only **File** menu entry that does not perform a file action. **NOTE:** You will be prompted to save any unsaved files that you have open.
- **Recently Closed Files** - The text editor files that were recently closed, while working on the current project, are added to the end of the **File** menu, and can be re-opened by choosing them from the menu.

4.3.2.1 Edit menu

The **Edit** menu, as the name suggests, is mainly used to assist you in editing files, and in changing your preferences. The **Edit** menu is shown below:



Edit Menu Entries

NOTE: The term *current file* is used below to mean the file in the active text editor window (or in other words the file you are currently editing).

- **Cut** - This menu entry is used to move selected text from the current file into the Windows Clipboard (see [moving a section of text](#)). **NOTE:** If there is no selected text in the current file, then this menu entry is disabled (it will appear greyed out).
- **Copy** - This menu entry is used to copy selected text from the current file into the Windows Clipboard (see [copying a section of text](#)). **NOTE:** If there is no select text in the current file, then this menu entry is disabled (it will appear greyed out).
- **Paste** - This menu entry is used to copy text from the Windows Clipboard into the current file (see [moving a section of text](#) and [copying a section of text](#)). **NOTE:** If there is no text in the Windows Clipboard then this menu entry is disabled (it will appear greyed out).
- **Clear** - This menu entry is used to delete selected text from the current file (see [selecting a section of text](#) and [deleting a section of text](#)). **NOTE:** If there is no selected text in the current file, then this menu entry is disabled (it will appear greyed out).
- **Select Line** - This menu entry is used to select the line in the current file with the text caret (see [selecting a section of text](#)).
- **Select Block** - This menu entry is used to select a block of text in the current file (see [selecting a section of text](#)).
- **Select All** - This menu entry is used to select all of the text in the current file (see [selecting a section of text](#)).
- **Delete Line** - This menu entry is used to delete the line in the current file with the text caret.
- **Preferences...** - This menu entry is used to activate the [User Preferences dialog box](#) (which allows you to select your preferences for how text is displayed, and your preferences for how executables are generated and run).

4.3.2.2.1.1 The User Preferences dialog box

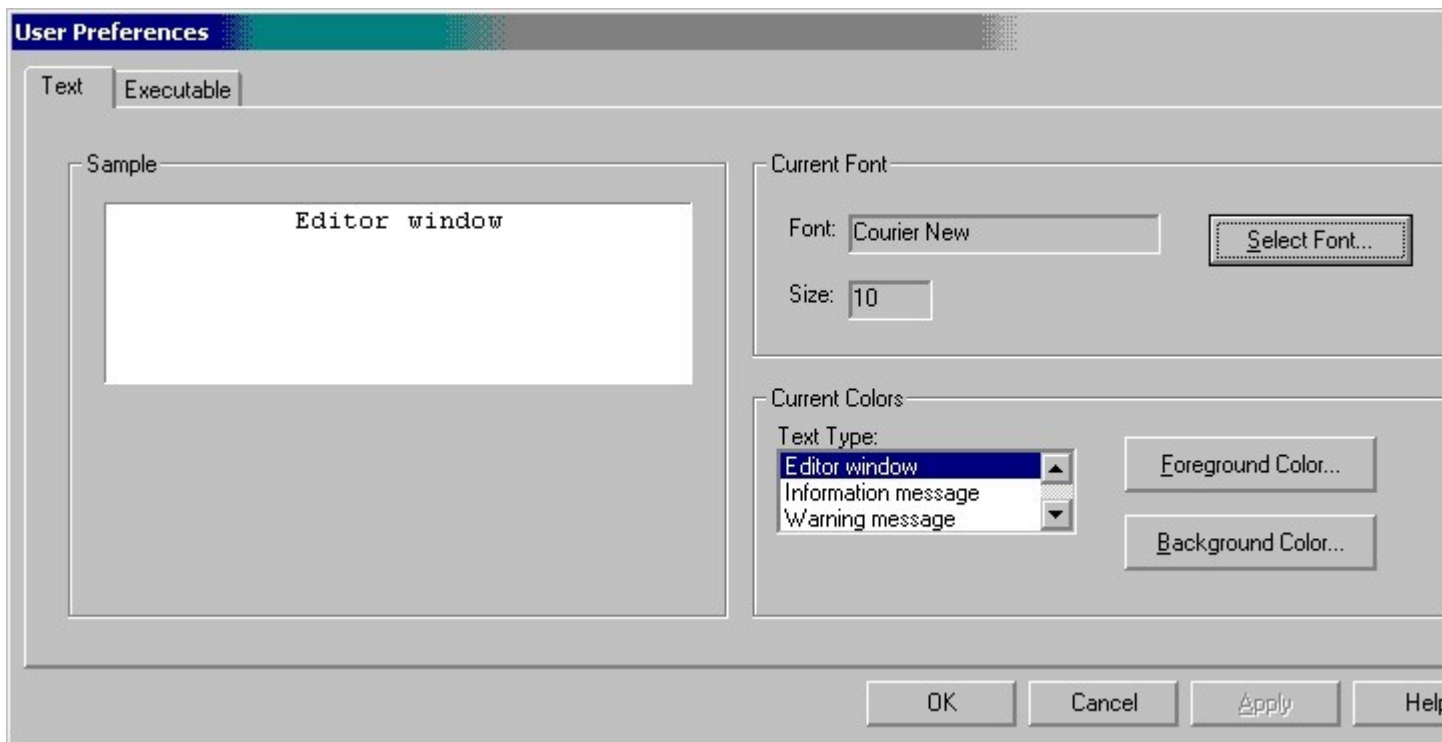
You activate the **User Preferences** dialog box by choosing **Preferences...** from the [Edit menu](#). This dialog box is divided into the following pages:

- **Text** - This page allows you to control the appearance of the following types of text: editor window, information messages, warning messages, and error messages (see [preferences for text](#) for more information).
- **Executable** - This page allows you to specify how Irie Pascal should generate and run executables (see [preferences for executables](#) for more information).

4.3.2.2.1.2 The Text Preferences page

The **Text Preferences** page is the first page of the [User Preferences dialog box](#), and is therefore the first page displayed when you activate this dialog box.

The **Text Preferences** page is used to control the appearance of different types of text in the Irie Pascal Integrated Development Environment (IDE). This page is shown below for easy reference:



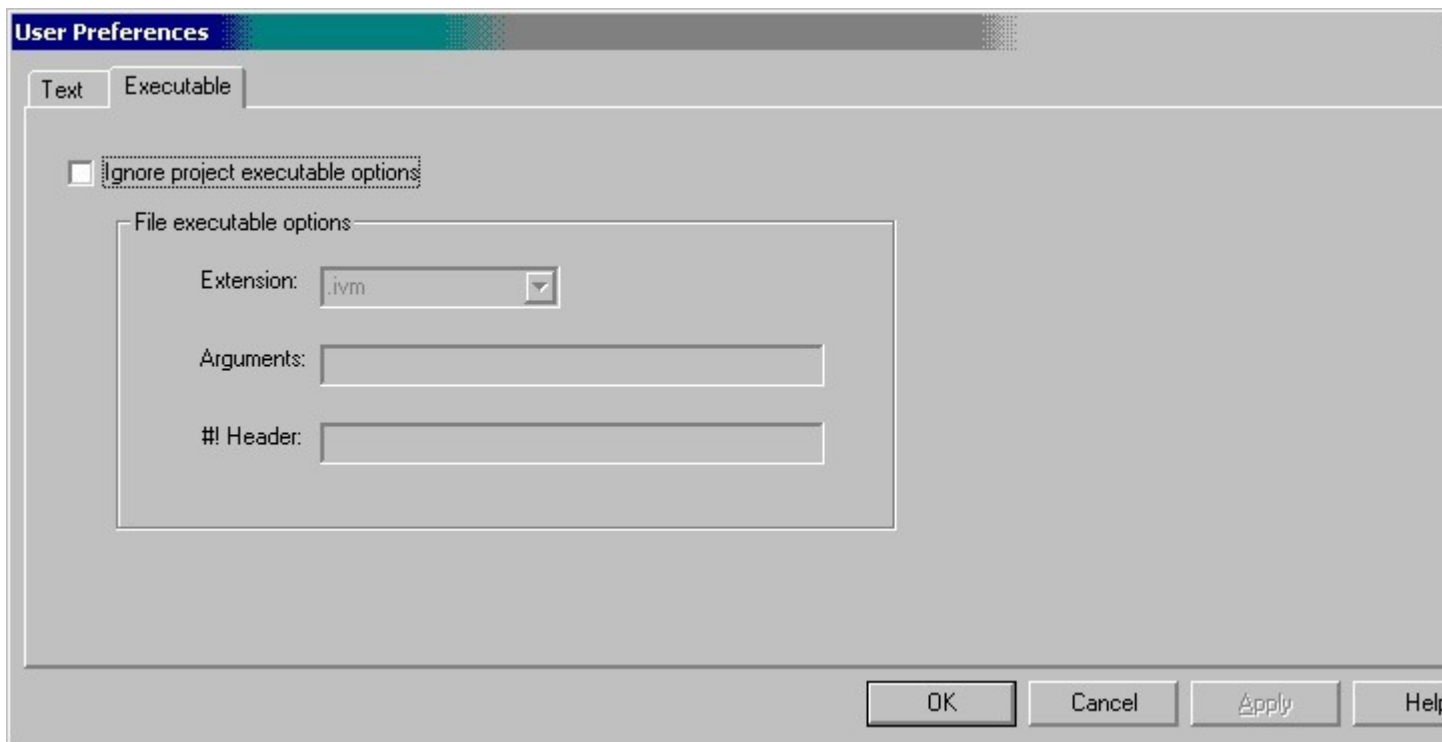
The **Text Preferences** page is divided into three sections:

1. **Sample** - This section displays a sample of what text will look like in the IDE.
2. **Current Font** - This section is where you select the font that will be used for text. **NOTE:** The same font is used for text in the editor windows, information messages, warning messages, and error messages.
3. **Current Colors** - This section is used to both select the type of text you are currently specifying colors for, and to select the foreground and background colors for the text.

4.3.2.2.1.3 The Executable Preferences page

The **Executable Preferences** page is the second page of the [User Preferences dialog box](#), and is therefore **not** the first page displayed when you activate this dialog box. To display the **Executable Preferences** page you must click on the **Executable** tab of the [User Preferences dialog box](#).

The **Executable Preferences** page is used to control what happens when you generate an executable (i.e. compile a program), or when you run an executable (i.e. run the program that was compiled). This page is shown below for easy reference:



The **Executable Preferences** page contains four components:

1. **File mode** - This check box controls whether Irie Pascal is in file mode (in which case the file executable options will be used), or in project mode (**the default**).
2. **Extension:** - This combo box allows you to specify the extension of the executable generated when you compile in file mode. Remember that you can create a true Windows EXE simply by using **.exe** as the extension. **NOTE:** The value in this combo box is ignored when Irie Pascal is in project mode.
3. **Arguments:** - This edit box allows you to specify the arguments (if any) that should be passed to executables when you run them from file mode. **NOTE:** The contents of this edit box are ignored when Irie Pascal is in project mode.
4. **#! Header:** - This edit box allows you to specify the **#! header** (if any) that should be added to executables generated from file mode. **NOTE:** The contents of this edit box are ignored when Irie Pascal is in project mode.

File Mode

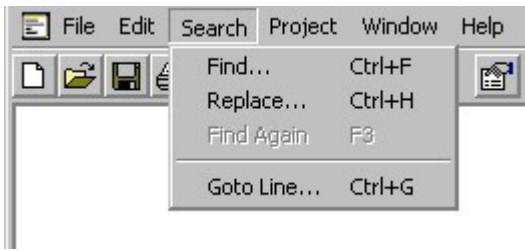
When in file mode, Irie Pascal will always compile the current file being edited (ignoring the file specified by the current project), and the file executable options (i.e. the other components in this page) will be used when compiling or running executables. **NOTE:** This mode is useful when you don't want to start a new project for every program that you create (instead you can just load a file into the editor and compile and run it immediately).

Project Mode

When in project mode, Irie Pascal will always compile the program specified by the current project, and the executable options (see the [Miscellaneous options page](#) of the [Project Options dialog box](#)) specified by the current project will be used when compiling or running executables. **NOTE:** This mode is useful when you intend to start a new project for every program that you create, since Irie Pascal will always remember which program to compile or run, and the executable options to use when doing so, regardless of what file you are currently editing. Also in this mode, each project can have different executable options, and Irie Pascal will always use the executable options for the currently open project.

4.3.3.1 Search menu

The **Search** menu is shown below:



Search Menu Entries

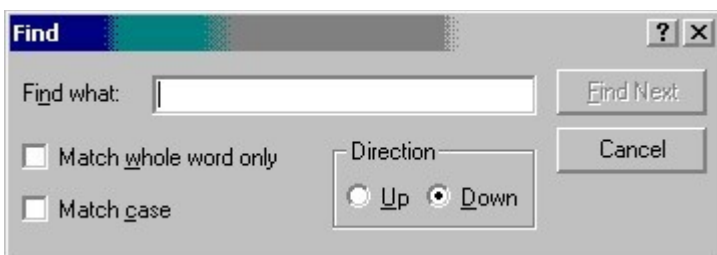
NOTE: The term *current file* is used below to mean the file in the active text editor window (or in other words the file you are currently editing).

- **Find...** - This menu entry displays the [Find](#) dialog box from the Windows Common Dialog Box Library, which is used to search for text in the current file.
- **Replace...** - This menu entry displays the [Replace](#) dialog box from the Windows Common Dialog Box Library, which is used to search for and replace text in the current file.
- **Find Again** - This menu entry is used to repeat a previous search for text in the current file
- **Goto Line...** - This menu entry displays the [Goto Line](#) dialog box which allows you to move to a particular line in the current file.

4.3.3.2.1 The Find Dialog

The **Find** dialog box is used to search for text in the current file, and is a part of the Windows Common Dialog Library. Since the **Find** dialog box is a part of Windows and not a part of the Irie Pascal IDE, its look and functionality is subject to change, and may not correspond exactly to the documentation below.

This **Find** dialog box should look similar to the image shown below:



This **Find** dialog box contains the following elements:

- **Find what text box** - This is where you would type the text you want to search for.
- **Match whole word only check box** - This is where you would specify whether you want the search to succeed only if the text is a whole word (i.e. when the text is not embedded in a word). For example, when this check box is checked, then a search for the text **for** will not find anything in the word **form** because the **for** in **form** is embedded. On the other hand, when this check box is not checked the same search will find the embedded **for** in **form**.
- **Match case check box** - This is where you would specify whether you want the search to be case sensitive. For example, when this check box is checked, then a search for the text **for** will not find **For**, or **FOR** because the case does not match. On the other hand, when this check box is not

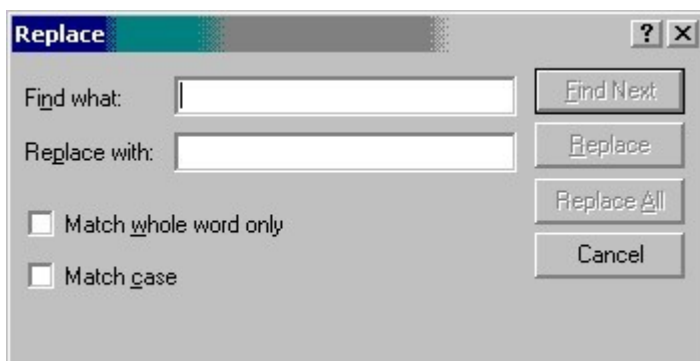
checked then same search will succeed.

- **Up option box** - This is where you indicate that you want the search to go upwards (i.e. go backwards in the file).
- **Down option box** - This is where you indicate that you want the search to go downwards (i.e. go forwards in the file).
- **Find Next button** - Use this button to search for the next instance of the text in the **Find what** text box. **NOTE:** The **Find** dialog box remains on the screen after the search is completed, so that you can easily repeat the search.
- **Cancel button** - Use this button to remove the **Find** dialog box from the screen. If you want to repeat a search after clicking the **Cancel** button, then you can use the **Find Again** menu entry in the [Search](#) menu.

4.3.3.3.1 The Replace Dialog

The **Replace** dialog box is used to search for and replace text in the current file, and is a part of the Windows Common Dialog Library. Since the **Replace** dialog box is a part of Windows and not a part of the Irie Pascal IDE, its look and functionality is subject to change, and may not correspond exactly to the documentation below.

This **Replace** dialog box should look similar to the image shown below:



This **Replace** dialog box contains the following elements:

- **Find what text box** - This is where you would type the text you want to search for.
- **Replace with text box** - This is where you would type the replacement text.
- **Match whole word only check box** - This is where you would specify whether you want the search to succeed only if the text is a whole word (i.e. when the text is not embedded in a word). For example, when this check box is checked, then a search for the text **for** will not find anything in the word **form** because the **for** in **form** is embedded. On the other hand, when this check box is not checked the same search will find the embedded **for** in **form**.
- **Match case check box** - This is where you would specify whether you want the search to be case sensitive. For example, when this check box is checked, then a search for the text **for** will not find **For**, or **FOR** because the case does not match. On the other hand, when this check box is not checked then same search will succeed.
- **Find Next button** - Use this button to search for the next instance of the text in the **Find what** text box. **NOTE:** The **Replace** dialog box remains on the screen after the search is completed, so that you can easily repeat the search.
- **Replace button** - Use this button to search for the next instance of the text in the **Find what** text box, and replace it with the text in the **Replace with** text box.
- **Replace All button** - Use this button to search for all instances of the text in the **Find what** text box, and replace them with the text in the **Replace with** text box.
- **Cancel button** - Use this button to remove the **Replace** dialog box from the screen.

4.3.3.4.1 The Goto Line dialog box

The **Goto Line** dialog box is used to move to a particular line in the current file, and is shown below:

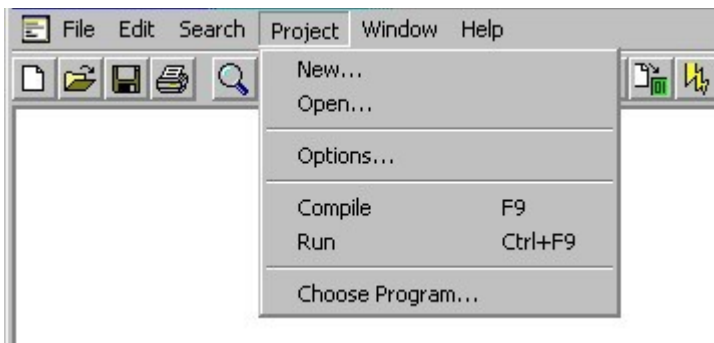


The **Goto Line** dialog box contains the following elements:

- **Line Number Text Box** - This is where you would type the number of the line you want to go to.
- **OK Button** - Use this button to go to the line specified in the **Line Number** text box.
- **Cancel Button** - Use this button to close the **Goto Line** dialog box without going to a new line.

