4.900

Prolog Examples

by Rebecca Shalfield

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.

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**

13 July, 2004

Contents

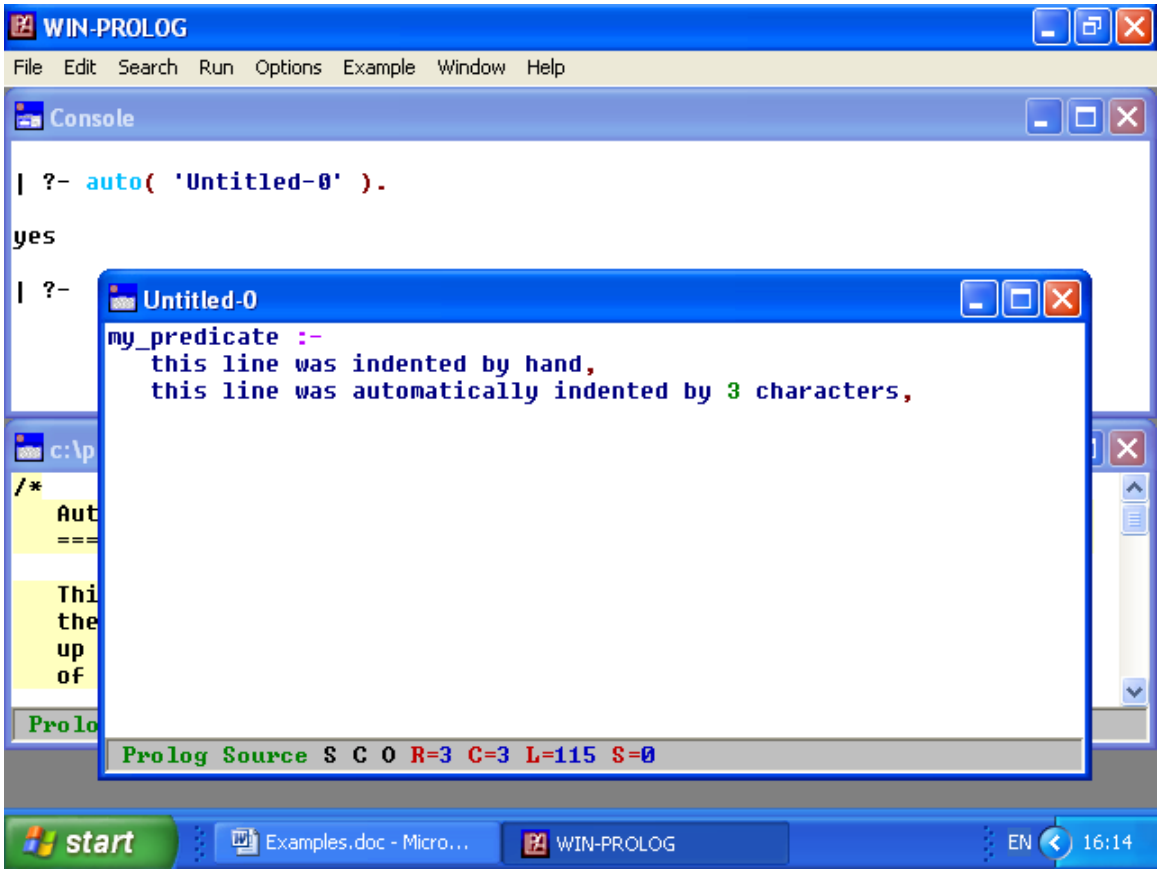
WIN-PROLOG EXAMPLE FILES

Introduction

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

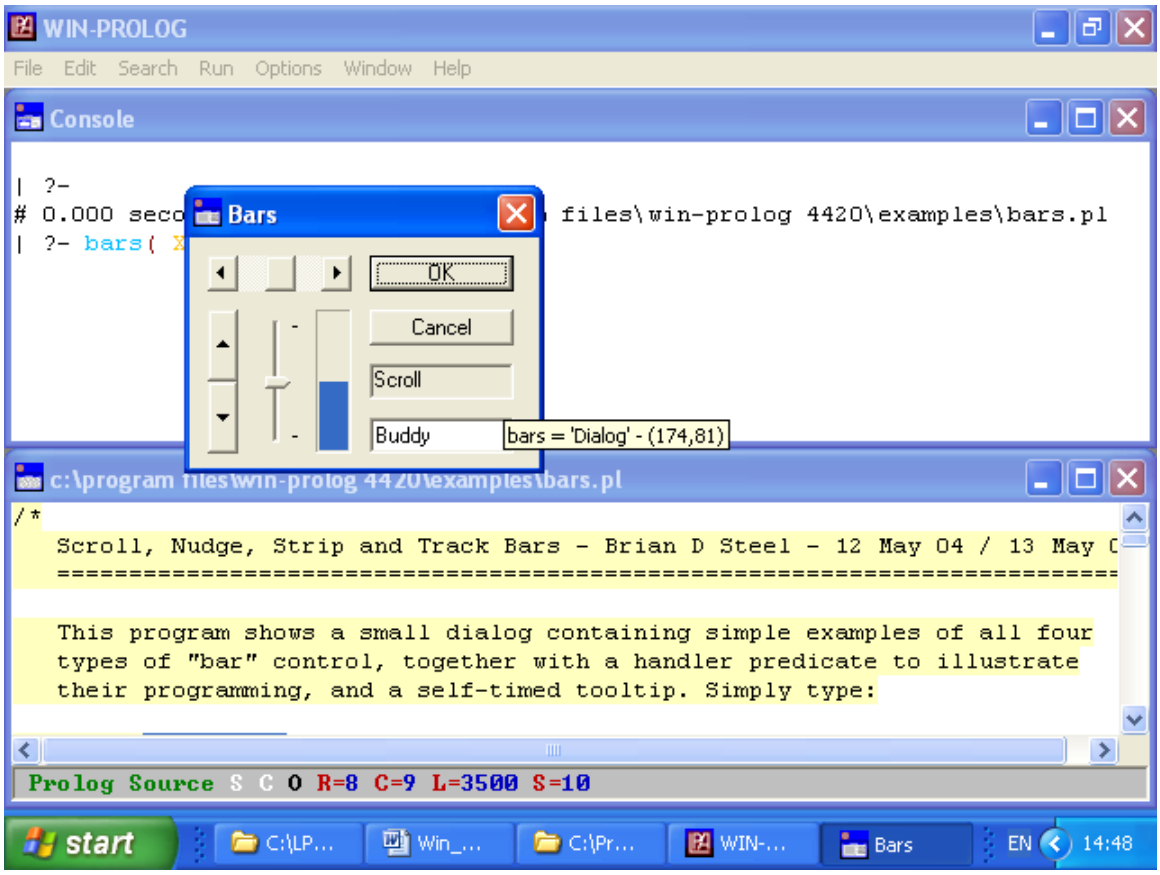
AUTO.PL - Auto Indent for the Editor

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.



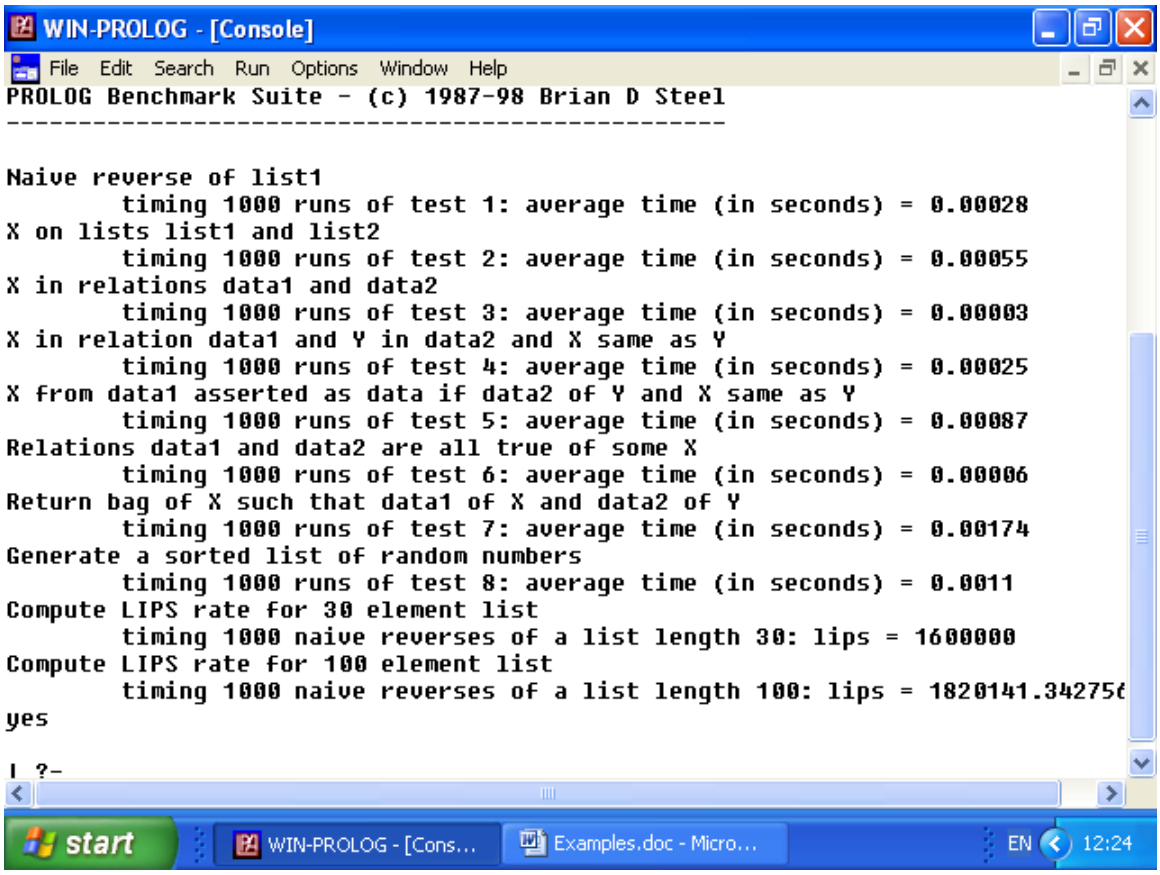
BARS.PL – Scroll, Nudge, Strip and Track Bars

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.



