


WIN-
PROLOG

4.900

**Prolog
Examples**

by Rebecca Shalfeld

WIN-PROLOG Example Files

This document describes the example files supplied with version 4.500 of **WIN-PROLOG** and are believed correct at the time of going to press. They do not embody a commitment on the part of Logic Programming Associates (LPA), who may from time to time make changes to the specification of the product, in line with their policy of continual improvement. No part of this manual may be reproduced or transmitted in any form, electronic or mechanical, for any purpose without the prior written agreement of LPA.

Copyright (c) 2004 Logic Programming Associates Ltd. All Rights Reserved.

Logic Programming Associates Ltd
Studio 30
The Royal Victoria Patriotic Building
Trinity Road
London SW18 3SX
England

phone: +44 (0) 20 8871 2016
fax: +44 (0) 20 8874 0449
web site: <http://www.lpa.co.uk>

LPA-PROLOG and **WIN-PROLOG** are trademarks of LPA Ltd., London England.

13 July, 2004

Examples

Contents

WIN-PROLOG Example Files	2
Contents	3
WIN-PROLOG EXAMPLE FILES	5
Introduction.....	5
AUTO.PL - Auto Indent for the Editor	6
BARS.PL – Scroll, Nudge, Strip and Track Bars	7
BENCHMRK.PL - Prolog Benchmark Suite.....	8
CALENDAR.PL - Perpetual Calendar.....	9
CARDS.PL - Interface for CARDS.DLL in WinNT	10
CHANGE.PL – Multiple File Text Edits	11
CLOCK.PL - Digital Clock	12
COMBOBOX.PL - Combobox Example	13
COMMAS.PL - Read a Comma-Separated Record	15
COMMENTS.PL – Read Comments from 386-PROLOG File	16
COMPLETE.PL and COMPDATA.PL – Keyword Completion Example	17
CRC_CHOP.PL - Create CRC32 Identical Files	18
DCG.PL - English Sentences and Expression DCG Examples.....	19
DCG_DRAW.PL - A Tree Drawing Package	20
DIALOG.PL - A Really Simple Dialog Example	21
EASTER.PL – Finding Easter Sunday.....	22
EXPERT.PL – A Natural Language Expert System	23
GFX_TOOL.PL - Graphics Tool Demonstration.....	24
GRAPHICS.PL - A Simple Graphics Demo.....	25
HANOI.PL - The Towers of Hanoi	26
KEYBOARD.PL - Key Interrogation Example	27
LISTBOX.PL - Program to Demonstrate List Boxes	28
LUNAR.PL – Lunar Phases	29
MCI.PL - MCI Functions for WIN-PROLOG.....	30
MEALS.PL – Meal Selection Example	31
MENUS.PL – A Simple Menu Demonstration.....	32
MESSAGES.PL – Listbox Example.....	33

META.PL - Creating a Metafile	34
PIECHART.PL - Pie-Chart Display	35
REVERSI.PL - Reversi Game.....	36
RGB.PL - Return the RGB Colour of a Single Pixel.....	37
ROUND.PL - Numerical Rounding Routines	38
SALESMAN.PL - The Travelling Salesman	39
SEMI.PL - Controlling Backtracking with Semi-Modal Dialogs	40
SHA256.PL - Generate the SHA-256 Constants	41
SHARE.PL - Shared Access Read-Only Files	Error! Bookmark not defined.
SIEVE.PL - Sieve of Erasthatones	42
SIEVE2.PL - Sieve of Erasthatones	43
STIRLING.PL - Stirling Approximation	44
STRINGS.PL - String Examples.....	45
TURTLE.PL - Turtle Graphics	46
INDEX	47

WIN-PROLOG EXAMPLE FILES

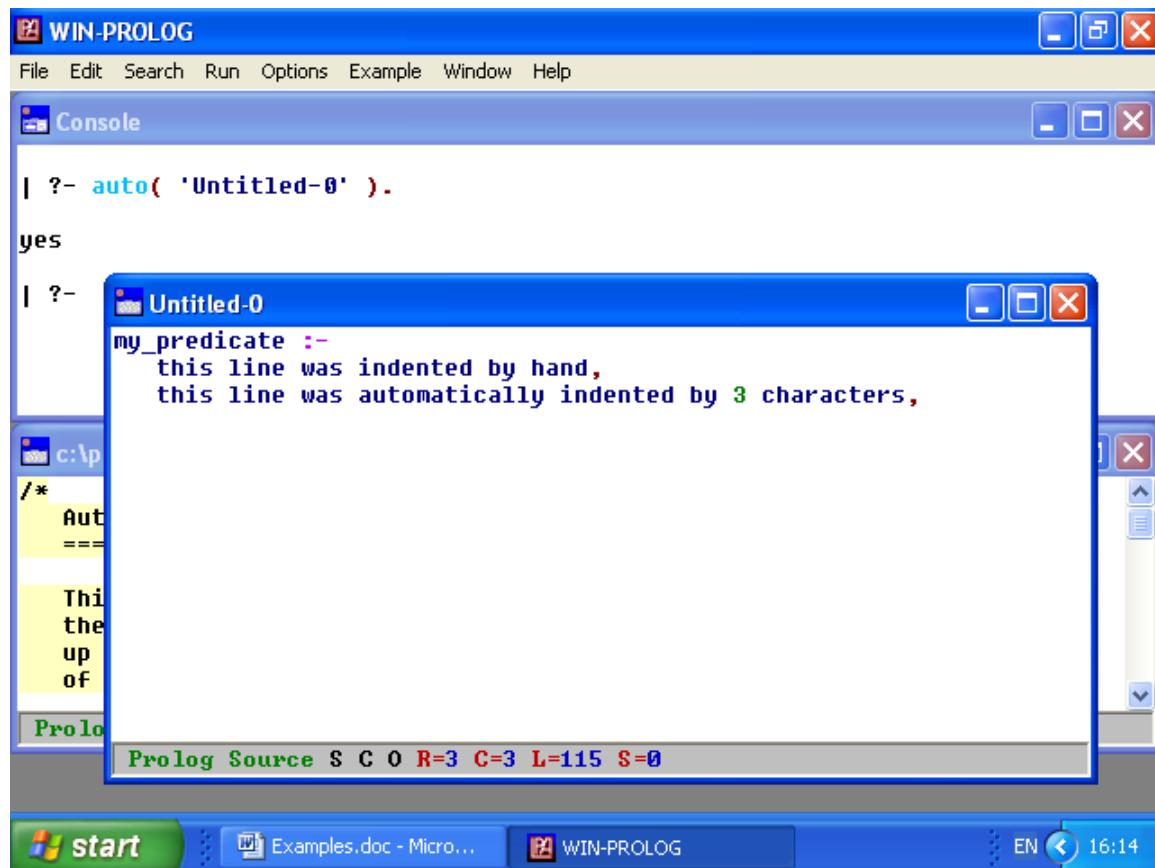
Introduction

The following **WIN-PROLOG** example files are all in the EXAMPLES directory.

Examples

AUTO.PL - Auto Indent for the Editor

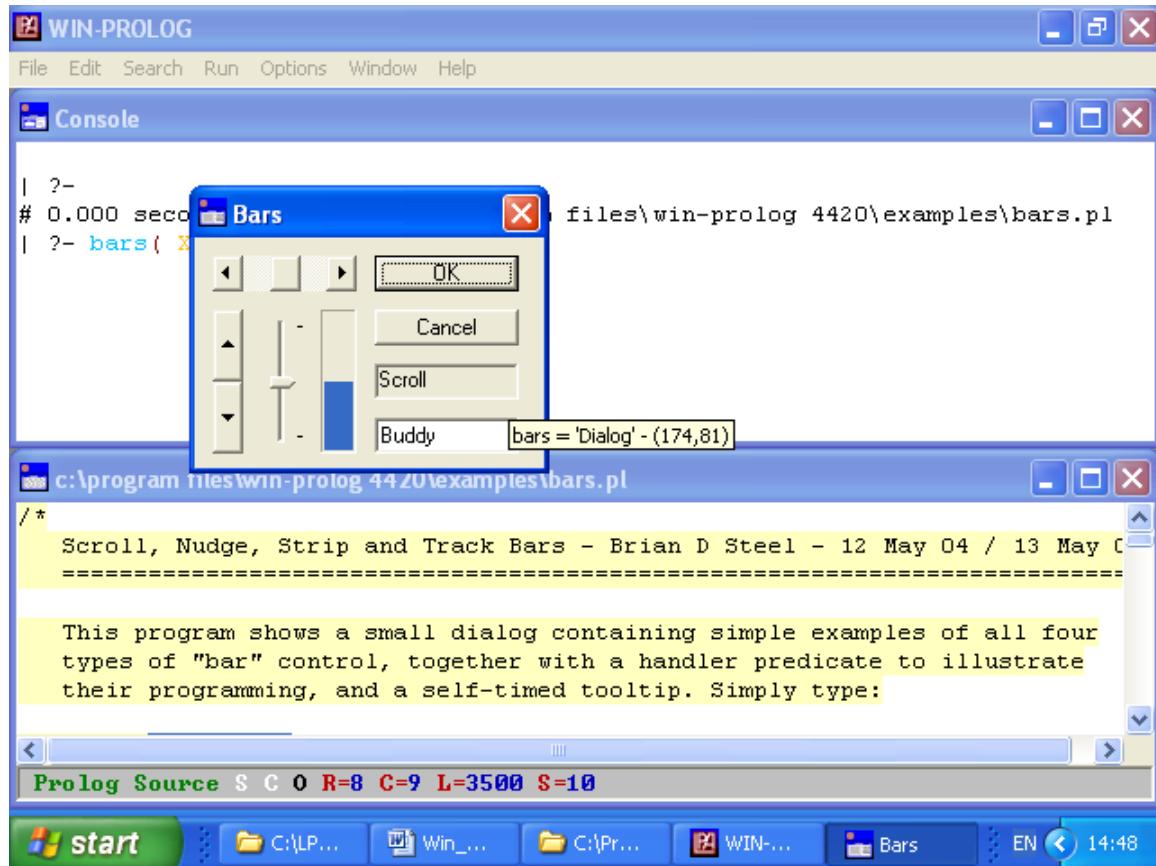
This program provides an "auto indent" feature in the editor. When the <enter> key is pressed, the auto indent is achieved by picking up the previous line, finding all the white space at the beginning of that line, and then inserting this space into the current line.



Examples

BARS.PL – Scroll, Nudge, Strip and Track Bars

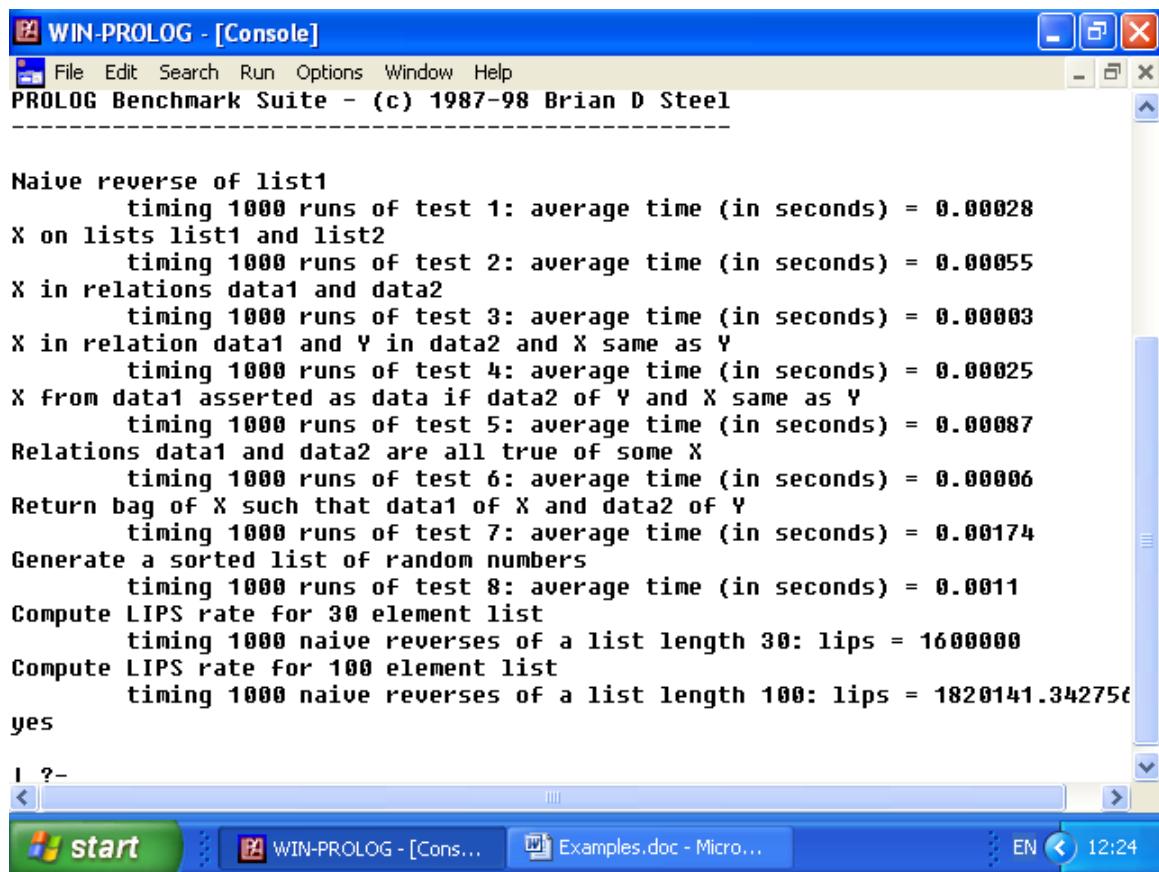
This program shows a small dialog containing simple examples of all four types of "bar" control, together with a handler predicate to illustrate their programming, and a self-timed tooltip.



Examples

BENCHMRK.PL - Prolog Benchmark Suite

This program is a PROLOG Benchmark Suite.



The screenshot shows a Windows-style application window titled "WIN-PROLOG - [Console]". The window contains a menu bar with File, Edit, Search, Run, Options, Window, and Help. Below the menu is a title bar reading "PROLOG Benchmark Suite - (c) 1987-98 Brian D Steel". The main area of the window displays the output of a Prolog benchmark suite. The output consists of several test cases followed by their average execution times in seconds. The tests include operations like reversing lists, finding relations between data1 and data2, generating sorted lists of random numbers, and computing LIPS rates for different list lengths. The text is in a monospaced font, and the overall interface is reminiscent of a classic Windows 9x/ME desktop environment.

```
Naive reverse of list1
    timing 1000 runs of test 1: average time (in seconds) = 0.00028
X on lists list1 and list2
    timing 1000 runs of test 2: average time (in seconds) = 0.00055
X in relations data1 and data2
    timing 1000 runs of test 3: average time (in seconds) = 0.00003
X in relation data1 and Y in data2 and X same as Y
    timing 1000 runs of test 4: average time (in seconds) = 0.00025
X from data1 asserted as data if data2 of Y and X same as Y
    timing 1000 runs of test 5: average time (in seconds) = 0.00087
Relations data1 and data2 are all true of some X
    timing 1000 runs of test 6: average time (in seconds) = 0.00006
Return bag of X such that data1 of X and data2 of Y
    timing 1000 runs of test 7: average time (in seconds) = 0.00174
Generate a sorted list of random numbers
    timing 1000 runs of test 8: average time (in seconds) = 0.0011
Compute LIPS rate for 30 element list
    timing 1000 naive reverses of a list length 30: lips = 1600000
Compute LIPS rate for 100 element list
    timing 1000 naive reverses of a list length 100: lips = 1820141.342756
yes

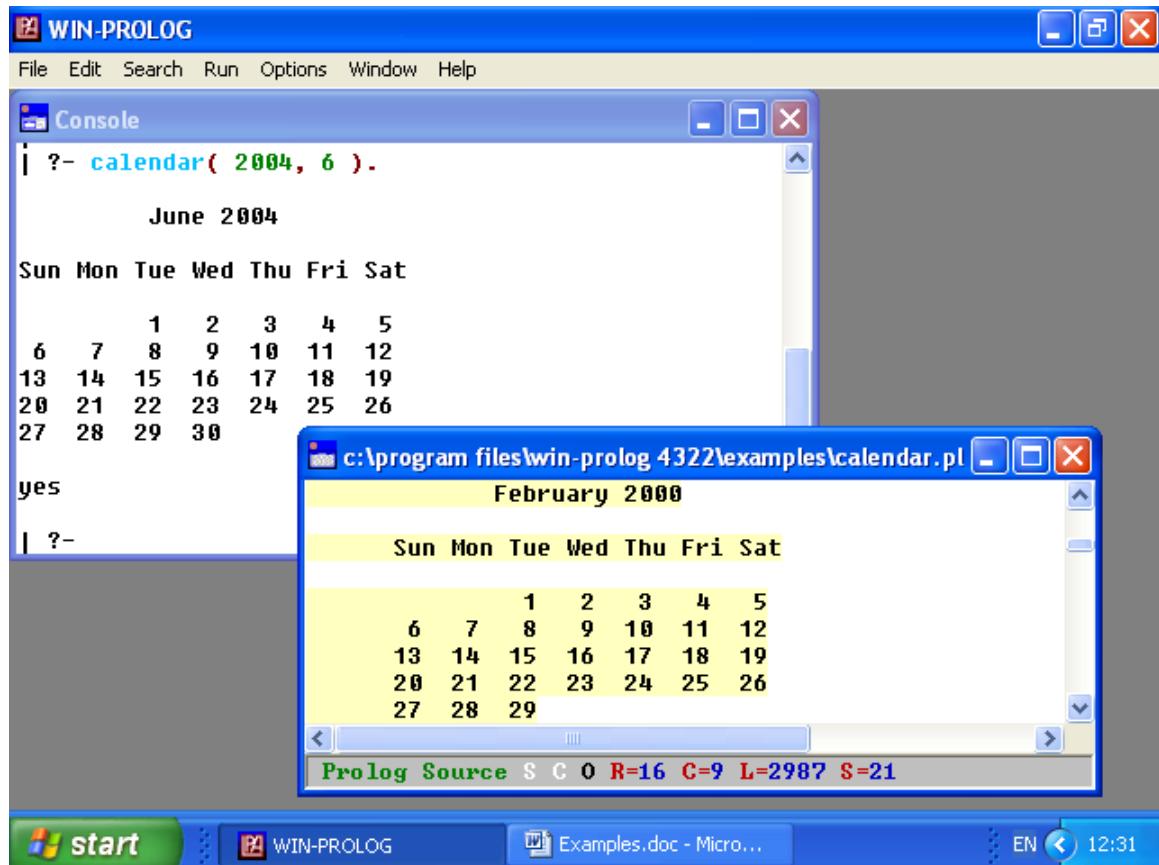
I ?-
```

Examples

CALENDAR.PL - Perpetual Calendar

This simple program illustrates the use of the time/4 predicate for generating day numbers from dates and vice versa, allowing a very simple program to generate correct calendars for any month from January 1600 (1600,1) onwards, according to the Gregorian calendar.

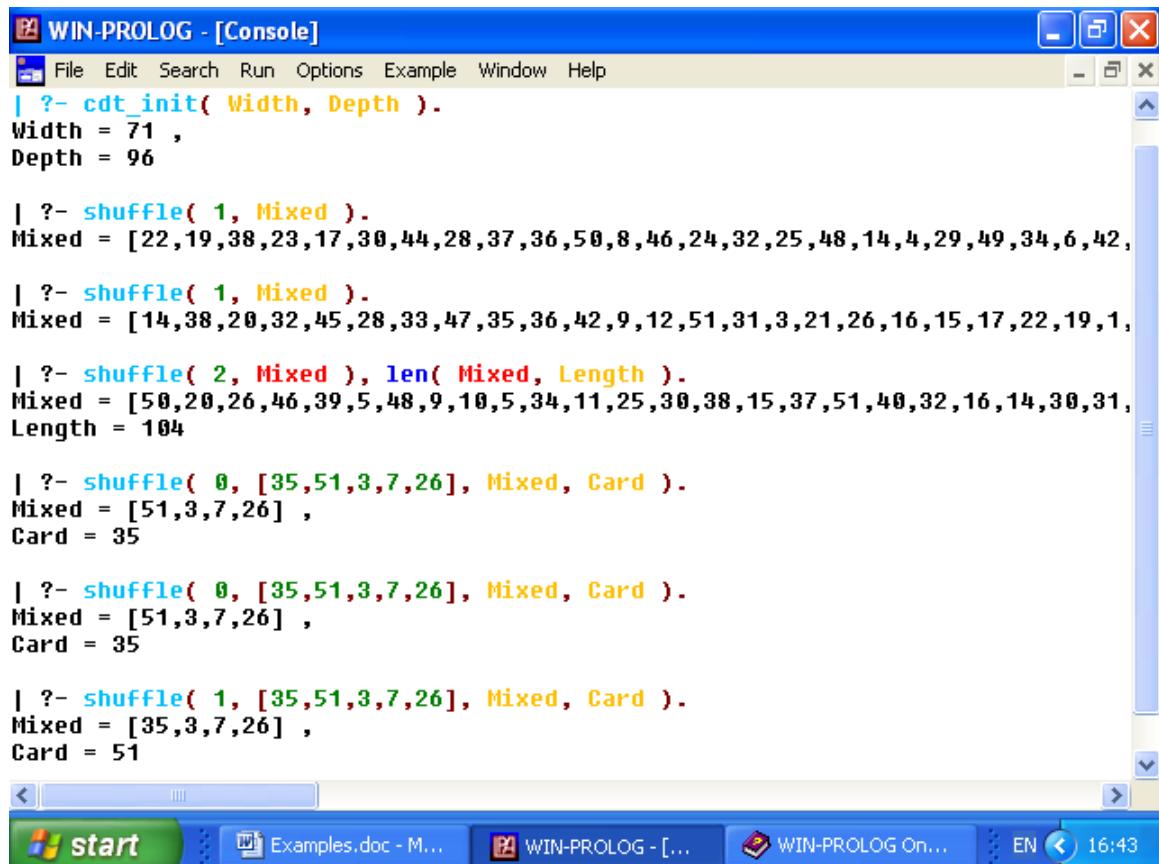
Several simple techniques are illustrated, including the centering of text based on its length, formatted output for displaying numbers in fixed, right-justified columns, and the powerful time/4 predicate itself for day/date calculations.