4.3.4.1 Project menu


The **Project** menu, as the name suggests, is mainly used to perform actions on [projects](#), and is shown below:



Project Menu Entries


- **New...** - This menu entry is used to create a new [project](#), and opens a dialog box which allows you to specify the name and location of the new [project](#). See [creating new projects](#) for more information.
- **Open...** - This menu entry is used to open an existing [project](#), and opens a dialog box which allows you to select the [project](#) file to open. See [opening existing projects](#) for more information.
- **Options...** - This menu entry is used to change the options for the current [project](#) (see the [Project Options dialog box](#) for more information).
- **Compile** - When the IDE is in *project mode* this menu entry is used to compile the program, which is a part of the current [project](#). When the IDE is in *file mode* this menu entry is used to compile the current editor file. See [the Executable Preferences page](#) for more information.
- **Run** - When the IDE is in *project mode* this menu entry is used to run the program, which is a part of the current [project](#). When the IDE is in *file mode* this menu entry is used to run the current program. See [the Executable Preferences page](#) for more information.
- **Choose Program...** - This menu entry is used to change the program, which is a part of the current [project](#).
- **Recently Closed Projects** - Irie Pascal projects that were recently closed are added to the end of the **Project** menu, and can be re-opened by choosing them from the menu.

4.3.4.2.1 Creating new projects

You can create a new [project](#) by selecting the **New...** menu entry from the [Project menu](#) or by clicking on the **New Project** button  in the [Toolbar](#). Either method will activate a dialog box which allows you to specify the name and location of the new [project](#). After specifying the name and location, use the **Save** button to close the dialog box and create the [project](#).


When you create a new [project](#), a program source file with the same name as the [project](#) is automatically created (if a program source file with the same name already exists then this file is opened instead of being created). So for example if you create a new [project](#) called **first**, a [project](#) file named **first.ipj**, and a program source file named **first.pas** will be created. The program source file **first.pas** will be empty (unless it existed before the [project](#) was created), and you can start entering your program immediately. In addition, the list of recently closed files, located at the end of the [File menu](#) is cleared.

4.3.4.3.1 Opening existing projects

You can open an existing [project](#) by selecting the **Open...** menu entry from the [Project menu](#), or by clicking on the **Open Project** button  in the [Toolbar](#). Either method will activate a dialog box which allows you to select the [project](#) you want to open. After selecting the [project](#) to open, use the **Open** button to close the dialog box and open the [project](#).

When you open an existing [project](#) any files loaded into the editor are closed, and then the files that were loaded into the editor when the [project](#) was last closed are reloaded. In addition, the list of recently closed files, located at the end of the [File menu](#) is updated to reflect the recently closed files for the [project](#) being opened.

4.3.4.4.1.1 The Project Options dialog box

You activate the **Project Options** dialog box either by choosing **Options...** from the [Project menu](#), or clicking on the project options button  in the [Toolbar](#).

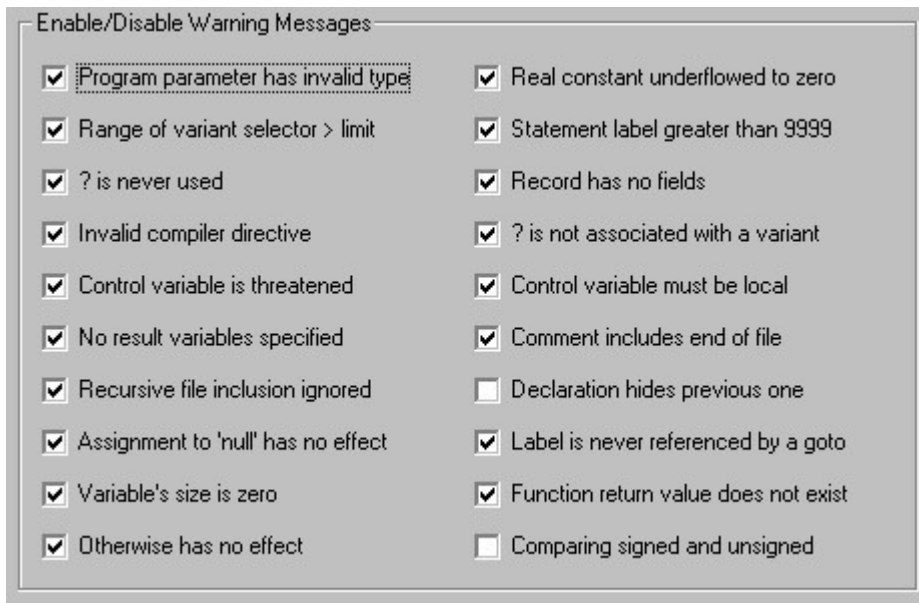
The **Project Options** dialog box contains the following pages:

- **Warnings** - This page allows you to enable and disable compiler warning messages (see [warning options](#)).
- **Extensions** - This page allows you to enable and disable Irie Pascal extensions to Standard Pascal. (see [extension options](#)).
- **Run-Time** - This page allows you to control how your program handles run-time errors. You can control which run-time errors are detected, and how run-time errors once detected are reported (see [run-time options](#) for more).
- **Code Generation** - This page allows you some control over the code generated by the [compiler](#) (see [code generation options](#)).
- **Environment** - This page allows you to control some aspects of the editor's behavior (see [environment options](#)).
- **Miscellaneous** - This page allows you to specify miscellaneous options (i.e. this is a catch-all category for those options that don't belong on any of the other pages). See [miscellaneous options](#).

4.3.4.4.1.2.1 Warning options

The **Warning Options** page is the first page of the [Project Options dialog box](#), and is therefore the first page displayed when you activate this dialog box.

The **Warnings** page allows you to enable and disable compiler warning messages.

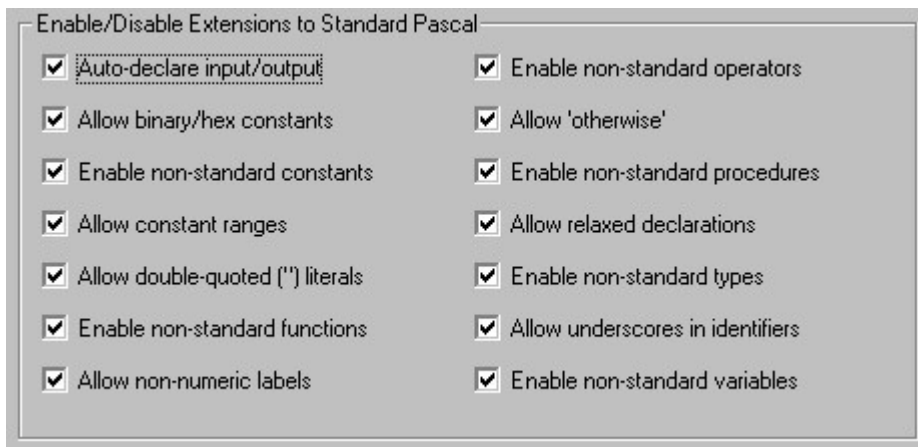


- [Program parameter has invalid type](#) [Real constant underflowed to zero](#)
- [Range of variant selector > limit](#) [Statement label greater than 9999](#)
- [? is never used](#) [Record has no fields](#)
- [Invalid compiler directive](#) [? is not associated with a variant](#)
- [Control variable is threatened](#) [Control variable must be local](#)
- [No result variables specified](#) [Comment includes end of file](#)
- [Recursive file inclusion ignored](#) [Declaration hides previous one](#)
- [Assignment to 'null' has no effect](#) [Label is never referenced by a goto](#)
- [Variable's size is zero](#) [Function return value does not exist](#)
- ['otherwise' has no effect](#) [Comparing signed and unsigned values](#)

4.3.4.4.1.3.1 Extension options

The **Extension Options** page is the second page of the [Project Options dialog box](#), and is therefore **not** the first page displayed when you activate this dialog box. To display the **Extension Options** page you must click on the **Extentions** tab of the [Project Options dialog box](#).

The **Extension Options** page allows you to enable and disable the Irie Pascal extensions to Standard Pascal.

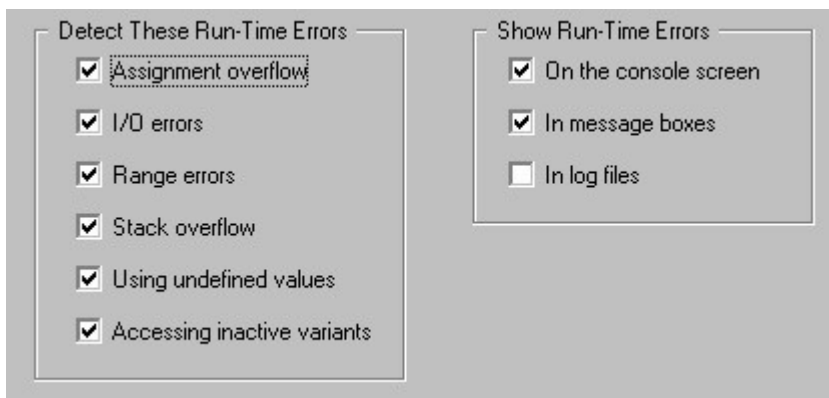


- [Auto declare input and output](#) [Enable non-standard operators](#)
- [Allow binary/hexadecimal constants](#) [Allow 'otherwise'](#)
- [Enable non-standard constants](#) [Enable non-standard procedures](#)
- [Allow constant ranges](#) [Allow relaxed declarations](#)
- [Allow double-quoted literals](#) [Enable non-standard types](#)
- [Enable non-standard functions](#) [Allow underscores \(\) in identifiers](#)
- [Allow non-numeric labels](#) [Enable non-standard variables](#)

4.3.4.4.1.4.1 Run-Time options

The **Run-Time Errors** page is the third page of the [Project Options dialog box](#), and is therefore **not** the first page displayed when you activate this dialog box. To display the **Run-Time Errors** page you must click on the **Run-Time Errors** tab of the [Project Options dialog box](#).

The **Run-Time Errors** page allows you to control how your program handles run-time errors. You can control which run-time errors are detected, and how detected run-time errors are reported.

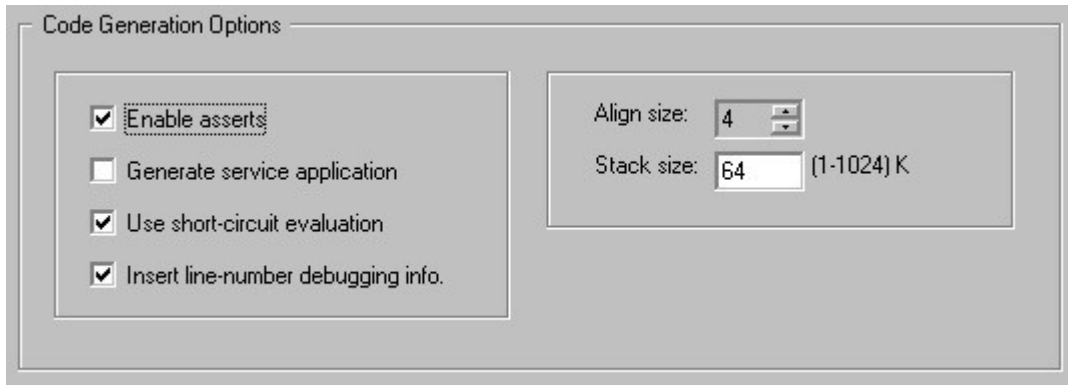


- [Assignment overflow](#)
- [I/O errors](#)
- [Range errors](#)
- [Stack overflow](#)
- [Using undefined values](#)
- [Using in-active variants](#)
- [On the console screen](#)
- [In message boxes](#)
- [In log files](#)

4.3.4.4.1.5.1 Code generation options

The **Code Generation Options** page is the fourth page of the [Project Options dialog box](#), and is therefore **not** the first page displayed when you activate this dialog box. To display the **Code Generation Options** page you must click on the **Code Generation** tab of the [Project Options dialog box](#).

The **Code Generation Options** page contains two sections and is shown below for easy reference:

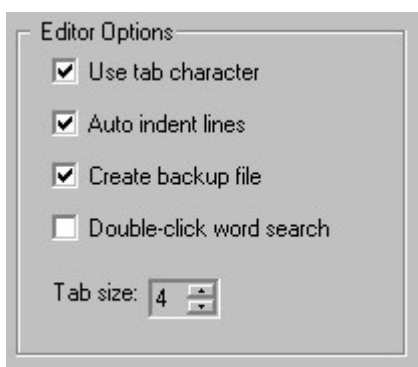


- [Enable asserts](#)
- [Generate Windows NT/2000 service application](#)
- [Use short-circuit evaluation](#)
- [Insert line-number debugging information](#)
- [Align size](#)
- [Stack size](#)

4.3.4.4.1.6.1 Environment options

The **Environment Options** page is the fifth page of the [Project Options dialog box](#), and is therefore **not** the first page displayed when you activate this dialog box. To display the **Environment Options** page you must click on the **Environment** tab of the [Project Options dialog box](#).

The **Environment Options** page contains one section and is shown below for easy reference:



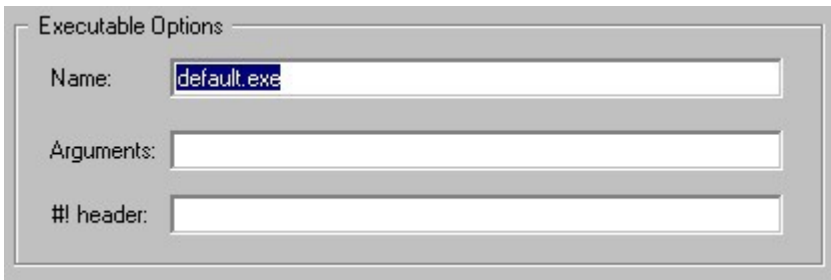
- [Use tab character](#)
- [Auto-indent lines](#)
- [Create backup files](#)
- [Double-click word search](#)
- [Tab size](#)

4.3.4.4.1.7.1 Miscellaneous options

The **Miscellaneous Options** page is the sixth page of the [Project Options dialog box](#), and is therefore **not** the first page displayed when you activate this dialog box. To display the **Miscellaneous Options** page you must click on the **Miscellaneous** tab of the [Project Options dialog box](#).

The **Miscellaneous Options** page contains three sections (**Executable Options**, **Language Options**, and **Message Limits**) and is shown below for easy reference:

Executable Options

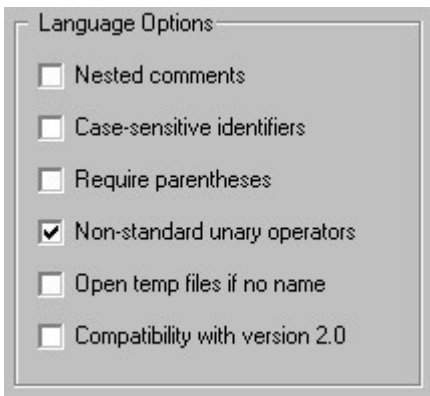


The screenshot shows a dialog box titled "Executable Options". It contains three input fields:

- Name:** A text box containing "default.exe".
- Arguments:** An empty text box.
- #! header:** An empty text box.

- [Name](#)
- [Arguments](#)
- [#! header](#)

Language Options

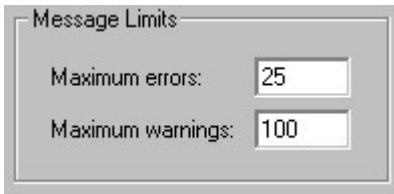


The screenshot shows a dialog box titled "Language Options". It contains six checkboxes:

- Nested comments
- Case-sensitive identifiers
- Require parentheses
- Non-standard unary operators
- Open temp files if no name
- Compatibility with version 2.0


- [Nested comments](#)
- [Case-sensitive identifiers](#)
- [Require parentheses](#)
- [Non-standard unary operators](#)
- [Open temp file if no name](#)
- [Compatibility with version 2.0](#)

Message Limits



- [Maximum errors](#)
- [Maximum warnings](#)

4.3.4.5.1 The Compile menu entry

You can [compile](#) a program by choosing this menu entry (i.e. selecting **Compile** from the [Project menu](#) or by pressing the **F9** key). **NOTE:** You can also [compile](#) a program by clicking on the **Compile** button  from the [IDE Toolbar](#).

See [Compiling programs](#) for more information.

Compiling programs

Your Irie Pascal programs must be [compiled](#) into executables, before they can be run (i.e. executed). **NOTE:** If you try to run your programs before they are [compiled](#), Irie Pascal will automatically compile them for you. While your programs are being [compiled](#) they are checked for errors, and if no errors are found an executable will be generated.


Compiling programs in project mode

If you compile when the IDE is in *project mode* the IDE will always compile the program associated with the current project (regardless of which files you are currently editing). In this mode the executable options from the [Miscellaneous options page](#) will be used when compiling the program. *Project mode* is recommended for most people, and works best when you create a new project for each new program you create (see [creating projects](#) for more information).

Compiling programs in file mode

If you compile when the IDE is in *file mode* the IDE will always compile the current editor file (regardless of the program associated with the current project). In this mode the executable options from [The Executable Preferences page](#) will be used when compiling the program. In *file mode* you don't have to create a new project for each new program you create.

4.3.4.6.1 Running programs

You can run a program by choosing this menu entry (i.e. selecting **Run** from the [Project menu](#) or by pressing the **Ctrl-F9** keys). **NOTE:** You can also run a program by clicking on the **Run** button  from the [IDE Toolbar](#).

If you have made changes to the program since the last time you [compiled](#) it, the program will automatically be [compiled](#) before it is run. The contents of the **Arguments:** text box in either the [Miscellaneous Options page](#) (*project mode* only) or the [The Executable Preferences page](#) (*file mode* only) will be passed to the program when it is run.

4.3.4.7.1 Selecting the program

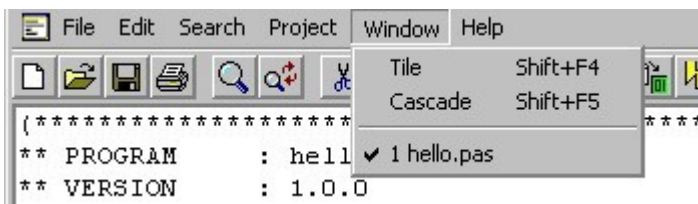
You can change the program associated with the current [project](#) using the **Choose Program...** menu entry from the [Project menu](#).