BENCHMRK.PL - Prolog Benchmark Suite

This program is a PROLOG Benchmark Suite.

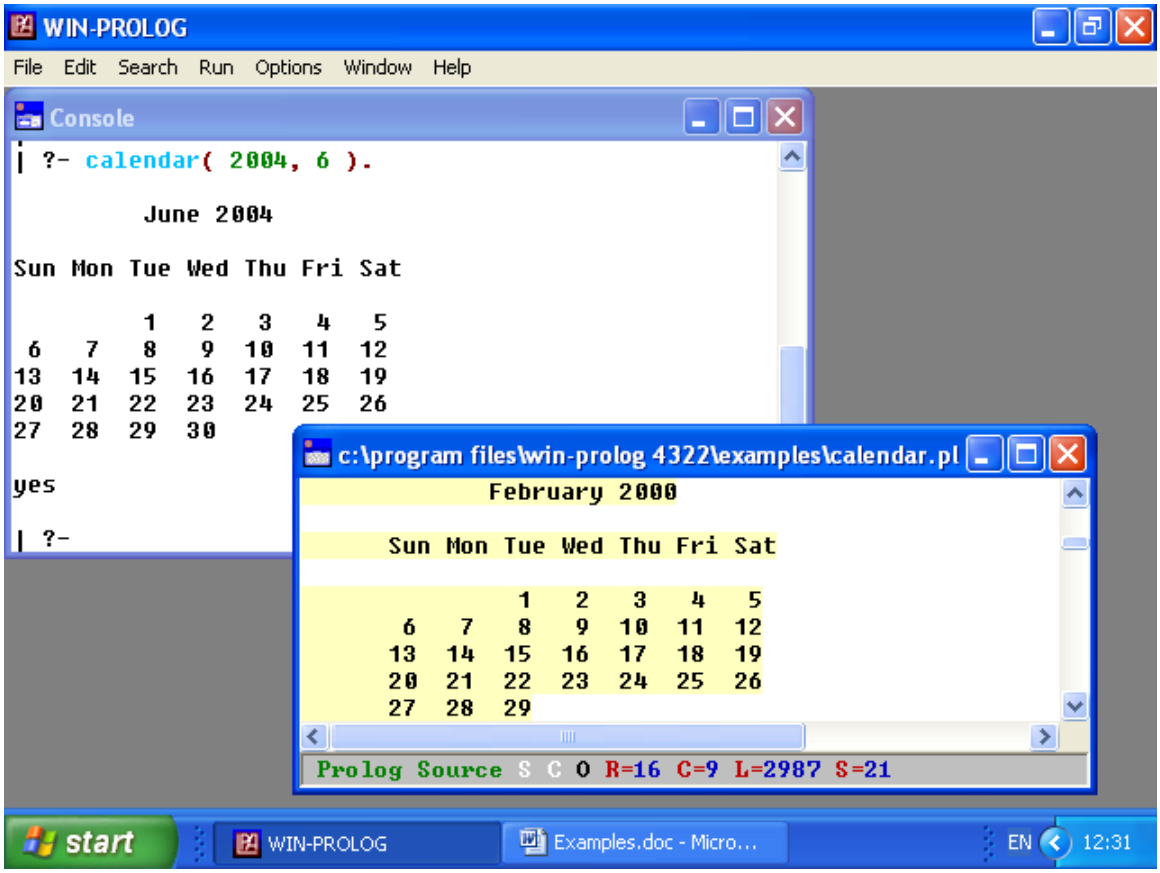


Examples

CALENDAR.PL - Perpetual Calendar

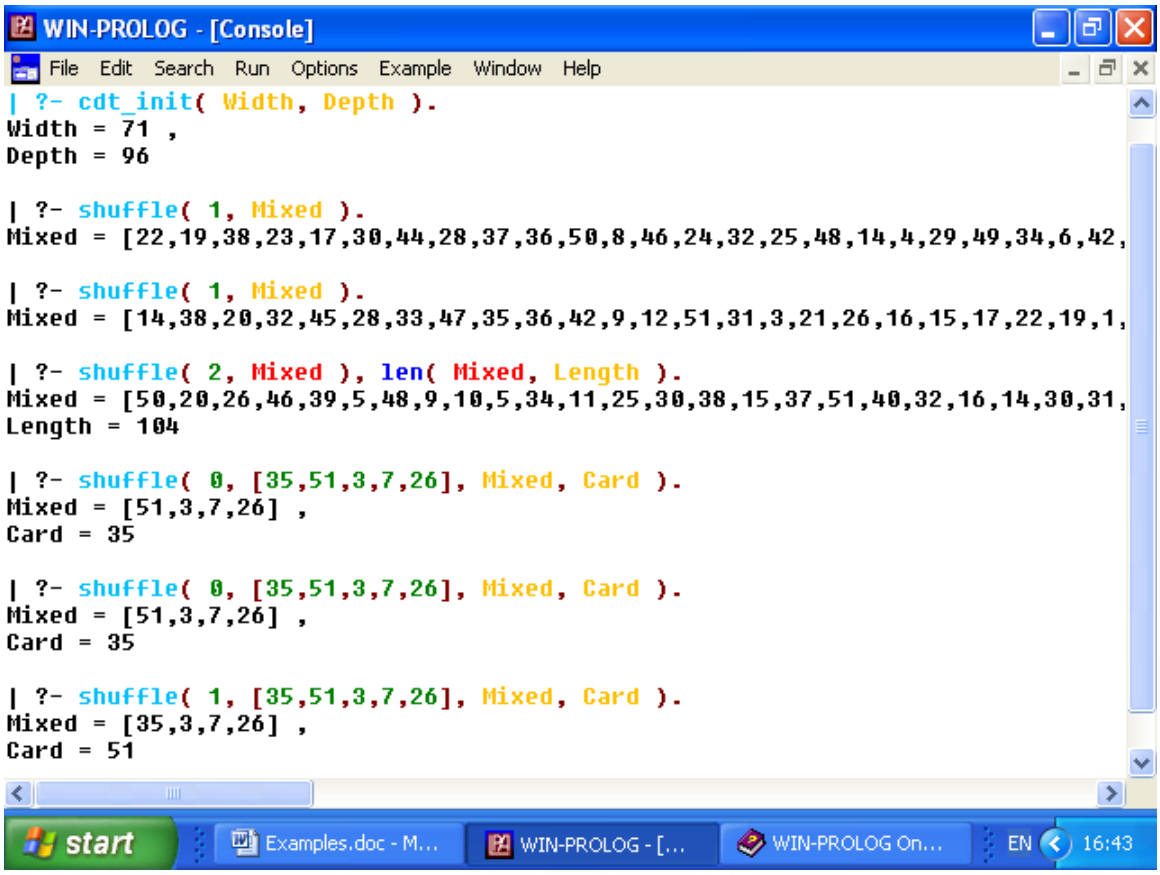
This simple program illustrates the use of the time/4 predicate for generating day numbers from dates and vice versa, allowing a very simple program to generate correct calendars for any month from January 1600 (1600,1) onwards, according to the Gregorian calendar.

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.



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.



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(Ptrn, Olds, News, Mode).

where:

Ptrn = 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:

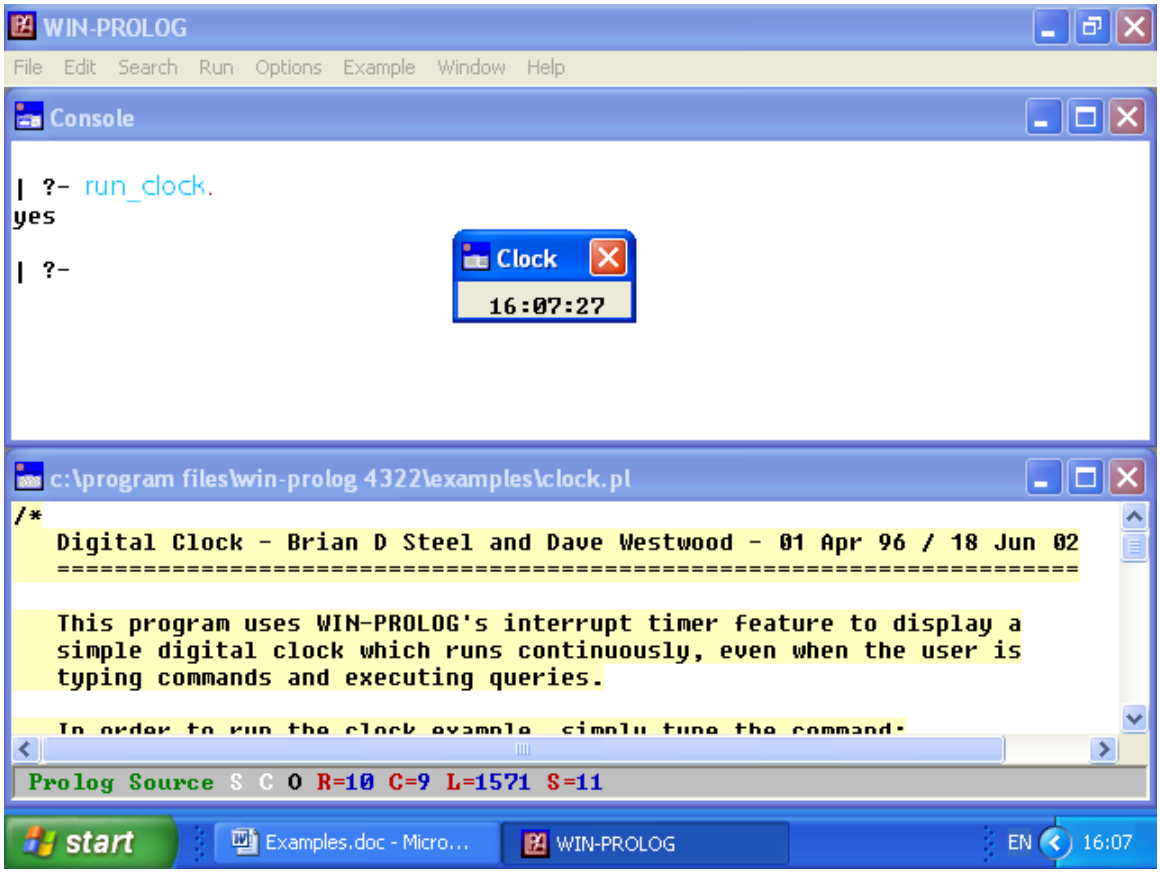
0 - perform an exact case (case-sensitive) search, and replace with exact given string