Examples

CARDS.PL - Interface for CARDS.DLL in WinNT

This interface allows easy direct access to the functions in the dynamic link library, CARDS.DLL, which is supplied with all NT-based versions of Windows.



The screenshot shows the WIN-PROLOG application window running on a Windows operating system. The window title is "WIN-PROLOG - [Console]". The menu bar includes File, Edit, Search, Run, Options, Example, Window, and Help. The main area displays the following Prolog code and its execution results:

```
| ?- cdt_init( Width, Depth ).  
Width = 71 ,  
Depth = 96  
  
| ?- shuffle( 1, Mixed ).  
Mixed = [22,19,38,23,17,30,44,28,37,36,50,8,46,24,32,25,48,14,4,29,49,34,6,42,  
| ?- shuffle( 1, Mixed ).  
Mixed = [14,38,20,32,45,28,33,47,35,36,42,9,12,51,31,3,21,26,16,15,17,22,19,1,  
| ?- shuffle( 2, Mixed ), len( Mixed, Length ).  
Mixed = [50,20,26,46,39,5,48,9,10,5,34,11,25,30,38,15,37,51,40,32,16,14,30,31,  
Length = 104  
  
| ?- shuffle( 0, [35,51,3,7,26], Mixed, Card ).  
Mixed = [51,3,7,26] ,  
Card = 35  
  
| ?- shuffle( 0, [35,51,3,7,26], Mixed, Card ).  
Mixed = [51,3,7,26] ,  
Card = 35  
  
| ?- shuffle( 1, [35,51,3,7,26], Mixed, Card ).  
Mixed = [35,3,7,26] ,  
Card = 51
```

The taskbar at the bottom shows the "start" button, "Examples.doc - M...", "WIN-PROLOG - [...]", "WIN-PROLOG On...", "EN", and the time "16:43".

Examples

CHANGE.PL – Multiple File Text Edits

This program uses the new memory-mapped files and find/3 predicate to implement an intelligent global file edit command:

change(Ptn, Olds, News, Mode).

where:

Ptn = an atom specifying a file search pattern (as used by dir/3)

Olds = a string with the search text (to be replaced)

News = a string with the replacement text

Mode = an integer in the range 0..2 to given search/replace behaviour

The "Mode" flag can have three settings:

0 - perform an exact case (case-sensitive) search, and replace with exact given string

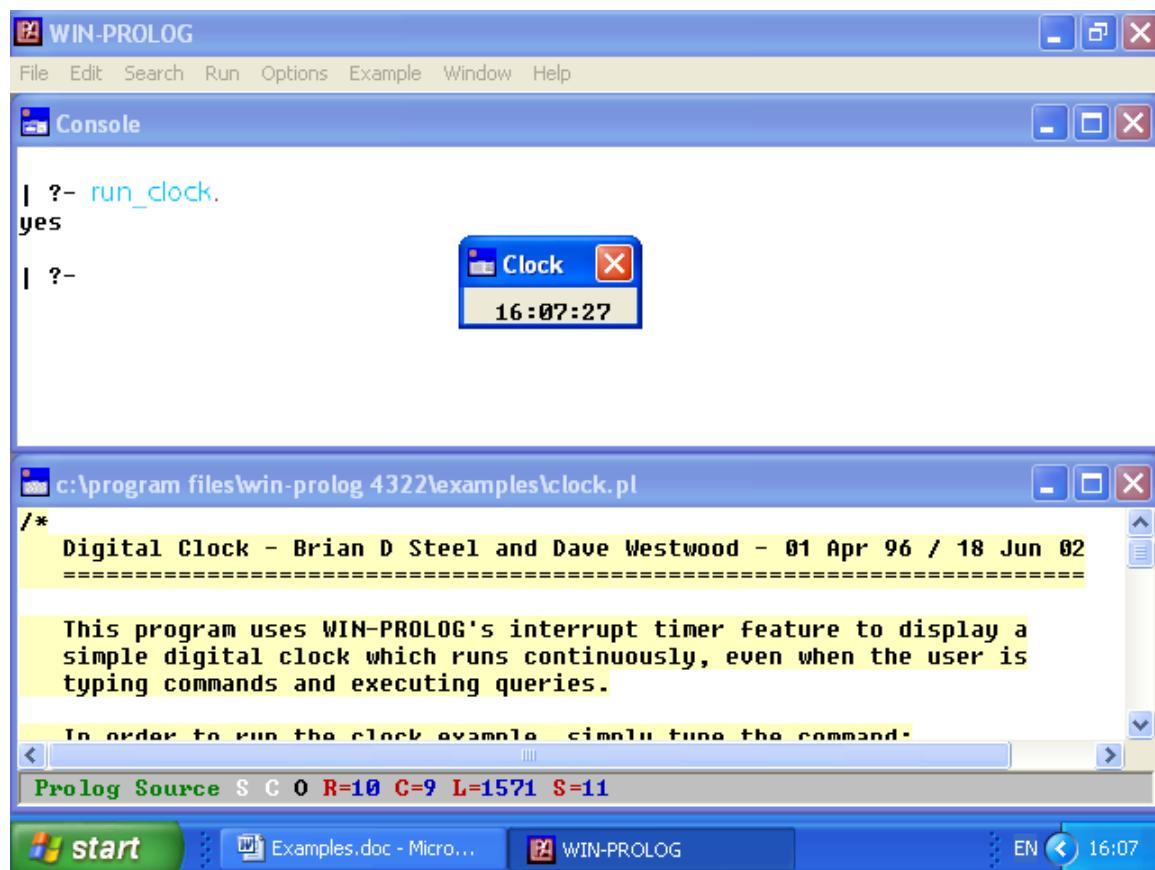
1 - perform a soft case (case-insensitive) search, and replace with case-synchronised version of given string

2 - perform a soft case (case-insensitive) search, and replace with exact given string (no case synchronisation)

Examples

CLOCK.PL - Digital Clock

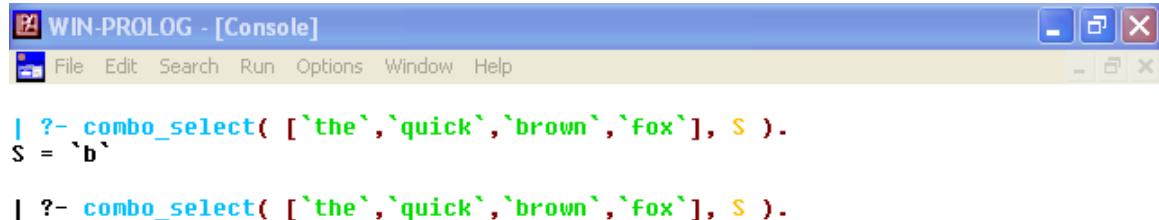
This program uses WIN-PROLOG's interrupt timer feature to display a simple digital clock which runs continuously, even when the user is typing commands and executing queries.



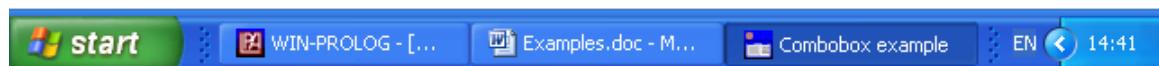
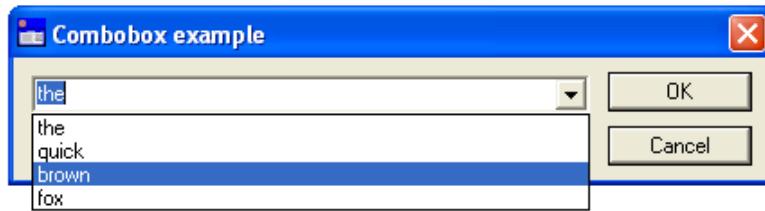
Examples

COMBOBOX.PL - Combobox Example

This example demonstrates the use of the windows messaging predicate `sndmsg/5` to control a combobox. A list of strings is given to the program, and these are displayed in a combobox. When the user selects one, it is returned by the program.



```
| ?- combo_select( ['the','quick','brown','fox'], S ).  
S = 'b'  
| ?- combo_select( ['the','quick','brown','fox'], S ).
```



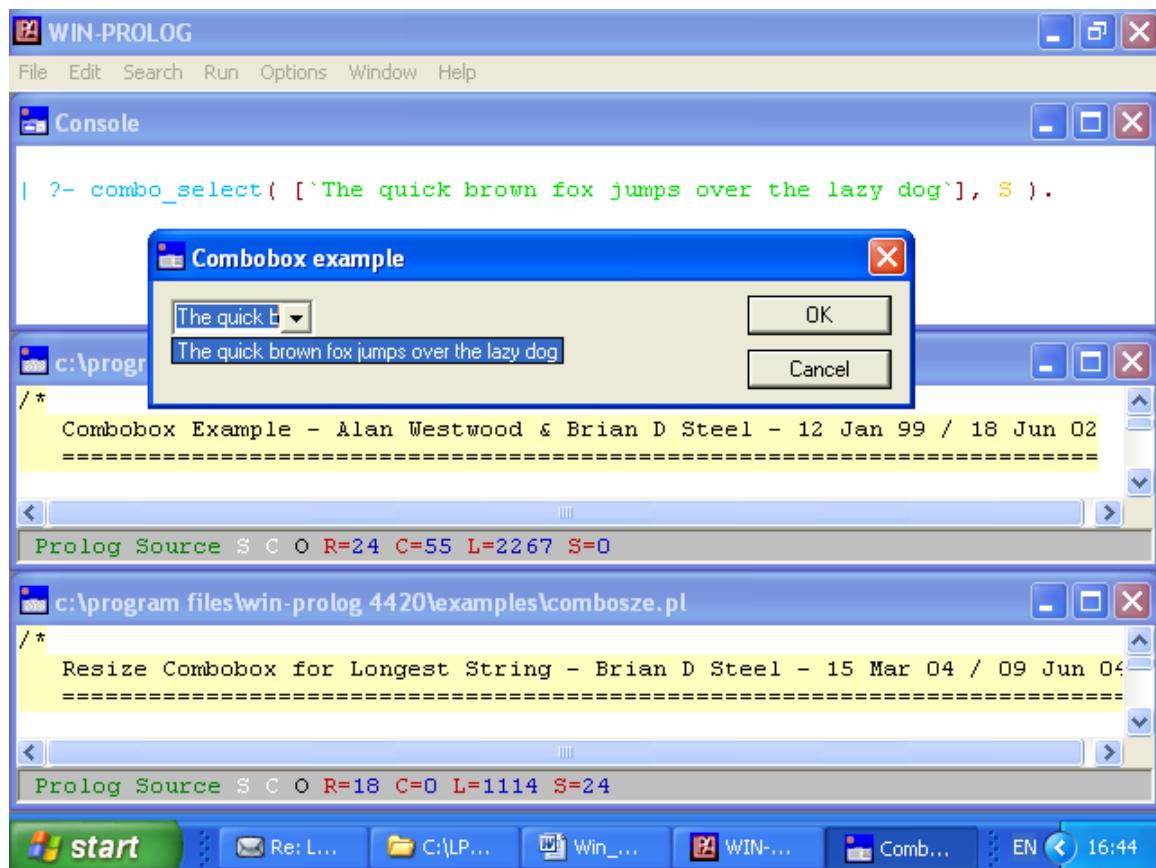
Examples

COMBOSZE.PL – Resize Combobox for Longest String

This program resizes the drop-down portion of a combobox that has the CBS_DROPDOWN or CBS_DROPDOWNLIST style, so that the list is wide enough to display its longest entry.

This program could, for example, be combined with COMBOBOX.PL:

```
combo_select( Items, Selected ) :-  
    ...  
    wccreate( (combo,500), combobox, `Combo`, 10, 10, 80, 150, Cstyle ),  
    ...  
    window_handler( combo, combo_handler ),  
    wcmbssze( (combo,500), _ ),  
    call_dialog( combo, Selected ),  
    ...
```



Examples

COMMAS.PL - Read a Comma-Separated Record

This simple program uses the advanced input/output capabilities of strings and the find/3 predicate to return a list of Prolog strings representing the tokens in a single record in a comma-separated file that has been opened for input.

The screenshot shows the WIN-PROLOG application window with three main panes:

- Console pane:** Displays the Prolog query `?- see('test.txt').`, followed by the response `yes`. Below this, two queries using the `read_record/1` predicate are shown, each returning a list of strings:
 - `?- read_record(R).` Returns `R = [`one`, `two`, `three`, `four`, `5.0`]`.
 - `?- read_record(R).` Returns `R = [`6`, `seven`, `eight`, `neuf`, `dix`]`.
- Output pane:** Shows the source code of the `commas.pl` file. The code includes a header comment:

```
/*  
 * Read a Comma-Separated Record - (c) Brian D Steel - 02 Sep 98 / 10 Feb 99  
 * =====
```

and a descriptive text block:

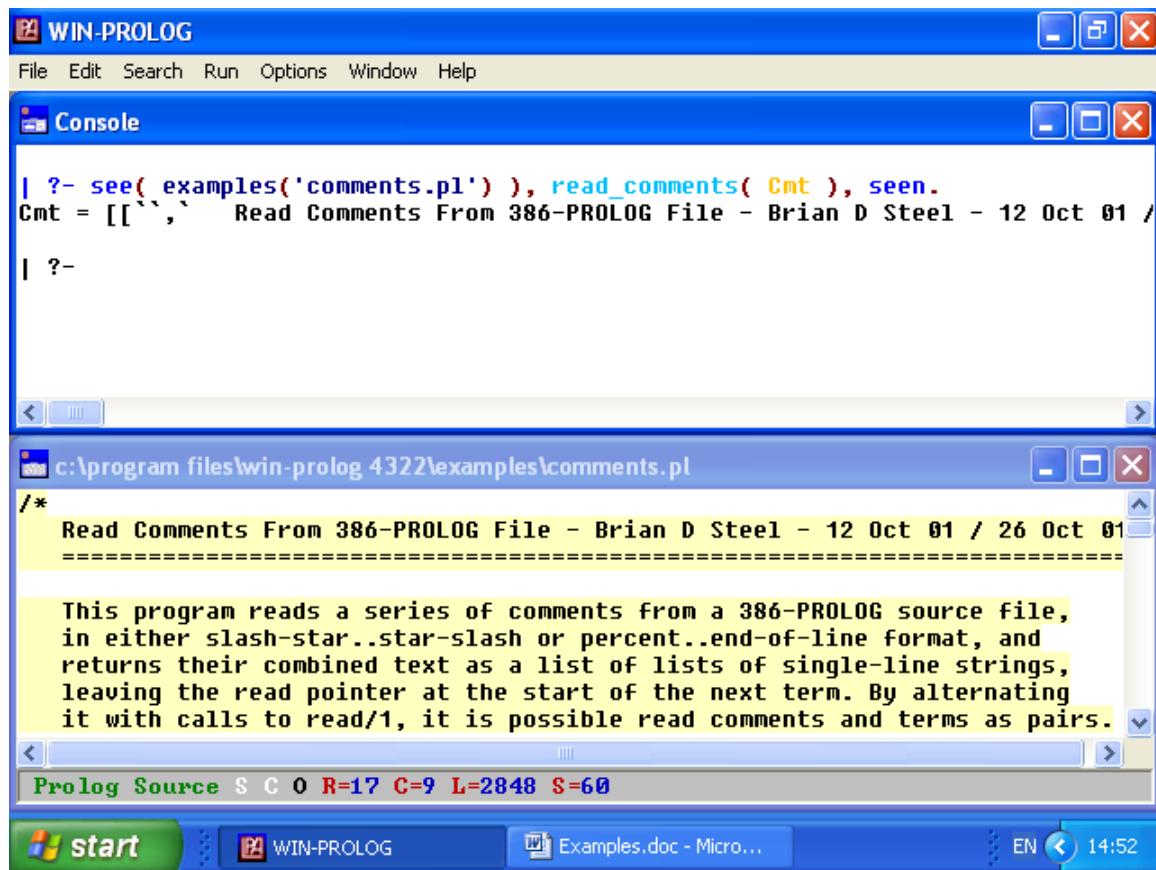
```
This simple program uses the advanced input/output capabilities of  
strings and the find/3 predicate to return a list of Prolog strings  
representing the tokens in a single record in a comma-separated file  
that has been opened for input.
```
- Status bar:** At the bottom, it displays the status `Prolog Source S C O R=0 C=2 L=2130 S=0`.

Examples

COMMENTS.PL – Read Comments from 386-PROLOG File

This program reads a series of comments from a 386-PROLOG source file, in either slash-star..star-slash or percent..end-of-line format, and returns their combined text as a list of lists of single-line strings, leaving the read pointer at the start of the next term. By alternating it with calls to read/1, it is possible read comments and terms as pairs.

Each slash-star..star-slash comment is treated as being a self-contained comment, irrespective of how many lines it covers, while a sequence of percent..end-of-line comments on consecutive lines of input are treated as comprising a single multiline comment, and are combined accordingly.



The screenshot shows the WIN-PROLOG application window. It consists of three main panes:

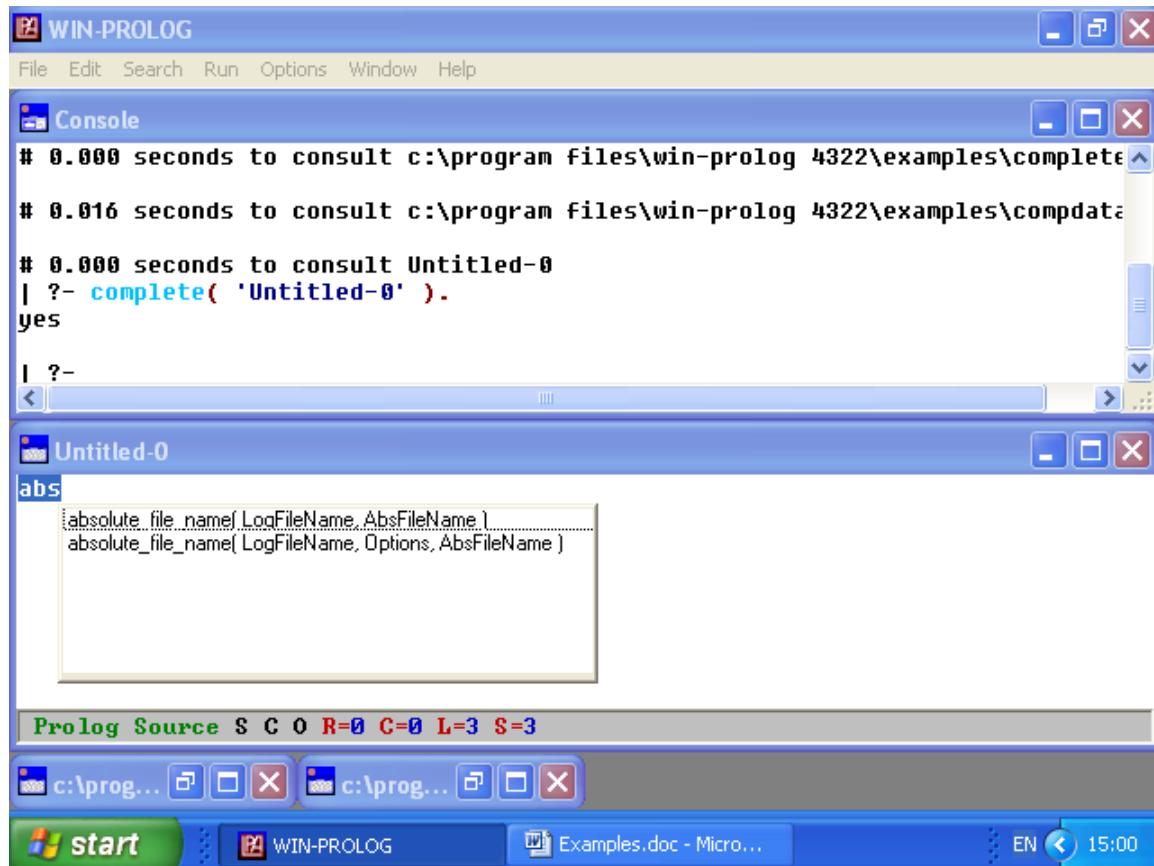
- Top Pane (WIN-PROLOG):** A menu bar with File, Edit, Search, Run, Options, Window, Help. Below it is a toolbar with standard window controls (minimize, maximize, close).
- Middle Pane (Console):** A text area titled "Console" containing the Prolog code and its output. The output shows the execution of `see('comments.pl')`, `read_comments(Cnt)`, and `seen`. It also displays the value of `Cmt` as a list of lists of strings, followed by a prompt `| ?-`.
- Bottom Pane (Editor):** A text editor window titled "c:\program files\win-prolog 4322\examples\comments.pl". It contains the source code for the `read_comments/1` predicate, which includes a detailed explanatory comment about reading comments from a file.