4.3.4.8.1 Opening recent projects

Recently closed projects are added to the end of the [Project menu](#), and can be re-opened by choosing them from this menu.

4.3.5.1 Window menu

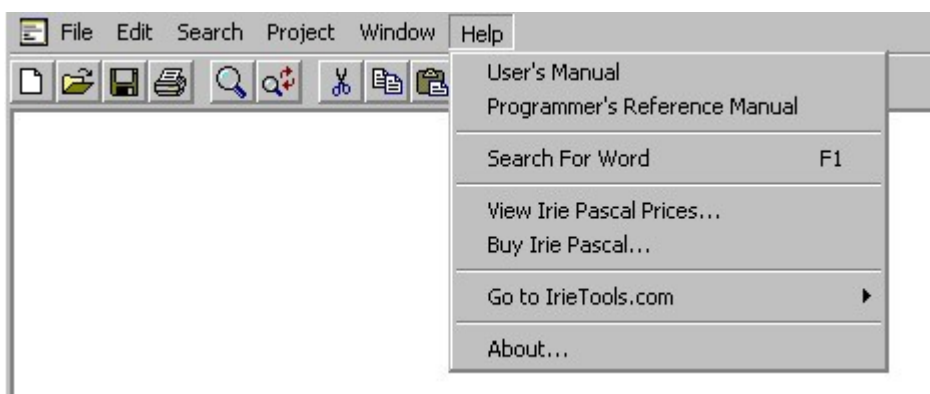
The **Window** menu is used to manage the open text editor windows, and is shown below:



- **Tile** - This menu entry is used to arrange the open text editor windows in a non-overlapping tiled arrangement.
- **Cascade** - This menu entry is used to arrange the open text editor windows in an overlapping arrangement.
- **Open Windows** - Open text editor windows appear at the end of the **Window** menu (the active text editor window is indicated with a tick). You can change the active text editor window by choosing it from the menu.

4.3.6.1 Help menu

The **Help** menu is mainly used to get information about Irie Pascal, and is shown below:

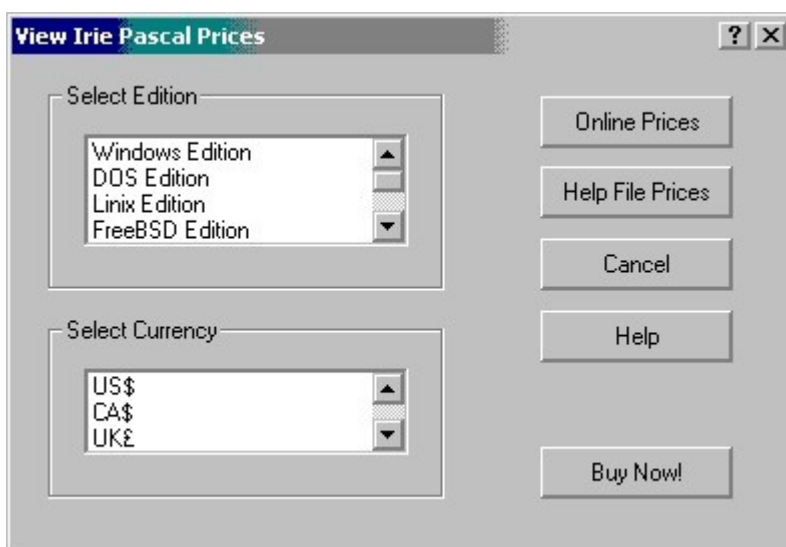


- **User's Manual** - This menu entry displays the Irie Pascal User's Manual, which describes how to use Irie Pascal, and is the manual you are now reading.
- **Programmer's Reference Manual** - This menu entry displays the Irie Pascal Programmer's Reference Manual (in "progref.html"), which describes the Pascal language implemented by Irie Pascal.
- **Search For Word** - This menu entry will search, for the word under the text caret, in the Irie Pascal Programmer's Reference Manual (in "progref.html").
- **View Irie Pascal Prices...** - This menu entry displays the [View Irie Pascal Prices dialog box](#), which you can use to get Irie Pascal prices.
- **Buy Now...** - This menu entry displays the [Buy Now! dialog box](#), which will guide you through the process of buying licenses to use Irie Pascal.
- **Go to IrieTools.com** - This menu entry displays the [Go to IrieTools.com sub-menu](#), which will take you to the Irie Tools website.
- **About...** - This menu entry displays the **About Irie Pascal** dialog box which displays Irie Pascal version and copyright information.

4.3.6.2 View Irie Pascal prices...

You activate the **View Irie Pascal Prices** dialog box by choosing **View Irie Pascal Prices...** from the [Help menu](#).

The **View Irie Pascal Prices** dialog box is used to get prices for Irie Pascal licenses, and is shown below for easy reference:



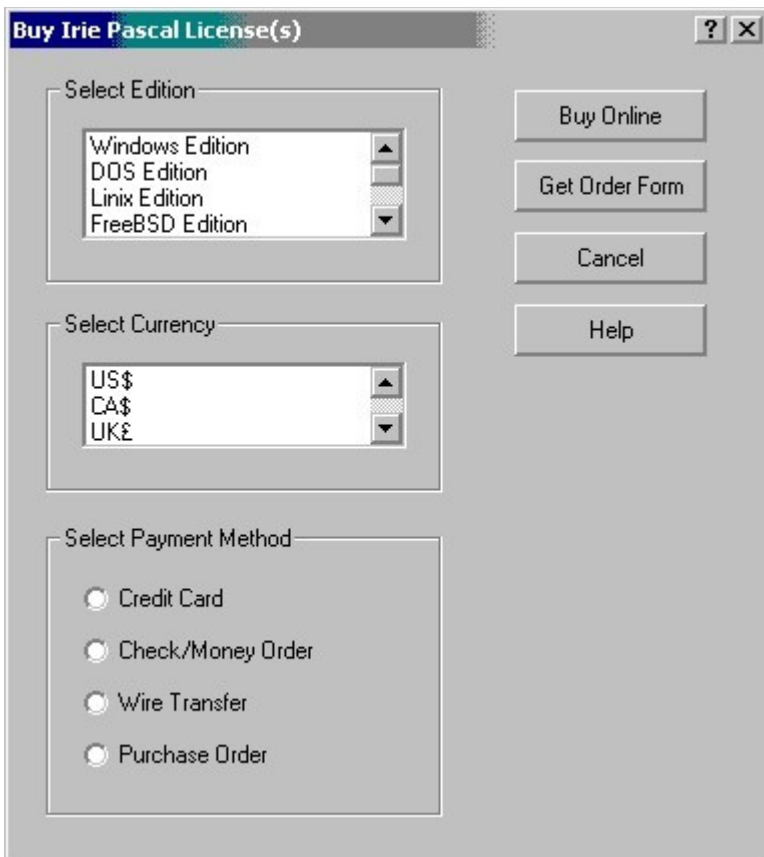
This **View Irie Pascal Prices** dialog box contains the following elements:

- [Select Edition List Box](#)
- [Select Currency List Box](#)
- [Online Prices Button](#)
- [Help File Prices Button](#)
- [Cancel Button](#)
- [Help Button](#)
- [Buy Now! Button](#)

4.3.6.3 Buy Now!...

You activate the **Buy Now!** dialog box by choosing **Buy Now!...** from the [Help menu](#).

The **Buy Now!** dialog box is used to buy licenses to use Irie Pascal, and is shown below:

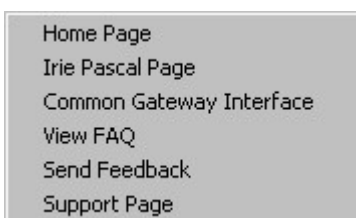


The **Buy Now!** dialog box contains the following elements:

- [Select Edition List Box](#)
- [Select Currency List Box](#)
- [Buy with a credit card](#)
- [Buy with a cheque](#)
- [Buy by wire transfer](#)
- [Buy with a purchase order](#)
- [Buy Online Button](#)
- [Get Order Form Button](#)
- [Cancel Button](#)
- [Help Button](#)

4.3.6.4 Go to IrieTools.com

The **Go to IrieTools.com** sub-menu takes you to the Irie Tools website, and is shown below:



- **Home Page** - This menu entry takes you to the home page of the Irie Tools website.
- **Irie Pascal Page** - This menu entry takes you to the main Irie Pascal page on the Irie Tools website.
- **Common Gateway Interface** - This menu entry takes you to the Common Gateway Interface section of the Irie Tools website.
- **View FAQ** - This menu entry takes you to the Frequently Asked Questions section of the Irie Tools website.
- **Send Feedback** - This menu entry allows you to give your feedback using the Irie Tools website.
- **Support Page** - This menu entry takes you to the Irie Pascal support section of the Irie Tools website.

4.3.7.1 Context menu

When you right-click in a text editor file window, a pop-up menu is displayed which allows you to perform actions on the current file. The pop-up menu is shown below:

















- **Cut** - This menu entry is used to move selected text from the current file into the Windows Clipboard (see [moving a section of text](#)). **NOTE:** If there is no selected text in the current file, then this menu entry is disabled (it will appear greyed out).
- **Copy** - This menu entry is used to copy selected text from the current file into the Windows Clipboard (see [copying a section of text](#)). **NOTE:** If there is no selected text in the current file, then this menu entry is disabled (it will appear greyed out).
- **Paste** - This menu entry is used to copy text from the Windows Clipboard into the current file (see [moving a section of text](#) and [copying a section of text](#)). **NOTE:** If there is no text in the Windows Clipboard then this menu entry is disabled (it will appear greyed out).
- **Clear** - This menu entry is used to delete selected text from the current file (see [selecting a section of text](#) and [deleting a section of text](#)). **NOTE:** If there is no selected text in the current file, then this menu entry is disabled (it will appear greyed out).
- **Find...** - This menu entry displays the [Find](#) dialog box from the Windows Common Dialog Box Library, which is used to search for text in the current file.
- **Replace...** - This menu entry displays the [Replace](#) dialog box from the Windows Common Dialog Box Library, which is used to search for and replace text in the current file.
- **Find Again** - This menu entry is used to repeat a previous search for text in the current file.
- **Goto Line...** - This menu entry displays the [Goto Line](#) dialog box which allows you to move to a particular line in the current file.
- **Print...** - This menu entry displays the Standard Print dialog box from the Windows Common Dialog Library, which gives you a chance to set printer options and print some or all of the current file.
- **Save** - This menu entry saves the current file, and depending on the setting of the **Create backup file** Environment Project Option a backup copy of the original contents of the file may be created. If the file being saved is new, and has never been saved before, then this menu entry works like the **Save As** menu entry described next.

- **Close** - This menu entry closes the current file. **NOTE:** You will be prompted to save any unsaved changes that have been made to the file.

4.4.1 IDE Toolbar

The Irie Pascal Integrated Development Environment (IDE) toolbar provides quick shortcuts for the most frequently used IDE menu entries. Each button in the toolbar is associated with a frequently used IDE menu entry, and you can access the menu entry by clicking on the button. The toolbar contains the following buttons:

-  This button is a shortcut for the **New** menu entry in the [File menu](#).
-  This button is a shortcut for the **Open...** menu entry in the [File menu](#).
-  This button is a shortcut for the **Save** menu entry in the [File menu](#).
-  This button is a shortcut for the **Print...** menu entry in the [File menu](#).
-  This button is a shortcut for the **Find...** menu entry in the [Search menu](#).
-  This button is a shortcut for the **Replace...** menu entry in the [Search menu](#).
-  This button is a shortcut for the **Cut** menu entry in the [Edit menu](#).
-  This button is a shortcut for the **Copy** menu entry in the [Edit menu](#).
-  This button is a shortcut for the **Paste** menu entry in the [Edit menu](#).
-  This button is a shortcut for the **Options...** menu entry in the [Project menu](#).
-  This button is a shortcut for the **New...** menu entry in the [Project menu](#).
-  This button is a shortcut for the **Open...** menu entry in the [Project menu](#).
-  This button is a shortcut for the **Compile** menu entry in the [Project menu](#).
-  This button is a shortcut for the **Run** menu entry in the [Project menu](#).

4.5.1 Bookmarks

The Integrated Development Environment (IDE) allows you to use keyboard short-cuts to bookmark up to ten positions in each editor window, and quickly return to the bookmarked positions. See [keyboard short-cuts](#) for more information.

4.6.1 Using mouse wheels

The Integrated Development Environment (IDE) allows you to use mouse wheels to do the following:

- Rolling the mouse wheel, without holding down the **Ctrl** or **Shift** keys, will scroll the active editor or message window.
- Rolling the mouse wheel, while holding down the **Ctrl** key, will zoom the size of the text in the editor and message windows.
- Rolling the mouse wheel, while holding down the **Shift** key, will change the active window.

NOTE: Mouse wheels are not supported in Windows 95.

4.7.1 Using the keyboard

The Integrated Development Environment (IDE) allows you to use the keyboard to perform a number of actions. See the list below for more information:

- [Editing text](#).
- [Manipulate menus](#).
- [Manipulate dialog boxes](#).
- [Keyboard short-cuts](#).

4.7.2 Editing text

The main action you can perform with the IDE is editing text (i.e. editing pascal source code), and this action is primarily performed using the keyboard. Using the IDE to edit text is very similiar to using other Windows programs to edit text, so this process will not be described in detail.

One thing you may not be aware of, although many other Windows programs support it, is that you can select text using the keyboard (see [Selecting a section of text](#) for more information). Also worth mentioning is that the IDE allows you to indent and unindent selected text using the **Tab** key, as long as the selection starts at the beginning of the first line (see [Indenting a section of text](#) and [Unindenting a section of text](#) for more information).

4.7.3 Using menus

Windows provides a standard menu keyboard interface that allows you to manipulate menus using the keyboard instead of the mouse. This interface allows you to use the keyboard to enter and exit menu bar mode, activate menus and menu items, and to choose menu items. **NOTE:** Choosing a menu item with the keyboard has the same effect as choosing the menu item with the mouse.

Menu Bar Mode

The IDE is in menu bar mode whenever one of the top-level application menus is selected but the menu has not been activated. The **Alt** and **F10** keys, allow you to toggle in and out of menu bar mode. When menu bar mode is on, the following keys behave as described below:

- The **LEFT** and **RIGHT** arrow keys change the selected top-level menu.
- The **UP** and **DOWN** arrow keys, and the **Enter** key activate the currently selected menu.
- The **Esc** exits menu bar mode.
- [Menu access keys](#) activate the menu in which they are located.

When the IDE is in menu bar mode, the title of the top-level application menu selected is usually highlighted to give a visual indication that it is selected. However this is not the case if the title of the selected menu is an icon rather than text. **NOTE:** This means that if you enter menu bar mode when the editor windows are maximized, no menu is initially highlighted. This is because, if you enter menu bar mode with the editor windows maximized, the intial menu selected is the system menu of the active editor window, and since the title of the system menu is an icon it is not highlighted. In this case, even though no menu is highlighed, you can still see that you are in menu bar mode because the caret in the editor window will either stop blinking or disappear entirely.

Using the Keyboard with Active Menus

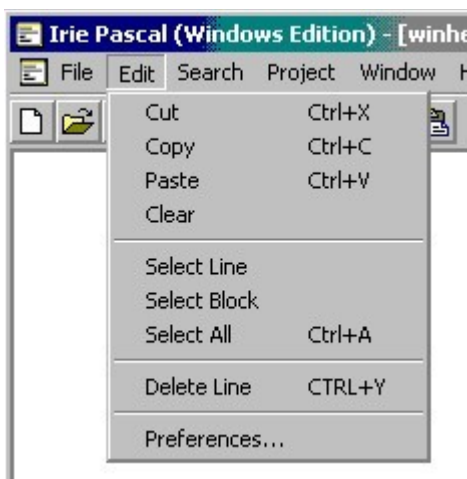
When a menu is active, the following keys behave as described below:

- The **LEFT** and **RIGHT** arrow keys activate the menu to the left and right of the currently active menu.
- The **UP** and **DOWN** arrow keys activate the menu item above and below the currently active menu item.
- The **Enter** key chooses the currently active menu.
- If the currently active menu is a sub-menu, the **Esc** key de-activates it and activates it's parent menu. If the currently active menu is **not** a sub-menu, the **Esc** key de-activates it and enters menu bar mode.
- [Menu item access keys](#) choose the menu items in which they are located.

Menu Short-Cut Keys

Menu short-cut keys are keys or combination of keys that are associated with some menu items. Those menu items that have menu short-cut keys associated with them can be chosen by pressing the key or combination of keys that make up the menu short-cut.

For example in the **Edit** menu (which is shown below for easy reference):



The **Delete Line** menu item is associated with the **Ctrl+Y** menu short-cut key, and this menu item can be selected by holding down the **Ctrl** key and pressing the **Y** key.

NOTE: Unlike menu item access keys, you can use menu short-cut keys even when the menu containing their menu items are not active. So for example, it is possible to delete the current line, in the active editor window, using the **Ctrl+Y** menu short-cut key, without having to activate the **Edit** menu.

4.7.4 Using dialog boxes

Windows provides a standard keyboard interface for dialog boxes that allows you to manipulate dialog boxes using the keyboard instead of the mouse. This interface is not specific to the dialog boxes used by the IDE, so it will not be described in detail, however some of the features of this interface are listed below:

- The **Tab** key can be used to move the input focus among the controls in a dialog box.
- The **Space** bar will toggle, between checked and unchecked, the check box with the input focus.
- The **Space** bar will select the option box with the input focus.

- The **F1** will usually display help about the control with the input focus.
- The **Enter** key will activate the default command button (usually named "OK") for the dialog box.
- The **Esc** key will activate the default cancel button (usually named "Cancel") for the dialog box.