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)

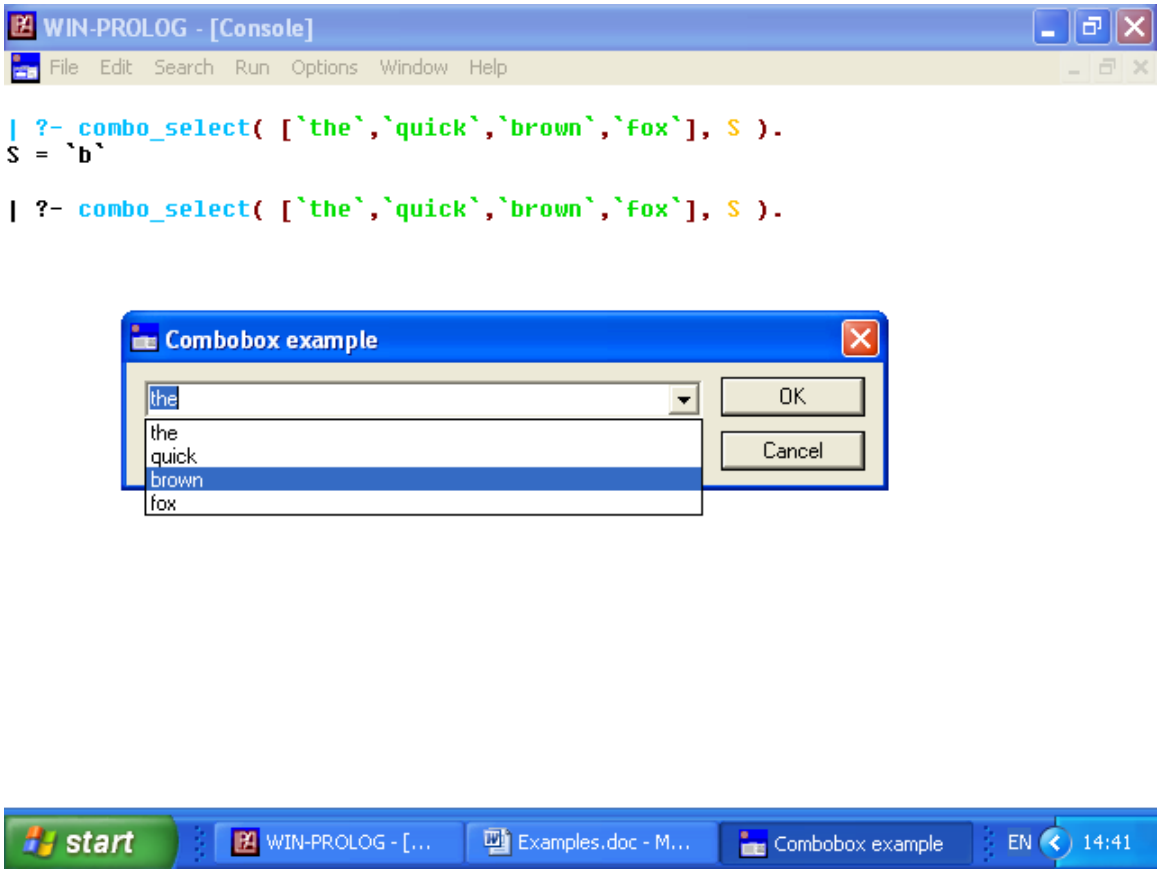
CLOCK.PL - Digital Clock

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.



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.

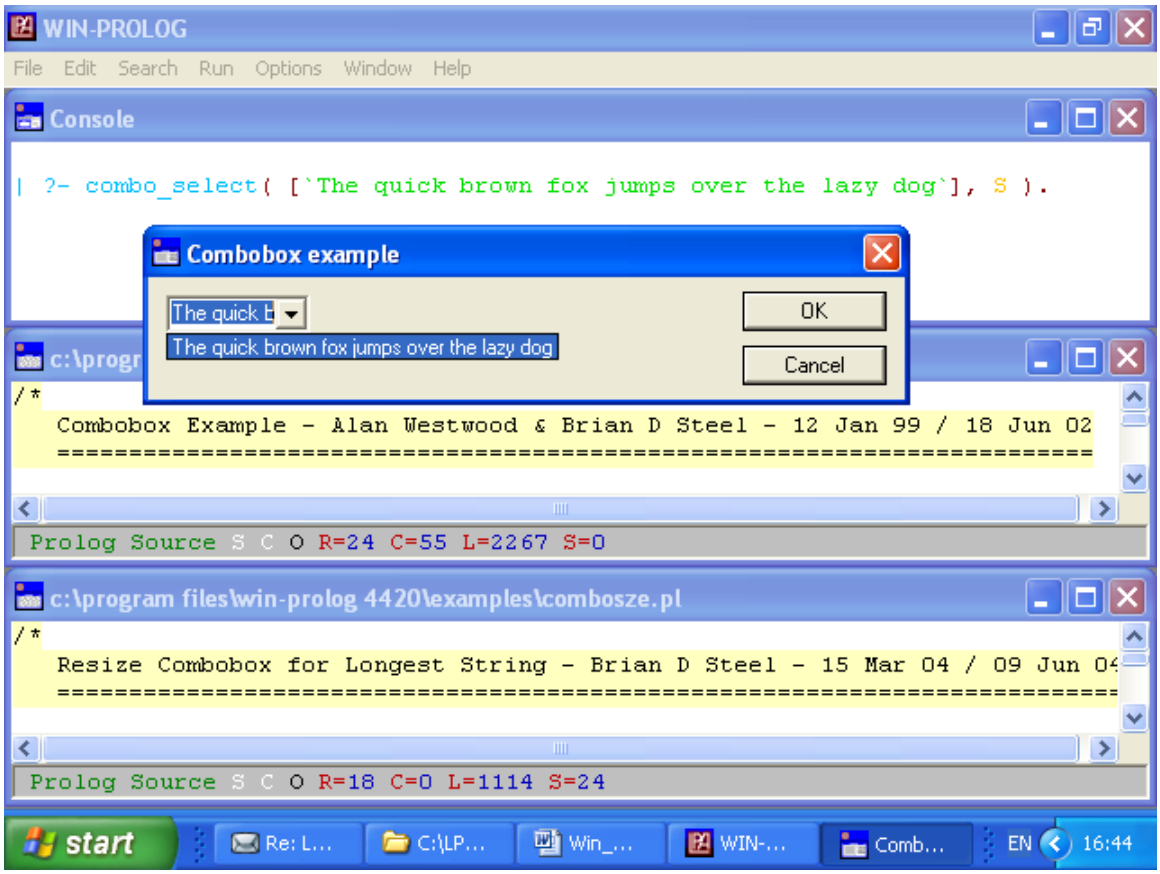


COMBOSZE.PL – Resize Combobox for Longest String

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.

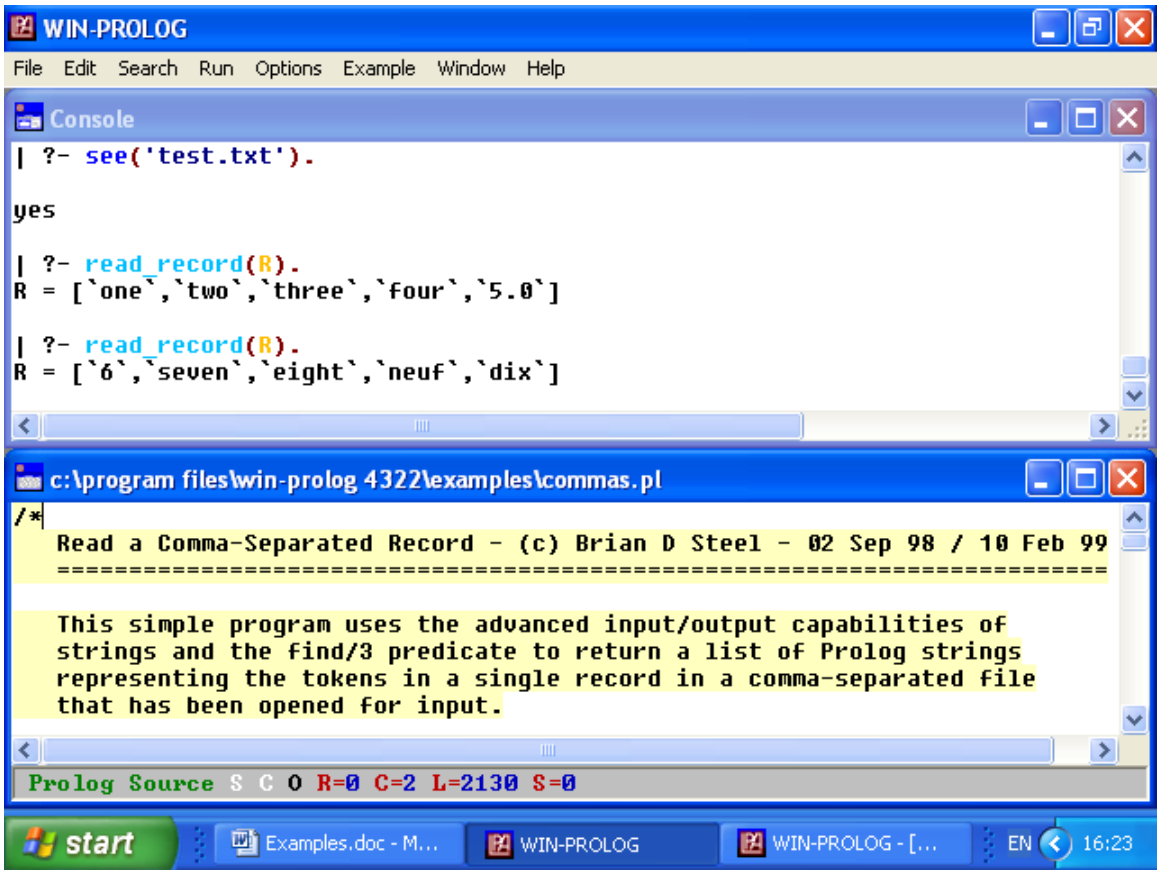
This program could, for example, be combined with COMBOBOX.PL:

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



COMMAS.PL - Read a Comma-Separated Record

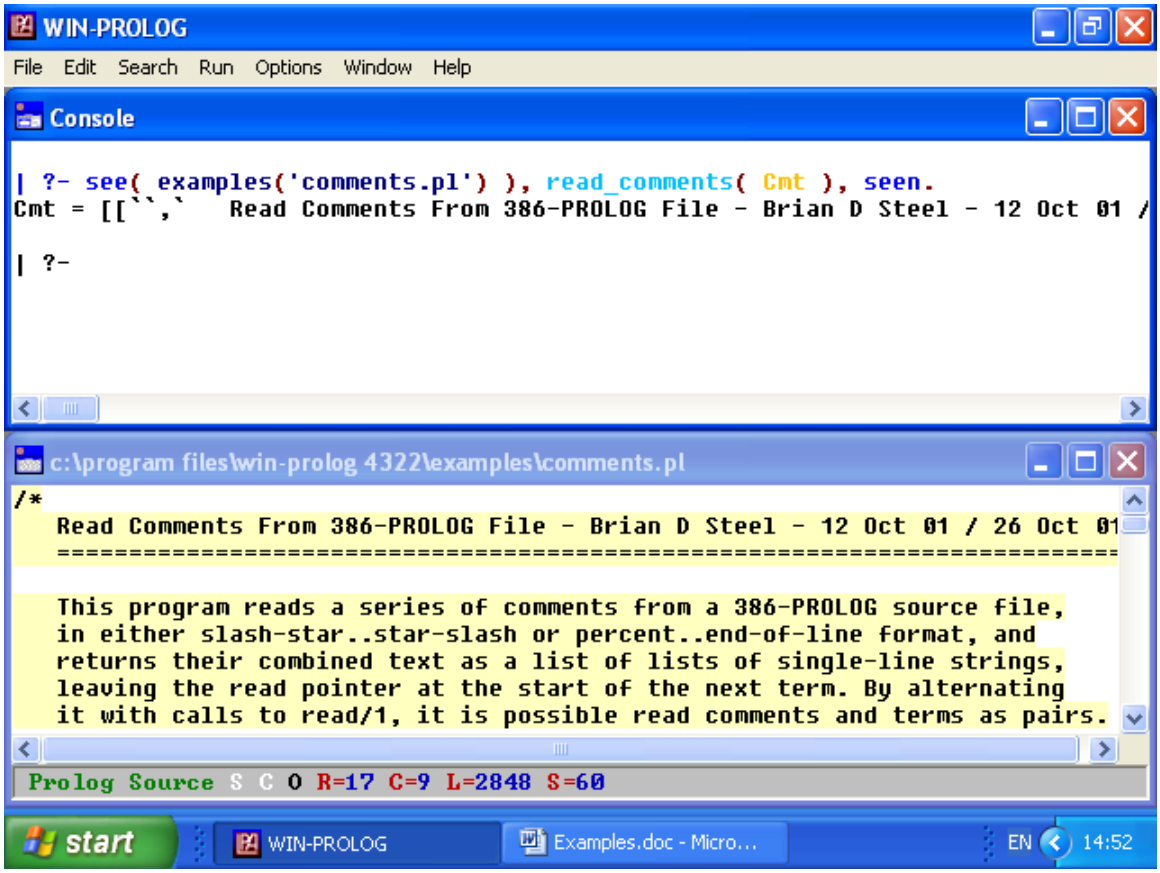
This simple program uses the advanced input/output capabilities of strings and the find/3 predicate to return a list of Prolog strings representing the tokens in a single record in a comma-separated file that has been opened for input.



COMMENTS.PL – Read Comments from 386-PROLOG File

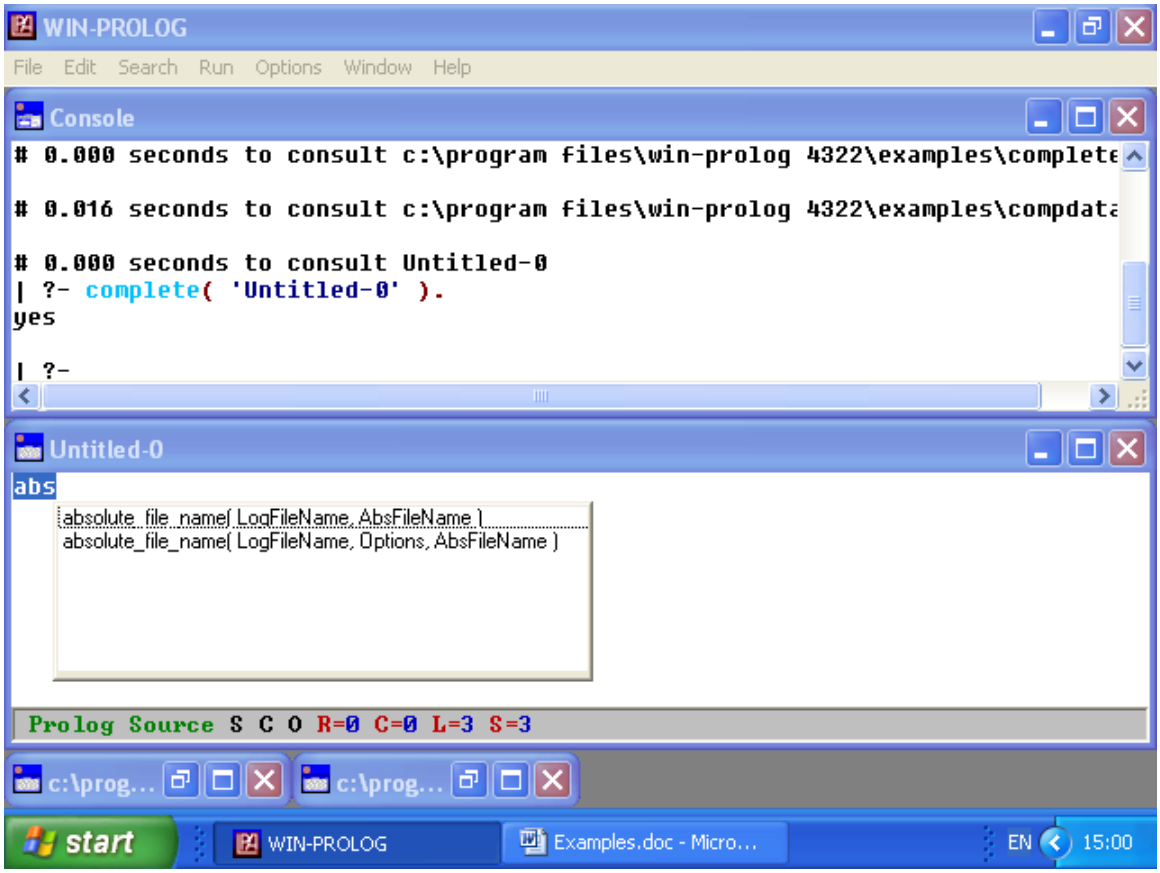
This program reads a series of comments from a 386-PROLOG source file, in either slash-star..star-slash or percent..end-of-line format, and returns their combined text as a list of lists of single-line strings, leaving the read pointer at the start of the next term. By alternating it with calls to read/1,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



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.