The status bar at the bottom shows "Prolog Source S C O R=17 C=9 L=2848 S=60".

Examples

COMPLETE.PL and COMPDATA.PL – Keyword Completion Example

This program implements keyword completion. The word immediately before the cursor in an edit window is "completed" by referring to a database of known words.



Examples

CRC_CHOP.PL - Create CRC32 Identical Files

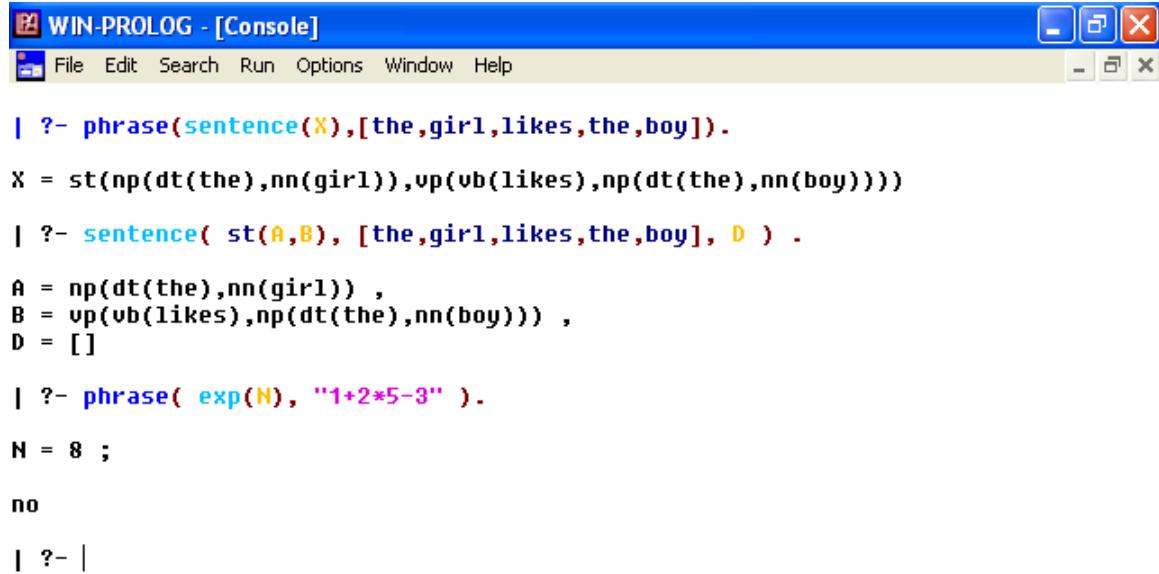
This program demonstrates both a strength and a weakness of the CRC32 algorithm, as implemented in the `crc/3` predicate, in that it is relatively easy to create any number of different files with exactly the same CRC32.

The strength of this feature is in that any arbitrary data file can be "tagged" with a modified CRC32, so that the resulting file generates a predetermined CRC32 value, making it self-verifiable; the weakness is that any malicious attack on the file can be concealed hidden simply by recomputing and replacing the CRC32 tag.

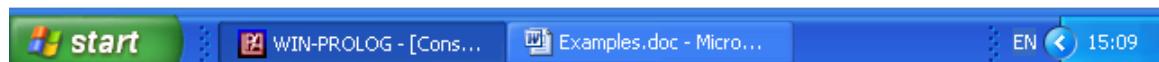
Examples

DCG.PL - English Sentences and Expression DCG Examples

This file contains two examples of using Definite Clause Grammars.



```
| ?- phrase(sentence(X),[the,girl,likes,the,boy]).  
X = st(np(dt(the),nn(girl)),vp(vb(likes),np(dt(the),nn(boy))))  
| ?- sentence( st(A,B), [the,girl,likes,the,boy], D ) .  
A = np(dt(the),nn(girl)) ,  
B = vp(vb(likes),np(dt(the),nn(boy))) ,  
D = []  
| ?- phrase( exp(N), "1+2*5-3" ).  
N = 8 ;  
no  
| ?-
```



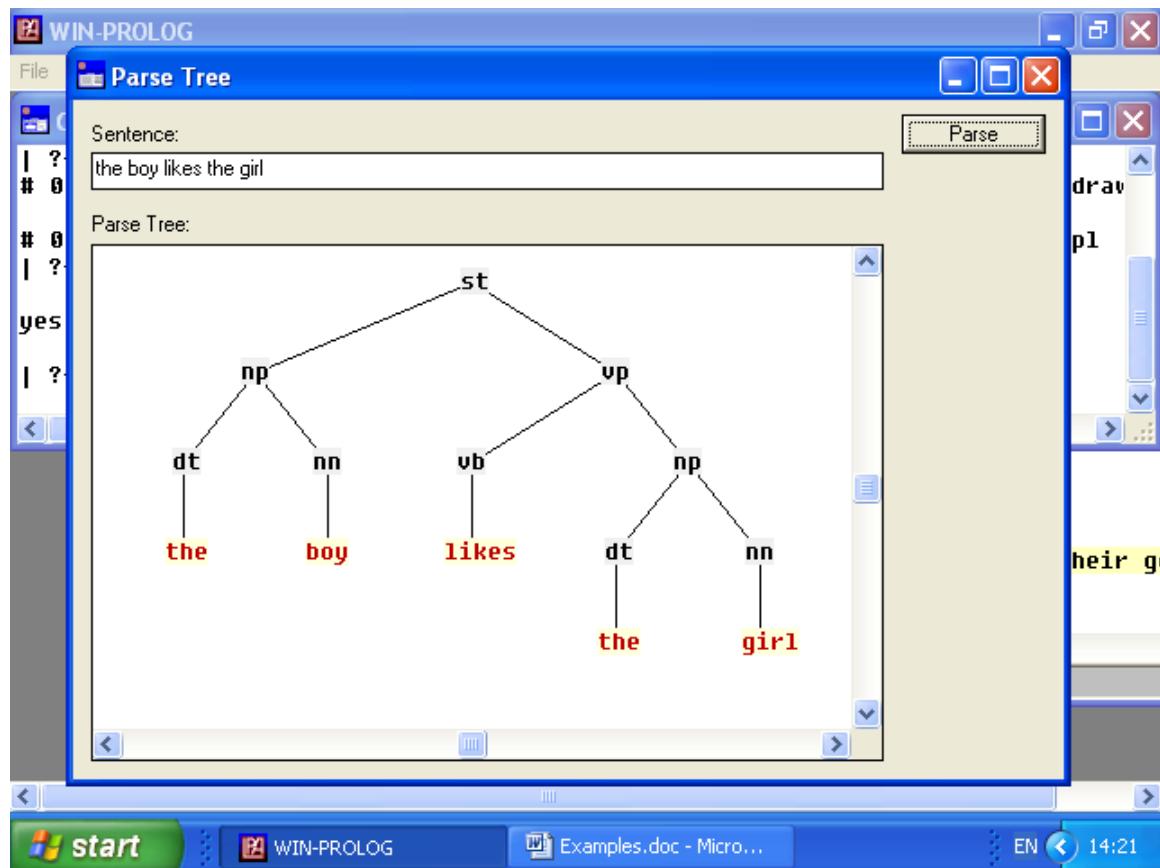
Examples

DCG_DRAW.PL - A Tree Drawing Package

This program will draw appropriate parse trees given a Definite Clause Grammar. The base of the grammar is assumed to be sentence(Tree), as defined in the file, DCG.PL, in the \EXAMPLES directory:

```
sentence( Tree ) -->
```

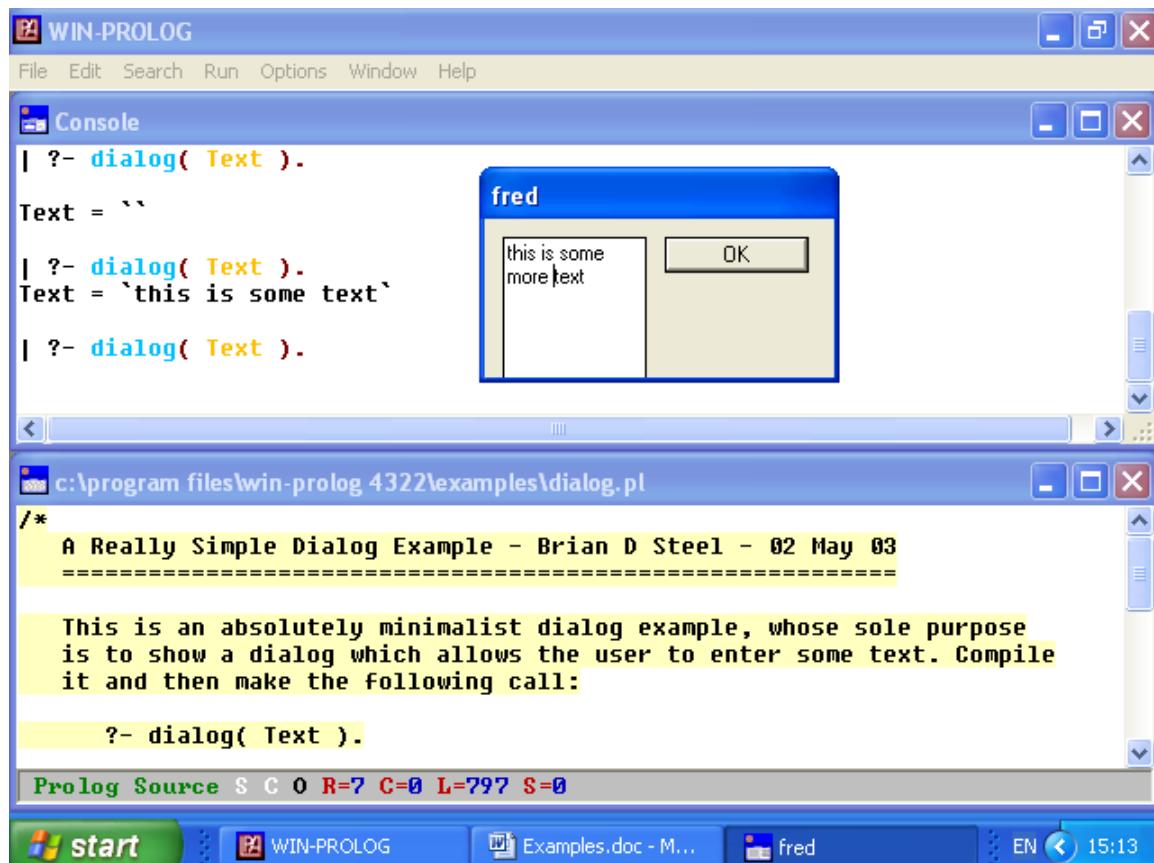
```
...
```



Examples

DIALOG.PL - A Really Simple Dialog Example

This is an absolutely minimalist dialog example, whose sole purpose is to show a dialog which allows the user to enter some text.



Examples

EASTER.PL – Finding Easter Sunday

Given a year number in the Gregorian calendar, this program uses the ‘official’ algorithm for computing the date of Easter in that year.

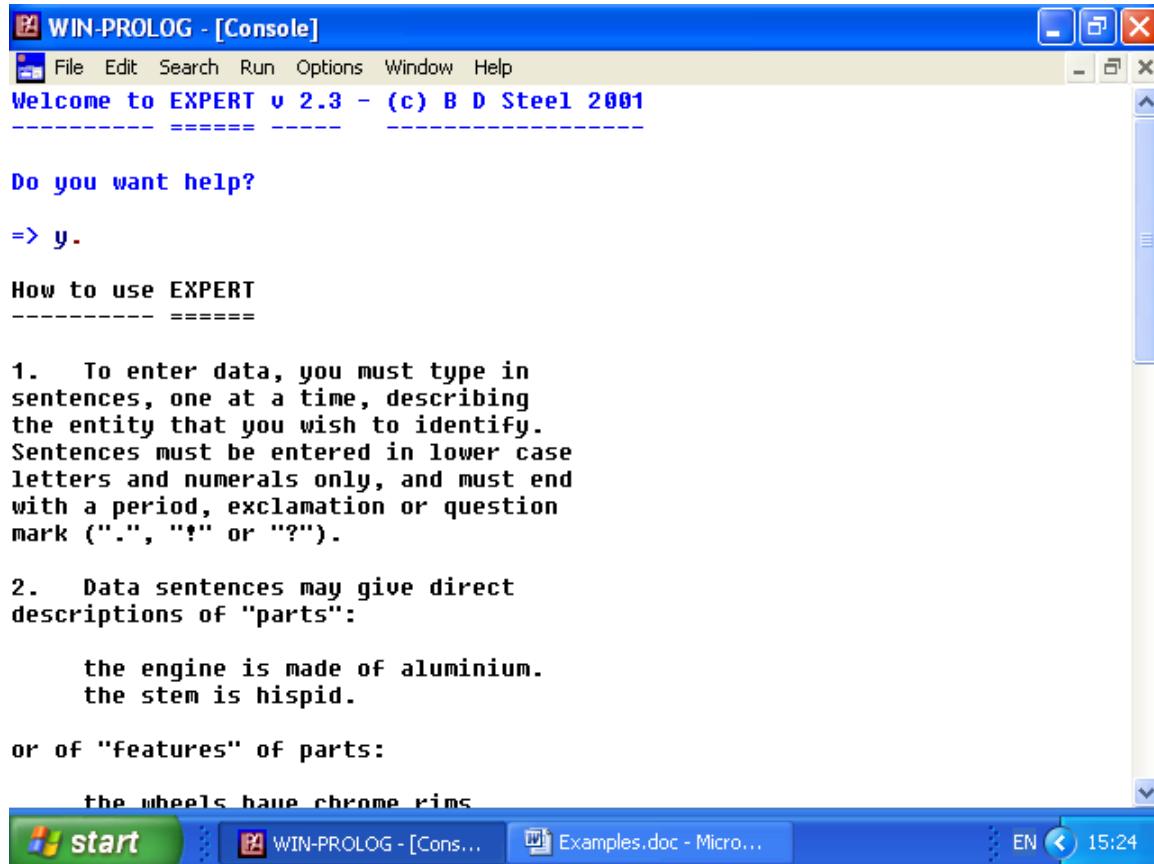


```
WIN-PROLOG - [Console]
File Edit Search Run Options Window Help
?- easter( 2004, M, D ).  
M = 4 ,  
D = 11  
  
?- easter( 2005, M, D ).  
M = 3 ,  
D = 27  
  
?- easter( 2006, M, D ).  
M = 4 ,  
D = 16  
  
?- easter( 2007, M, D ).  
M = 4 ,  
D = 8  
  
?- easter( 2008, M, D ).  
M = 3 ,  
D = 23  
  
?- easter( 2009, M, D ).  
M = 4 ,  
D = 12  
  
?-
```

Examples

EXPERT.PL – A Natural Language Expert System

A good example of a simple natural language expert system.



The screenshot shows a Windows application window titled "WIN-PROLOG - [Console]". The menu bar includes File, Edit, Search, Run, Options, Window, and Help. The title bar displays "Welcome to EXPERT v 2.3 - (c) B D Steel 2001". The main window contains the following text:

```
Do you want help?  
=> y.  
  
How to use EXPERT  
-----  
  
1. To enter data, you must type in sentences, one at a time, describing the entity that you wish to identify. Sentences must be entered in lower case letters and numerals only, and must end with a period, exclamation or question mark (".", "!" or "?").  
  
2. Data sentences may give direct descriptions of "parts":  
  
the engine is made of aluminium.  
the stem is hispid.  
  
or of "features" of parts:  
  
the wheels have chrome rims
```

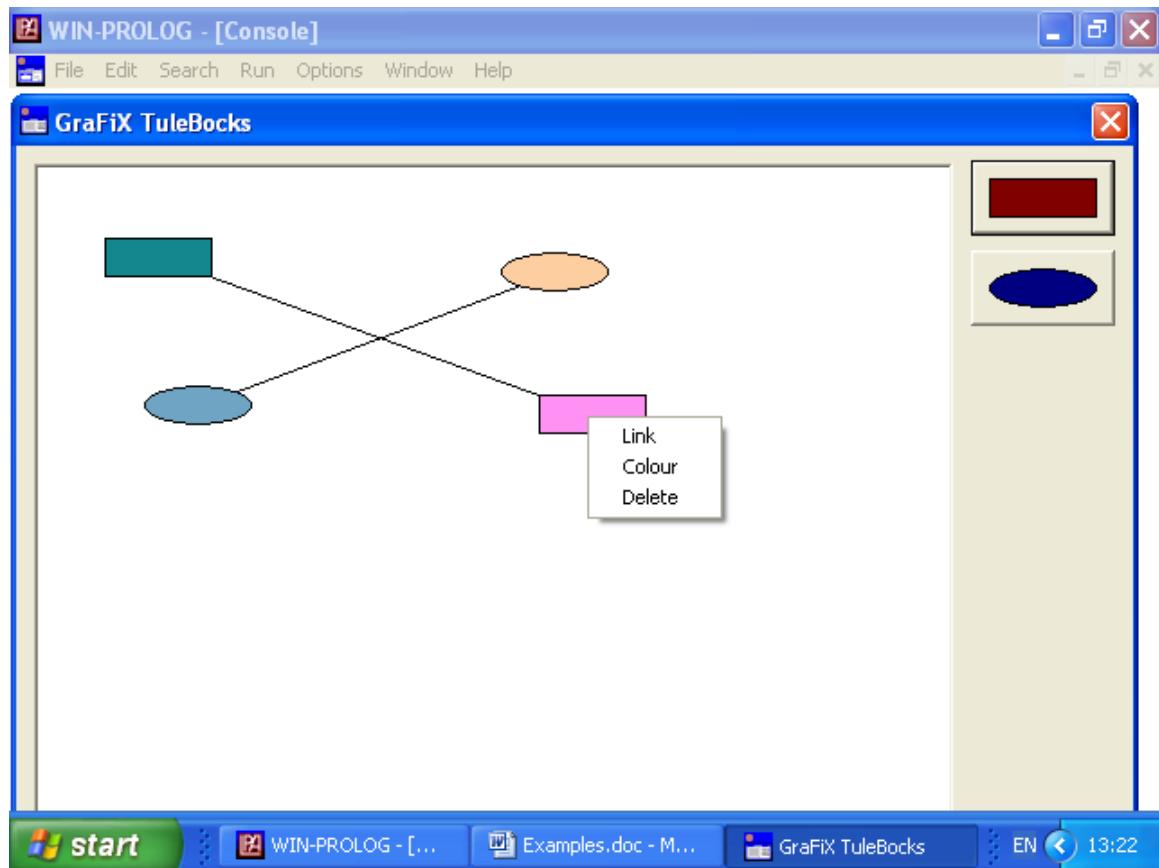
The taskbar at the bottom shows the "start" button, the application icon, and the file "Examples.doc - Micro...". The system tray indicates the language is set to EN and the time is 15:24.

Examples

GFX_TOOL.PL - Graphics Tool Demonstration

The purpose of this program is to show how to write a simple graphics tool, in which shapes can be created, moved, modified and linked.