4.7.5 Keyboard short-cuts

The Integrated Development Environment (IDE) allows you to use keyboard short-cuts to perform a number of actions. See below for more information:

Bookmarks

Short-cut	Action
Alt+0	Sets bookmark 0
Alt+1	Sets bookmark 1
Alt+2	Sets bookmark 2
Alt+3	Sets bookmark 3
Alt+4	Sets bookmark 4
Alt+5	Sets bookmark 5
Alt+6	Sets bookmark 6
Alt+7	Sets bookmark 7
Alt+8	Sets bookmark 8
Alt+9	Sets bookmark 9
Shift+Ctrl+0	Sets bookmark 0
Shift+Ctrl+1	Sets bookmark 1
Shift+Ctrl+2	Sets bookmark 2

Short-cut**Action**

Shift+Ctrl+3

Sets bookmark 3

Shift+Ctrl+4

Sets bookmark 4

Shift+Ctrl+5

Sets bookmark 5

Shift+Ctrl+6

Sets bookmark 6

Shift+Ctrl+7

Sets bookmark 7

Shift+Ctrl+8

Sets bookmark 8

Shift+Ctrl+9

Sets bookmark 9

Ctrl+0

Goto bookmark 0

Ctrl+1

Goto bookmark 1

Ctrl+2

Goto bookmark 2

Ctrl+3

Goto bookmark 3

Ctrl+4

Goto bookmark 4

Ctrl+5

Goto bookmark 5

Ctrl+6

Goto bookmark 6

Ctrl+7

Goto bookmark 7

Ctrl+8

Goto bookmark 8

Short-cut**Action**

Ctrl+9

Goto bookmark 9

Goto**Short-cut****Action**

Ctrl+Left

Move one word left

Ctrl+Right

Move one word right

Ctrl+Home

Move to the top of the file

Ctrl+End

Move to the end of the file

Ctrl+PageDown

Move to bottom of window retaining column

Ctrl+PageUp

Move to the top of the window retaining column

Ctrl+Up

Scroll window up one line

Ctrl+Down

Scroll window down one line

Ctrl+G

Goto line

Find**Short-cut****Action**

F3

Find again

F5

Find

F6

Replace

Short-cut**Action**

Ctrl+F

Find

Ctrl+H

Replace

Clipboard**Short-cut****Action**

Ctrl+C

Copy text to clipboard

Ctrl+Insert

Copy text to clipboard

Ctrl+V

Paste text from clipboard

Shift+Insert

Paste text from clipboard

Ctrl+X

Cut text to clipboard

File**Short-cut****Action**

Ctrl+N

New

Ctrl+O

Open

Ctrl+W

Close

Ctrl+S

Save

Ctrl+P

Print

Other

Short-cut	Action
F1	Search help for a word
Ctrl+F1	Search help for a word
F9	Compile
Ctrl+F9	Run
F10	Toggle menu bar mode
Shift+F4	Tile editor window
Shift+F5	Cascade editor windows
Ctrl+A	Select all
Ctrl+Y	Delete line
Ctrl+Backspace	Delete on word left
Shift+Ctrl+Y	Delete to the end of the line
Ctrl+F4	Exit IDE
Ctrl+F6	Next window

5.1.1 Using the command-line compiler

The Irie Pascal command-line Pascal compiler translates Pascal source programs into Irie Virtual Machine executables (**.IVM** executables), or (**.EXE** executables).

Command-line Compiler (ipc) Syntax

The syntax for the command-line compiler is as follow:

- **ipc** - The compiler displays a brief help screen showing the proper syntax.
- **ipc ?** - The compiler displays a more detailed help screen listing all the options you can use.
- **ipc [options] filename** - The compiler attempts to compile the file specified by **filename**, using the options specified. NOTE: **filename** can include path information, and **[x]** indicates that **x** is optional.

For example to compile the sample program **hello.pas** enter

```
ipc hello
```

or

```
ipc hello.pas
```

(assuming of course that **hello.pas** is in the current directory).

The compiler will generate a file called **hello.ivm** which contains an IVM executable. You can run it by entering

```
ivm hello
```

or

```
ivm hello.ivm
```

NOTE: If you do not specify a path for the executable (as in the example above) the interpreter will first look in the current directory and if not found the interpreter will search the path.

5.1.2.1 Compiler options overview

Compiler options are instructions to the compiler to somehow modify its behavior (usually they are instructions to enable or disable a particular compiler feature). Compiler options are entered on the command-line when you invoke the command-line compiler. **NOTE:** Compile options are case-sensitive so for example **i** and **I** are different compiler options.

There are two kinds of compiler options

1. Flag options
2. Value options

Flag Options

Flag options are used to enable or disable a compiler feature.

To enable the feature use

```
-option or -option+
```

where **option** is the particular compiler option.

To disable the feature use

`-option-`

For example the [nc](#) option is used to enable/disable the processing of nested comments. So `-nc` or `-nc+` is used to enable the processing of nested comments and `-nc-` is used to disable the processing of nested comments.

Value options are used to specify the value of some quantity.

To specify a value option use

`-optionVALUE`

where **option** is the particular compiler option and **VALUE** is the value being specified.

For example the [mw](#) option is used to specify the maximum number of warnings that the compiler should process. So `-mw2` is used to set the maximum number of warnings to 2.

More than one option can be specified so for example, if you want to compile the program **bad.pas** using brief messages and nested comments then you can enter

```
ipc -b -nc bad.pas
```

Options can be combined so you can also enter

```
ipc -bnc bad
```

or

```
ipc -ncb bad
```

To turn off nested comment processing and brief message then you can enter

```
ipc -nc- -b- bad
```

or

```
ipc -ncb- bad
```

Options List

The command-line compiler options are listed below:

- [-aN Align on N bytes](#)
- [-ao Trap assignment overflow errors](#)
- [-A Enable Asserts](#)
- [-b Use brief messages](#)
- [-C Identifiers are case-sensitive](#)
- [-cm20 Compatibility mode](#)
- [-ead Auto-declare input & output](#)
- [-ebh Allow binary & hex integers](#)
- [-eco Enable non-standard constants](#)
- [-ecr Allow constant ranges](#)
- [-edq Allow double-quoted literals](#)
- [-efn Enable non-standard functions](#)

- [-enn](#) Allow non-numeric labels
- [-eop](#) Enable non-standard operators
- [-eow](#) Allow otherwise
- [-epr](#) Enable non-standard procedures
- [-erd](#) Allow relaxed declarations
- [-ety](#) Enable non-standard types
- [-eui](#) Allow underscores in identifiers
- [-eva](#) Enable non-standard variables
- [-E](#) Enable all extensions
- [-gs](#) Generate a WinNT/2000 service
- [-hTEXT](#) Add #!TEXT header
- [-i](#) Trap I/O errors
- [-I](#) Control informatory messages
- [-ln](#) Insert line-number debug info
- [-mb](#) Generate Borland compatible messages
- [-mc](#) Display message context
- [-meN](#) Set maximum errors
- [-mm](#) Generate Microsoft compatible messages
- [-mwN](#) Set maximum warnings
- [-nc](#) Allow nested comments
- [-nu](#) Non-standard unary operators
- [-oNAME](#) Set output filename
- [-p](#) Require parentheses
- [-r](#) Trap range errors
- [-rtlf](#) Send run-time errors to log file
- [-rtmb](#) Send run-time errors to message box
- [-rtsc](#) Send run-time errors to screen
- [-s](#) Strict var strings
- [-sc](#) Use short-circuit evaluation
- [-so](#) Maximum stack overflow checking
- [-Snn](#) Set stack size in K
- [-u](#) Trap use of undefined values
- [-v](#) Trap use of inactive variants
- [-W](#) Control warning messages

5.1.2.2.1 -aN Align on N bytes

This command-line compiler option specifies the maximum alignment used by the compiler.

Syntax: **-aN** (Sets maximum alignment to **N**)

Default: Maximum alignment is 4.

Notes: See also [specify align size](#)

5.1.2.2.2 -ao* Trap assignment overflow errors

This command-line compiler option enables/disables assignment overflow checking.

Syntax: **-ao[+|-]**

Default: Enabled

Notes: See also [Assignment overflow checking](#).

5.1.2.2.3 -A* Enable Asserts

This command-line compiler option enables/disables asserts.

Syntax: **-A[+|-]**

Default: Enabled

Notes: See also [enable asserts](#).

5.1.2.2.4 -b Use brief messages

This command-line compiler option enables/disables the brief format for messages.

Syntax: **-b[+|-]**

Default: Disabled

Notes:

The verbose format for Fatal error and Error messages (which is used by default) is

Error #nn: "name" (Line l, Col c): text

where

- **nn** is a number identifying the message
- **name** is the file where the problem as detected
- **l** is the line where the problem was detected
- **c** is the column where the problem was detected
- **text** is the text of the message

The verbose format for warning messages is similar except that **Warning** is used instead of **Error**.

The brief format for Fatal error and Error messages is:

Enn: "name" l:text

where

- **nn** is a number identifying the message
- **name** is the file where the problem as detected
- **l** is the line number of the line where the problem was detected
- **text** is the text of the message

The brief format for warning messages are similar except that **W** is used instead of **E**.

5.1.2.2.5 -C Identifiers are case-sensitive

This command-line compiler option enables/disables case-sensitive identifiers.

Syntax: **-C[+|-]**

Default: Disabled

Notes: See also [make identifiers case-sensitive](#).

5.1.2.2.6 -cm20 Compatibility mode

This command-line compiler option controls whether the compiler is in version 2.0 compatibility mode.

Syntax: **-cm20** [+|-]

Default: Disabled

Notes: See also [Compatibility mode \(with version 2.0\)](#).

5.1.2.2.7 -ead* Auto-declare input & output

This command-line compiler option enables/disables auto-declaration of input and output.

Syntax: **-ead** [+|-]

Default: Enabled

Notes: See also [auto-declare input/output](#).

5.1.2.2.8 -ebh* Allow binary & hex integers

This command-line compiler option enables/disables binary and hexadecimal integer constants.

Syntax: **-ebh** [+|-]

Default: Enabled

Notes: See also [allow binary/hexadecimal constants](#).

5.1.2.2.9 -eco* Enable non-standard constants

This command-line compiler option enables/disables non-standard constants.

Syntax: **-eco** [+|-]

Default: Enabled

Notes: See also [enable non-standard constants](#).

5.1.2.2.10 -ecr* Allow constant ranges

This command-line compiler option enables/disables constant ranges.

Syntax: **-ecr** [+|-]

Default: Enabled

Notes: See also [allow constant ranges](#).

5.1.2.2.11 -edq* Allow double-quoted literals

This command-line compiler option enables/disables double quoted literals.

Syntax: **-edq** [+|-]

Default: Enabled

Notes: See also [allow double-quoted \("\) literals](#).

5.1.2.2.12 -efn* Enable non-standard functions

This command-line compiler option enables/disables non-standard functions.

Syntax: **-efn**[+|-]

Default: Enabled

Notes: See also [enable non-standard functions](#).

5.1.2.2.13 -enn* Allow non-numeric labels

This command-line compiler option enables/disables support for non-numeric statement labels.

Syntax: **-enn**[+|-]

Default: Enabled

Notes: See also [allow non-numeric labels](#).

5.1.2.2.14 -eop* Enable non-standard operators

This command-line compiler option enables/disables support for non-standard operators.

Syntax: **-eop**[+|-]

Default: Enabled

Notes: See also [enable non-standard operators](#).

5.1.2.2.15 -eow* Allow otherwise

This command-line compiler option enables/disables support for the keyword **otherwise**.

Syntax: **-eow**[+|-]

Default: Enabled

Notes: See also [allow otherwise](#).

5.1.2.2.16 -epr* Enable non-standard procedures

This command-line compiler option enables/disables non-standard procedures.

Syntax: **-epr**[+|-]

Default: Enabled

Notes: See also [enable non-standard procedures](#).

5.1.2.2.17 -erd* Allow relaxed declarations

This command-line compiler option enables/disables relaxed declarations.

Syntax: **-erd**[+|-]

Default: Enabled

Notes: See also [allow relaxed declarations](#).

5.1.2.2.18 -ety* Enable non-standard types

This command-line compiler option enables/disables non-standard types.

Syntax: **-ety**[+|-]

Default: Enabled

Notes: See also [enable non-standard types](#).

5.1.2.2.19 -eui* Allow underscores in identifiers

This command-line compiler option enables/disables underscores (`_`) in identifiers.

Syntax: **-eui**[+|-]

Default: Enabled

Notes: See also [allow underscores \(`_` \) in identifiers](#).

5.1.2.2.20 -eva* Enable non-standard variables

This command-line compiler option enables/disables non-standard variables.

Syntax: **-eva**[+|-]

Default: Enabled

Notes: See also [enable non-standard variables](#).

5.1.2.2.21 -E* Enable all extensions

This command-line compiler option enables/disables all Irie Pascal extensions to Standard Pascal.

Syntax: **-E**[+|-]

Default: Enabled

5.1.2.2.22 -gs Generate a WinNT/2000 service

This command-line compiler option makes the compiler generate a Windows NT/2000 service application.

Syntax: **-gs**[+|-]

Default: Disabled

Notes: See also [generate Windows NT/2000 service application](#).

5.1.2.2.23 -hTEXT Add `#!TEXT` header

This command-line compiler option specifies a `#!` header to be inserted in front of the generated executable.

Syntax: **-hTEXT** (Sets the `#!` header to TEXT)

Default: By default no `#!` header is used.

NOTES: See also [#! header](#).

5.1.2.2.24 -i* Trap I/O errors

This command-line compiler option enables/disables I/O trapping.

Syntax: **-i**[+|-]

Default: Enabled

Notes: When I/O trapping is enabled the compiler generates code which checks each I/O operation and issues a run-time error if an I/O error is detected.

5.1.2.2.25 -I Control informatory messages

This command-line compiler option enables/disables information messages.

Syntax: **-I**[nn][+|-]

Default: All information message enabled.

- Use **-I** or **-I+** to enable all information messages.
- Use **-I-** to disable all information messages.
- Use **-Inn** or **-Inn+** to enable the information message with message number **nn**.
- Use **-Inn-** to disable the information message with message number **nn**.

5.1.2.2.26 -ln* Insert line-number debug info

This command-line compiler option enables/disables line number debugging information.

Syntax: **-ln**[+|-]

Default: Enabled

Notes: See also [insert line-number debugging information](#).

5.1.2.2.27 -mb Generate Borland compatible messages

This command-line compiler option enables/disables Borland compatible messages.

Syntax: **-mb**[+|-]

Default: Disabled

Notes: When this compiler option is enabled the compiler displays messages in the format used by Borland compilers. This option can be used to make it easier to integrate the Irie Pascal command-line compiler with third-party editors and IDE's which already know how to process Borland compiler messages. This option automatically suppresses the [mc compiler option](#).

5.1.2.2.28 -mc* Display message context

This command-line compiler option enables/disables message context information.

Syntax: **-mc[+|-]**

Default: Enabled

Notes: When this compiler option is enabled the compiler shows the position in the source file referred to by error and warning messages. NOTE: Some error and warning messages do not refer to a particular position in the source file and therefore context information is not displayed for those messages.

5.1.2.2.29 -meN Set maximum errors

This command-line compiler option specifies the maximum number of error messages that the compiler should allow.

Syntax: **-meN**

Default: N = 25

Notes: Suppose you want the compiler to stop after 5 error messages then use:

-me5

5.1.2.2.30 -mm Generate Microsoft compatible messages

This command-line compiler option enables/disables Microsoft compatible messages.

Syntax: **-mm[+|-]**

Default: Disabled

Notes: When this compiler option is enabled the compiler displays messages in the format used by Microsoft compilers. This option can be used to make it easier to integrate the Irie Pascal command-line compiler with third-party editors and IDE's which already know how to process Microsoft compiler messages. This option automatically suppresses the [mc compiler option](#).

5.1.2.2.31 -mwN Set maximum warnings

This command-line compiler option specifies the maximum number of warning messages that the compiler should allow.

Syntax: **-mwN**

Default: N = 100

Notes: Suppose you want the compiler to stop after 5 warning messages then use

-mw5

5.1.2.2.32 -nc Allow nested comments

This command-line compiler option enables/disables support for nested comments.

Syntax: **-nc**[+|-]

Default: Disabled

Notes: See also [allow nested comments](#).

5.1.2.2.33 -nu Non-standard unary operators

This command-line compiler option enables/disables non-standard use of unary plus and minus operators.

Syntax: **-nu**[+|-]

Default: Enabled

Notes: See also [non-standard unary operators](#).

5.1.2.2.34 -oNAME Set output filename

This command-line compiler option specifies the name of the executable generated by the compiler.

Syntax: **-oNAME**

Default: The name of the executable is the name of the Pascal source file with the extension changed to **.ivm**.

5.1.2.2.35 -p Require parentheses

This command-line compiler option enables/disables mandatory parentheses mode.

Syntax: **-p**[+|-]

Default: Disabled

Notes: See also [require parentheses](#).

5.1.2.2.36 -r* Trap range errors

This command-line compiler option enables/disables range checking.

Syntax: **-r**[+|-]

Default: Enabled

Notes: See also [values out of range](#).

5.1.2.2.37 -rtlf Send run-time errors to log file

This command-line compiler option sends run-time errors to a log file.

Syntax: **-rtlf**[+|-]

Default: Disabled

Notes: See also [run-time errors in log file](#).

5.1.2.2.38 -rtmb* Send run-time errors to message box

This command-line compiler option sends run-time errors to a message box. NOTE: Only executables running under the Windows platform can display run-time errors in message boxes, on all other platforms this option has no effect.

Syntax: **-rtmb**[+|-]

Default: Enabled

Notes: See also [run-time errors in message box](#).

5.1.2.2.39 -rtsc* Send run-time errors to screen

This command-line compiler option sends run-time errors to the console screen.

Syntax: **-rtsc**[+|-]

Default: Enabled

Notes: See also [run-time errors to console screen](#).

5.1.2.2.40 -s* Strict var strings

This command-line compiler option enables/disables strict checking of var string parameters.

Syntax: **-s**[+|-]

Default: Enabled

Notes:

When strict checking is enabled, it is an error to pass a string variable by reference if the length of the variable's string type is not equal to the length of the formal parameter's string type.

When strict checking is disabled you can pass a string variable by reference even if the length of the string variable is not equal to the length of the formal parameter.

For example if you compile the following program and strict checking is enabled:

```
program p(output) ;
type
string16 = string[16];
string8 = string[8];
var
x : string16;
procedure print(var s : string8);
begin
writeln(s)
end;
begin
x := 'Hello';
print(x)          (* Error only if strict checking is enabled *)
end.
```

then the compiler will report an error with the call **print(x)** since the length of **x**'s string type is 16 and the length of the formal parameter's string type is 8 (i.e. they are not equal).

If strict checking is disabled then no errors are reported.

5.1.2.2.41 -sc* Use short-circuit evaluation

This command-line compiler option enables/disables short-circuit evaluation for the boolean operators **and** and **or**.

Syntax: **-sc**[+|-]

Default: Enabled

Notes: See also [use short-circuit evaluation](#).