CRC_CHOP.PL - Create CRC32 Identical Files

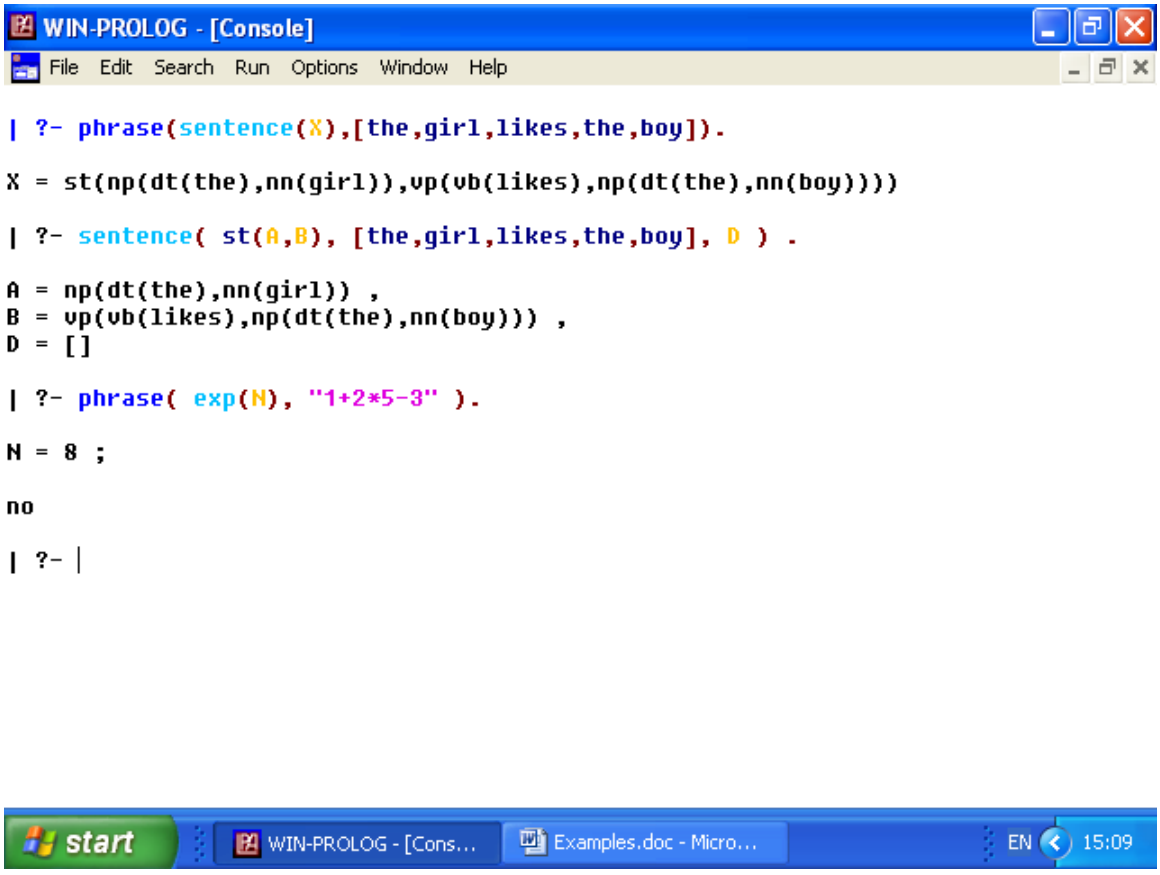
This program demonstrates both a strength and a weakness of the CRC32 algorithm, as implemented in the crc/3 predicate, in that it is relatively easy to create any number of different files with exactly the same CRC32.

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

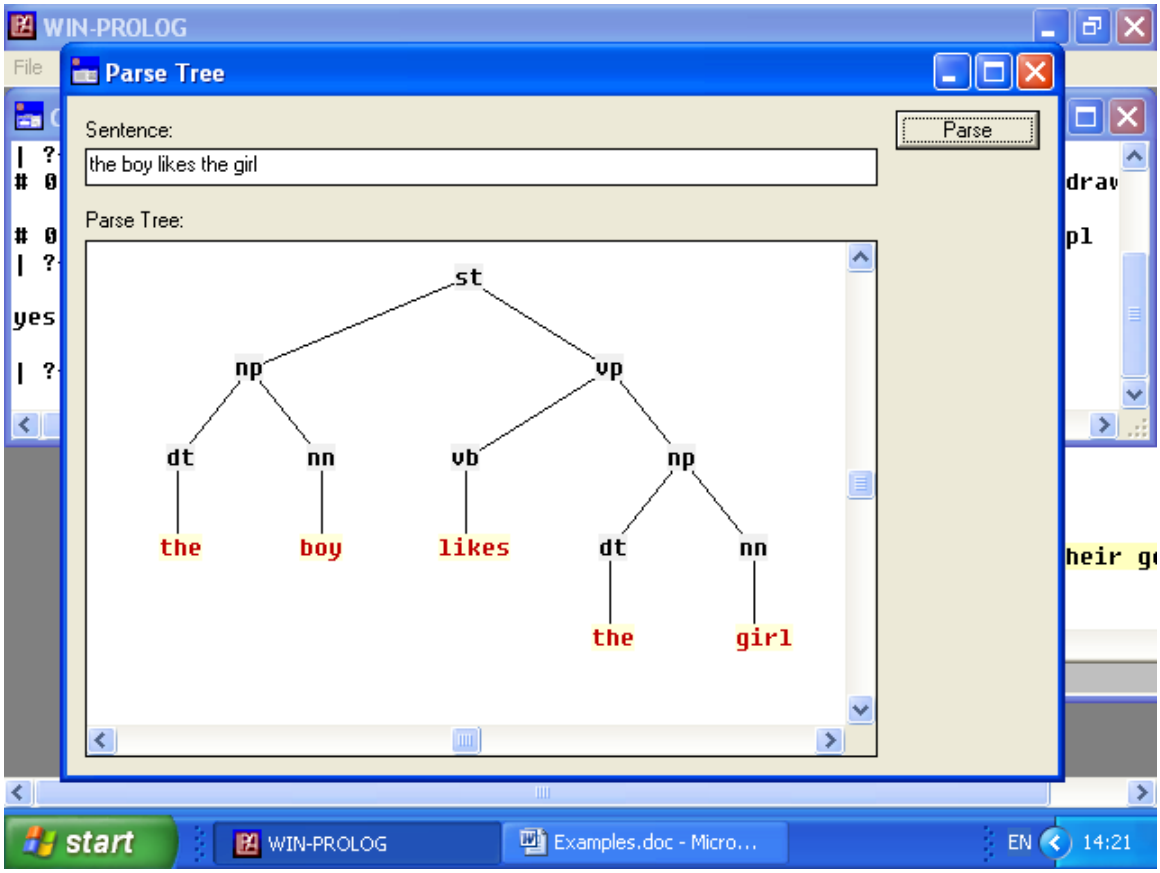
This file contains two examples of using Definite Clause Grammars.



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:

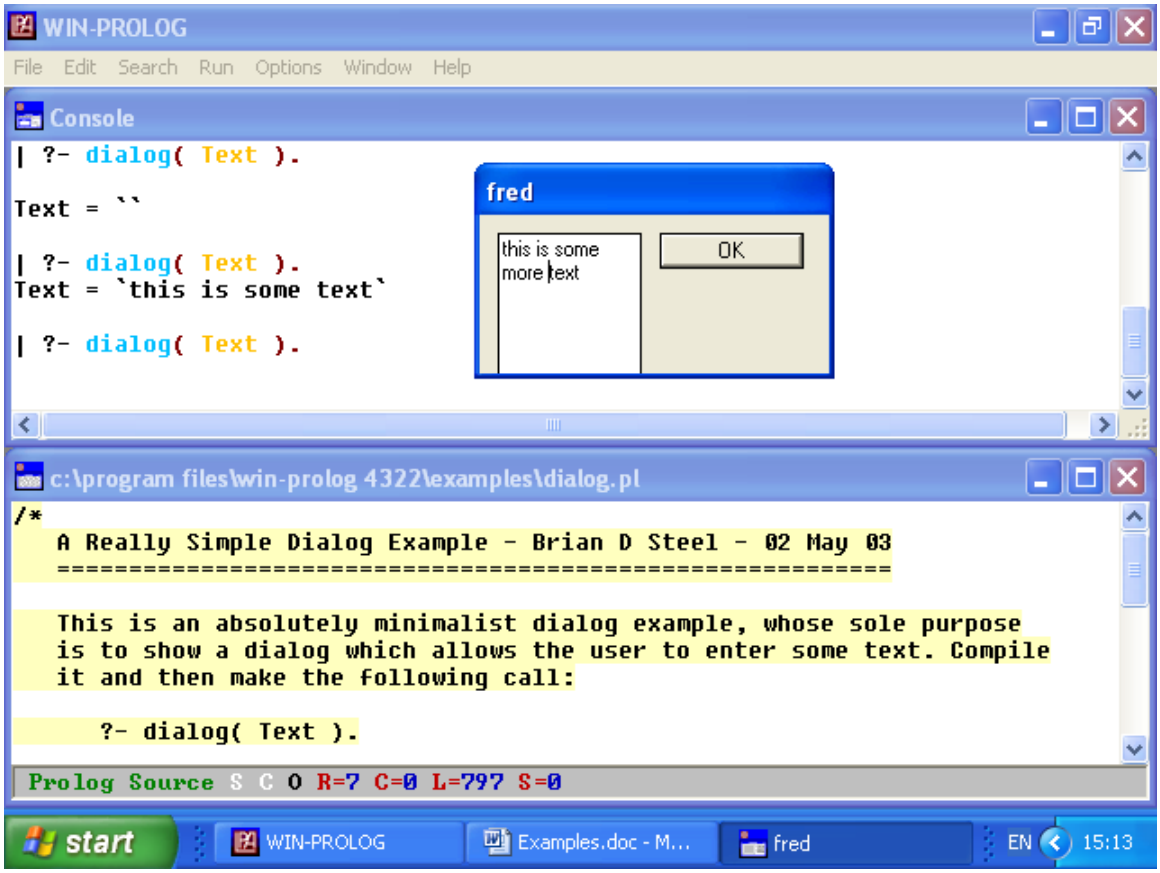
```
sentence( Tree ) -->
   ...
```



Examples

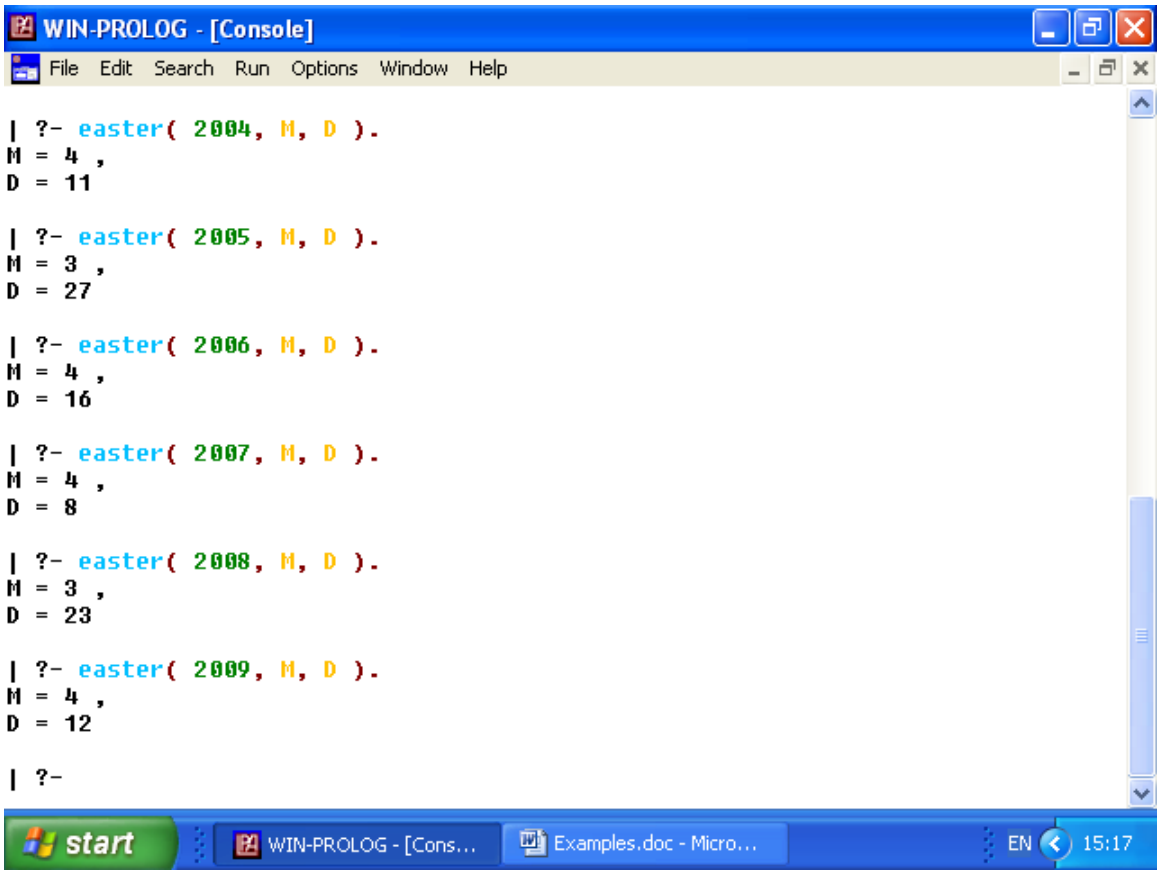
DIALOG.PL - A Really Simple Dialog Example

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



EASTER.PL – Finding Easter Sunday

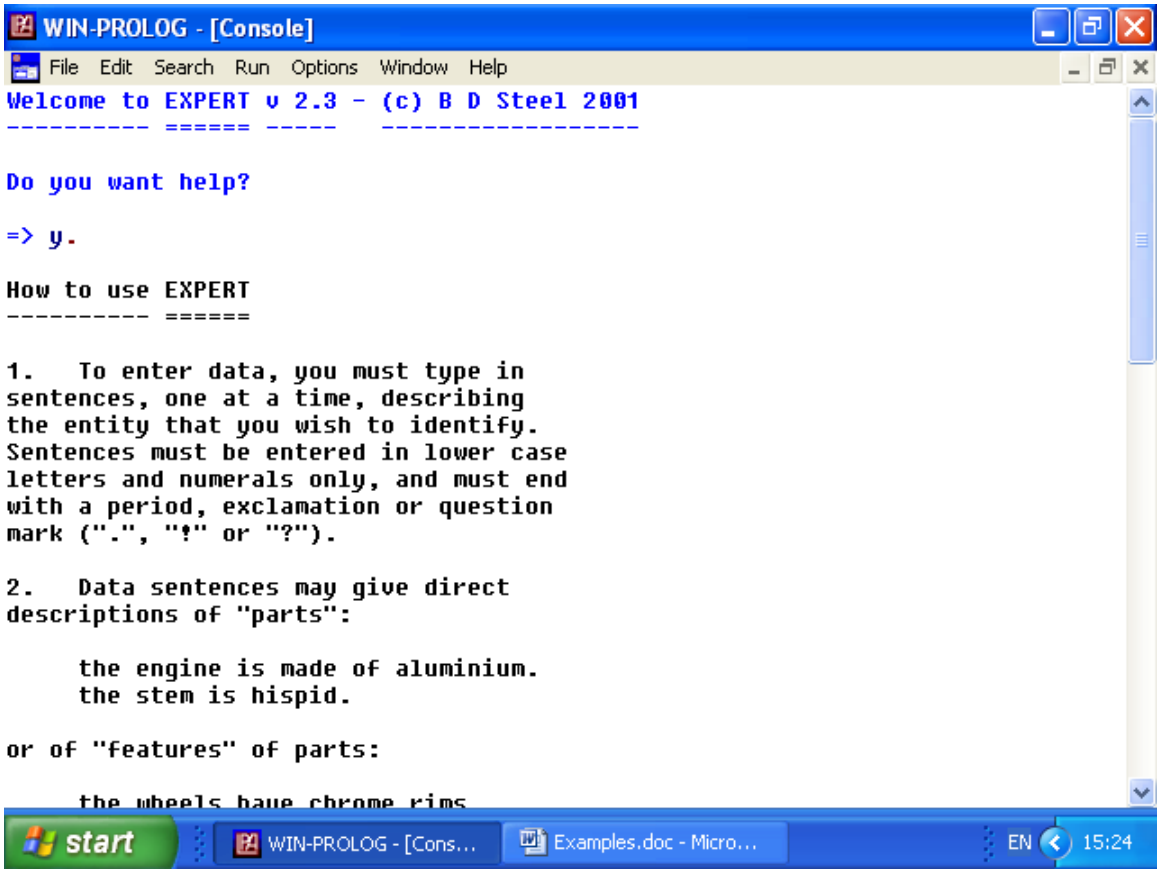
Given a year number in the Gregorian calendar, this program uses the 'official' algorithm for computing the date of Easter in that year.



Examples

EXPERT.PL – A Natural Language Expert System

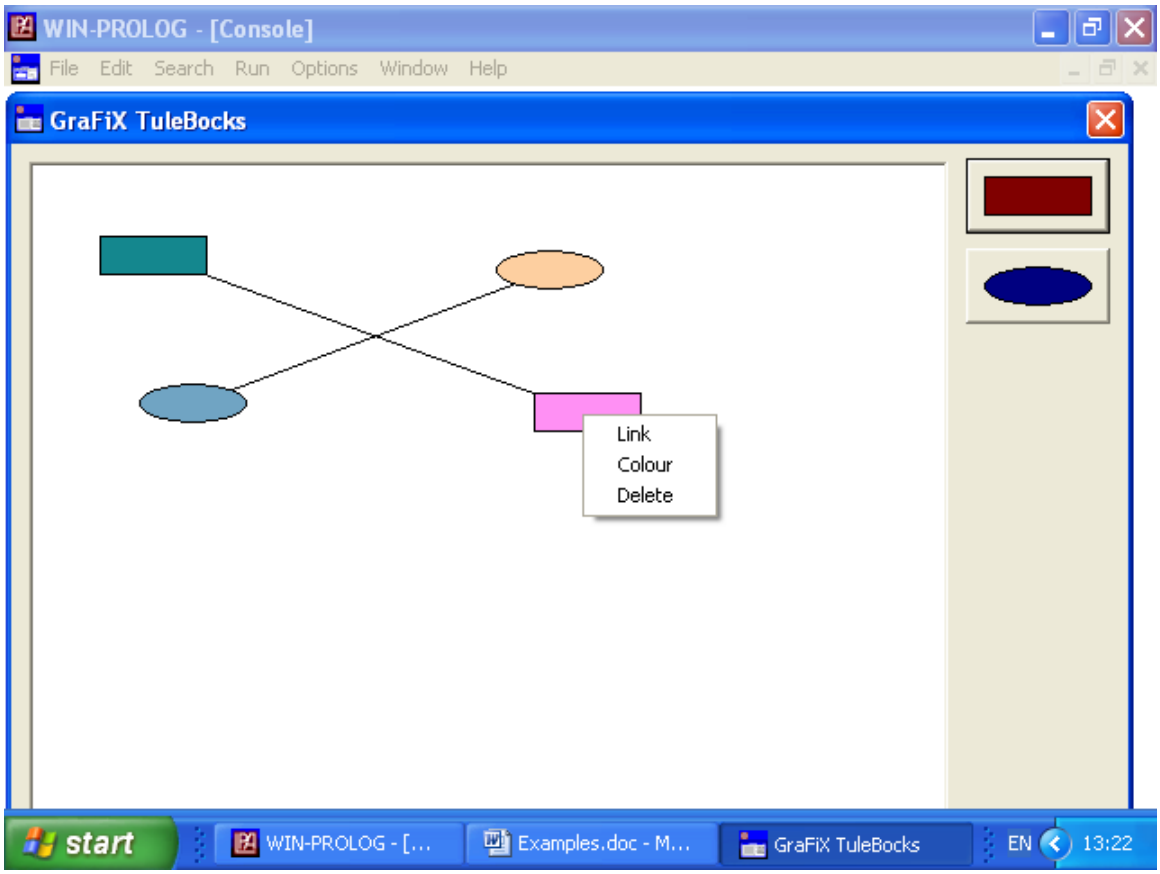
A good example of a simple natural language expert system.