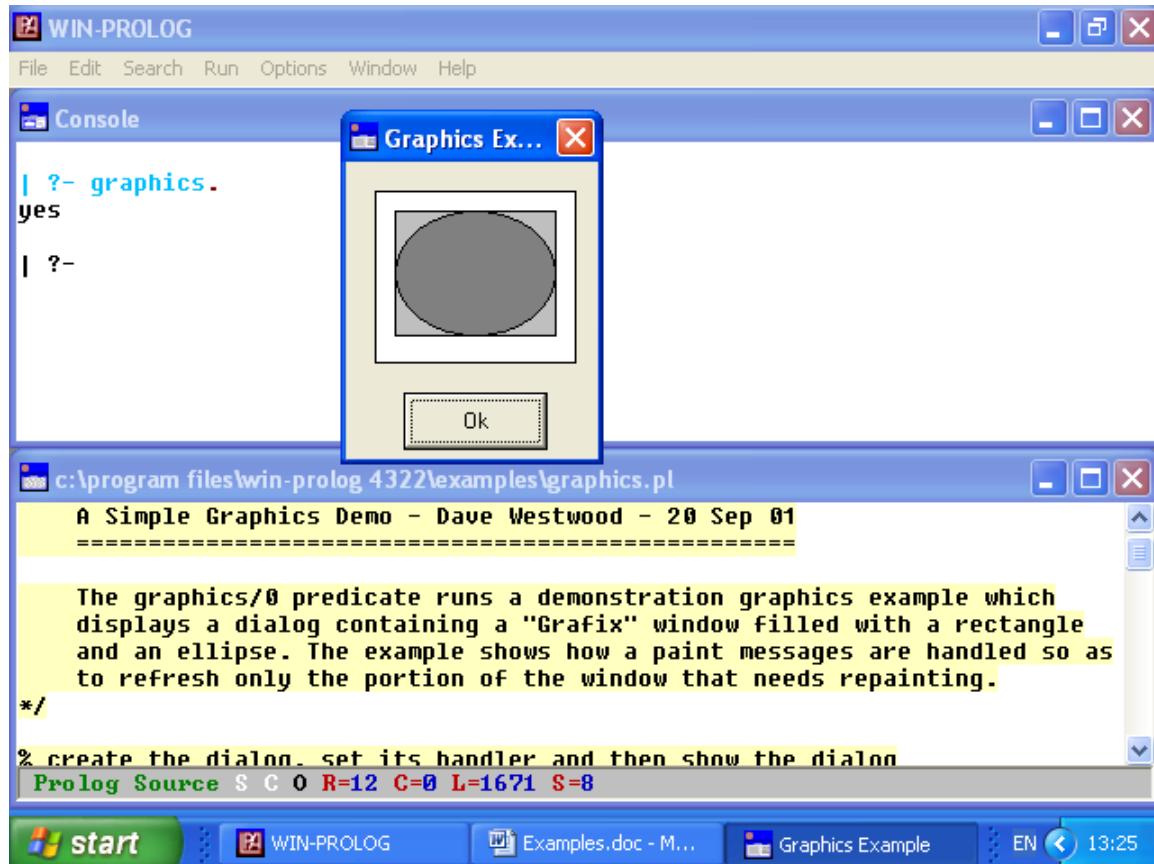
This program displays a simple dialog with two "tool" buttons, one to draw ellipses and the other to draw rectangles. Once clicked, the mouse can then be used to place these in a work area. Once placed, the objects can be recoloured, dragged and/or linked to other objects to create a simple network diagram using left and right clicks.



Examples

GRAPHICS.PL - A Simple Graphics Demo

The graphics/0 predicate runs a demonstration graphics example which displays a dialog containing a "Grafix" window filled with a rectangle and an ellipse. The example shows how paint messages are handled so as to refresh only the portion of the window that needs repainting.



Examples

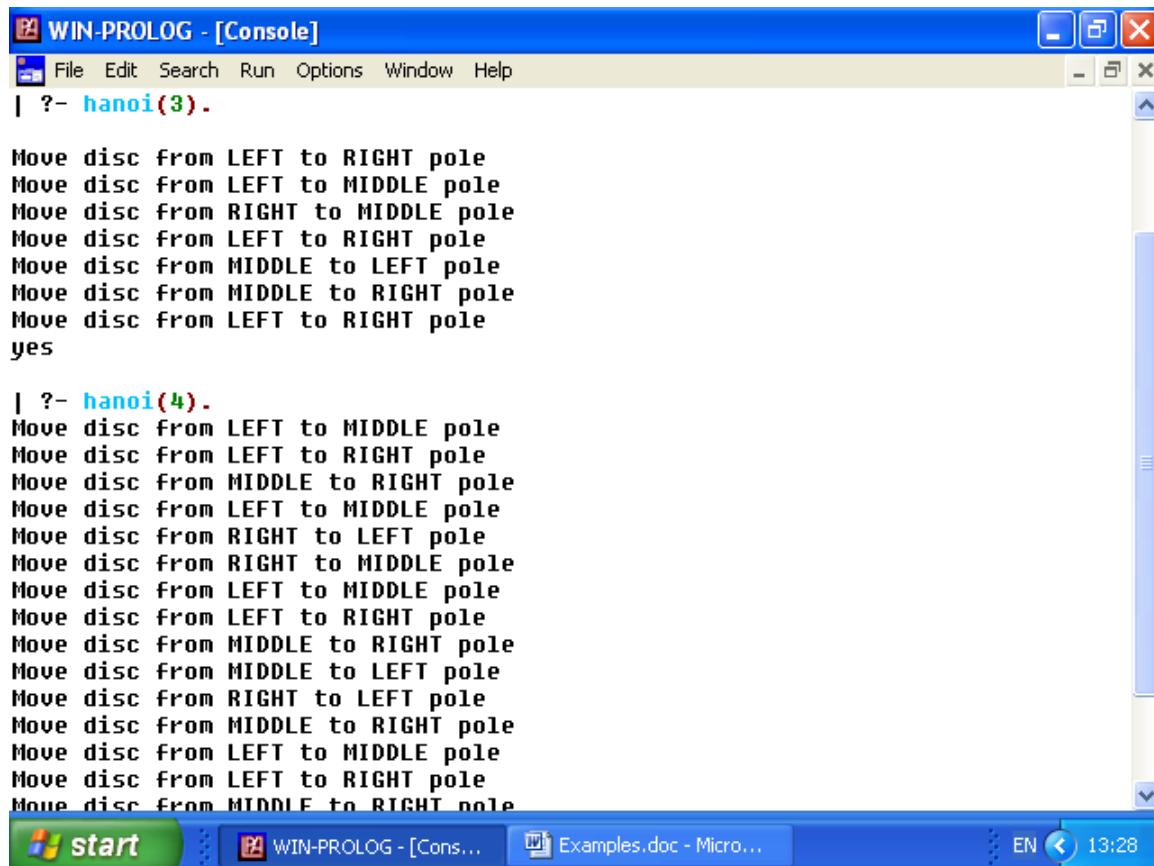
HANOI.PL - The Towers of Hanoi

In this classic children's puzzle, a pile of discs is stored on the left three poles, with each disc being smaller than the one beneath it.

The object of the game is to transfer the whole pile to the right hand pole, one disc at a time, with the help of the middle pole, but at no stage may a larger disc be placed on a smaller one.

The idea is simple. Suppose you have a pile of discs on the LEFT pole, and want to transfer them legally to the RIGHT pole. If you can somehow transfer all but the very bottom disc to the MIDDLE pole, then all you need to do is move the remaining disc from LEFT to RIGHT, and then somehow transfer the other discs from MIDDLE to RIGHT. The same is true of any pile of discs on any one pole that you want on another pole: transfer all but the last disc to a spare pole, move the last one directly to your chosen target, and then transfer everything from the spare pole to your target.

This is a classic candidate for recursion: figure out how to do one step, as described above, and then trust a recursive call to handle the rest.



A screenshot of a Windows desktop environment. In the center is a window titled "WIN-PROLOG - [Console]". The window has a menu bar with "File", "Edit", "Search", "Run", "Options", "Window", and "Help". Below the menu is a command line input field containing "?- hanoi(3).". The main body of the window shows the output of the Prolog query, which consists of a series of moves: "Move disc from LEFT to RIGHT pole", "Move disc from LEFT to MIDDLE pole", "Move disc from RIGHT to MIDDLE pole", "Move disc from LEFT to RIGHT pole", "Move disc from MIDDLE to LEFT pole", "Move disc from MIDDLE to RIGHT pole", "Move disc from LEFT to RIGHT pole", followed by "yes". Below this, another query "?- hanoi(4)." is shown, followed by a long list of moves for 4 discs, including many more "Move disc from ... to ..." commands and intermediate states like "Move disc from MIDDLE to MIDDLE pole". At the bottom of the screen, the taskbar is visible with icons for "start", "WIN-PROLOG - [Cons...]", "Examples.doc - Micro...", and system status indicators like "EN", "13:28", and a battery icon.

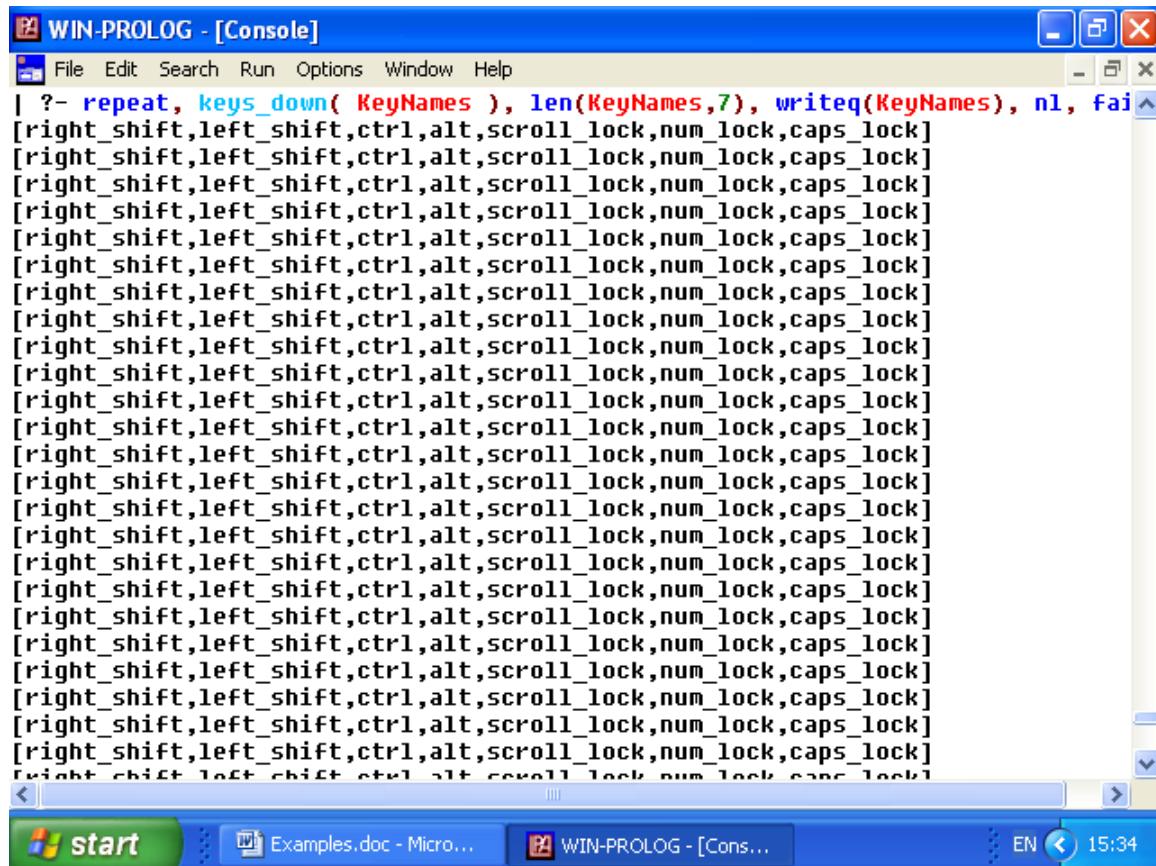
```
?- hanoi(3).
Move disc from LEFT to RIGHT pole
Move disc from LEFT to MIDDLE pole
Move disc from RIGHT to MIDDLE pole
Move disc from LEFT to RIGHT pole
Move disc from MIDDLE to LEFT pole
Move disc from MIDDLE to RIGHT pole
Move disc from LEFT to RIGHT pole
yes

?- hanoi(4).
Move disc from LEFT to MIDDLE pole
Move disc from LEFT to RIGHT pole
Move disc from MIDDLE to RIGHT pole
Move disc from LEFT to MIDDLE pole
Move disc from RIGHT to LEFT pole
Move disc from RIGHT to MIDDLE pole
Move disc from LEFT to MIDDLE pole
Move disc from LEFT to RIGHT pole
Move disc from MIDDLE to RIGHT pole
Move disc from MIDDLE to LEFT pole
Move disc from RIGHT to LEFT pole
Move disc from MIDDLE to RIGHT pole
Move disc from LEFT to MIDDLE pole
Move disc from LEFT to RIGHT pole
Move disc from MIDDLE to RIGHT pole
```

Examples

KEYBOARD.PL - Key Interrogation Example

This example shows the use of the keys/1 predicate and the '`\`' (and) bitwise operator available with is/2, to get the status of the system keys.

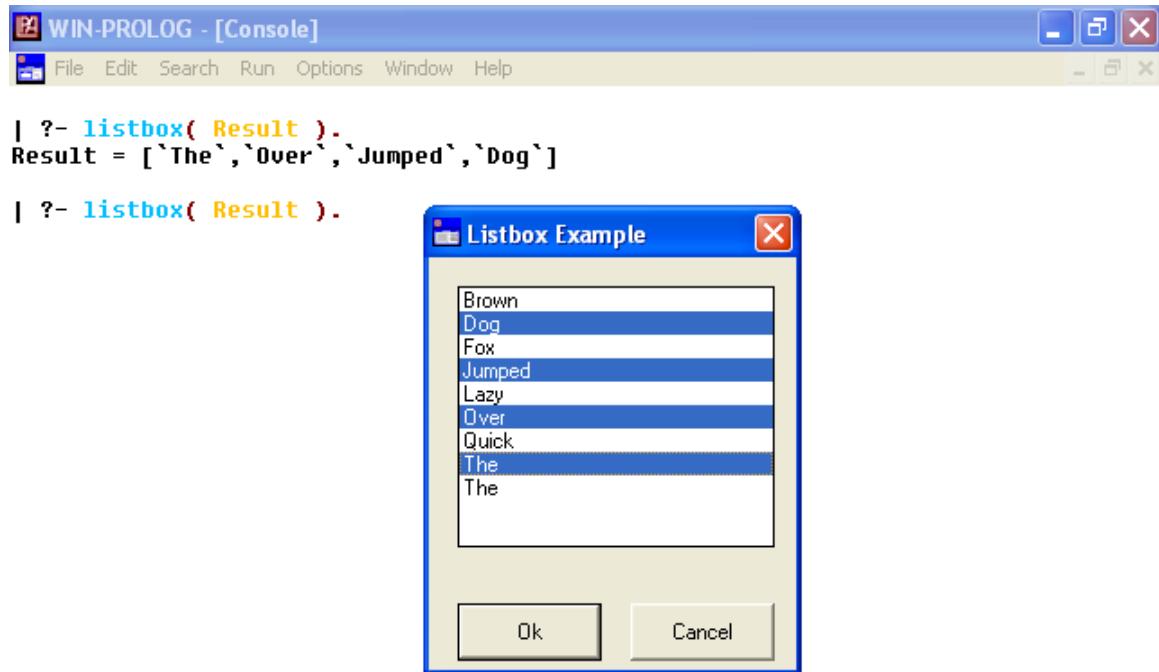


```
| ?- repeat, keys_down( KeyNames ), len(KeyNames,7), writeq(KeyNames), nl, fail
[right_shift,left_shift,ctrl,alt,scroll_lock,num_lock,caps_lock]
```

Examples

LISTBOX.PL - Program to Demonstrate List Boxes

The listbox/1 predicate runs a demonstration listbox example, that returns the selections in the listbox when the dialog is finished.

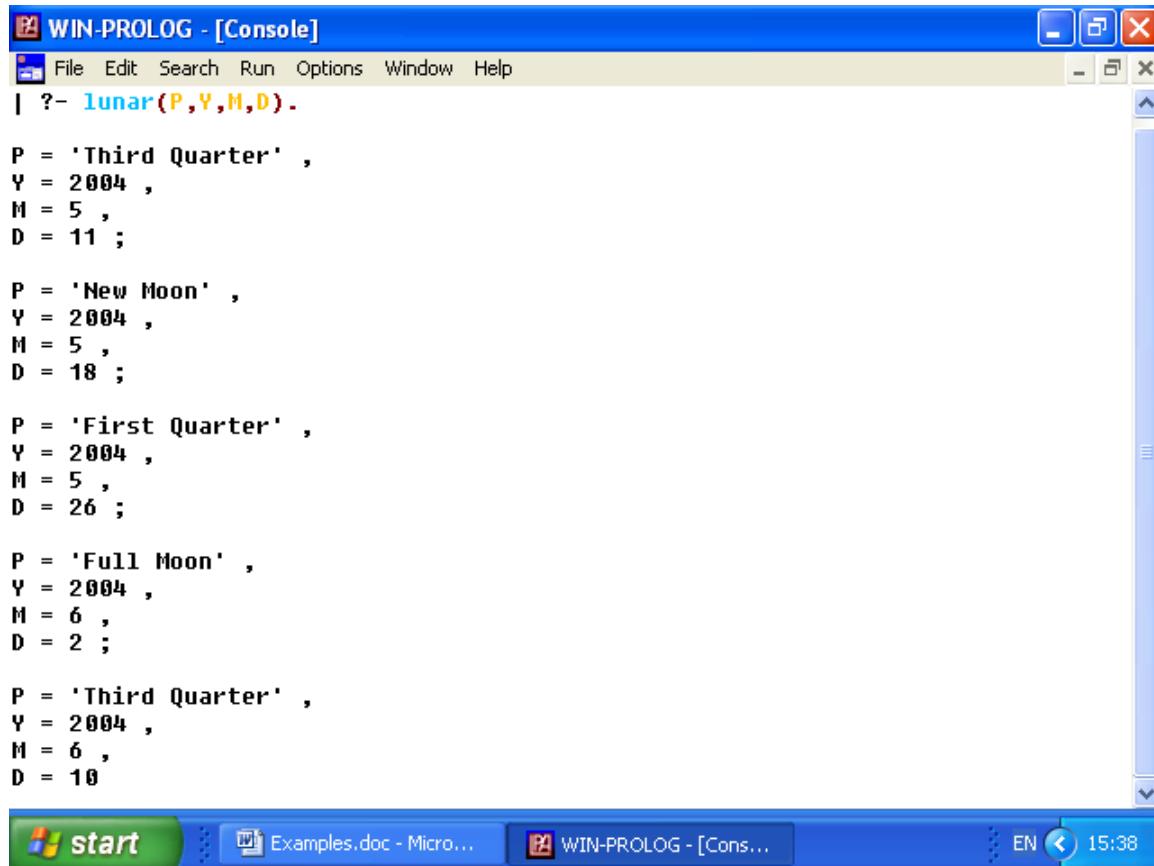


Examples

LUNAR.PL – Lunar Phases

This program computes the next quarter phase of the moon occurring on or after the given date, based on two assumptions: (i) a new moon occurred on 01 Jan 1900, and (ii) the lunar month is 29.53059 days.

The program uses the time/2 predicate to obtain a starting day number, and the time/4 predicate to compute year/month/day dates from these numbers.



The screenshot shows the WIN-PROLOG [Console] window. The menu bar includes File, Edit, Search, Run, Options, Window, and Help. The main area displays the following Prolog code:

```
P = 'Third Quarter' ,
Y = 2004 ,
M = 5 ,
D = 11 ;

P = 'New Moon' ,
Y = 2004 ,
M = 5 ,
D = 18 ;

P = 'First Quarter' ,
Y = 2004 ,
M = 5 ,
D = 26 ;

P = 'Full Moon' ,
Y = 2004 ,
M = 6 ,
D = 2 ;

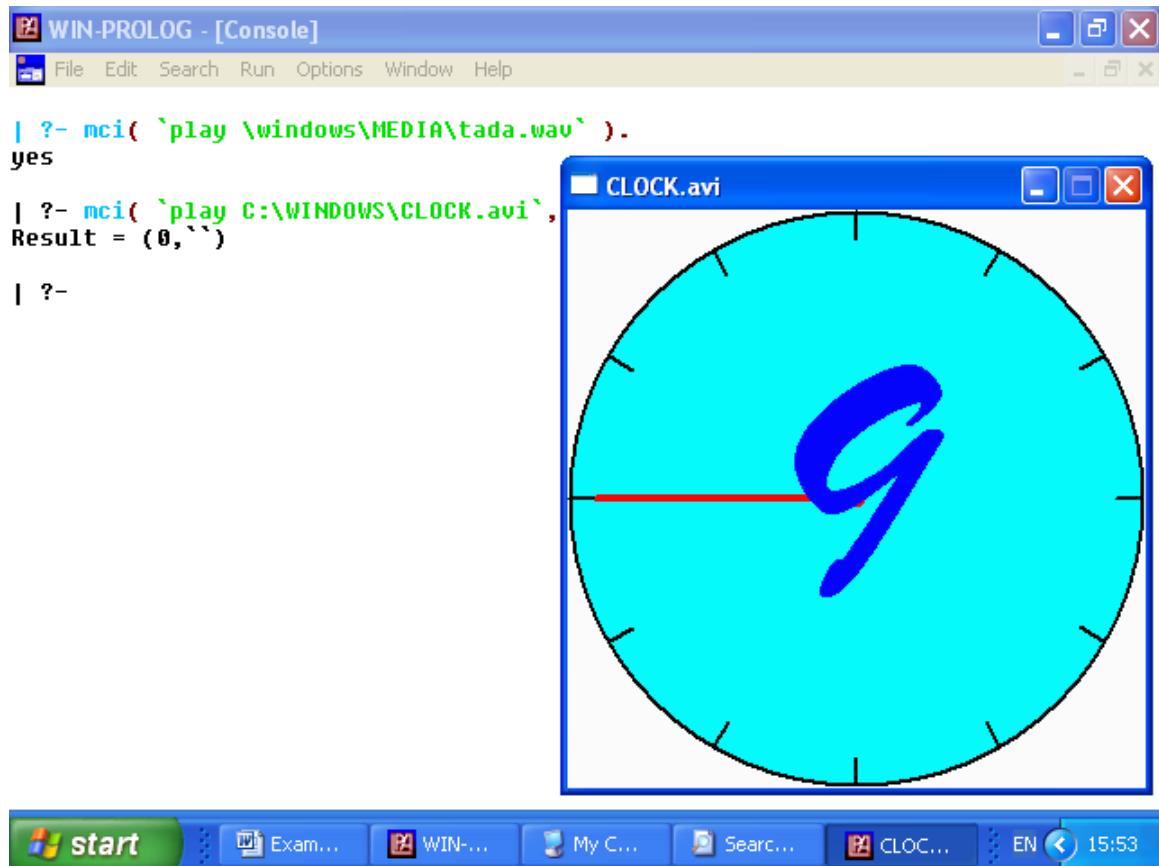
P = 'Third Quarter' ,
Y = 2004 ,
M = 6 ,
D = 18
```

The window is part of a Windows desktop environment, with other icons like 'start' and 'Examples.doc - Micro...' visible in the taskbar.