5.1.2.2.42 -so Maximum stack overflow checking

This command-line compiler option enables/disables detailed stack overflow checking.

Syntax: **-so**[+|-]

Default: Disabled

Notes: The interpreter automatically checks for stack overflow at the beginning and end of each procedure/function call and when large values are placed on the stack. These checks should suffice for most purposes, however you can enable this option if you want the interpreter to check for stack overflow before every statement is executed.

5.1.2.2.43 -Snn Set stack size in K

This command-line compiler option specifies the number of kilobytes to allocate for your program's stack.

Syntax: **-Snn**

Default: nn = 64

Notes: See also [specify stack size](#).

5.1.2.2.44 -u* Trap use of undefined values

This command-line compiler option enables/disables checking for undefined values.

Syntax: **-u[+|-]**

Default: Enabled

Notes: When this option is enabled the compiler generates code which checks each time your program gets a value from a variable to make sure that the value is not undefined. If your program does get an undefined value from a variable then the code generated by the compiler will issue a run-time error message and terminate your program.

So for example if you compile and run the following program

```
program bad(output) ;
var
r : real;
begin
writeln(r) (* The value of "r" is undefined *)
end.
```

The program will terminate with a run-time error message because of the attempt to print the value of **r** which is undefined. Unfortunately not all variable accesses can be checked, checks are only made for accesses to variables of the following types:

- enumerated types (including boolean)
- subranges of enumerated types
- subranges of integer that do not include -1
- file
- list
- object
- pointer
- real
- set (array representation)

Accesses to variables of types **char**, **integer**, **record** and **set** (Bit set representation) are not checked.

Checking for undefined values is performed as follows:

- All memory is initialized by setting all bits to 1.
- When accessing a variable a check is performed to see if all bits are set and if they are this variable is undefined. This doesn't work for **char**, **integer** or **set** (bit set representation) variables since a value with all bits set to 1 is valid for these variables. Record variables are not checked because they may contain fields which can not be checked.

5.1.2.2.45 -v* Trap use of inactive variants

This command-line compiler option enables/disables variant checking.

Syntax: **-v[+|-]**

Default: Enabled

Notes: When variant checking is enabled the compiler generates code to check each time a field of a variant, is accessed, to make sure that the variant is active.

5.1.2.2.46 -W Control warning messages

This command-line compiler option enables/disables warning messages.

Syntax: **-W[nn][+|-]**

Default: All warning messages enabled.

- Use **-W** or **-W+** to enable all warning messages.
- Use **-W-** to disable all warning messages.
- Use **-Wnn** or **-Wnn+** to enable warning message number **nn**.
- Use **-Wnn-** to disable warning message number **nn**.

5.2.1 Using the interpreter

The Irie Virtual Machine Interpreter is used to run Irie Virtual Machine executables.

The interpreter should be run from a command-line prompt.

Once you are at a command-line prompt use the following syntax:

Syntax: **ivm [filename] [arguments]**

where **filename** specifies the Irie Virtual Machine Executable to run.

and **[arguments]** are program arguments passed to the executable.

NOTE: **[x]** indicates that **x** is optional.

For example if you compile the sample program **hello.pas**, the compiler will generate a file called **hello.ivm** which contains an Irie Virtual Machine Executable.

You can run it by entering **ivm hello** or **ivm hello.ivm**.

NOTE: If you do not specify a path for the executable (as in the example above) the interpreter will first look in the current directory and if not found the interpreter will search the path.

5.3.1 Using the header utility

NOTE: The command-line and IDE compilers can put `#!` headers into the generated executables so using the Irie Header Utility is not necessary anymore unless you can't or don't want to recompile the executable.

The Irie Header Utility is used to create IVM executables with `#!` headers. `#!` headers can be used by UNIX or UNIX-like operating systems (such as Solaris, Linux or FreeBSD) to locate the interpreters that should be used to execute scripts. When UNIX or a UNIX-like operating system attempts to execute a file and the first two characters in the file are `#!`, the operating system will assume that the file is a script that needs to be interpreted, and it will expect the location of the interpreter to follow the `#!`.

The syntax is:

```
ivm header input-executable output-executable [location]
```

where

- **input-executable** - is the ivm executable you want to use to create the new executable from.
- **output-executable** - is the ivm executable you want to create
- **location** - if specified is the location of the interpreter to put in the `#!` header.

So for example if the interpreter was installed in `/usr/local/bin`, you would store `#!/usr/local/bin/ivm` in front of the executable.

So for example if you had an IVM executable named `hello.ivm` you could enter

```
ivm header hello.ivm hello /usr/local/bin/ivm
```

which creates a new executable called `hello` which has

```
#!/usr/local/bin/ivm
```

in front (`#!` is automatically inserted). In order to be able to execute `hello` you also need to set the executable permission bit (using

```
chmod a+x hello
```

for example).

6.1 Overview of extensions to Standard Pascal

Standard Pascal (i.e. ISO/IEC 7185) allows implementations, such as Irie Pascal, to add new features, called extensions. However an extension to Standard Pascal can not invalidate any Standard Pascal program that would otherwise be valid, except possibly by adding new keyword(s) that would invalidate programs that used those keyword(s) as normal identifiers. **NOTE:** ISO/IEC 7185 is the standard for the Pascal programming language published by the [International Organization for Standardization](#).

Irie Pascal supports a number of extensions to Standard Pascal. Some of these extensions were added for compatibility with Turbo Pascal or Extended Pascal, while others were added for other reasons and are likely to be specific to Irie Pascal.

The Irie Pascal extensions to Standard Pascal are listed below:

- [Auto declare input and output](#)
- [Allow binary/hexadecimal constants](#)
- [Enable non-standard constants](#)
- [Allow constant ranges](#)
- [Allow double-quoted literals](#)
- [Enable non-standard functions](#)
- [Allow non-numeric labels](#)
- [Enable non-standard operators](#)
- [Allow 'otherwise'](#)
- [Enable non-standard procedures](#)
- [Allow relaxed declaratons](#)
- [Enable non-standard types](#)
- [Allow underscores \(_ \) in identifiers](#)
- [Enable non-standard variables](#)

"6.2 Auto declare input and output"

When this extension is enabled (it is by **default**), the Irie Pascal compiler will automatically declare the built-in identifiers **input** and **output**, when they are not declared in your program.

Standard Pascal (i.e. ISO/IEC 7185) specifies that whenever the required identifiers **input** and **output** are referenced in a program, that they must be declared (i.e. appear as program parameters). However because some Pascal compilers do not enforce this requirement many Pascal programs do not meet this specification. **NOTE:** ISO/IEC 7185 is the standard for the Pascal programming language published by the [Internation Organization for Standardization](#).

"6.3 Allow binary/hexadecimal constants"

When this extension is enabled (the **default**), the Irie Pascal compiler will recognize binary and hexadecimal constants.

Binary Constants

Binary constants begin with **%**, and are followed by one or more binary (**0** or **1**) digits.

The following are examples of valid binary constants

```
%0    %1    %01110101010101010111101
```

The following are not valid binary constants

```
%2    %    %151    %g
```

Hexadecimal Constants

Hexadecimal constants begin with **\$**, and are followed by one or more hexadecimal (**one of 0123456789ABCDEF**) digits.

The following are examples of valid hexadecimal constants

```
$9    $A123    $ffff
```

The following is not a valid hexadecimal constant

```
$abcd
```

since **g** is not a hexadecimal digit.

"6.4 Enable non-standard constants"

When this extension is enabled (it is by **default**), the Irie Pascal compiler will recognize a number of non-standard constants, which are not part of Standard Pascal (i.e. ISO/IEC 7185). **NOTE:** ISO/IEC 7185 is the standard for the Pascal programming language published by the [International Organization for Standardization](#).

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for a complete list of non-standard constants.

"6.5 Allow constant ranges"

When this extension is enabled (the **default**), the Irie Pascal compiler will recognize constant ranges in **case statements** and **variant records**.

Constant Ranges

You can use constant ranges to specify a number of consecutive case constants. To use a constant range you specify the first constant, and the last constant, separated by **..** as follows:

```
first..last
```

So for example you could use the constant range

```
1..5
```

to specify the following constants

```
1, 2, 3, 4, 5
```

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more information.

"6.6 Allow double-quoted literals"

When this extension is enabled (the **default**), the Irie Pascal compiler will allow you to use double-quotes to delimitate character and string literals.

For example when this extension is enabled you could use

```
"Hello world"
```

instead of

```
'Hello world'
```

Double-quoted literals can be especially useful if you want to create literals with single quotes in them, since you don't have to use two single quotes to represent one single quote.

For example you could use

```
"Don't go away"
```

which is equivalent to

```
'Don''t do away'
```

but much more readable.

"6.7 Enable non-standard functions"

When this extension is enabled (it is by **default**), the Irie Pascal compiler will recognize a number of non-standard functions, which are not part of Standard Pascal (i.e. ISO/IEC 7185). **NOTE:** ISO/IEC 7185 is the standard for the Pascal programming language published by the [International Organization for Standardization](#).

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for a complete list of non-standard functions.

"6.8 Allow non-numeric labels"

When this extension is enabled (it is by **default**), the Irie Pascal compiler will allow non-numeric statement labels. In Standard Pascal (i.e. ISO/IEC 7185), statement labels must be numeric and between 0 and 9999. When this extension is enabled, you can declare and use labels containing letters and underscores. **NOTE:** ISO/IEC 7185 is the standard for the Pascal programming language published by the [International Organization for Standardization](#).

For example in the following program, **loop** is used as a statement label.

```
program name(output);
label loop;
var
i : integer;
begin
i := 1;
loop:
writeln(i);
i := i + 1;
if i <= 20 then goto loop;
end.
```

"6.9 Enable non-standard operators"

When this extension is enabled (it is by **default**), the Irie Pascal compiler will recognize a number of non-standard operators, which are not a part of Standard Pascal (i.e. ISO/IEC 7185), and it will allow some of the boolean operators to be used to perform bitwise operations. **NOTE:** ISO/IEC 7185 is the standard for the Pascal programming language published by the [International Organization for Standardization](#).

The non-standard operators are:

```
and_then, or_else, xor, shl, shr
```

The boolean operators that can perform bitwise operations are:

```
not, and, or
```

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more information.

"6.10 Allow 'otherwise'"

When this extension is enabled (the **default**), the Irie Pascal compiler will allow you to use the keyword **otherwise** in case statements and variant records to specify "all values that haven't been used yet". You can also use the keyword **else** instead of **otherwise** (this feature was added in order to improve compatibility with Turbo Pascal).

"6.11 Enable non-standard procedures"

When this extension is enabled (it is by **default**), the Irie Pascal compiler will recognize a number of non-standard procedures, which are not part of Standard Pascal (i.e. ISO/IEC 7185). **NOTE:** ISO/IEC 7185 is the standard for the Pascal programming language published by the [International Organization for Standardization](#).

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for a complete list of non-standard procedures.

"6.12 Allow relaxed declarations"

Standard Pascal (i.e. ISO/IEC 7185) requires that all declarations/definitions of the same kind must be made together in a single group and that the groups must appear in a specific order. **NOTE:** ISO/IEC 7185 is the standard for the Pascal programming language published by the [International Organization for Standardization](#).

The order of declarations/definitions required by Standard Pascal is:

```
Label declaration group
Constant definition group
Type definition group
Variable declaration group
sub-block declaration group
```

When this extension is enabled (it is by **default**), there can be more than one of each kind of group and groups can appear in any order except that, for declarations local to a function or procedure, the sub-block declaration group must be last.

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more information about declarations/definitions.

"6.13 Enable non-standard types"

When this extension is enabled (it is by **default**), the Irie Pascal compiler will recognize a number of non-standard types, which are not part of Standard Pascal (i.e. ISO/IEC 7185). **NOTE:** ISO/IEC 7185 is the standard for the Pascal programming language published by the [International Organization for Standardization](#).

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for more a complete list of non-standard types.

"6.14 Allow underscores (_) in identifiers"

When this extension is enabled (the **default**), the Irie Pascal compiler will allow underscores in identifiers.

So for example the following would be valid identifiers:

```
_name  
last_name  
_all_names_
```

"6.15 Enable non-standard variables"

When this extension is enabled (it is by **default**), the Irie Pascal compiler will recognize a number of non-standard variables, which are not part of Standard Pascal (i.e. ISO/IEC 7185). **NOTE:** ISO/IEC 7185 is the standard for the Pascal programming language published by the [International Organization for Standardization](#).

See the Irie Pascal Programmer's Reference Manual (in "progref.html") for a complete list of non-standard variables.

7.1 What is Irie Pascal?

Irie Pascal is a Pascal compiler and interpreter. The compiler translates Pascal programs into Irie Virtual Machine (IVM) executables, which are then executed by the interpreter. The IVM is an abstract computer platform that is implemented in software (by the interpreter), and runs executables on many different computer platforms. The IVM has been implemented on the following computer platforms (Win95/98/NT/2000/XP, Linux, FreeBSD, Solaris/x86, and Solaris/Sparc) so far. IVM executables developed on any platform, run on all the other platforms.

Irie Pascal's ability to generate executables which run on multiple platforms make it ideally suited for creating internet applications. The Common Gateway Interface (CGI) is a simple but powerful protocol for creating server side internet applications. Irie Pascal assists the creation of CGI scripts with built-in support for decoding and parsing URL encoded strings, as well as support for databases, and sending email. Irie Pascal also supports the UNIX #! trick that allows the location of the interpreter to be embedded inside the script making it easier to execute the script from a URL, since the URL need only refer to the script and not the interpreter.

Irie Pascal is highly compatible with Standard Pascal (i.e. ISO/IEC 7185). This high level of compatibility means that Irie Pascal shares Standard Pascal's strengths as a first language for beginners. These strengths include readable syntax, and extensive compile-time and run-time checking. **NOTE:** ISO/IEC 7185 is the standard for the Pascal programming language published by the [International Organization for Standardization](#).

Irie Pascal supports many extensions to Standard Pascal, particularly in the areas of string, file/folder processing, and database programming, which make it useful for creating scripts and utilities. Irie Pascal's support for automatic run-time checking make it useful for creating **quick and dirty** programs (i.e. programs that are expected to be run only a few times or by only a few people and may not be worth spending a lot of time on).

7.2 Compliance

Irie Pascal complies with the requirements of level 0 of ISO/IEC 7185, with the following exceptions: (see the Irie Pascal Reference Manual, Appendix B - Deviations from ISO/IEC 7185). **NOTE:** ISO/IEC 7185 is the standard for the Pascal programming language published by the [International Organization for Standardization](#).

NOTE: Irie Pascal compliance with ISO/IEC 7185 has not been formally certified by an external body.

7.3 License and distribution rights

IRIE PASCAL EVALUATION VERSION (WINDOWS EDITION)

LICENSE STATEMENT AND DISCLAIMER OF WARRANTY

IMPORTANT - READ CAREFULLY This license statement and disclaimer of warranty constitutes a legal agreement ("License Agreement") between you (either as an individual or a single entity) and Stuart King ("Author") for the software product ("Software") identified above, including any software, media, and accompanying on-line or printed documentation.

BY INSTALLING, COPYING, OR OTHERWISE USING THE SOFTWARE, YOU AGREE TO BE BOUND BY ALL OF THE TERMS AND CONDITIONS OF THE LICENSE AGREEMENT.

Upon your acceptance of the terms and conditions of the License Agreement, the Author grants you the right to use the Software in the manner provided below.

This Software is owned by the Author and is protected by copyright law and international copyright treaty. Therefore, you must treat this Software like any other copyrighted material (e.g. a book), except that you may either make one copy of the Software solely for backup or archival purposes or transfer the Software to a single hard disk provided you keep the original solely for backup or archival purposes.

The Author grants to you as an individual, a personal, nonexclusive, non-transferable license to install and use the Software for evaluation purposes only. In particular, you may not distribute or cause to be distributed the Software or any programs you develop using the Software. You may install a copy of the Software on a computer and freely move the Software from one computer to another, provided that you are the only individual using the Software. If you are an entity, the Author grants you the right to designate one individual within your organization ("Named User") to have the right to use the Software in the manner provided above.

The Software might include source code, redistributable files, and/or other files provided by a third party vendor (Third Party Software). Since use of Third Party Software might be subject to license restrictions imposed by the third party vendor, you should refer to the on-line documentation (if any) provided with Third Party Software for any license restrictions imposed by the third party vendor. In any event, any license restrictions imposed by a third party vendor are in addition to, not in lieu of, the terms and conditions of the License Agreement.

DISCLAIMER OF WARRANTY

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE AUTHOR DISCLAIMS ALL WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NON-INFRINGEMENT, WITH REGARD TO THE SOFTWARE, AND THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES. THIS

WARRANTY DISCLAIMER AFFECTS YOUR LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM JURISDICTION TO JURISDICTION. SOME JURISDICTIONS DO NOT ALLOW EXCLUSIONS OF IMPLIED WARRANTIES OR LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU.

HIGH RISK ACTIVITIES The Software is not fault-tolerant and is not designed, manufactured or intended for use or resale as on-line control equipment in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of the Software could lead directly to death, personal injury, or severe physical or environmental damage ("High Risk Activities"). The Author specifically disclaims any express or implied warranty of fitness for High Risk Activities.