GFX_TOOL.PL - Graphics Tool Demonstration

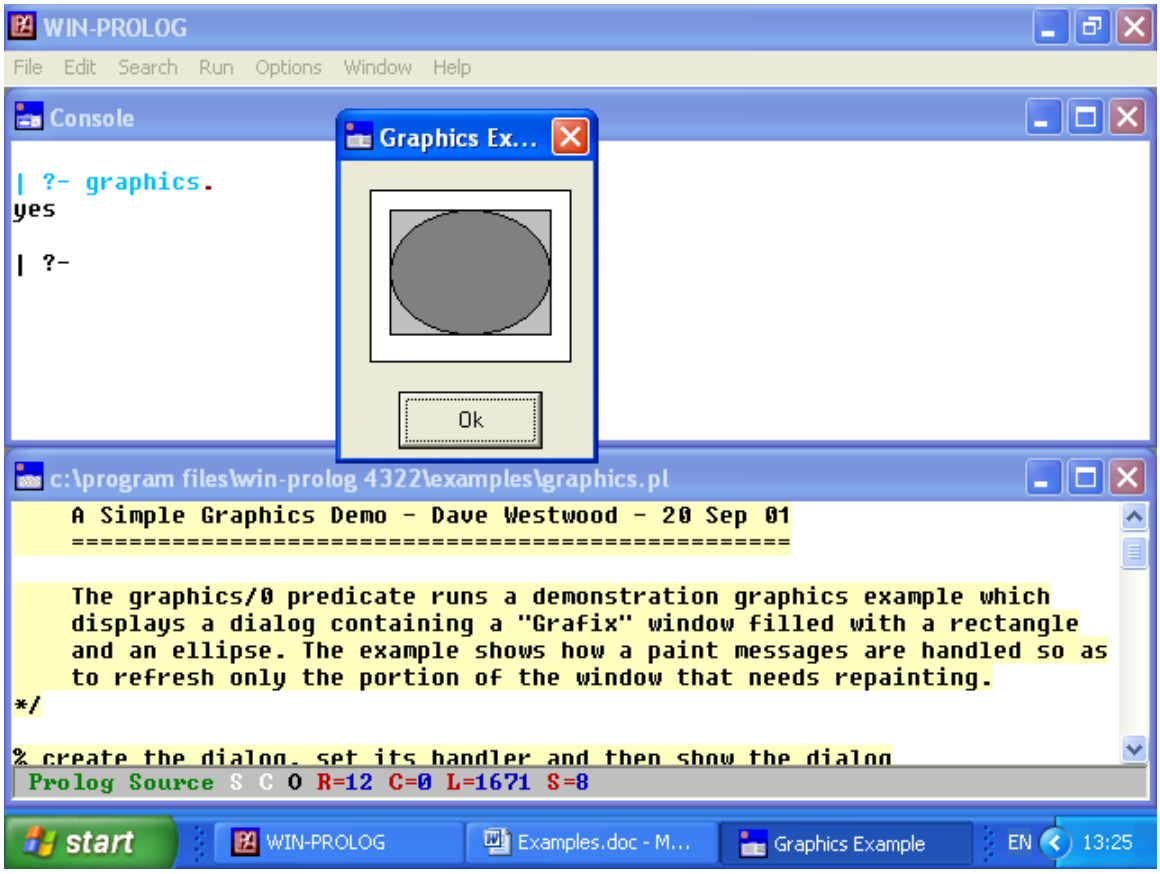
The purpose of this program is to show how to write a simple graphics tool, in which shapes can be created, moved, modified and linked.

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.



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 a paint messages are handled so as to refresh only the portion of the window that needs repainting.



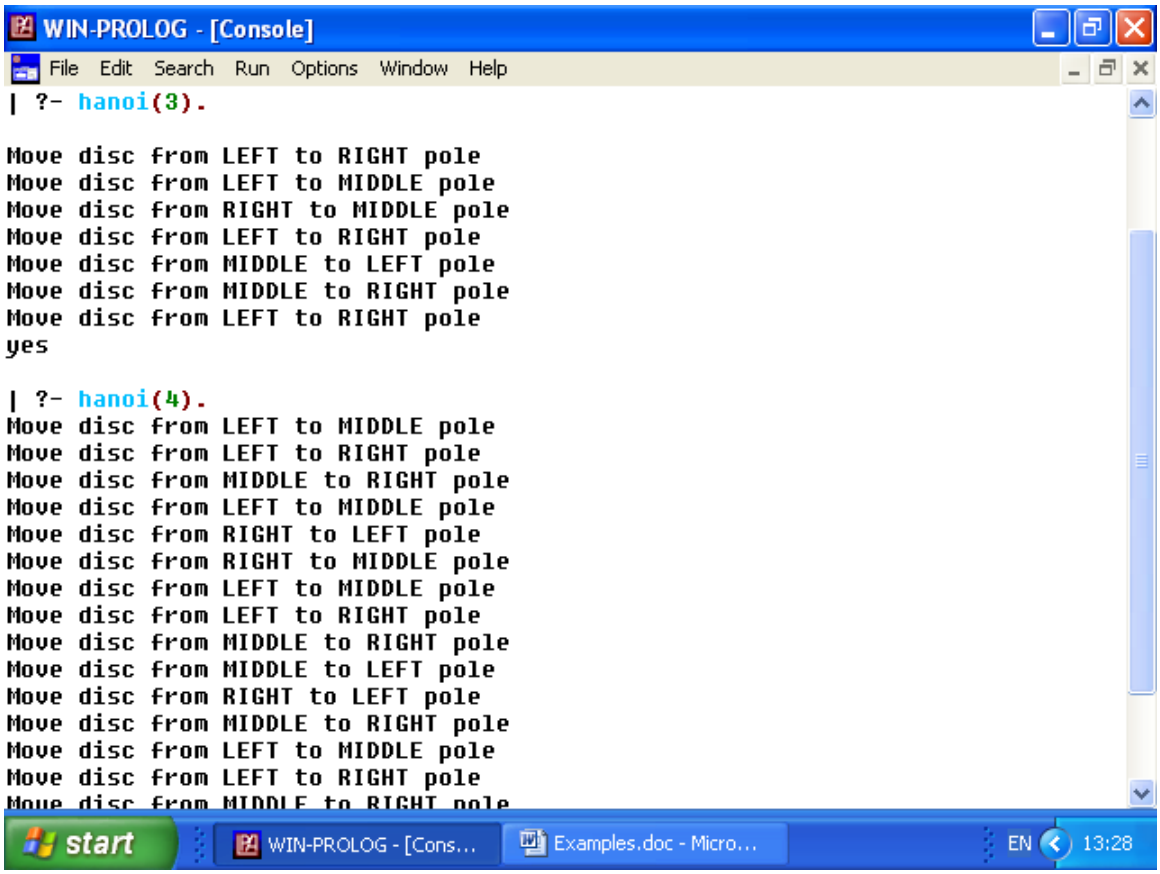
HANOI.PL - The Towers of Hanoi

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.

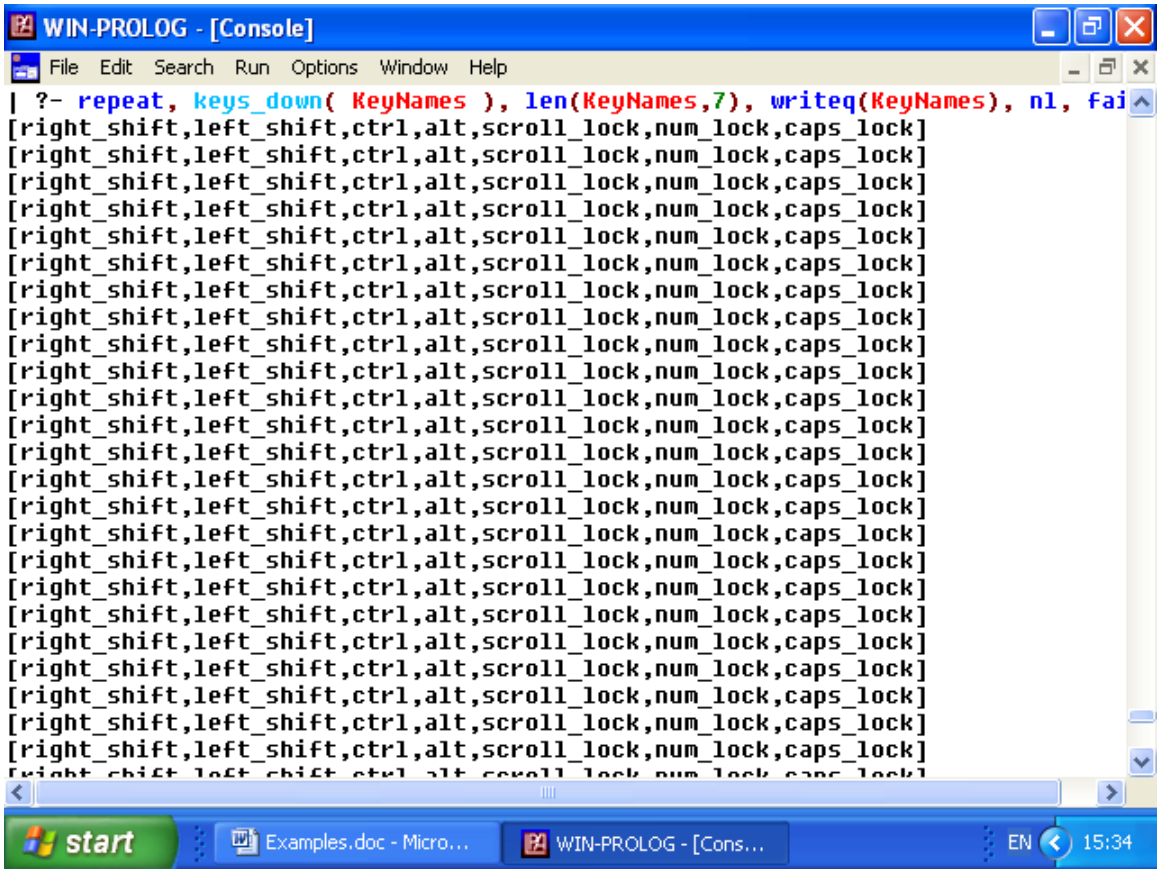
This is a classic candidate for recursion: figure out how to do one step, as described above, and then trust a recursive call to handle the rest.