Examples

MCI.PL - MCI Functions for WIN-PROLOG

The mci/1 and mci/2 predicate are used to send command strings to the WINMM.DLL (Windows MultiMedia) API, and permits supported files to be played under Prolog control.



A screenshot of a Windows desktop environment. At the top, there's a taskbar with icons for Start, Exam..., WIN-..., My C..., Search..., CLOC..., EN, and a clock showing 15:53. In the center, a Win-Prolog console window titled "WIN-PROLOG - [Console]" is open, displaying the following Prolog code:

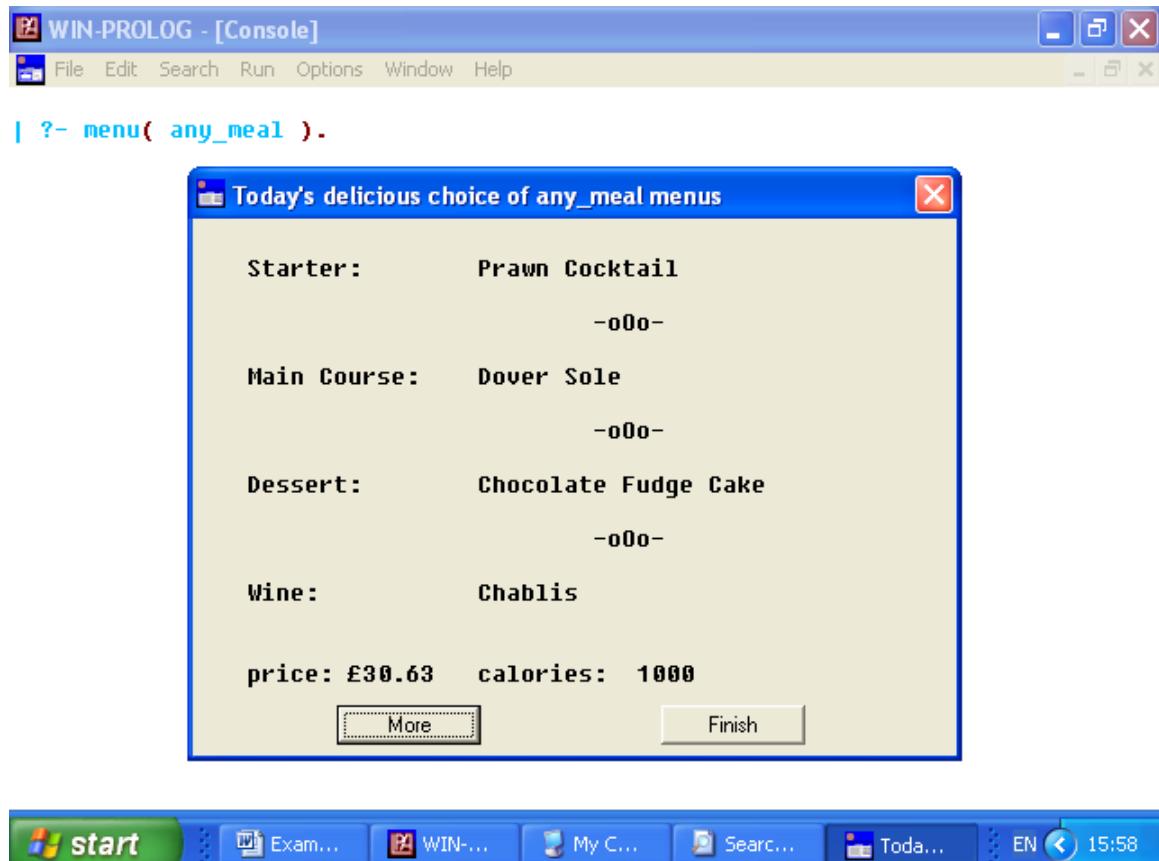
```
| ?- mci( `play \windows\MEDIA\tada.wav` ).  
yes  
| ?- mci( `play C:\WINDOWS\CLOCK.avi` ,  
Result = (0,'`'))  
| ?-
```

To the right of the console, a video player window titled "CLOCK.avi" is running, showing a blue digital clock face with the number "9" in the center. The desktop background is white.

Examples

MEALS.PL – Meal Selection Example

This program is designed to help you select meals of various sorts from a database of dishes.

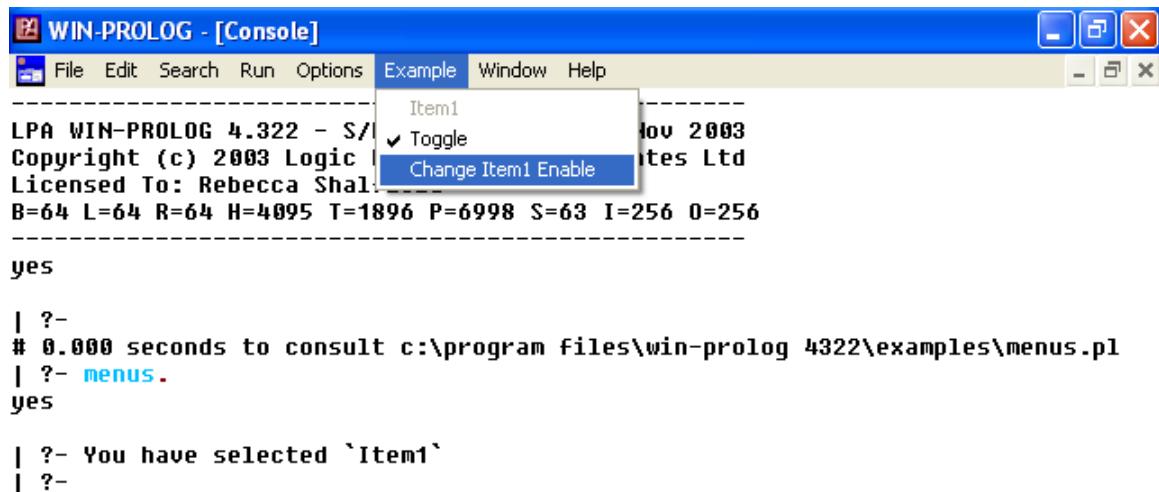


Examples

MENUS.PL – A Simple Menu Demonstration

The menus/0 predicate creates and installs a simple menu example.

Once installed, the menu messages are handled by adding a handler to the console window.



A screenshot of the WIN-PROLOG [Console] window. The window title is "WIN-PROLOG - [Console]". The menu bar includes File, Edit, Search, Run, Options, Example, Window, and Help. The "Example" menu is currently selected and has a dropdown menu open with items: Item1, Toggle, and Change Item1 Enable. The main console area displays the following text:

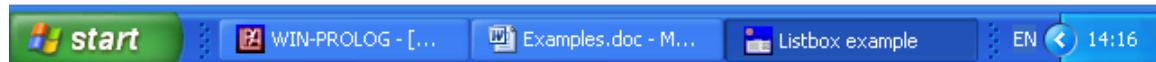
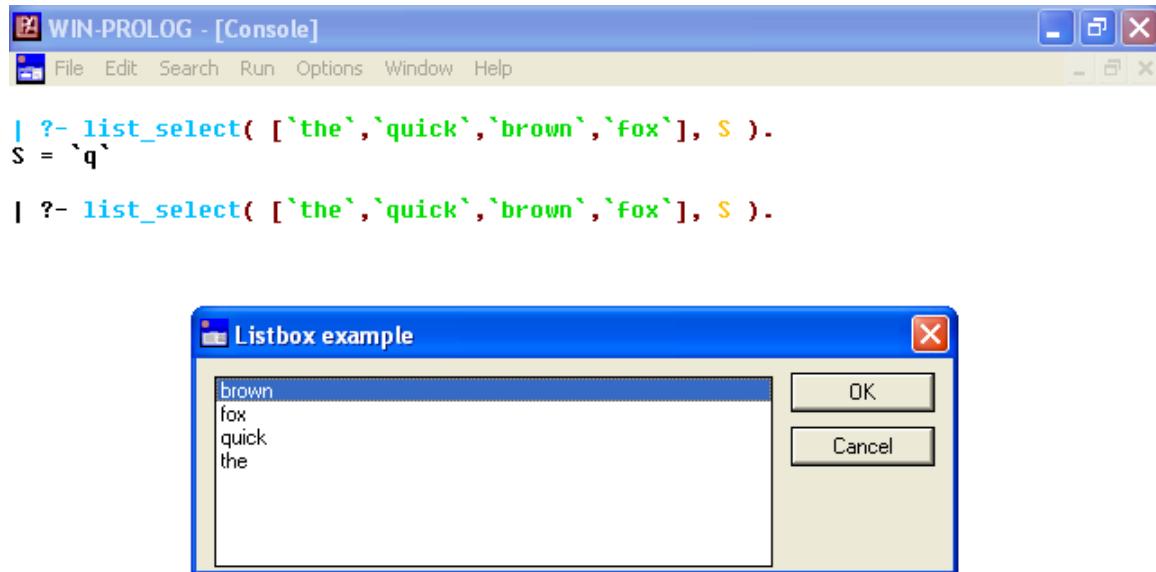
```
LPA WIN-PROLOG 4.322 - S/ 2003 Nov 2003
Copyright (c) 2003 Logic & Mathematics Ltd
Licensed To: Rebecca Shallit
B=64 L=64 R=64 H=4095 T=1896 P=6998 S=63 I=256 O=256
-----
yes
| ?-
# 0.000 seconds to consult c:\program files\win-prolog 4322\examples\menus.pl
| ?- menus.
yes
| ?- You have selected `Item1'
| ?-
```



Examples

MESSAGES.PL – Listbox Example

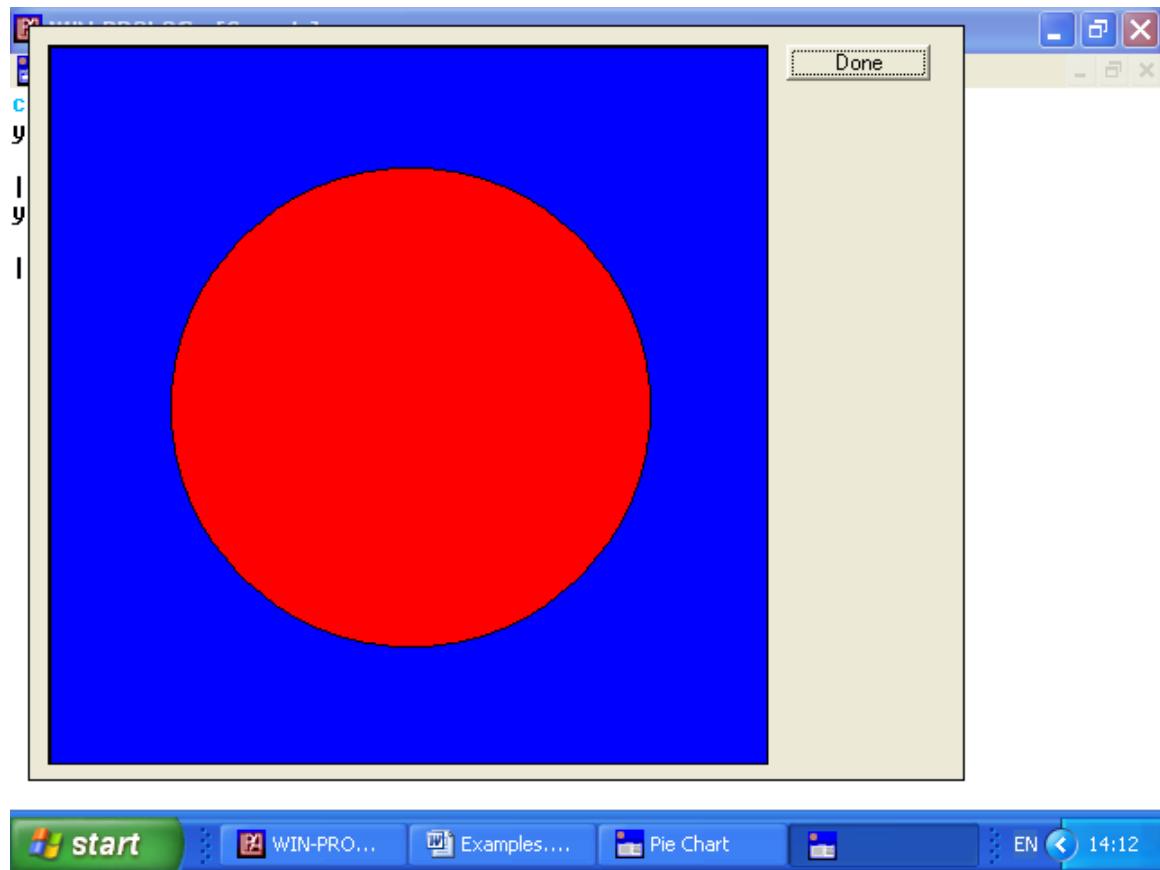
This example demonstrates the use of the windows messaging predicate `sndmsg/5` to control a listbox. A list of strings is given to the program, and these are displayed in a listbox. When the user selects one, it is returned by the program.



Examples

META.PL - Creating a Metafile

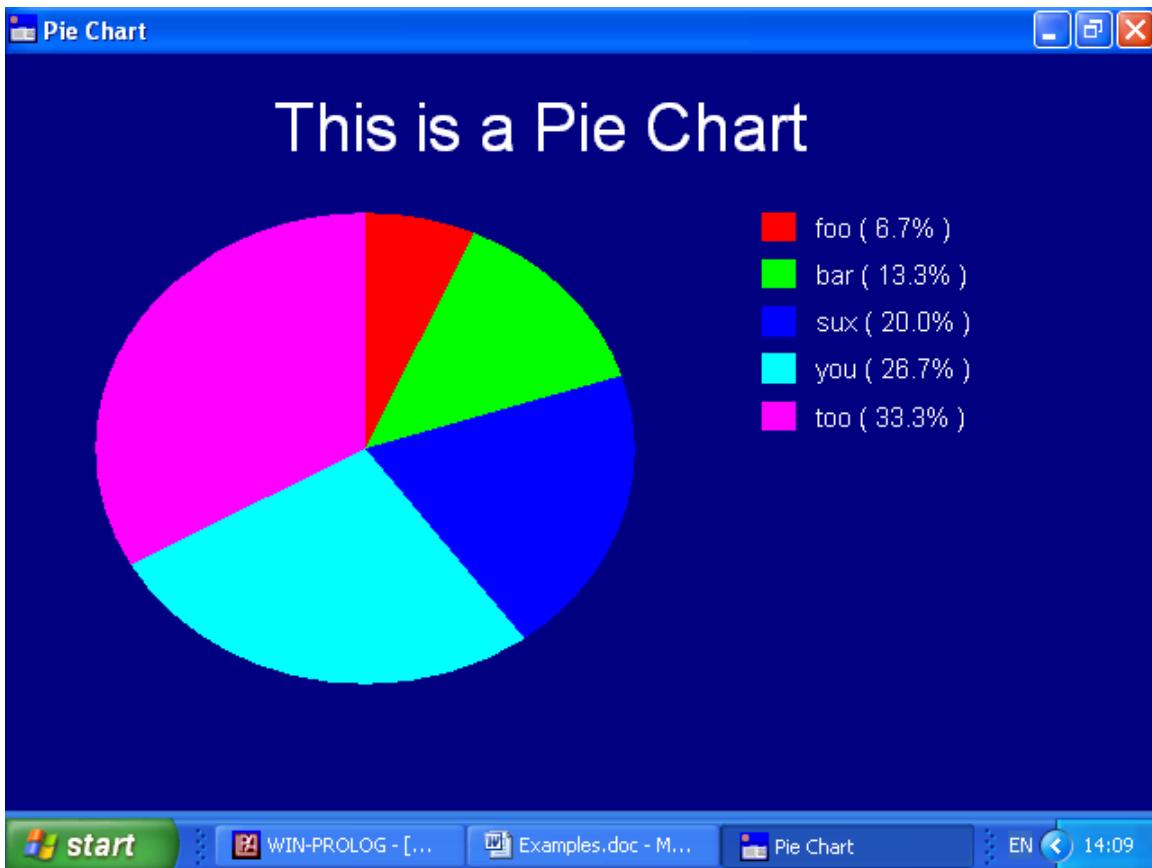
This program demonstrates how to create a Windows "enhanced metafile" (.wmf file) using "gfx" style graphics. Use is made of winapi/4 to create, close, save and delete the metafile, but all other graphics are handled by the gfx/n predicates. Please note that as create_metafile/0 has called the "CloseEnhMetaFile" function, the expected call to gfx_end/1 would generate an error: gfx_cleanup/0 is called instead to restore the grafix stack.



Examples

PIECHART.PL - Pie-Chart Display

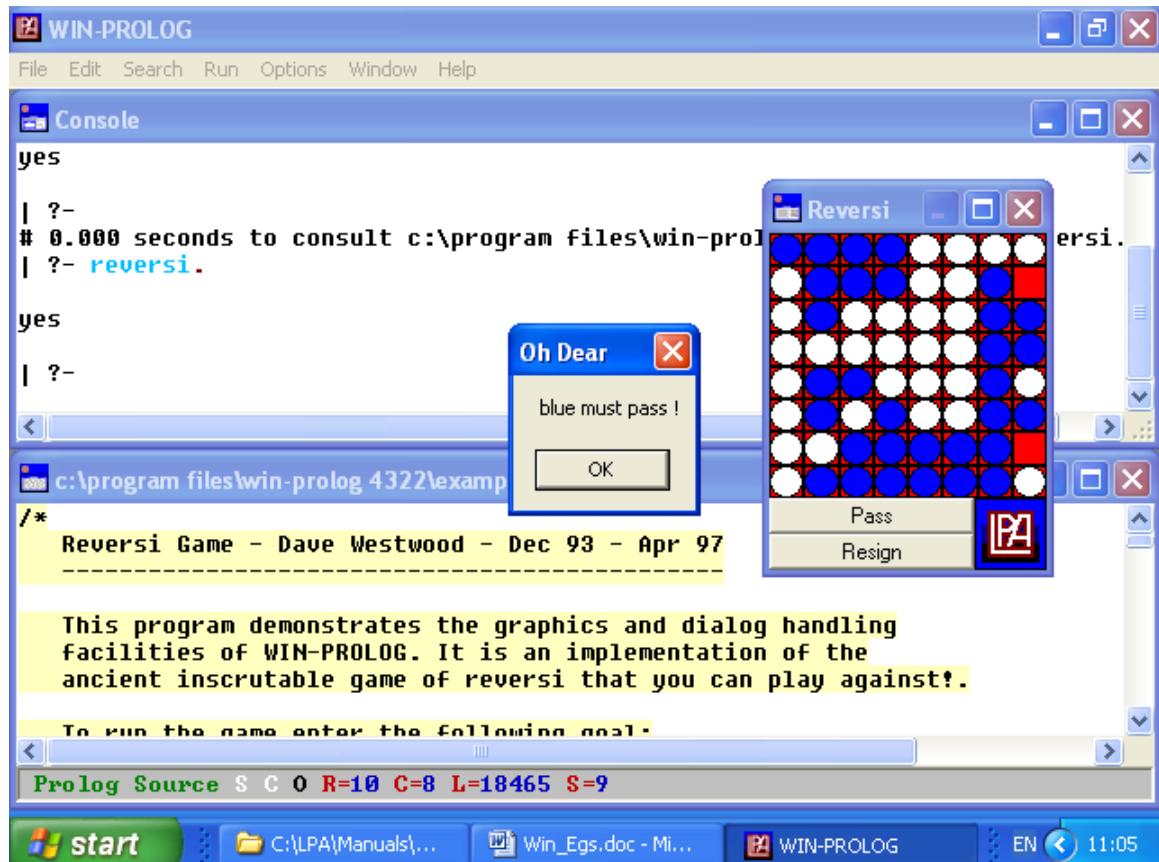
This program takes data in the form of a chart title and a list of name(number) terms, and displays the result in a pie chart. The numbers are automatically totalled, and segment sizes are computed according to the percentage ratio of each respective number to the total.