LIMITATION OF LIABILITY

IN NO EVENT SHALL THE AUTHOR BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY CONSEQUENTIAL, SPECIAL, INCIDENTAL OR INDIRECT DAMAGES OF ANY KIND ARISING OUT OF OR RELATING TO THE SOFTWARE OR THE USE THEREOF (INCLUDING BUT NOT LIMITED TO LOST PROFITS OR OTHER ECONOMIC LOSS), EVEN IF THE AUTHOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO EVENT WILL THE AUTHOR'S LIABILITY FOR ANY CLAIM, WHETHER IN CONTRACT, TORT, OR ANY OTHER THEORY OF LIABILITY, EXCEED THE FEE PAID BY YOU FOR THE SOFTWARE THAT IS THE SUBJECT OF SUCH CLAIM. IF THE RELEVANT SOFTWARE WAS PROVIDED TO YOU AT NO CHARGE YOU AGREE THAT THE AUTHOR SHALL NOT BE LIABLE TO YOU FOR ANY DAMAGES. YOU AGREE THAT YOU ARE SOLELY RESPONSIBLE FOR ADEQUATE PROTECTION AND BACKUP OF THE DATA AND EQUIPMENT USED IN CONNECTION WITH THE SOFTWARE OR SUBSCRIPTION SERVICES, AND FURTHER AGREE THAT THE AUTHOR WILL NOT BE LIABLE FOR ANY DAMAGES THAT YOU MAY SUFFER IN CONNECTION WITH DOWNLOADING, INSTALLING, OR USING THE SOFTWARE. IF YOU ELECT NOT TO PURCHASE A LICENSE TO THE SOFTWARE, YOU FURTHER ACKNOWLEDGE THAT YOU ARE PROVIDED A REASONABLE TIME FRAME TO EVALUATE THE SOFTWARE AND AT THE END OF SUCH EVALUATION PERIOD YOU MAY ONLY ACCESS AND USE THE SOFTWARE IF YOU PURCHASE A LICENSE TO THE SOFTWARE. YOU AGREE THAT THE AUTHOR WILL NOT BE LIABLE FOR ANY DAMAGE THAT YOU MAY SUFFER IN CONNECTION WITH THE TERMINATION OF SUCH EVALUATION PERIOD AND YOUR INABILITY TO ACCESS AND USE THE SOFTWARE. THIS LIMITATION SHALL APPLY TO CLAIMS OF PERSONAL INJURY TO THE EXTENT PERMITTED BY LAW. THE LIMITATIONS IN THIS SECTION ARE SEPARATE AND INDEPENDENT OF ANY OTHER REMEDY LIMITATIONS IN THIS AGREEMENT AND SHALL NOT FAIL IF SUCH OTHER LIMITATION OR REMEDY FAILS. THE FEES AND OTHER PROVISIONS IN THIS AGREEMENT REFLECT THE ALLOCATION OF RISKS BETWEEN THE PARTIES. THIS SECTION IS AN ESSENTIAL ELEMENT OF THE BASIS OF THE BARGAIN BETWEEN THE PARTIES. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE EXCLUSIONS OR LIMITATIONS MAY NOT APPLY TO YOU.

TERMINATION

This Agreement shall terminate automatically if you fail to comply with the terms of this Agreement. This Agreement shall terminate if you do not purchase a license to the Software within a period of 30 calendar days from the date the Software is first installed by you. No notice shall be required from the Author to effect such termination. You may also terminate this Agreement at any time by uninstalling and destroying all copies of the Software.

ENTIRE AGREEMENT

You agree that this is the entire agreement between you and the Author, and that it supersedes any prior agreement, whether written or oral, and all other communications between the Author and you relating to the subject matter of this Agreement. This Agreement may be amended, modified or supplemented only by a writing that is signed by the authorized representatives of both parties.

RESERVATION OF RIGHTS

All rights not expressly granted in this Agreement are reserved by the Author. ©1998-2005

7.4 Disclaimer-Agreement

See the [license statement](#) for the warranty disclaimer.

7.5.1 Getting help from the IDE

The Irie Pascal User's Manual

The Irie Pascal User's Manual, which is the manual you are currently reading, provides help on using Irie Pascal. This manual can be accessed by choosing **User's Manual** from the [Help menu](#).

The Programmer's Reference Manual

The Irie Pascal Programmer's Reference Manual (in "progref.html") provides help on the Irie Pascal programming language. This manual can be accessed by choosing **Programmer's Reference Manual** from the [Help menu](#).

Word search

Word search is a convenient way to search the Irie Pascal Programmer's Reference Manual (in "progref.html") for help on words that have special meaning to Irie Pascal. These words include keywords, built-in constants, types, variables, functions, and procedures. When a word search is performed, the Irie Pascal Programmer's Reference Manual (in "progref.html") is searched for the current word in the active edit window (i.e. the word under the caret). A word search can be performed in a number of ways. One way to perform a word search is to choose **Search For Word** from the [Help menu](#). Another way to perform a word search is to press the F1 key. Yet another way to perform a word search is to double-click on a word in an editor window. If the **double-click word search** project option is selected (which is one of the Environment project options) then a word search will be performed.

Dialog Box Help

The dialog boxes used by Irie Pascal provide help in two ways.

- With a help button which displays help on the entire dialog box when clicked.
- With a question mark in the title bar of the dialog box. If you click on the question mark the mouse pointer will change to a question mark with a pointer. If you then click on a control in the dialog box, a short description of what the control does (called a tool tip) will be displayed.

Toolbar Help

When you leave the mouse pointer over a button in the toolbar for more than a second or so, a brief description (called a tool tip) of what the button does will appear. When you move the mouse pointer away the tool tip will disappear.

Status Line Help

When you leave the mouse pointer over a menu item or over a button in the tool bar the status line will display a short description of what the menu item or button does. When you move the mouse pointer away the description is removed from the status line.

7.5.2 Getting help from the website

The Irie Tools Website

The Irie Tools website is an online source of up-to-date information about Irie Pascal. You can access this website at www.iriertools.com.

You can also access this website by choosing **Go to IrieTools.com** from the [Help menu](#) and then choosing one of the menu items (**Home Page**, **Irie Pascal Page**, **Common Gateway Interface**, **View FAQ**, **Send Feedback**, or **Support Page**).

7.5.3 Contacting customer support

Email Help

Irie Pascal help is also available at support@iriertools.com. Queries to this email address are normally answered within 24 hours.

7.6.1 Checking prices

Irie Pascal is available in a number of different editions, and at the time this help file was created the available editions were: **Windows**, **Linux**, **FreeBSD**, **Solaris/x86**, **Solaris/Sparc**, and **Universal**. The price for Irie Pascal licenses varies depending on the edition, the number of licenses, and the currency.

After you decide on the Irie Pascal edition you are interested in and the currency you would like to use for payment, finding the price is easy. To find the price select **View Irie Pascal Prices...** from the [Help menu](#). You will be presented with a dialog box and asked to select the edition you are interested in and the currency. After making those selections you just click on the **Prices (Web)** button or the **Prices (Help)** button. If you click on the **Prices (Web)** button you will be taken to iriertools.com to view the prices (this assumes you have an internet connection). If you click the **Prices (Help)** button you will shown prices from this manual. The prices from this manual are guaranteed until December 31, 2005, after that date you should either use the **Prices (Web)** button or send an email to sales@iriertools.com.

To find out more about the Irie Pascal Prices dialog box click [here](#).

7.6.2.1 Irie Pascal Windows Edition Prices (in US\$)

No. Users	Unit Price (US\$)	Total Price (US\$)
1	79.99	79.99
5	69.99	349.95
10	59.99	599.90
20	49.99	999.80
40	39.99	1,599.60
100	29.99	2,999.00

All prices are quoted in US\$ and are guaranteed until December 31, 2005.

7.6.2.2 Irie Pascal Windows Edition Prices (in CA\$)

No. Users	Unit Price (CA\$)	Total Price (CA\$)
1	99.99	99.99
5	87.49	437.45
10	74.99	749.90

No. Users	Unit Price (CA\$)	Total Price (CA\$)
20	62.49	1,249.80
40	49.99	1,999.60
100	37.49	3,749.00

All prices are quoted in CA\$ and are guaranteed until December 31, 2005.

7.6.2.3 Irie Pascal Windows Edition Prices (in UK)

No. Users	Unit Price (UK£)	Total Price (UK£)
1	39.99	39.99
5	34.99	174.95
10	29.99	299.90
20	24.99	499.80
40	19.99	799.60
100	14.99	1,499.00

All prices are quoted in UK£ and are guaranteed until December 31, 2005.

7.6.2.4 Irie Pascal Windows Edition Prices (in Euros)

No. Users	Unit Price (Euros)	Total Price (Euros)
1	59.99	59.99
5	52.99	264.95
10	45.99	459.90

No. Users	Unit Price (Euros)	Total Price (Euros)
20	38.99	779.80
40	31.99	1,279.60
100	24.99	2,499.00

All prices are quoted in Euros and are guaranteed until December 31, 2005.

7.6.3.1 Irie Pascal Linux Edition Prices (in US\$)

No. Users	Unit Price (US\$)	Total Price (US\$)
1	79.99	79.99
5	69.99	349.95
10	59.99	599.90
20	49.99	999.80
40	39.99	1,599.60
100	29.99	2,999.00

All prices are quoted in US\$ and are guaranteed until December 31, 2005.

7.6.3.2 Irie Pascal Linux Edition Prices (in CA\$)

No. Users	Unit Price (CA\$)	Total Price (CA\$)
1	99.99	99.99
5	87.49	437.45

No. Users	Unit Price (CA\$)	Total Price (CA\$)
10	74.99	749.90
20	62.49	1,249.80
40	49.99	1,999.60
100	37.49	3,749.00

All prices are quoted in CA\$ and are guaranteed until December 31, 2005.

7.6.3.3 Irie Pascal Linux Edition Prices (in UK)

No. Users	Price (UK£)	Total Price (UK£)
1	39.99	39.99
5	34.99	174.95
10	29.99	299.90
20	24.99	499.80
40	19.99	799.60
100	14.99	1,499.00

All prices are quoted in UK£ and are guaranteed until December 31, 2005.

7.6.3.4 Irie Pascal Linux Edition Prices (in Euros)

No. Users	Unit Price (Euros)	Total Price (Euros)
1	59.99	59.99

No. Users	Unit Price (Euros)	Total Price (Euros)
5	52.99	264.95
10	45.99	459.90
20	38.99	779.80
40	31.99	1,279.60
100	24.99	2,499.00

All prices are quoted in Euros and are guaranteed until December 31, 2005.

7.6.4.1 Irie Pascal FreeBSD Edition Prices (in US\$)

No. Users	Unit Price (US\$)	Total Price (US\$)
1	79.99	79.99
5	69.99	349.95
10	59.99	599.90
20	49.99	999.80
40	39.99	1,599.60
100	29.99	2,999.00

All prices are quoted in US\$ and are guaranteed until December 31, 2005.

7.6.4.2 Irie Pascal FreeBSD Edition Prices (in CA\$)

No. Users	Unit Price (CA\$)	Total Price (CA\$)
1	99.99	99.99
5	87.49	437.45
10	74.99	749.90
20	62.49	1,249.80
40	49.99	1,999.60
100	37.49	3,749.00

All prices are quoted in CA\$ and are guaranteed until December 31, 2005.

7.6.4.3 Irie Pascal FreeBSD Edition Prices (in UK)

No. Users	Price (UK£)	Total Price (UK£)
1	39.99	39.99
5	34.99	174.95
10	29.99	299.90
20	24.99	499.80
40	19.99	799.60
100	14.99	1,499.00

All prices are quoted in UK£ and are guaranteed until December 31, 2005.

7.6.4.4 Irie Pascal FreeBSD Edition Prices (in Euros)

No. Users	Unit Price (Euros)	Total Price (Euros)
1	59.99	59.99
5	52.99	264.95
10	45.99	459.90
20	38.99	779.80
40	31.99	1,279.60
100	24.99	2,499.00

All prices are quoted in Euros and are guaranteed until December 31, 2005.

7.6.5.1 Irie Pascal Solaris/x86 Edition Prices (in US\$)

No. Users	Unit Price (US\$)	Total Price (US\$)
1	79.99	79.99
5	69.99	349.95
10	59.99	599.90
20	49.99	999.80
40	39.99	1,599.60
100	29.99	2,999.00

All prices are quoted in US\$ and are guaranteed until December 31, 2005.

7.6.5.2 Irie Pascal Solaris/x86 Edition Prices (in CA\$)

No. Users	Unit Price (CA\$)	Total Price (CA\$)
1	99.99	99.99
5	87.49	437.45
10	74.99	749.90
20	62.49	1,249.80
40	49.99	1,999.60
100	37.49	3,749.00

All prices are quoted in CA\$ and are guaranteed until December 31, 2005.

7.6.5.3 Irie Pascal Solaris/x86 Edition Prices (in UK)

No. Users	Price (UK£)	Total Price (UK£)
1	39.99	39.99
5	34.99	174.95
10	29.99	299.90
20	24.99	499.80
40	19.99	799.60
100	14.99	1,499.00

All prices are quoted in UK£ and are guaranteed until December 31, 2005.

7.6.5.4 Irie Pascal Solaris/x86 Edition Prices (in Euros)

No. Users	Unit Price (Euros)	Total Price (Euros)
1	59.99	59.99
5	52.99	264.95
10	45.99	459.90
20	38.99	779.80
40	31.99	1,279.60
100	24.99	2,499.00

All prices are quoted in Euros and are guaranteed until December 31, 2005.

7.6.6.1 Irie Pascal Solaris/Sparc Edition Prices (in US\$)

No. Users	Unit Price (US\$)	Total Price (US\$)
1	79.99	79.99
5	69.99	349.95
10	59.99	599.90
20	49.99	999.80
40	39.99	1,599.60
100	29.99	2,999.00

All prices are quoted in US\$ and are guaranteed until December 31, 2005.

7.6.6.2 Irie Pascal Solaris/Sparc Edition Prices (in CA\$)

No. Users	Unit Price (CA\$)	Total Price (CA\$)
1	99.99	99.99
5	87.49	437.45
10	74.99	749.90
20	62.49	1,249.80
40	49.99	1,999.60
100	37.49	3,749.00

All prices are quoted in CA\$ and are guaranteed until December 31, 2005.

7.6.6.3 Irie Pascal Solaris/Sparc Edition Prices (in UK)

No. Users	Price (UK£)	Total Price (UK£)
1	39.99	39.99
5	34.99	174.95
10	29.99	299.90
20	24.99	499.80
40	19.99	799.60
100	14.99	1,499.00

All prices are quoted in UK£ and are guaranteed until December 31, 2005.

7.6.6.4 Irie Pascal Solaris/Sparc Edition Prices (in Euros)

No. Users	Unit Price (Euros)	Total Price (Euros)
1	59.99	59.99
5	52.99	264.95
10	45.99	459.90
20	38.99	779.80
40	31.99	1,279.60
100	24.99	2,499.00

All prices are quoted in Euros and are guaranteed until December 31, 2005.

7.6.7.1 Irie Pascal Universal Edition Prices (in US\$)

No. Users	Unit Price (US\$)	Total Price (US\$)
1	149.99	149.99
5	131.99	659.95
10	113.99	1,139.90
20	95.99	1,919.80
40	77.99	3,119.60
100	59.99	5,999.00

All prices are quoted in US\$ and are guaranteed until December 31, 2005.

7.6.7.2 Irie Pascal Universal Edition Prices (in CA\$)

No. Users	Unit Price (CA\$)	Total Price (CA\$)
1	187.49	187.49
5	164.99	824.95
10	142.49	1,424.90
20	119.99	2,399.80
40	97.49	3,899.60
100	74.99	7,499.00

All prices are quoted in CA\$ and are guaranteed until December 31, 2005.

7.6.7.3 Irie Pascal Universal Edition Prices (in UK)

No. Users	Unit Price (UK£)	Total Price (UK£)
1	74.99	74.99
5	65.99	329.95
10	56.99	569.90
20	47.99	959.80
40	38.99	1,559.60
100	29.99	2,999.00

All prices are quoted in UK£ and are guaranteed until December 31, 2005.

7.6.7.4 Irie Pascal Universal Edition Prices (in Euros)

No. Users	Unit Price (Euros)	Total Price (Euros)
1	112.49	112.49
5	98.99	494.95
10	85.49	854.90
20	71.99	1,439.80
40	58.49	2,339.60
100	44.99	4,499.00

All prices are quoted in Euros and are guaranteed until December 31, 2005.

7.7.1 Why you should buy a license

There are basically two reasons why you should buy an Irie Pascal license.

- The first reason is that it is the right thing to do. When you buy an Irie Pascal license you are purchasing the legal right to use Irie Pascal under the terms specified in the license.
- The second reason is that it is in your own enlightened self-interest. When you buy an Irie Pascal license you contribute to the continued development of Irie Pascal. If you use Irie Pascal you are making an investment of your time and effort, and as Irie Pascal develops the value of your investment increases (i.e. you can do more of what you want to do with less effort).

So if you find Irie Pascal useful, it makes sense to buy a license to use it.

7.7.2 How do you buy a license

Buying an Irie Pascal license is quick, easy, and secure. You can choose from a number of options when making your purchase. These options are described below.

• **Payment Methods:**

You can buy Irie Pascal licenses using a variety of payment methods (Credit Card, Check/Money Order, Wire Transfer, Purchase Order).

- **Payment Sites:**

Payments for Irie Pascal licenses are accepted at a number of different sites. You can make your purchase directly to Irie Tools by fax or mail. You can also purchase from one of the following payment processors, ShareIt (at www.ShareIt.com), or RegNow (at www.RegNow.com).

- **Currencies:**

You can choose to pay for licenses in a variety of currencies (US\$, CA\$, UK£, and Euros).

Buying Licenses:

Not all combinations of options are possible (for example all purchase orders must be sent directly to Irie Tools and not to the other payment processors). For this reason it is recommended that you allow the IDE or the Irie Tools website to guide you through the buying process. Don't worry it is easier to buy a license than it is to describe how to buy a license.

The IDE includes a dialog box that will guide you through the buying process. To access this dialog box choose **Buy Now!...** from the [Help menu](#). See [Buy Now!](#) for more information on how to use this dialog box.

The Irie Tools website (at www.iriertools.com/iriepascal/buy.html) can also guide you through the process of an Irie Pascal license. From the website you will be able to have an order form generated and emailed to you (you would then complete the order form and fax it in or mail it in along with your payment). From the website you will also be able to go to one of the online credit card processors which can accept your payment. The online credit card processors have been carefully selected to ensure that your payment information will be secure, and allow you to download Irie Pascal immediately after your payment is processed.

Sales enquiries can be sent to sales@iriertools.com.

7.7.3 Buying using the IDE

The Irie Pascal IDE includes a dialog box that will guide you through the buying process. To access this dialog box choose **Buy Now!...** from the [Help menu](#). See [Buy Now!](#) for more information on how to use this dialog box.

Sales enquiries can be sent to sales@iriertools.com.

7.7.4 Buying through the web

The Irie Tools website at www.iriertools.com/iriepascal/buy.html can guide you through the process of buying an Irie Pascal license.

From the website you can have an order form generated (which you can print using your browser). You can have the order form emailed to you (you can then print it using your email program). After printing the order form you would then complete it and fax it (see [buying by fax](#) for more information) or mail it (see [buying by mail](#)) for more information.

From the website you can be transferred to one of the online payment processors which can accept your payment. The online payment processors have been carefully selected to ensure that your payment

information will be secure. If you pay through an online payment processor then you will have the option of downloading Irie Pascal immediately after your payment is processed.

Sales enquiries can be sent to sales@irietools.com.

7.7.5 Buying by telephone

If you are paying by credit card then it is possible to phone in your order. Telephone orders are accepted by ShareIt (one of the online payment processors authorized to accept payments for Irie Pascal licenses). You can access ShareIt through the Irie Tools website at www.irietools.com/iriepascal/buy.html or you can go directly to ShareIt at www.ShareIt.com and search for Irie Pascal.

NOTE: ShareIt has been carefully selected as an authorized payment processor to ensure that your payment information will be secure.

Sales enquiries can be sent to sales@irietools.com.

7.7.6 Buying by Fax

If you are paying with a credit card or if you are sending in a purchase order then it is possible to fax in your order.

First you will need to get an order form. You can get an order form from the Irie Pascal IDE. Choose **Buy Now!...** from the [Help menu](#) which will bring up a dialog box you can use to get an order form. See [Buy Now!](#) for more information on how to use this dialog box. You can also get an order form from the Irie Tools website (at www.irietools.com/iriepascal/buy.html). You will be able to have an order form generated for you, which you can then print or have it emailed to you.

If you are paying with a credit card then you will need to fill in the order form and fax it to ShareIt (one of the online payment processors authorized to accept payments for Irie Pascal licenses). Go to www.ShareIt.com.

If you sending in a purchase order then you must fax it along with the completed order form. **NOTE:** If you are not already a customer of Irie Tools it is best to check whether purchase orders from your company/institution will be accepted before ordering. See [Purchase orders](#) for information on using purchase orders.

Sales enquiries can be sent to sales@irietools.com.

7.7.7 Buying through the mail

Irie Pascal orders can be sent in by postal mail.

First you will need to get an order form. You can get an order form from the Irie Pascal IDE. Choose **Buy Now!...** from the [Help menu](#) which will bring up a dialog box you can use to get an order form. See [Buy Now!](#) for more information on how to use this dialog box. You can also get an order form from the Irie Tools website (at www.irietools.com/iriepascal/buy.html). You will be able to have an order form generated for you, which you can then print or have it emailed to you.

Then you will need to fill in the order form and mail it to:

Attn: Stuart King
Irie Tools
221 S. State Road 7, MB #247
Plantation, FL, 33317, USA

If you are using a purchase order then you must mail it along with the completed order form. NOTE: If you are not already a customer of Irie Tools it is best to check whether purchase orders from your company/institution will be accepted before ordering. See [Purchase orders](#) for information on using purchase orders.

If you are using a check or money order then you should make it out to **Irie Tools** and mail it along with the completed order form.

Sales enquiries can be sent to sales@irietools.com.

7.7.8 Buying by wire transfer

If you are paying by wire transfer card then you should order through ShareIt (one of the online payment processors authorized to accept payments for Irie Pascal licenses). You can access ShareIt through the Irie Tools website at www.irietools.com or you can go directly to ShareIt at www.ShareIt.com and search for Irie Pascal.

NOTE: ShareIt has been carefully selected as an authorized payment processor to ensure that your payment information will be secure.

Sales enquiries can be sent to sales@irietools.com.

7.7.9 Purchase orders

Purchase orders are accepted from government and accredited educational institutions and major corporations, provided that they are submitted on purchase order forms with a purchase order number. You must include an Irie Pascal order form along with your purchase order. NOTE: If you are not already a customer of Irie Tools it is best to check whether purchase orders from your company/institution will be accepted before ordering.

Purchase orders can be mailed to:

Attn: Stuart King
Irie Tools
221 S. State Road 7, MB #247
Plantation, FL, 33317, USA

or faxed to **1-876-946-2703**.

Sales enquiries can be sent to sales@irietools.com.

7.7.10.1.1 Irie Pascal (Windows edition) US\$ Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.irietools.com
Email: sales@irietools.com

No. Users	Unit Price	Quantity	Total Price
1	79.99	_____	\$ _____
5	69.99	_____	\$ _____
10	59.99	_____	\$ _____
20	49.99	_____	\$ _____
40	39.99	_____	\$ _____
100	29.99	_____	\$ _____

Shipping

& Handling 5.00 \$ _____

TOTAL \$ _____

Check/Money Order enclosed Charge credit card

AmEx MasterCard Visa

Account #: _____

Expiration Date: _____

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in US\$ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be*

notified by email when your order is shipped.

7.7.10.1.2 Irie Pascal (Windows Edition) CAS Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	99.99	_____	\$ _____
5	87.49	_____	\$ _____
10	74.99	_____	\$ _____
20	62.49	_____	\$ _____
40	49.99	_____	\$ _____
100	37.49	_____	\$ _____
Shipping			
& Handling	7.00		\$ _____
TOTAL			\$ _____

[] Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

[] Tick this box to join the Irie Pascal mailing list.

Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in CA\$ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.

7.7.10.1.3 Irie Pascal (Windows Edition) UK Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	39.99	_____	_____
5	34.99	_____	_____
10	29.99	_____	_____
20	24.99	_____	_____
40	19.99	_____	_____
100	14.99	_____	_____
Shipping			
& Handling	3.00		_____
TOTAL			_____

[] Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

[] Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in UK£ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.1.4 Irie Pascal (Windows Edition) Euro Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	59.99	_____	_____
5	52.99	_____	_____
10	45.99	_____	_____
20	38.99	_____	_____
40	31.99	_____	_____
100	24.99	_____	_____
Shipping			
& Handling	4.00		_____
TOTAL			_____

[] Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in Euros and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.2.1 Irie Pascal (Linux Edition) US\$ Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	79.99	_____	\$ _____
5	69.99	_____	\$ _____
10	59.99	_____	\$ _____
20	49.99	_____	\$ _____
40	39.99	_____	\$ _____
100	29.99	_____	\$ _____
Shipping			
& Handling	5.00		\$ _____
TOTAL			\$ _____

Check/Money Order enclosed

Charge credit card

AmEx

MasterCard

Visa

Account #: _____

Expiration Date: _____

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

[] Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in US\$ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.2.2 Irie Pascal (Linux Edition) CAS Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	99.99	_____	\$ _____
5	87.49	_____	\$ _____
10	74.99	_____	\$ _____
20	62.49	_____	\$ _____
40	49.99	_____	\$ _____
100	37.49	_____	\$ _____

Shipping

& Handling 7.00 \$ _____

No. Users	Unit Price	Quantity	Total Price
TOTAL			\$ _____

Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in CA\$ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.2.3 Irie Pascal (Linux Edition) UK Order Form



Irie Tools
 221 S. State Rd 7, #247
 Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	39.99	_____	_____
5	34.99	_____	_____
10	29.99	_____	_____
20	24.99	_____	_____
40	19.99	_____	_____

No. Users	Unit Price	Quantity	Total Price
100	14.99	_____	_____
Shipping			
& Handling	3.00		_____
TOTAL			_____

Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in UK£ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.2.4 Irie Pascal (Linux Edition) Euro Order Form



Irie Tools
 221 S. State Rd 7, #247
 Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	59.99	_____	_____
5	52.99	_____	_____

No. Users	Unit Price	Quantity	Total Price
10	45.99	_____	_____
20	38.99	_____	_____
40	31.99	_____	_____
100	24.99	_____	_____

Shipping

& Handling 4.00 _____

TOTAL _____

Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in Euros and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.3.1 Irie Pascal (FreeBSD Edition) US\$ Order Form



Irie Tools
 221 S. State Rd 7, #247
 Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	79.99	_____	\$ _____
5	69.99	_____	\$ _____
10	59.99	_____	\$ _____
20	49.99	_____	\$ _____
40	39.99	_____	\$ _____
100	29.99	_____	\$ _____

Shipping

& Handling 5.00 \$ _____

TOTAL \$ _____

Check/Money Order enclosed Charge credit card

AmEx MasterCard Visa

Account #: _____

Expiration Date: _____

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in US\$ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be

notified by email when your order is shipped.

7.7.10.3.2 Irie Pascal (FreeBSD Edition) CAS Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	99.99	_____	\$ _____
5	87.49	_____	\$ _____
10	74.99	_____	\$ _____
20	62.49	_____	\$ _____
40	49.99	_____	\$ _____
100	37.49	_____	\$ _____
Shipping			
& Handling	7.00		\$ _____
TOTAL			\$ _____

[] Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in CA\$ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.

7.7.10.3.3 Irie Pascal (FreeBSD Edition) UK Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	39.99	_____	_____
5	34.99	_____	_____
10	29.99	_____	_____
20	24.99	_____	_____
40	19.99	_____	_____
100	14.99	_____	_____
Shipping			
& Handling	3.00		_____
TOTAL			_____

Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

[] Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in UK£ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.3.4 Irie Pascal (FreeBSD Edition) Euro Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	59.99	_____	_____
5	52.99	_____	_____
10	45.99	_____	_____
20	38.99	_____	_____
40	31.99	_____	_____
100	24.99	_____	_____
Shipping			
& Handling	4.00		_____
TOTAL			_____

[] Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in Euros and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.4.1 Irie Pascal (Solaris/x86 Edition) US\$ Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	79.99	_____	\$ _____
5	69.99	_____	\$ _____
10	59.99	_____	\$ _____
20	49.99	_____	\$ _____
40	39.99	_____	\$ _____
100	29.99	_____	\$ _____
Shipping			
& Handling	5.00		\$ _____
TOTAL			\$ _____

Check/Money Order enclosed

Charge credit card

AmEx

MasterCard

Visa

Account #: _____

Expiration Date: _____

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

[] Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in US\$ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.4.2 Irie Pascal (Solaris/x86 Edition) CA\$ Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	99.99	_____	\$ _____
5	87.49	_____	\$ _____
10	74.99	_____	\$ _____
20	62.49	_____	\$ _____
40	49.99	_____	\$ _____
100	37.49	_____	\$ _____

Shipping

& Handling 7.00 \$ _____

No. Users	Unit Price	Quantity	Total Price
TOTAL			\$ _____

Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in CA\$ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.4.3 Irie Pascal (Solaris/x86 Edition) UK Order Form



Irie Tools
 221 S. State Rd 7, #247
 Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	39.99	_____	_____
5	34.99	_____	_____
10	29.99	_____	_____
20	24.99	_____	_____
40	19.99	_____	_____

No. Users	Unit Price	Quantity	Total Price
100	14.99	_____	_____
Shipping			
& Handling	3.00		_____
TOTAL			_____

Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in UK£ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.4.4 Irie Pascal (Solaris/x86 Edition) Euro Order Form



Irie Tools
 221 S. State Rd 7, #247
 Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	59.99	_____	_____
5	52.99	_____	_____

No. Users	Unit Price	Quantity	Total Price
10	45.99	_____	_____
20	38.99	_____	_____
40	31.99	_____	_____
100	24.99	_____	_____

Shipping

& Handling 4.00 _____

TOTAL _____

Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in Euros and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.5.1 Irie Pascal (Solaris/Sparc Edition) US\$ Order Form



Irie Tools
 221 S. State Rd 7, #247
 Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	79.99	_____	\$ _____
5	69.99	_____	\$ _____
10	59.99	_____	\$ _____
20	49.99	_____	\$ _____
40	39.99	_____	\$ _____
100	29.99	_____	\$ _____

Shipping

& Handling 5.00 \$ _____

TOTAL \$ _____

Check/Money Order enclosed Charge credit card

AmEx MasterCard Visa

Account #: _____

Expiration Date: _____

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in US\$ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be*

notified by email when your order is shipped.

7.7.10.5.2 Irie Pascal (Solaris/Sparc Edition) CAS Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	99.99	_____	\$ _____
5	87.49	_____	\$ _____
10	74.99	_____	\$ _____
20	62.49	_____	\$ _____
40	49.99	_____	\$ _____
100	37.49	_____	\$ _____
Shipping			
& Handling	7.00		\$ _____
TOTAL			\$ _____

[] Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in CA\$ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.

7.7.10.5.3 Irie Pascal (Solaris/Sparc Edition) UK Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	39.99	_____	_____
5	34.99	_____	_____
10	29.99	_____	_____
20	24.99	_____	_____
40	19.99	_____	_____
100	14.99	_____	_____
Shipping			
& Handling	3.00		_____
TOTAL			_____

Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

[] Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in UK£ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.5.4 Irie Pascal (Solaris/Sparc Edition) Euro Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	59.99	_____	_____
5	52.99	_____	_____
10	45.99	_____	_____
20	38.99	_____	_____
40	31.99	_____	_____
100	24.99	_____	_____
Shipping			
& Handling	4.00		_____
TOTAL			_____

[] Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in Euros and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.6.1 Irie Pascal (Universal Edition) US\$ Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	149.99	_____	\$ _____
5	131.99	_____	\$ _____
10	113.99	_____	\$ _____
20	95.99	_____	\$ _____
40	77.99	_____	\$ _____
100	59.99	_____	\$ _____
Shipping			
& Handling	5.00		\$ _____
TOTAL			\$ _____

Check/Money Order enclosed

Charge credit card

AmEx

MasterCard

Visa

Account #: _____

Expiration Date: _____

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

[] Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in US\$ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.6.2 Irie Pascal (Universal Edition) CAS Order Form



Irie Tools
221 S. State Rd 7, #247
Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	187.49	_____	\$ _____
5	164.99	_____	\$ _____
10	142.49	_____	\$ _____
20	119.99	_____	\$ _____
40	97.49	_____	\$ _____
100	74.99	_____	\$ _____

Shipping

& Handling 7.00 \$ _____

No. Users	Unit Price	Quantity	Total Price
TOTAL			\$ _____

Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in CA\$ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.6.3 Irie Pascal (Universal Edition) UK Order Form



Irie Tools
 221 S. State Rd 7, #247
 Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	74.99	_____	_____
5	65.99	_____	_____
10	56.99	_____	_____
20	47.99	_____	_____
40	38.99	_____	_____

No. Users	Unit Price	Quantity	Total Price
100	29.99	_____	_____
Shipping			
& Handling	3.00		_____
TOTAL			_____

Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in UK£ and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

7.7.10.6.4 Irie Pascal (Universal Edition) Euro Order Form



Irie Tools
 221 S. State Rd 7, #247
 Ft. Lauderdale, FL 33317, USA

web: www.iriertools.com
Email: sales@iriertools.com

No. Users	Unit Price	Quantity	Total Price
1	112.49	_____	_____
5	98.99	_____	_____

No. Users	Unit Price	Quantity	Total Price
10	85.49	_____	_____
20	71.99	_____	_____
40	58.49	_____	_____
100	44.99	_____	_____
Shipping			
& Handling	4.00		_____
TOTAL			_____

Check/Money Order enclosed

Name: _____

Address: _____

City/State/Province: _____

Country/Postal Code: _____

Telephone: _____ [optional]

Email Address: _____ [optional]

How did you hear about Irie Pascal? _____

_____ [optional]

Tick this box to join the Irie Pascal mailing list.

*Please do **not** send cash. Make checks or money orders payable to **Irie Tools**. All prices are quoted in Euros and are guaranteed until December 31, 2005. Please allow two to four weeks for delivery. If you fill in your email address you will be notified by email when your order is shipped.*

" Warning: Program parameter X has invalid type"

This warning message is displayed when a parameter declared by your program will not be able to access command-line arguments passed to your program, because the type of the parameter is invalid (i.e. is not a file or string type).

" Warning: Real constant underflowed to zero"

This warning message is displayed when the value of a real constant is so small that it can not be represented by any non-zero real number, and as a result was converted to zero (0.0).

" Warning: Implementation limit exceeded (range of variant selector)"

This warning message is displayed when the range of a record's variant selector is larger than the largest range supported by Irie Pascal. **NOTE:** Variant selectors determine which variant of a variant record is active. Currently Irie Pascal limits the range of possible values of variant selectors to 1024.

" Warning: Statement label is greater than 9999"

This warning message is displayed when the value of a statement label is greater than the limit set by Standard Pascal.

" Warning: ? is never used"

This warning message is displayed when your program declares a **label**, **constant**, **type**, **variable**, **function**, or **procedure**, but does not use it.

" Warning: Record has no fields"

This warning message is displayed when a record is declared without any fields.

" Warning: 'otherwise' can never be used"

This warning message is displayed when one of the following occurs:

- **otherwise** is used in a variant record but all the possible values of the variant selector have already been linked to a variant, and therefore the variant linked to **otherwise** can never become active. For example

```
t = record case a : boolean of
true : ( x : integer);
false: ( y : char);
otherwise
(z : boolean)
end;
```

will generate this warning, because the variant selector **a** can have only two values (**true** and **false**), and these values are linked to the first two variants (**x : integer**) and (**y : char**), and therefore the third variant (**z : boolean**) can never become active.

- **otherwise** is used in a case statement but all the possible values of the case index have already been linked to one of the case list element, and therefore the case list element linked to **otherwise** can never be executed.

" Warning: Invalid compiler directive"

This warning message is displayed when a compiler directive is unknown or has a syntax error.

" Warning: ? is not associated with a variant"

This warning message is displayed when one more more possible values of the variant selector in a variant record is not linked to a variant. For example:

```
t = record case a : boolean of
true : ( x : integer);
end;
```

" Warning: Control variable is threatened"

Standard Pascal has a number of rules that are meant to ensure that the control variable of a **for** statement is modified only by the **for** statement, and any other statement that can modify the control variable, while the **for** statement is executing is said to *threaten* the control variable. In Standard Pascal it is an error to *threaten* the control variable of a **for** statement.

When all extensions are disabled (which can be done only with the command-line compiler), Irie Pascal also considers it an error to *threaten* the control variable of a **for** statement. When all extensions are **not** disabled (which is always the case when using the IDE), Irie Pascal does not consider it an error to *threaten* the control variable of a **for** statement, instead this warning message is displayed.

" Warning: Control variable must be local"

In Standard Pascal, there is a rule that the control variable of a **for** statement must either be local to the function or procedure that contains the **for** statement, or be global if the **for** statement is not contained in a function or procedure.

When all extensions are disabled (which can be done only with the command-line compiler), Irie Pascal displays an error message whenever this rule is violated. When all extensions are **not** disabled (which is always the case when using the IDE), Irie Pascal does enforce this rule, instead this warning message is displayed.

" Warning: No result variables specified"

This warning message is displayed when the built-in procedure **fsplit** is called but no variables are passed to retrieve the results of the call.

" Warning: Comment includes end of file"

If a comment in an include file is not closed inside the include file, then the comment continues into the file that included the include file, and this warning message is displayed.

" Warning: Recursive file inclusion ignored"

This warning message is displayed when an include file, directly or indirectly, includes itself.

" Warning: Declaration of X hides previous declaration"

This warning message is displayed when the declaration of an identifier hides a declaration of the same identifier in an outer scope.

" Warning: Assignment to 'null' has no effect"

This warning message is displayed when a value is assigned to the built-in variable **null**.