Examples

REVERSI.PL - Reversi Game

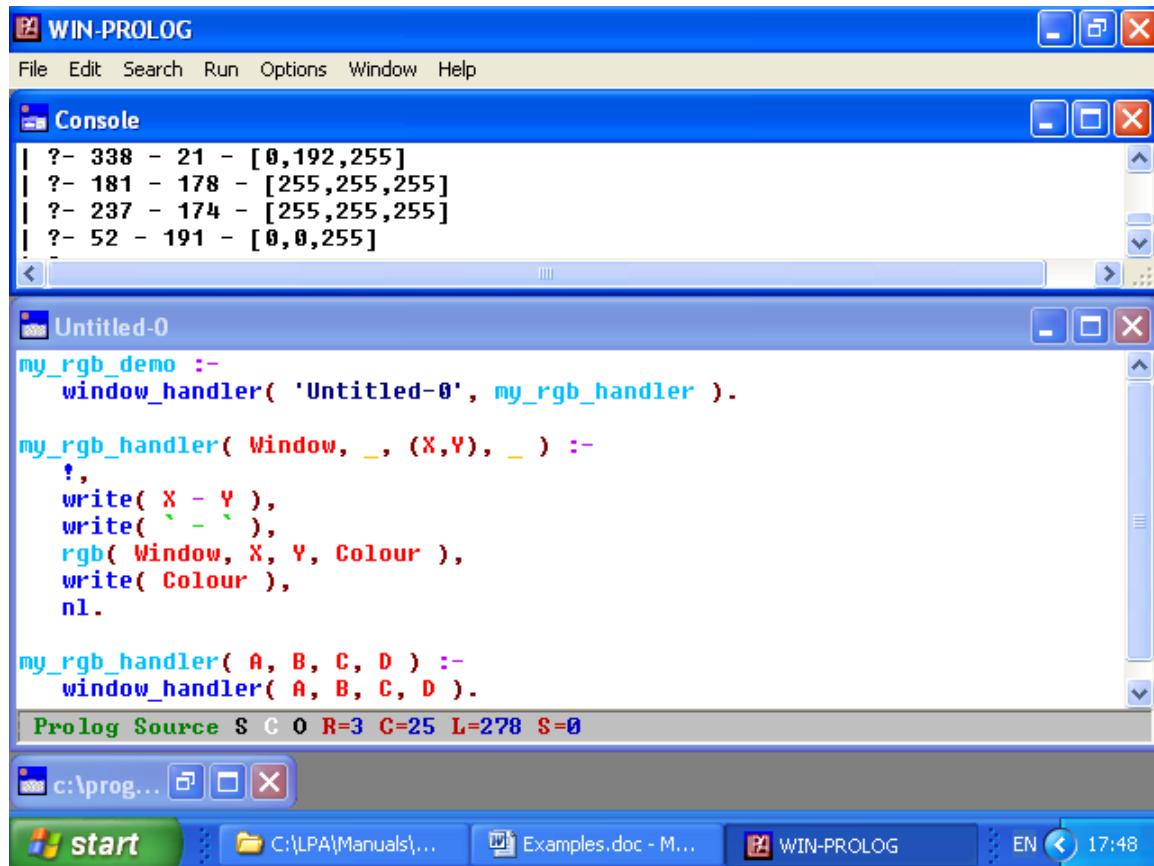
This program demonstrates the graphics and dialog handling facilities of WIN-PROLOG. It is an implementation of the ancient inscrutable game of reversi that you can play against!.



Examples

RGB.PL - Return the RGB Colour of a Single Pixel

This program uses the Windows API, "GetPixel", to obtain the Red, Green and Blue (RGB) values associated with a single pixel at the given (X,Y) offset within a given Window. If the Window is not visible, or the offset outside of its clipping region, an empty list, "[]", is returned; otherwise, a list of the form, "[R,G,B]", is returned, containing the individual Red, Green and Blue values.



```

WIN-PROLOG
File Edit Search Run Options Window Help
Console
?- 338 - 21 - [0,192,255]
?- 181 - 178 - [255,255,255]
?- 237 - 174 - [255,255,255]
?- 52 - 191 - [0,0,255]
Untitled-0
my_rgb_demo :-  

    window_handler( 'Untitled-0', my_rgb_handler ).  

my_rgb_handler( Window, _, (X,Y), _ ) :-  

    !,  

    write( X - Y ),  

    write( ' - ' ),  

    rgb( Window, X, Y, Colour ),  

    write( Colour ),  

    nl.  

my_rgb_handler( A, B, C, D ) :-  

    window_handler( A, B, C, D ).  

Pro log Source S C O R=3 C=25 L=278 S=0
c:\prog...
start C:\LPA\Manuals\... Examples.doc - M... WIN-PROLOG EN 17:48

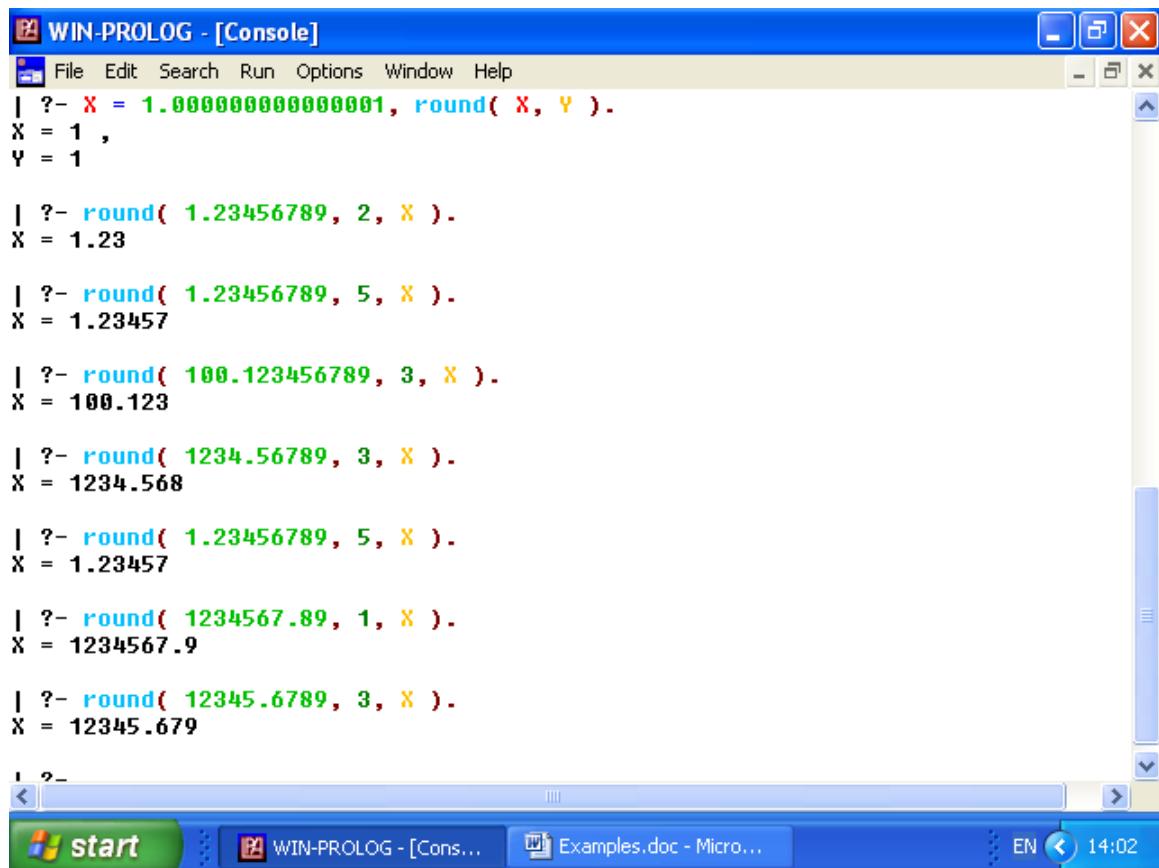
```

Examples

ROUND.PL - Numerical Rounding Routines

This file contains two simple predicates which round numbers down to a given precision. The first is used to round numbers down to their displayed precision, hiding any "guard digits" that might be present after floating point arithmetic operations, while the second rounds a value to given number of decimal places.

Both predicates use input and output to perform the rounding, and the second is limited to numbers that can be displayed using conventional decimal notation (eg 123.456), being unable to handle numbers which require exponential notation (eg 1.23456e78).



The screenshot shows the WIN-PROLOG [Console] window with the following interactions:

```
| ?- X = 1.00000000000001, round( X, Y ).  
X = 1 ,  
Y = 1  
  
| ?- round( 1.23456789, 2, X ).  
X = 1.23  
  
| ?- round( 1.23456789, 5, X ).  
X = 1.23457  
  
| ?- round( 180.123456789, 3, X ).  
X = 180.123  
  
| ?- round( 1234.56789, 3, X ).  
X = 1234.568  
  
| ?- round( 1.23456789, 5, X ).  
X = 1.23457  
  
| ?- round( 1234567.89, 1, X ).  
X = 1234567.9  
  
| ?- round( 12345.6789, 3, X ).  
X = 12345.679
```

The window has a blue title bar with the text "WIN-PROLOG - [Console]". The menu bar includes File, Edit, Search, Run, Options, Window, and Help. The status bar at the bottom shows "EN 14:02".

Examples

SALESMAN.PL - The Travelling Salesman

This program displays a map of the mainland UK, showing a number of towns. These can be selected using the mouse, and then the shortest route found between them.

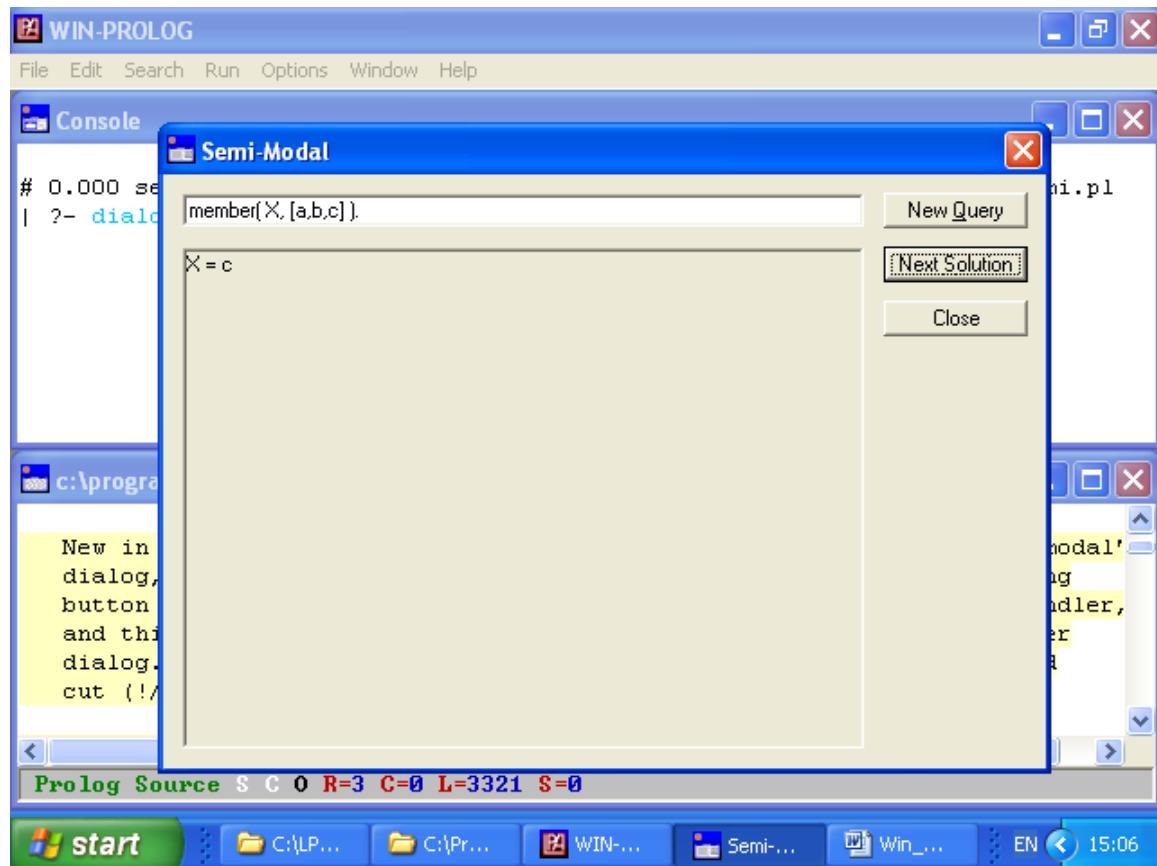
Two algorithms are defined: the "exhaustive" one finds every possible route and returns the shortest; the "heuristic" one takes each selected town in turn, and inserts it into the optimal location in the route as it grows. The former routine is combinatorial in nature, and takes far too long to compute complex routes; the latter is an n-square algorithm, and works reasonably well even with large numbers of towns, however, it does not always return the very best route.



Examples

SEMI.PL - Controlling Backtracking with Semi-Modal Dialogs

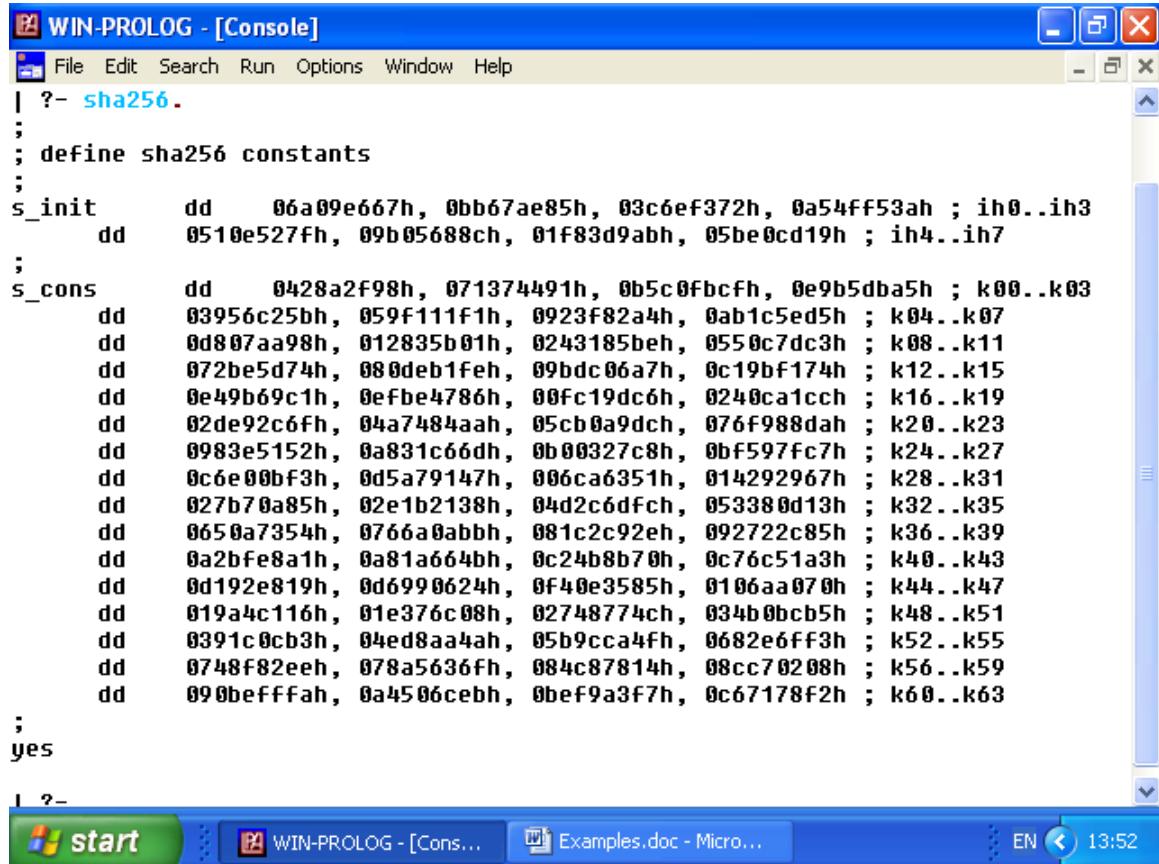
New in version 4.400 of **WIN-PROLOG** is the ability to define a "semi-modal" dialog, in which it is easy to control backtracking of a program using button events. This can be achieved without even writing a dialog handler, and this example shows how to do this, using a simple query-and-answer dialog. The only "messyness" is caused by the need to use `catch/2` and `cut (!/0)` to make the program immune to user-typed errors.



Examples

SHA256.PL - Generate the SHA-256 Constants

This program uses the Sieve of Erasthenes to generate prime numbers and then outputs the most significant 32 bits of the fractional parts of their square and cube roots as required by the 256-bit Secure Hash Algorithm (SHA-256), as defined in FIPS 180-2, in MASM format.



```

WIN-PROLOG - [Console]
File Edit Search Run Options Window Help
?- sha256.
;
; define sha256 constants
;
s_init    dd 06a09e667h, 0bb67ae85h, 03c6ef372h, 0a54ff53ah ; ih0..ih3
          dd 0510e527fh, 09b05688ch, 01f83d9abh, 05be0cd19h ; ih4..ih7
;
s_cons    dd 0428a2f98h, 071374491h, 0b5c0fbcfh, 0e9b5dba5h ; k00..k03
          dd 03956c25bh, 059f111f1h, 0923f82a4h, 0ab1c5ed5h ; k04..k07
          dd 0d807aa98h, 012835b01h, 0243185beh, 0550c7dc3h ; k08..k11
          dd 072be5d74h, 080deb1feh, 09bdc06a7h, 0c19bf174h ; k12..k15
          dd 0e49b69c1h, 0efbe4786h, 00fc19dc6h, 0240ca1cch ; k16..k19
          dd 02de92c6fh, 04a7484aah, 05cb0a9dch, 076f988dah ; k20..k23
          dd 0983e5152h, 0a831c66dh, 0b00327c8h, 0bf597fc7h ; k24..k27
          dd 0c6e00bf3h, 0d5a79147h, 006ca6351h, 014292967h ; k28..k31
          dd 027b70a85h, 02e1b2138h, 04d2c6dfch, 053380d13h ; k32..k35
          dd 0650a7354h, 0766a0abbh, 081c2c92eh, 092722c85h ; k36..k39
          dd 0a2bfe8a1h, 0a81a664bh, 0c24b8b70h, 0c76c51a3h ; k40..k43
          dd 0d192e819h, 0d6990624h, 0f40e3585h, 0106aa070h ; k44..k47
          dd 019a4c116h, 01e376c08h, 02748774ch, 034b0bcb5h ; k48..k51
          dd 0391c0cb3h, 04ed8aa4ah, 05b9cca4fh, 0682e6ff3h ; k52..k55
          dd 0748f82eeh, 078a5636fh, 084c87814h, 08cc70208h ; k56..k59
          dd 090befffah, 0a4506cebh, 0bef9a3f7h, 0c67178f2h ; k60..k63
;
yes

```

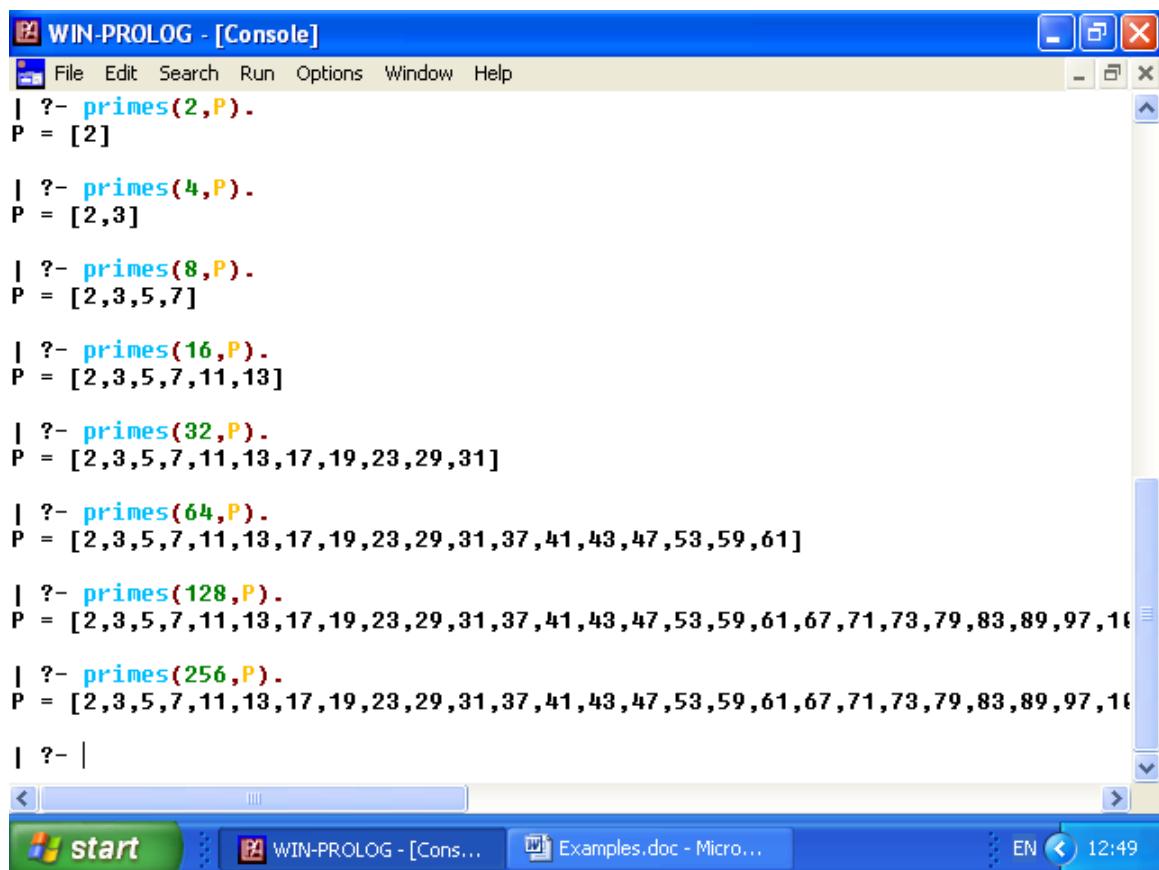
Examples

SIEVE.PL - Sieve of Erasthatones

In this classic algorithm, a list of prime numbers is generated by sifting a list of all integers between 2 and the given limit, and discarding those which are directly divisible by a known prime number.