" Warning: Label is never referenced by a goto statement"

This warning message is displayed when a label is declared but not referenced by a **goto** statement.

NOTE: This warning message is displayed even if the label has been used to mark a statement.

" Warning: Variable's size is zero"

This warning message is displayed when a variable is declared, but because of its type is zero bytes long.

" Warning: Function return value does not exist"

This warning message is displayed when your program contains a function call, that the compiler is able to determine, can not return a legal value given the function argument(s).

For example the following function call

```
succ (x) ;
```

will cause this warning message to be displayed, if **x** is a value whose type has only one value.

" Warning: Comparing signed and unsigned integral values"

This warning message is when signed and unsigned integral values are used in the same expression. In this situation it can be difficult for the compiler to know the appropriate type for the expression.

" Warning: This program is too big for the demo version of Irie Pascal"

The demo version of Irie Pascal, is limited in the size of the program it can compile, and when the limit is exceeded this warning message is displayed and the executable is **not** generated.

Displays a color choice dialog box that allows you to choose the text color used in the edit windows.

Displays a color choice dialog box that allows you to choose the background color used in the edit windows.

Shows a sample of what your edit window color choices will look like.

Displays a color choice dialog box that allows you to choose the text color used for informational messages in the Message Window.

Displays a color choice dialog box that allows you to choose the background color used for information messages in the Message Window.

Shows a sample of what your information message color choices will look like.

Displays a color choice dialog box that allows you to choose the text color used for warning messages in the Message Window.

Displays a color choice dialog box that allows you to choose the background color used for warning messages in the Message Window.

Shows a sample of what your warning message color choices will look like.

Displays a color choice dialog box that allows you to choose the text color used for error messages in the Message Window.

Displays a color choice dialog box that allows you to choose the background color used for error messages in the Message Window.

Shows a sample of what your error message color choices will look like.

Saves the color choices you have made.

Closes this dialog box without saving the color choices you have made.

Displays an overview of this dialog box.

Provides a space for you to type in the number of the line, in the current file, that you want to go to.

Moves you to the line specified.

Closes the dialog box without moving to another line.

Shows a sample of what the current text type looks like with your text preferences.

Shows the name of your preferred font.

Shows the size of your preferred font.

Displays a dialog box that allows you to select your preferred font.

Allows you to select the text type that you want to specify preferences for.

Displays a dialog box that allows you to choose the foreground color of the selected text type.

Displays a dialog box that allows you to choose the background color of the selected text type.

Check this option if you don't want to start a new project for every program that you create. When this option is checked, the executable options specified by the current project will be ignored, and the file executable options specified in this dialog box will be used instead. Also when this option is checked, the program specified by the current project will be ignored, and Irie Pascal will compile and run whichever program is currently being edited.

Allows you to specify the file extension of the executable generated or executed, when project executable options are being ignored, and you compile or run the current project. **NOTE:** The file extension should begin with a period (.).

Allows you to specify the arguments (if any) that should be passed to the executable, when project executable options are being ignored, and you run the current project.

Allows you to specify the #! header that should be prefixed to the executable (may be blank), when project executable options are being ignored, and you compile the current project.

" Assignment overflow"

When this option is enabled, your program will check each time a value is assigned to a string or set variable, to make sure that the value is not too long to fit. If the value is too long to fit in the variable then an assignment overflow error is generated.

" I/O errors"

When this option is enabled, your program will check each I/O operation for errors.

" Range errors"

When this option is enabled, your program will check for range errors. Some possible causes of range errors are:

- Attempts to assign or read in values which are not assignment compatible with a particular type to a variable of that type. For example given the declaration below:

```
var bit : 0..1;
```

the assignment below will cause a range error:

```
bit := 2;
```

- Attempts to access element outside of array bounds. For example given the declaration below:

```
var x : array[-4..10] of integer;
```

the attempt below to reference an out-of bounds array element will cause a range error:

```
x[-5]
```

" Stack overflow"

When this option is enabled, your program checks for stack overflow or underflow before each program statement is executed. When this option is not enabled, your program checks for stack overflow or underflow only at the beginning and end of each function/procedure call and when large values are placed on the stack.

" Using undefined values"

When this option is enabled, your program checks each time a value, from a variable, or returned by a function call, is accessed, to make sure that the value is not undefined. **NOTE:** Not all values can be checked, checks are only made for values of the following types:

- enumerated types (including boolean)
- subranges of enumerated types
- subranges of integer that do not include -1
- file types
- list types
- object types
- pointer types
- real

- set types (array representation)

Accesses to values of types: **char**, **integer**, **record types** and **set types (Bit set representation)** are not checked.

" Using in-active variants"

This check box controls whether your program checks each time a field of a variant is accessed, to make sure that the variant is active.

" On the console screen"

This check box controls whether run-time errors are displayed on your program's console window.

" in message boxes"

This check box controls whether run-time errors are displayed in Windows' message boxes.

" in log files"

This check box controls whether run-time errors are logged to a file named **name.log** (where **name** is the name of your program).

" Enable asserts"

This check box controls whether the compiler generates code for the built-in procedure **assert**. **NOTE:** Since the compiler parses **assert** procedures whether or not this check box is checked, compile-time errors in **assert** procedures are reported regardless of the setting of this check box.

" Generate Windows NT/2000 service application"

Check this box to compile your program as a Windows NT/2000 service. **NOTE:** Your program must be a EXE executable.

" Use short-circuit evaluation"

This check box controls whether your program uses short-circuit evaluation for the boolean operators **and** and **or**. When short-circuit evaluation is used for the boolean **and** operator, your program evaluates the left-hand operand and if the result is **false**, then the right-hand operand is not evaluated because the result of the operation must be **false**. When short-circuit evaluation is used for the boolean **or** operator, your program evaluates the left-hand operand and if the result is **true**, then the right-hand operand is not evaluated because the result of the operation must be **true**.

You might want to disable short-circuit evaluation if you need the side-effects of evaluating the right operand. For example, suppose the right operand is a call to a function which modifies some global variables (a side-effect) in addition to returning a boolean value, then short-circuit evaluation might cause the function not to be called and therefore the global variables will not get modified. If you want to make sure that the right operand is always evaluated then disable short-circuit evaluation.

" Insert line-number debugging information"

This check box controls whether line-number debugging information is inserted into your program. Line number debugging information makes your program larger, but allows for more meaningful run-time error messages since they will include number of the source line that caused the error.

" Specify align size"

This allows you to control the maximum alignment used by the compiler. Some CPUs (including those in the Intel 80x86 family) access data faster if it is aligned on an address which is a multiple of the size of the data. Suppose the CPU is accessing a real which is 8 bytes long, then for fastest access, the real should be on an address which is a multiple of 8 (for example 0, 8, 16, 24, 32, etc).

The compiler stores variables in memory at the lowest available address which is a multiple of either the variable's size or the maximum alignment, whichever is smaller.

For example suppose you compile the following program and the maximum alignment is 4.

```
program x(output) ;
var
c : char;
i : integer;
b : boolean;
r : real;
begin
end.
```

The compiler needs to decide where to store the variables **c**, **i**, **b** and **r**. The first variable **c** gets stored at address 0, and since **c** is a char variable (which are 1 byte long) the available addresses are from 1 upwards. The second variable **i** is an integer variable (which are 4 bytes long). The lowest available address which is a multiple of the variable size is 4 and the lowest available address which is a multiple of the maximum alignment is also 4. So the compiler stores **i** at address 4, and the available address are from 8 upwards (since **i** is 4 bytes long). The third variable **b** is a boolean variable (which are 4 bytes long). The lowest available address which is a multiple of the variable size is 8 and the lowest available address which is a multiple of the maximum alignment is also 8. So the compiler stores **b** at address 8, and the available address are from 12 upwards (since **b** is 4 bytes long). The fourth variable **r** is a real variable (which are 8 bytes long). The lowest available address which is a multiple of the variable size is 16 but the lowest available address which is a multiple of the maximum alignment is 12. So the compiler stores **r** at address 12 (since 12 is less than 16).

In general if you set maximum alignment to 1 then you waste no memory but you get fastest access only for chars. If you set the maximum alignment to 4 you may waste memory when storing all variables except char, but you get the fastest access to all variables except real. If you set the maximum alignment to 8 you may waste memory when storing all variables except char, but you get the fastest access to variables of all types.

If you are not sure about the maximum alignment to use just leave the default (4).

" Specify stack size"

This allows you to control the size of your program's stack (in Kilobytes). The default stack size (64K) should be more than enough for the vast majority of programs. However if your program is heavily recursive or has functions that need a large amount of space for local variables or parameters then you can increase this value up to a maximum of 1024K (1MB). On the other hand you can also reduce the size of your program's stack (not recommended).

" Use tab character"

When this check box is checked the editor will use the tab character to store tabs. If tab characters are used then you can adjust the size of the tabs using the **tab size** checkbox.

When this check box is not checked, the editor will convert tabs to spaces before storing them. Tabs are converted to spaces whether they are entered using the keyboard, or occur in files opened by the editor.

" Auto-indent lines"

This check box controls whether the editor will automatically indent new lines to match the indentation of the previous line.

" Create backup files"

This check box controls whether the editor will create a backup when files are saved. The backup file has the same name as the original file, except with the extension (.bak). If the original file already has a (.bak) extension then no backup is created.

" Double-click performs word search"

When this check box is checked double-clicking on a word in the editor will cause the IDE to search the Irie Pascal Programmer's Reference Manual (in "progref.html") for help on that word.

When this check box is not checked double-clicking on a word in the editor will cause the editor to select the word.

" Tab size"

This allows you to specify the number of characters between tab stops. The default tab size is 8. If you use tabs to indent your programs you might want to reduce this size so that more of your code fits on the screen.

" Name of your program's executable"

If you want to change the name of the executable generated by the compiler, enter the new name here. By default, the name of the executable generated by the compiler will be the same as the name of the project, except with the extension **.ivm**. If you enter a name with the extension **.exe**, the compiler will generate a true EXE executable. If the name here does not have an extension or has an extension other than **.exe** then the compiler will generate an IVM executable.

" Arguments passed by the IDE"

This is where you would enter any arguments you want to pass to your program when it is run from inside the IDE.

" #! header"

This is where you would enter any #! header that you want in the executable generated by the compiler. UNIX like operating systems such as Linux, FreeBSD, and Solaris will use the #! header to locate the

interpreter. This allows you to run your executable without specifying the interpreter (a necessity for CGI applications). NOTE: You also need to set the executable permission bit of the executable for this to work. IMPORTANT: This header is ignored under non-UNIX like operating systems, so including this header does not prevent the executable from running under any supported operating system.

For example, if you plan to deploy your executable on Linux, and you know that the full pathname of the interpreter is `/usr/local/bin/ivm`, then you would just enter that pathname into this text box, and the compiler will generate the `#!` header for you.

NOTE: The command-line compiler can also generate `#!` headers (see the [-h compiler option](#)). If you wish to add, change, or delete the `#!` header of an existing executable then you can use the Irie Header Utility (see [using the Irie Header Utility](#)). Which can be useful if you can't, or don't want to recompile the program.

" Allow nested comments"

This check box controls whether nested comments (which are comments inside other comments) are supported. For example

```
(* outer (* inner comment *) comment *)
```

When nested comments are not supported the example comment above will terminate at the first `*)` so only

```
(* outer (* inner comment *)
```

will be treated as a comment. When nested comments are supported the compiler recognizes the end of comments only when the number of close comment markers matches the number of open comment markers. So the example comment above will terminate only after the second `*)`.

Both open comment markers (`*` and `{` are considered to be equivalent, and both close comment markers `*)` and `}` are considered to be equivalent. So attempting to trick the compiler into accepting nested comments with something like

```
(* outer { inner comment } comment *)
```

will not work.

Nested comments are disabled by default since in Standard Pascal comments do not nest.

" Make identifiers case-sensitive"

When this check box is checked identifiers are case-sensitive, so for example the following are all different identifier (**hi**, **Hi**, **hI**, and **HI**).

When this check box is not checked, identifiers are not case-sensitive, so the identifiers in the previous example are all the same identifier.

Case-sensitive identifiers are disabled by default, since Pascal is normally not a case-sensitive language. If you enable this option then remember to use all lowercase for keywords and built-in identifiers (such as **var**, **integer**, **input**, **writeln**).

" Require parentheses"

This check box controls whether parentheses are mandatory in all function and procedure calls and after the program name. In Pascal parentheses are not normally used when calling or declaring functions or procedures with no parameters. So for example in the following statement:

```
a := b;
```

it is not clear whether **b** is a function that takes no parameters or whether **b** is a variable. When mandatory parentheses mode is enabled then parentheses are required when declaring or calling all functions and procedures even those with no parameters. Parentheses are also required after the program name even if there are no program parameters.

" Non-standard unary operators"

This check box controls whether the compiler handles the unary plus and unary minus operators in a non-standard way (i.e. different from the way specified by Standard Pascal). The compiler will allow the unary plus and unary minus operators to appear before any numeric operand, and their precedence is higher than any other operator.

" Open temp file if no name assigned"

Normally, if you open a file variable, of text type, without first assigning it a name, then the file variable will become associated with the the standard input stream or the standard output stream, depending on whether you are opening the file variable for reading or writing. In which case reading from the file variable will read from the standard input stream, and writing to the file variable will write to the standard output stream.

When this check box is checked and you open a file variable, of text type, without first assigning it a name, then the file variable will become associated with a temporary disk file and not the standard input stream or the standard output stream.

NOTE: Opening a file variable, that is **not** of text type, without first assigning it a name, will always associated the file variable with a temporary disk file, regardless of whether this check box is checked.

" Compatibility mode (with version 2.0)"

This check box controls whether the compiler is in version 2.0 compatibility mode. When you compile your program with the compiler in this mode, your program should behave in the same way it did when compiled with verions 2.0 or 2.1 of the compiler. Compatibility with versions earlier than 2.0 is not guaranteed.

In this version of the compiler the only effect of version 2.0 compatibility mode, is to make the compiler continue to incorrectly treat sets of subrange types as if they were sets of the subrange's host type. So for example

```
set of 0..1
```

is treated like

```
set of integer
```

which in this case will affect how values of the set are represented. Normally your program is not affected by the way set values are represented, however if your program stores set values in data files then the

representation of the set values is important.

Compatibility mode was introduced so that you can maintain the format of data files created by programs compiled with previous versions of the compiler, even when you compile those programs with this version of the compiler.

" Maximum number of errors allowed"

This is where you would enter the maximum number of errors you will allow the compiler to report before it should give up and stop compiling.

" Maximum number of warnings allowed"

This is where you would enter the maximum number of warning you will allow the compiler to report before it should give up and stop compiling.

" Select Edition List Box"

This list box is where you would choose the Irie Pascal edition you want to view prices for. Irie Pascal supports a number of different operating system, with a different edition for each supported operating system. The supported operating systems are: Windows (Windows 95 or later), Linux, FreeBSD, Solaris/Sparc, and Solaris/x86. There is also an edition (the Universal edition), that includes all of the other editions.

" Select Currency List Box"

This list box is where you would choose the currency that you want view prices in. Irie Pascal license fees can be paid in up to four different currencies (depending on the payment method used), so prices may be viewed in different currencies.

" Online Prices Button"

Use this button to go to the Irie Tools website to view prices, after selecting the Irie Pascal edition you are interested in and the currency you want to view the prices in.

" Help File Prices Button"

Use this button to view prices from this help file, after selecting the Irie Pascal edition you are interested in and the currency you want to view the prices in. **NOTE:** Prices from the help file are guaranteed until December 31, 2005. After that date you should contact Irie Tools for updated pricing.

" Cancel Button"

Use this button to close the **View Irie Pascal Prices** dialog box without viewing prices.

" Help Button"

Use this button to display help information about the **View Irie Pascal Prices** dialog box.

" Buy Now! Button"

Use this button to display the **Buy Now!** dialog box, which will guide you through the process of buying licenses to use Irie Pascal.

" Select Edition List Box"

This list box is where you would choose the Irie Pascal edition you want to buy licenses for. Irie Pascal supports a number of different operating system, with a different edition for each supported operating system. The supported operating systems are: Windows (Windows 95 or later), Linux, FreeBSD, Solaris/Sparc, and Solaris/x86. There is also an edition (the Universal edition), that includes all of the other editions.

" Select Currency List Box"

This list box is where you would choose the currency that you want to use to pay for the Irie Pascal licenses.

" Buy with a credit card"

Select this check box if you want to pay for Irie Pascal licenses with a credit card.

" Buy with a cheque"

Select this check box if you want to pay for Irie Pascal licenses with a check or international money order.

" Buy by wire transfer"

Select this check box if you want to pay for Irie Pascal licenses with a wire transfer.

" Buy with a purchase order"

Select this check box if you want to send in a purchase order pay for Irie Pascal licenses. **NOTE:** If you are not already a customer of Irie Tools it is best to check whether purchase orders from your company/institution will be accepted before ordering.

Sales enquiries can be sent to sales@iriertools.com.

" Buy Online Button"

Use this button to go to the Irie Tools website to buy Irie Pascal licenses.

From the website you can have an order form generated (which you can print using your browser). You can have the order form emailed to you (you can then print it using your email program). After printing the order form you would then complete it and fax it (see [buying by fax](#) for more information) or mail it (see [buying by mail](#)) for more information.

From the website you can be transferred to one of the online payment processors which can accept your payment. The online payment processors have been carefully selected to ensure that your payment

information will be secure. If you pay through an online payment processor then you will have the option of downloading Irie Pascal immediately after your payment is processed.

Sales enquiries can be sent to sales@irietools.com.

" Get Order Form Button"

Use this button to get an Irie Pascal order form for the edition and currency you have selected.

" Cancel Button"

Use this button to close the **Buy Now!** dialog box without buying any licenses.

" Help Button"

Use this button to display help information about the **Buy Now!** dialog box.

Access keys are associated with some menus and menu items, and are indicated by underlined letters in the menu titles and the text of the menu items.

A *compiler* translates programs written in one programming language into programs written in another programming language. The program being translated is usually written in a high-level language (like Pascal), and the program that results from the translation is usually written in a low-level language. This process of translating programs from one programming language to another is called *compiling*, and the program that performs this process is called a *compiler*.

Compiling is the process of translating programs written in one programming language into equivalent programs written in another programming language. The program being translated is usually written in a high-level language (like Pascal), and the program that results from the translation is usually written in a low-level language. The program that does the *compiling* is called a *compiler*.

Copyright © Stuart King, 2002-2005. All Rights Reserved.