The initial list of integers is created using a simple recursive program, and all entries directly divisible by the first number in the list (initially 2) are then removed; the head of the resulting list (2) is kept, and the tail (in this case, starting with 3) is processed recursively until all numbers have been stripped.

A shorter variant of this algorithm can be found in the file SIEVE2.PL.



The screenshot shows a Windows desktop environment with a 'WIN-PROLOG - [Console]' window open. The window title bar is blue with the text 'WIN-PROLOG - [Console]'. The menu bar includes 'File', 'Edit', 'Search', 'Run', 'Options', 'Window', and 'Help'. The main area of the window displays the output of a Prolog program. The code consists of several queries to the 'primes' predicate:

```
| ?- primes(2,P).  
P = [2]  
  
| ?- primes(4,P).  
P = [2,3]  
  
| ?- primes(8,P).  
P = [2,3,5,7]  
  
| ?- primes(16,P).  
P = [2,3,5,7,11,13]  
  
| ?- primes(32,P).  
P = [2,3,5,7,11,13,17,19,23,29,31]  
  
| ?- primes(64,P).  
P = [2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61]  
  
| ?- primes(128,P).  
P = [2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97,101]  
  
| ?- primes(256,P).  
P = [2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97,101,107,113,127,131,137,149,151,157,163,173,179,181,191,197,199,211,223,227,233,241,251,257,263,269,271,281,283,293,297,307,311,313,323,331,341,347,353,359,367,373,383,397,401,409,419,431,433,443,457,461,463,479,487,491,499,503,521,523,541,547,557,563,569,587,593,597,601,613,617,623,631,641,643,653,659,661,673,677,683,691,697,701,709,713,721,733,739,751,757,761,769,773,787,791,797,809,811,821,823,839,841,851,853,863,877,881,883,893,897,901,911,923,931,941,953,961,971,977,983,991,1001,1013,1021,1031,1043,1051,1061,1073,1081,1091,1103,1111,1121,1133,1141,1151,1163,1171,1181,1193,1201,1211,1223,1231,1241,1253,1261,1271,1283,1291,1301,1311,1321,1331,1341,1351,1361,1371,1381,1391,1401,1411,1421,1431,1441,1451,1461,1471,1481,1491,1501,1511,1521,1531,1541,1551,1561,1571,1581,1591,1601,1611,1621,1631,1641,1651,1661,1671,1681,1691,1701,1711,1721,1731,1741,1751,1761,1771,1781,1791,1801,1811,1821,1831,1841,1851,1861,1871,1881,1891,1901,1911,1921,1931,1941,1951,1961,1971,1981,1991,2001,2011,2021,2031,2041,2051,2061,2071,2081,2091,2101,2111,2121,2131,2141,2151,2161,2171,2181,2191,2201,2211,2221,2231,2241,2251,2261,2271,2281,2291,2301,2311,2321,2331,2341,2351,2361,2371,2381,2391,2401,2411,2421,2431,2441,2451,2461,2471,2481,2491,2501,2511,2521,2531,2541,2551,2561,2571,2581,2591,2601,2611,2621,2631,2641,2651,2661,2671,2681,2691,2701,2711,2721,2731,2741,2751,2761,2771,2781,2791,2801,2811,2821,2831,2841,2851,2861,2871,2881,2891,2901,2911,2921,2931,2941,2951,2961,2971,2981,2991,3001,3011,3021,3031,3041,3051,3061,3071,3081,3091,3101,3111,3121,3131,3141,3151,3161,3171,3181,3191,3201,3211,3221,3231,3241,3251,3261,3271,3281,3291,3301,3311,3321,3331,3341,3351,3361,3371,3381,3391,3401,3411,3421,3431,3441,3451,3461,3471,3481,3491,3501,3511,3521,3531,3541,3551,3561,3571,3581,3591,3601,3611,3621,3631,3641,3651,3661,3671,3681,3691,3701,3711,3721,3731,3741,3751,3761,3771,3781,3791,3801,3811,3821,3831,3841,3851,3861,3871,3881,3891,3901,3911,3921,3931,3941,3951,3961,3971,3981,3991,4001,4011,4021,4031,4041,4051,4061,4071,4081,4091,4101,4111,4121,4131,4141,4151,4161,4171,4181,4191,4201,4211,4221,4231,4241,4251,4261,4271,4281,4291,4301,4311,4321,4331,4341,4351,4361,4371,4381,4391,4401,4411,4421,4431,4441,4451,4461,4471,4481,4491,4501,4511,4521,4531,4541,4551,4561,4571,4581,4591,4601,4611,4621,4631,4641,4651,4661,4671,4681,4691,4701,4711,4721,4731,4741,4751,4761,4771,4781,4791,4801,4811,4821,4831,4841,4851,4861,4871,4881,4891,4897,4909,4919,4929,4939,4949,4959,4969,4979,4989,4999,5009,5019,5029,5039,5049,5059,5069,5079,5089,5099,5109,5119,5129,5139,5149,5159,5169,5179,5189,5199,5209,5219,5229,5239,5249,5259,5269,5279,5289,5299,5309,5319,5329,5339,5349,5359,5369,5379,5389,5399,5409,5419,5429,5439,5449,5459,5469,5479,5489,5499,5509,5519,5529,5539,5549,5559,5569,5579,5589,5599,5597,5609,5619,5629,5639,5649,5659,5669,5679,5689,5699,5697,5709,5719,5729,5739,5749,5759,5769,5779,5789,5799,5797,5809,5819,5829,5839,5849,5859,5869,5879,5889,5899,5897,5909,5919,5929,5939,5949,5959,5969,5979,5989,5999,5997,6009,6019,6029,6039,6049,6059,6069,6079,6089,6099,6097,6109,6119,6129,6139,6149,6159,6169,6179,6189,6199,6197,6209,6219,6229,6239,6249,6259,6269,6279,6289,6299,6297,6309,6319,6329,6339,6349,6359,6369,6379,6389,6399,6397,6409,6419,6429,6439,6449,6459,6469,6479,6489,6499,6497,6509,6519,6529,6539,6549,6559,6569,6579,6589,6599,6597,6609,6619,6629,6639,6649,6659,6669,6679,6689,6699,6697,6709,6719,6729,6739,6749,6759,6769,6779,6789,6799,6797,6809,6819,6829,6839,6849,6859,6869,6879,6889,6899,6897,6909,6919,6929,6939,6949,6959,6969,6979,6989,6999,6997,7009,7019,7029,7039,7049,7059,7069,7079,7089,7099,7097,7109,7119,7129,7139,7149,7159,7169,7179,7189,7199,7197,7209,7219,7229,7239,7249,7259,7269,7279,7289,7299,7297,7309,7319,7329,7339,7349,7359,7369,7379,7389,7399,7397,7409,7419,7429,7439,7449,7459,7469,7479,7489,7499,7497,7509,7519,7529,7539,7549,7559,7569,7579,7589,7599,7597,7609,7619,7629,7639,7649,7659,7669,7679,7689,7699,7697,7709,7719,7729,7739,7749,7759,7769,7779,7789,7799,7797,7809,7819,7829,7839,7849,7859,7869,7879,7889,7899,7897,7909,7919,7929,7939,7949,7959,7969,7979,7989,7999,7997,8009,8019,8029,8039,8049,8059,8069,8079,8089,8099,8097,8109,8119,8129,8139,8149,8159,8169,8179,8189,8199,8197,8209,8219,8229,8239,8249,8259,8269,8279,8289,8299,8297,8309,8319,8329,8339,8349,8359,8369,8379,8389,8399,8397,8409,8419,8429,8439,8449,8459,8469,8479,8489,8499,8497,8509,8519,8529,8539,8549,8559,8569,8579,8589,8599,8597,8609,8619,8629,8639,8649,8659,8669,8679,8689,8699,8697,8709,8719,8729,8739,8749,8759,8769,8779,8789,8799,8797,8809,8819,8829,8839,8849,8859,8869,8879,8889,8899,8897,8909,8919,8929,8939,8949,8959,8969,8979,8989,8999,8997,9009,9019,9029,9039,9049,9059,9069,9079,9089,9099,9097,9109,9119,9129,9139,9149,9159,9169,9179,9189,9199,9197,9209,9219,9229,9239,9249,9259,9269,9279,9289,9299,9297,9309,9319,9329,9339,9349,9359,9369,9379,9389,9399,9397,9409,9419,9429,9439,9449,9459,9469,9479,9489,9499,9497,9509,9519,9529,9539,9549,9559,9569,9579,9589,9599,9597,9609,9619,9629,9639,9649,9659,9669,9679,9689,9699,9697,9709,9719,9729,9739,9749,9759,9769,9779,9789,9799,9797,9809,9819,9829,9839,9849,9859,9869,9879,9889,9899,9897,9909,9919,9929,9939,9949,9959,9969,9979,9989,9999,9997]
```

Examples

SIEVE2.PL - Sieve of Erasthatones

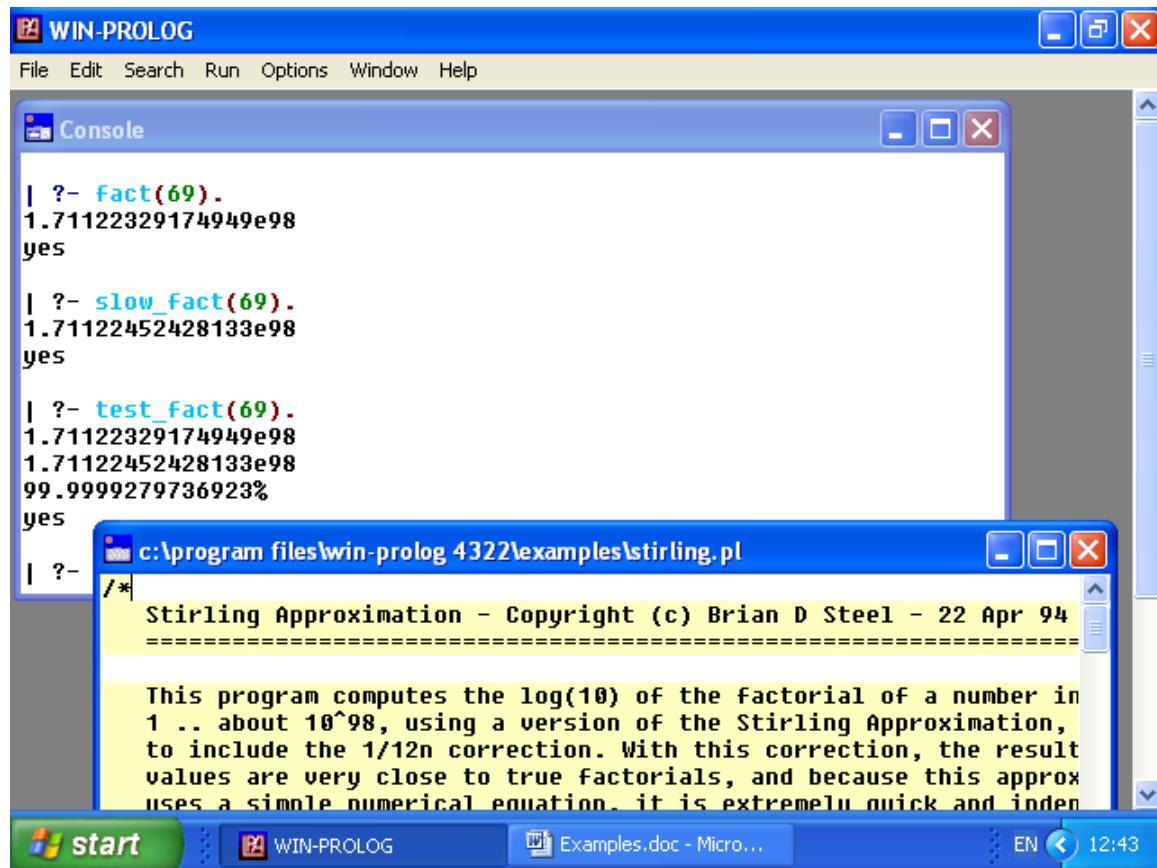
In this classic algorithm, a list of prime numbers is generated by sifting a list of all integers between 2 and the given limit, and discarding those which are directly divisible by a known prime number.

The initial list of integers is created using a combination of findall/3 and integer_bound/3, and all entries directly divisible by the first number in the list (initially 2) are then removed; the head of the resulting list (2) is kept, and the tail (in this case, starting with 3) is processed recursively until all numbers have been stripped.

<img alt="Screenshot of the WIN-PROLOG [Console] window showing the execution of the SIEVE2.PL program. The window title is 'WIN-PROLOG - [Console]'. The console output shows the recursive generation of prime lists for increasing limits: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 299, 311, 317, 323, 331, 347, 349, 353, 359, 367, 373, 379, 383, 397, 401, 409, 421, 431, 433, 437, 443, 457, 461, 463, 467, 479, 481, 491, 499, 503, 521, 523, 541, 547, 557, 563, 569, 581, 593, 599, 601, 611, 623, 631, 641, 643, 647, 653, 659, 661, 671, 673, 677, 683, 691, 697, 701, 709, 711, 721, 731, 733, 739, 743, 751, 757, 761, 763, 767, 773, 781, 787, 791, 797, 801, 809, 811, 821, 823, 831, 841, 851, 853, 859, 863, 871, 881, 891, 897, 901, 911, 923, 931, 941, 953, 961, 971, 977, 983, 991, 997, 1001, 1009, 1013, 1021, 1031, 1041, 1051, 1061, 1071, 1081, 1091, 1097, 1101, 1111, 1121, 1131, 1141, 1151, 1161, 1171, 1181, 1191, 1201, 1211, 1221, 1231, 1241, 1251, 1261, 1271, 1281, 1291, 1301, 1311, 1321, 1331, 1341, 1351, 1361, 1371, 1381, 1391, 1401, 1411, 1421, 1431, 1441, 1451, 1461, 1471, 1481, 1491, 1501, 1511, 1521, 1531, 1541, 1551, 1561, 1571, 1581, 1591, 1601, 1611, 1621, 1631, 1641, 1651, 1661, 1671, 1681, 1691, 1701, 1711, 1721, 1731, 1741, 1751, 1761, 1771, 1781, 1791, 1801, 1811, 1821, 1831, 1841, 1851, 1861, 1871, 1881, 1891, 1901, 1911, 1921, 1931, 1941, 1951, 1961, 1971, 1981, 1991, 2001, 2011, 2021, 2031, 2041, 2051, 2061, 2071, 2081, 2091, 2101, 2111, 2121, 2131, 2141, 2151, 2161, 2171, 2181, 2191, 2201, 2211, 2221, 2231, 2241, 2251, 2261, 2271, 2281, 2291, 2301, 2311, 2321, 2331, 2341, 2351, 2361, 2371, 2381, 2391, 2401, 2411, 2421, 2431, 2441, 2451, 2461, 2471, 2481, 2491, 2501, 2511, 2521, 2531, 2541, 2551, 2561, 2571, 2581, 2591, 2601, 2611, 2621, 2631, 2641, 2651, 2661, 2671, 2681, 2691, 2701, 2711, 2721, 2731, 2741, 2751, 2761, 2771, 2781, 2791, 2801, 2811, 2821, 2831, 2841, 2851, 2861, 2871, 2881, 2891, 2901, 2911, 2921, 2931, 2941, 2951, 2961, 2971, 2981, 2991, 3001, 3011, 3021, 3031, 3041, 3051, 3061, 3071, 3081, 3091, 3101, 3111, 3121, 3131, 3141, 3151, 3161, 3171, 3181, 3191, 3201, 3211, 3221, 3231, 3241, 3251, 3261, 3271, 3281, 3291, 3301, 3311, 3321, 3331, 3341, 3351, 3361, 3371, 3381, 3391, 3401, 3411, 3421, 3431, 3441, 3451, 3461, 3471, 3481, 3491, 3501, 3511, 3521, 3531, 3541, 3551, 3561, 3571, 3581, 3591, 3601, 3611, 3621, 3631, 3641, 3651, 3661, 3671, 3681, 3691, 3701, 3711, 3721, 3731, 3741, 3751, 3761, 3771, 3781, 3791, 3801, 3811, 3821, 3831, 3841, 3851, 3861, 3871, 3881, 3891, 3901, 3911, 3921, 3931, 3941, 3951, 3961, 3971, 3981, 3991, 4001, 4011, 4021, 4031, 4041, 4051, 4061, 4071, 4081, 4091, 4101, 4111, 4121, 4131, 4141, 4151, 4161, 4171, 4181, 4191, 4201, 4211, 4221, 4231, 4241, 4251, 4261, 4271, 4281, 4291, 4301, 4311, 4321, 4331, 4341, 4351, 4361, 4371, 4381, 4391, 4401, 4411, 4421, 4431, 4441, 4451, 4461, 4471, 4481, 4491, 4501, 4511, 4521, 4531, 4541, 4551, 4561, 4571, 4581, 4591, 4601, 4611, 4621, 4631, 4641, 4651, 4661, 4671, 4681, 4691, 4701, 4711, 4721, 4731, 4741, 4751, 4761, 4771, 4781, 4791, 4801, 4811, 4821, 4831, 4841, 4851, 4861, 4871, 4881, 4891, 4901, 4911, 4921, 4931, 4941, 4951, 4961, 4971, 4981, 4991, 5001, 5011, 5021, 5031, 5041, 5051, 5061, 5071, 5081, 5091, 5101, 5111, 5121, 5131, 5141, 5151, 5161, 5171, 5181, 5191, 5201, 5211, 5221, 5231, 5241, 5251, 5261, 5271, 5281, 5291, 5301, 5311, 5321, 5331, 5341, 5351, 5361, 5371, 5381, 5391, 5401, 5411, 5421, 5431, 5441, 5451, 5461, 5471, 5481, 5491, 5501, 5511, 5521, 5531, 5541, 5551, 5561, 5571, 5581, 5591, 5601, 5611, 5621, 5631, 5641, 5651, 5661, 5671, 5681, 5691, 5701, 5711, 5721, 5731, 5741, 5751, 5761, 5771, 5781, 5791, 5801, 5811, 5821, 5831, 5841, 5851, 5861, 5871, 5881, 5891, 5901, 5911, 5921, 5931, 5941, 5951, 5961, 5971, 5981, 5991, 6001, 6011, 6021, 6031, 6041, 6051, 6061, 6071, 6081, 6091, 6101, 6111, 6121, 6131, 6141, 6151, 6161, 6171, 6181, 6191, 6201, 6211, 6221, 6231, 6241, 6251, 6261, 6271, 6281, 6291, 6301, 6311, 6321, 6331, 6341, 6351, 6361, 6371, 6381, 6391, 6401, 6411, 6421, 6431, 6441, 6451, 6461, 6471, 6481, 6491, 6501, 6511, 6521, 6531, 6541, 6551, 6561, 6571, 6581, 6591, 6601, 6611, 6621, 6631, 6641, 6651, 6661, 6671, 6681, 6691, 6701, 6711, 6721, 6731, 6741, 6751, 6761, 6771, 6781, 6791, 6801, 6811, 6821, 6831, 6841, 6851, 6861, 6871, 6881, 6891, 6901, 6911, 6921, 6931, 6941, 6951, 6961, 6971, 6981, 6991, 7001, 7011, 7021, 7031, 7041, 7051, 7061, 7071, 7081, 7091, 7101, 7111, 7121, 7131, 7141, 7151, 7161, 7171, 7181, 7191, 7201, 7211, 7221, 7231, 7241, 7251, 7261, 7271, 7281, 7291, 7301, 7311, 7321, 7331, 7341, 7351, 7361, 7371, 7381, 7391, 7401, 7411, 7421, 7431, 7441, 7451, 7461, 7471, 7481, 7491, 7501, 7511, 7521, 7531, 7541, 7551, 7561, 7571, 7581, 7591, 7601, 7611, 7621, 7631, 7641, 7651, 7661, 7671, 7681, 7691, 7701, 7711, 7721, 7731, 7741, 7751, 7761, 7771, 7781, 7791, 7801, 7811, 7821, 7831, 7841, 7851, 7861, 7871, 7881, 7891, 7901, 7911, 7921, 7931, 7941, 7951, 7961, 7971, 7981, 7991, 8001, 8011, 8021, 8031, 8041, 8051, 8061, 8071, 8081, 8091, 8010, 8011, 8012, 8013, 8014, 8015, 8016, 8017, 8018, 8019, 8020, 8021, 8022, 8023, 8024, 8025, 8026, 8027, 8028, 8029, 8030, 8031, 8032, 8033, 8034, 8035, 8036, 8037, 8038, 8039, 8040, 8041, 8042, 8043, 8044, 8045, 8046, 8047, 8048, 8049, 8050, 8051, 8052, 8053, 8054, 8055, 8056, 8057, 8058, 8059, 8060, 8061, 8062, 8063, 8064, 8065, 8066, 8067, 8068, 8069, 8070, 8071, 8072, 8073, 8074, 8075, 8076, 8077, 8078, 8079, 8080, 8081, 8082, 8083, 8084, 8085, 8086, 8087, 8088, 8089, 8090, 8091, 8092, 8093, 8094, 8095, 8096, 8097, 8098, 8099, 80100, 80101, 80102, 80103, 80104, 80105, 80106, 80107, 80108, 80109, 80110, 80111, 80112, 80113, 80114, 80115, 80116, 80117, 80118, 80119, 80120, 80121, 80122, 80123, 80124, 80125, 80126, 80127, 80128, 80129, 80130, 80131, 80132, 80133, 80134, 80135, 80136, 80137, 80138, 80139, 80140, 80141, 80142, 80143, 80144, 80145, 80146, 80147, 80148, 80149, 80150, 80151, 80152, 80153, 80154, 80155, 80156, 80157, 80158, 80159, 80160, 80161, 80162, 80163, 80164, 80165, 80166, 80167, 80168, 80169, 80170, 80171, 80172, 80173, 80174, 80175, 80176, 80177, 80178, 80179, 80180, 80181, 80182, 80183, 80184, 80185, 80186, 80187, 80188, 80189, 80190, 80191, 80192, 80193, 80194, 80195, 80196, 80197, 80198, 80199, 80200, 80201, 80202, 80203, 80204, 80205, 80206, 80207, 80208, 80209, 80210, 80211, 80212, 80213, 80214, 80215, 80216, 80217, 80218, 80219, 80220, 80221, 80222, 80223, 80224, 80225, 80226, 80227, 80228, 80229, 80230, 80231, 80232, 80233, 80234, 80235, 80236, 80237, 80238, 80239, 80240, 80241, 80242, 80243, 80244, 80245, 80246, 80247, 80248, 80249, 80250, 80251, 80252, 80253, 80254, 80255, 80256, 80257, 80258, 80259, 80260, 80261, 80262, 80263, 80264, 80265, 80266, 80267, 80268, 80269, 80270, 80271, 80272, 80273, 80274, 80275, 80276, 80277, 80278, 80279, 80280, 80281, 80282, 80283, 80284, 80285, 80286, 80287, 80288, 80289, 80290, 80291, 80292, 80293, 80294, 80295, 80296, 80297, 80298, 80299, 80300, 80301, 80302, 80303, 80304, 80305, 80306, 80307, 80308, 80309, 80310, 80311, 80312, 80313, 80314, 80315, 80316, 80317, 80318, 80319, 80320, 80321, 80322, 80323, 80324, 80325, 80326, 80327, 80328, 80329, 80330, 80331, 80332, 80333, 80334, 80335, 80336, 80337, 80338, 80339, 80340, 80341, 80342, 80343, 80344, 80345, 80346, 80347, 80348, 80349, 80350, 80351, 80352, 80353, 80354, 80355, 80356, 80357, 80358, 80359, 80360, 80361, 80362, 80363, 80364, 80365, 80366, 80367, 80368, 80369, 80370, 80371, 80372, 80373, 80374, 80375, 80376, 80377, 80378, 80379, 80380, 80381, 80382, 80383, 80384, 80385, 80386, 80387, 80388, 80389, 80390, 80391, 80392, 80393, 80394, 80395, 80396, 80397, 80398, 80399, 80400, 80401, 80402, 80403, 80404, 80405, 80406, 80407, 80408, 80409, 80410, 80411, 80412, 80413, 80414, 80415, 80416, 80417, 80418, 80419, 80420, 80421, 80422, 80423, 80424, 80425, 80426, 80427, 80428, 80429, 80430, 80431, 80432, 80433, 80434, 80435, 80436, 80437, 80438, 80439, 80440, 80441, 80442, 80443, 80444, 80445, 80446, 80447, 80448, 80449, 80450, 80451, 80452, 80453, 80454, 80455, 80456, 80457, 80458, 80459, 80460, 80461, 80462, 80463, 80464, 80465, 80466, 80467, 80468, 80469, 80470, 80471, 80472, 80473, 80474, 80475, 80476, 80477, 80478, 80479, 80480, 80481, 80482, 80483, 80484, 80485, 80486, 80487, 80488, 80489, 80490, 80491, 80492, 80493, 80494, 80495, 80496, 80497, 80498, 80499, 80500, 80501, 80502, 80503, 80504, 80505, 80506, 80507, 80508, 80509, 80510, 80511, 80512, 80513, 80514, 80515, 80516, 80517, 80518, 80519, 80520, 80521, 80522, 80523, 80524, 80525, 80526, 80527, 80528, 80529, 80530, 80531, 80532, 80533, 80534, 80535, 80536, 80537, 80538, 80539, 80540, 80541, 80542, 80543, 80544, 80545, 80546, 80547, 80548, 80549, 80550, 80551, 80552, 80553, 80554, 80555, 80556, 80557, 80558, 80559, 80560, 80561, 80562, 80563, 80564, 80565, 80566, 80567, 80568, 80569, 80570, 80571, 80572, 80573, 80574, 80575, 80576, 80577, 80578, 80579, 80580, 80581, 80582, 80583, 80584, 80585, 80586, 80587, 80588, 80589, 80590, 80591, 80592, 80593, 80594, 80595, 80596, 80597, 80598, 80599, 80600, 80601, 80602, 80603, 80604, 80605, 80606, 80607, 80608, 80609, 80610, 80611, 80612, 80613, 80614, 80615, 80616, 80617, 80618, 80619, 80620, 80621, 80622, 80623, 80624, 80625, 80626, 80627, 80628, 80629, 80630, 80631, 80632, 80633, 80634, 80635, 80636, 80637, 80638, 80639, 80640, 80641, 80642, 80643, 80644, 80645, 80646, 80647, 80648, 80649, 80650, 80651, 80652, 80653, 80654, 80655, 80656, 80657, 80658, 80659, 80660, 80661, 80662, 80663, 80664, 80665, 80666, 80667, 80668, 80669, 80670, 80671, 80672, 80673, 80674, 80675, 80676, 80677, 80678, 80679, 80680, 80681, 80682, 80683, 80684, 80685, 80686, 80687, 80688, 80689, 80690, 80691, 80692, 80693, 80694, 80695, 80696, 80697, 80698, 80699, 80700, 80701, 80702, 80703, 80704, 80705, 80706, 80707, 80708, 80709, 80710, 80711, 80712, 80713, 80714, 80715, 80716, 80717, 80718, 80719, 80720, 80721, 80722, 80723, 80724, 80725, 80726, 80727, 80728, 80729, 80730, 80731, 80732, 80733, 80734, 80735, 80736, 80737, 80738, 80739, 80740, 80741, 80742, 80743, 80744, 80745, 80746, 80747, 80748, 80749, 80750, 80751, 80752, 80753, 80754, 80755, 80756, 80757, 80758, 80759, 80760, 80761, 80762, 80763, 80764, 80765, 80766, 80767, 80768, 80769, 80770, 80771, 80772, 80773, 80774, 80775, 80776, 80777, 80778, 80779, 80780, 80781, 80782, 80783, 80784, 80785, 80786, 80787, 80788, 80789, 80790, 80791, 80792, 80793, 80794, 80795, 80796, 80797, 80798, 80799, 80800, 80801, 80802, 80803, 80804, 80805, 80806, 80807, 80808, 80809, 80810, 80811, 80812, 80813, 80814, 80815, 80816, 80817, 80818, 80819, 80820, 80821, 80822, 80823, 80824, 80825, 80826, 80827, 80828, 80829, 80830, 80831, 80832, 80833, 80834, 80835, 80836, 80837, 80838, 80839, 80840, 80841, 80842, 80843, 80844, 80845, 80846, 80847, 80848, 80849, 80850, 80851, 80852, 80853, 80854, 80855, 80856, 80857, 80858, 80859, 80860, 80861, 80862, 80863, 80864, 80865, 80866, 80867, 80868, 80869, 80870, 80871, 80872, 80873, 8

STIRLING.PL - Stirling Approximation

This program computes the log(10) of the factorial of a number in the range 1 .. about 10^{98} , using a version of the Stirling Approximation, modified to include the $1/12n$ correction. With this correction, the resulting values are very close to true factorials, and because this approximation uses a simple numerical equation, it is extremely quick and independent of the size of the input value. This makes it very useful for statistical programs.



The screenshot shows the WIN-PROLOG application window. The menu bar includes File, Edit, Search, Run, Options, Window, and Help. A sub-menu window titled "Console" is open, showing the following Prolog queries and their results:

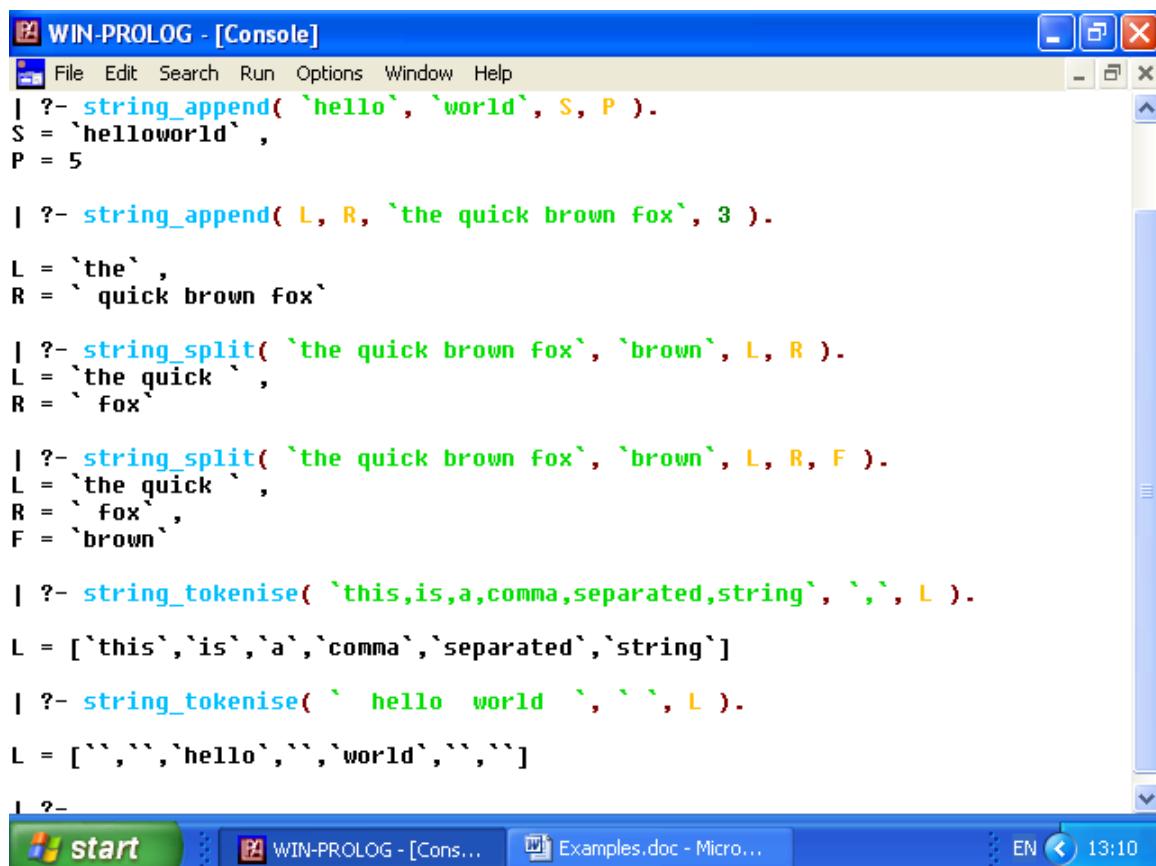
```
| ?- fact(69).  
1.71122329174949e98  
yes  
  
| ?- slow_fact(69).  
1.71122452428133e98  
yes  
  
| ?- test_fact(69).  
1.71122329174949e98  
1.71122452428133e98  
99.9999279736923%  
yes
```

Below the console window, a file browser window titled "c:\program files\win-prolog 4.3.2\examples\stirling.pl" is displayed, showing the source code of the Stirling Approximation program. The code includes a copyright notice and a detailed description of the program's purpose and performance.

Examples

STRINGS.PL - String Examples

386-PROLOG contains a special string data type, which provides it with exceptional text handling capabilities. More compact than conventional "char lists", and considerably more flexible than atoms, strings allow large amounts of text to be stored and processed as Prolog terms. This file contains examples to show some of the uses of the following: (1) The `cat/3` predicate, which can join and split strings and atoms at multiple known positions (2) The `find/3` predicate, which can search for text with or without output, and with or without case sensitivity (3) The `copy/2` predicate, which can be used to mop up and transfer remaining text after a call to `find/3` (4) The `</2` and `/>/2` predicates, which allow input or output predicates to read from and write to strings directly and (5) The `repeat/0`, `fail/0`, `!/0`, `integer_bound/3` and `findall/3` predicates, which provide support for backtracking.



```

WIN-PROLOG - [Console]
File Edit Search Run Options Window Help
| ?- string_append( `hello`, `world`, S, P ).  

S = `helloworld` ,  

P = 5  

| ?- string_append( L, R, `the quick brown fox` , 3 ).  

L = `the` ,  

R = ` quick brown fox`  

| ?- string_split( `the quick brown fox` , `brown` , L, R ).  

L = `the quick` ,  

R = ` fox`  

| ?- string_split( `the quick brown fox` , `brown` , L, R, F ).  

L = `the quick` ,  

R = ` fox` ,  

F = `brown`  

| ?- string_tokenise( `this,is,a,comma,separated,string` , ` ,` , L ).  

L = [ `this` , `is` , `a` , `comma` , `separated` , `string` ]  

| ?- string_tokenise( ` hello world ` , ` `` , L ).  

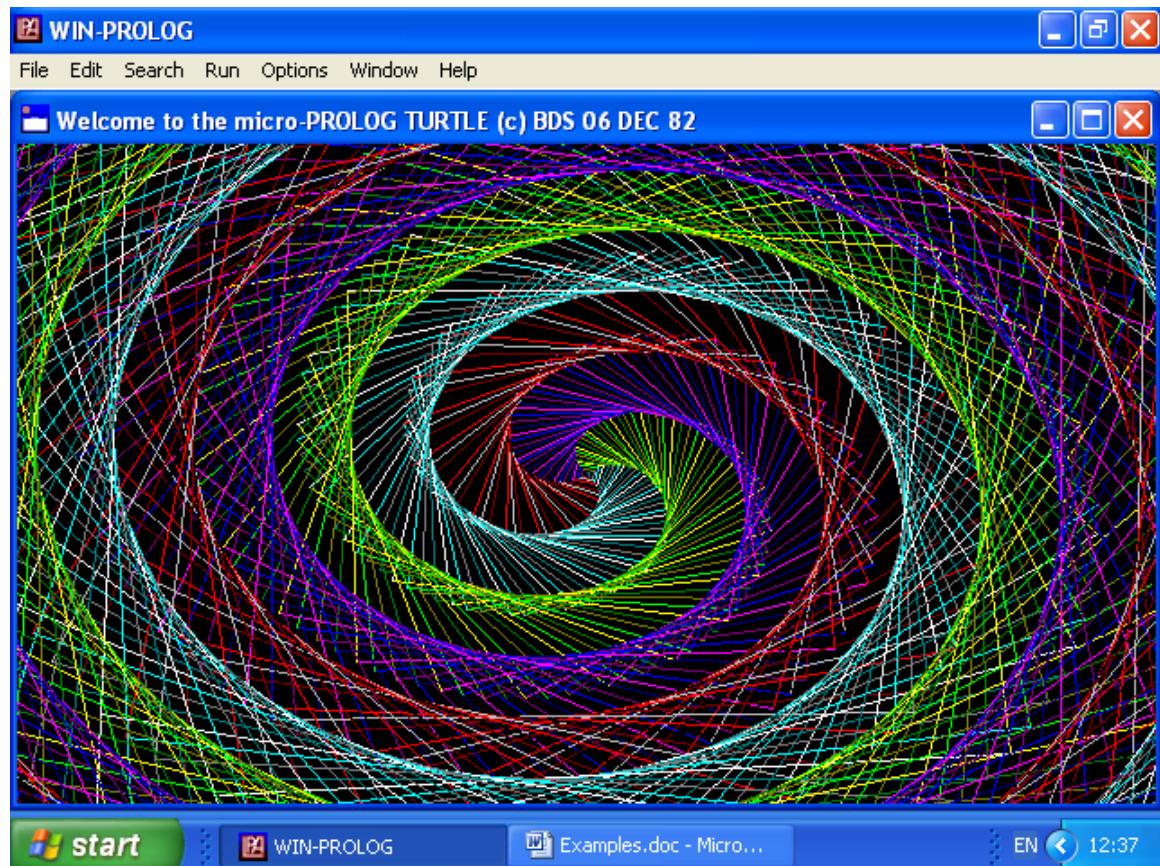
L = [ `` , `` , `hello` , `` , `world` , `` , `` ]
| ?-

```

Examples

TURTLE.PL - Turtle Graphics

This program implements the "Turtle" graphics originally made popular by the programming language "LOGO".



Examples

INDEX

I/O	45	is/2	26
~>/2	45	Key Interrogation.....	26
<~/2	45	keys	26
Auto Indent	6	keys/1	26
Benchmark	8	Keyword Completion	16
Calendar	9	listbox example	27
CARDS.DLL.....	10	LOGO.....	46
cat/3.....	45	Meal Selection	30
Clock.....	12	memory-mapped files	11
comma-separated file.....	14	menu example	31
copy/2	45	natural language	22
crc/3.....	17	natural language expert system....	22
CRC32 algorithm	17	network diagram.....	23
Definite Clause Grammar	18, 19	Nudge Bars	7
dialog example	20	paint message	24
Digital Clock	12	parse tree	19
Easter Sunday	21	Perpetual Calendar	9
enhanced metafile	33	Pie-Chart.....	34
expert system	22	prime numbers.....	40, 42, 43
factorial.....	44	Read Comments.....	15
fail/0	45	read/1.....	15
find/3.....	11, 14, 45	repaint.....	24
findall/3.....	43, 45	repeat/0.....	45
gfx/n.....	33	Reversi Game.....	35
integer_bound/3	43, 45	Rounding	37
interrupt timer	12	Scroll Bars.....	7

Examples

Semi-Modal Dialog	39	time/4	9, 28
SHA-256.....	40	Towers of Hanoi	25
Shared Access Read-Only File.....	41	Track Bars.....	7
Sieve of Erasthatones.....	42, 43	Travelling Salesman	38
sndmsg/5	13, 32	Tree Drawing	19
statistical.....	44	Turtle Graphics	46
Stirling Approximation.....	44	winapi/4	33
String Examples.....	45	windows messaging.....	13, 32
Strip Bars	7	WINMM.DLL.....	29
time/2	28	wmf file.....	33

Examples