Examples

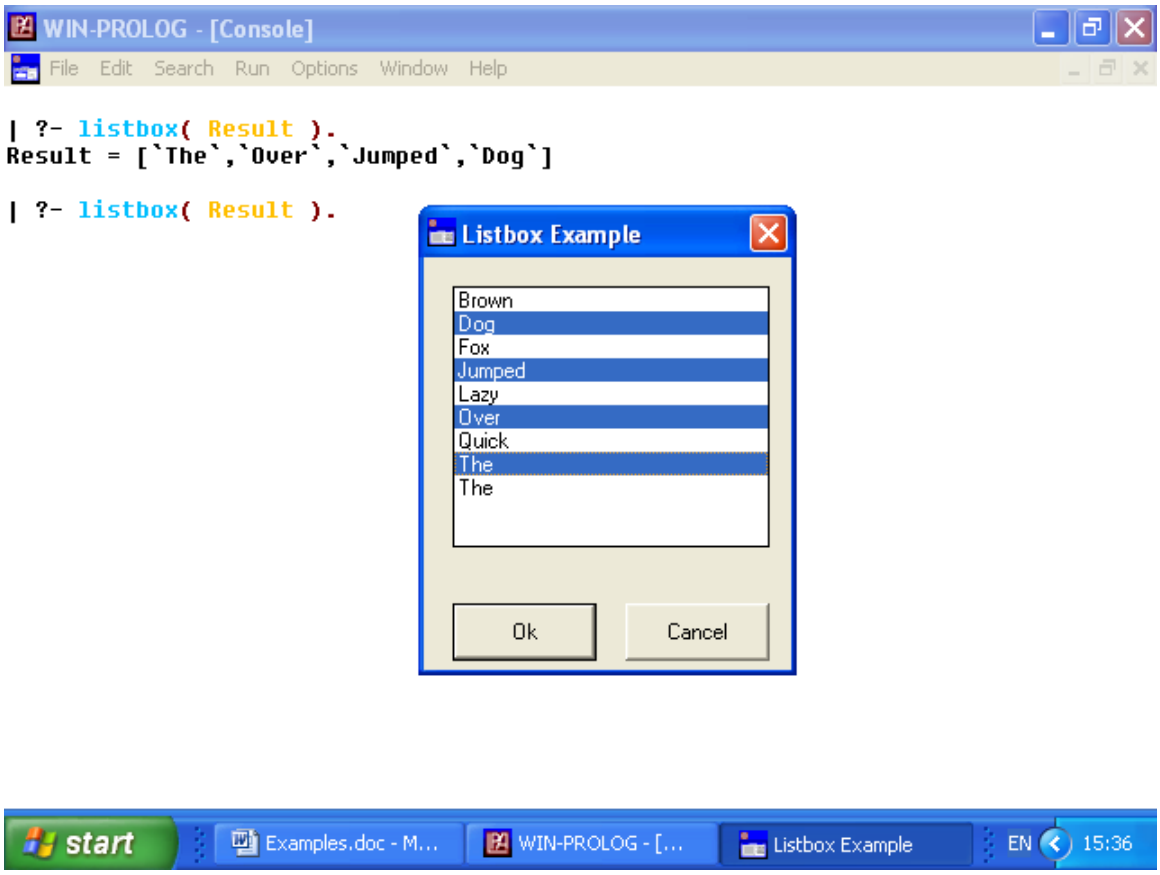
KEYBOARD.PL - Key Interrogation Example

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.



LISTBOX.PL - Program to Demonstrate List Boxes

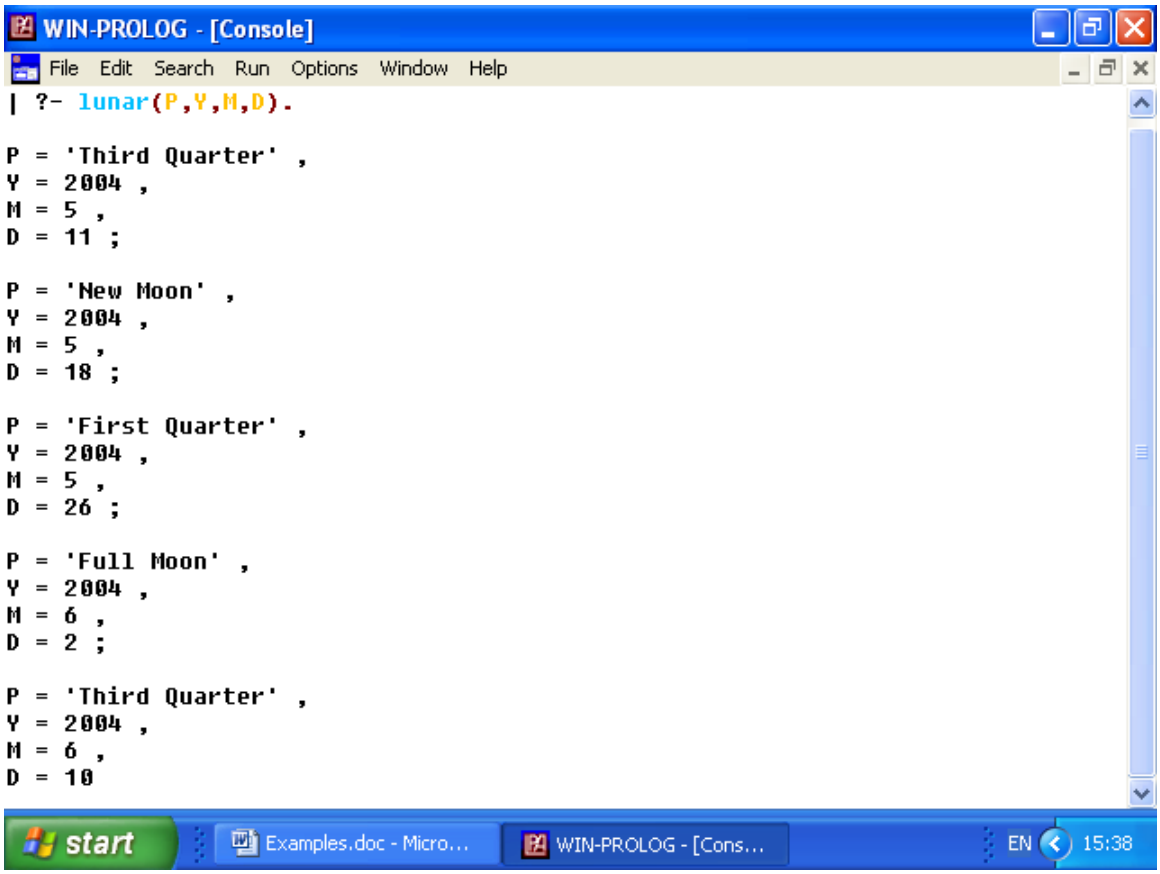
The listbox/1 predicate runs a demonstration listbox example, that returns the selections in the listbox when the dialog is finished.



LUNAR.PL – Lunar Phases

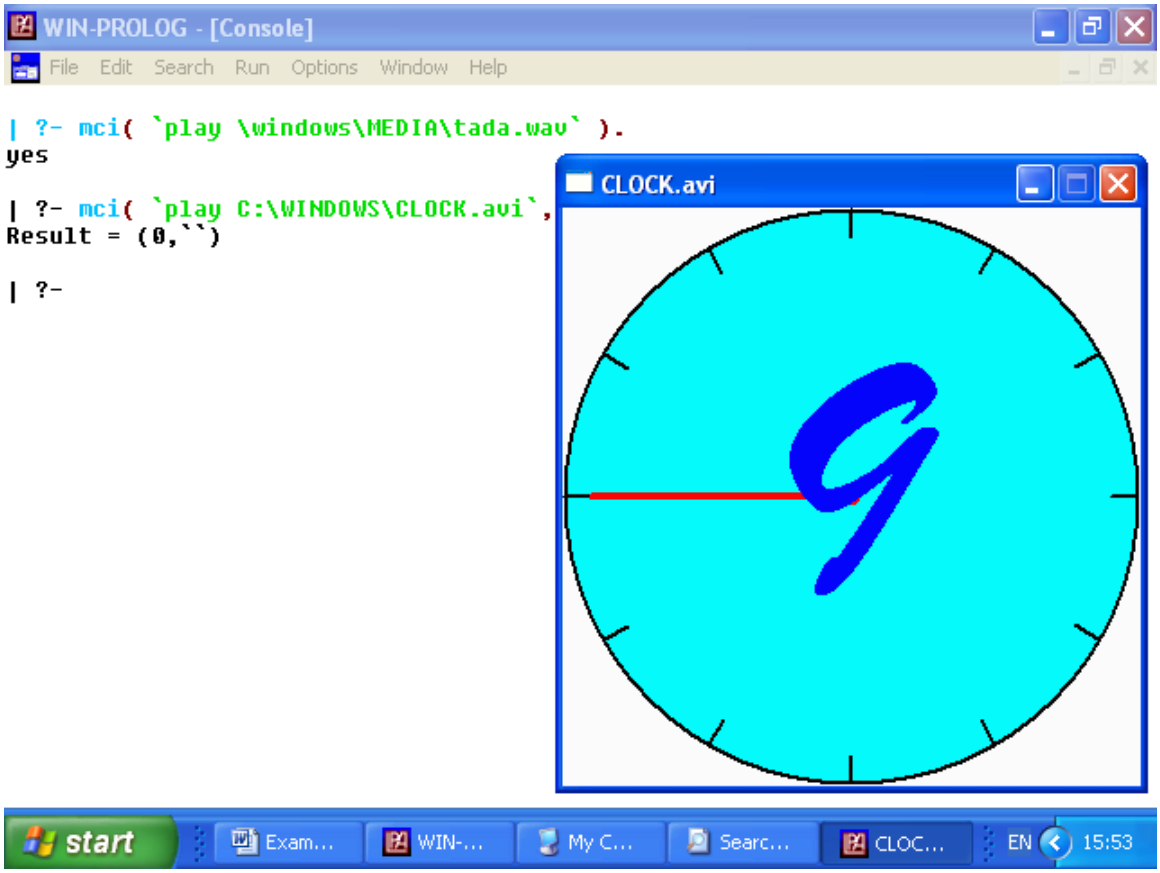
This program computes the next quarter phase of the moon occurring on or after the given date, based on two assumptions: (i) a new moon occurred on 01 Jan 1900, and (ii) the lunar month is 29.53059 days.

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.



MCI.PL - MCI Functions for WIN-PROLOG

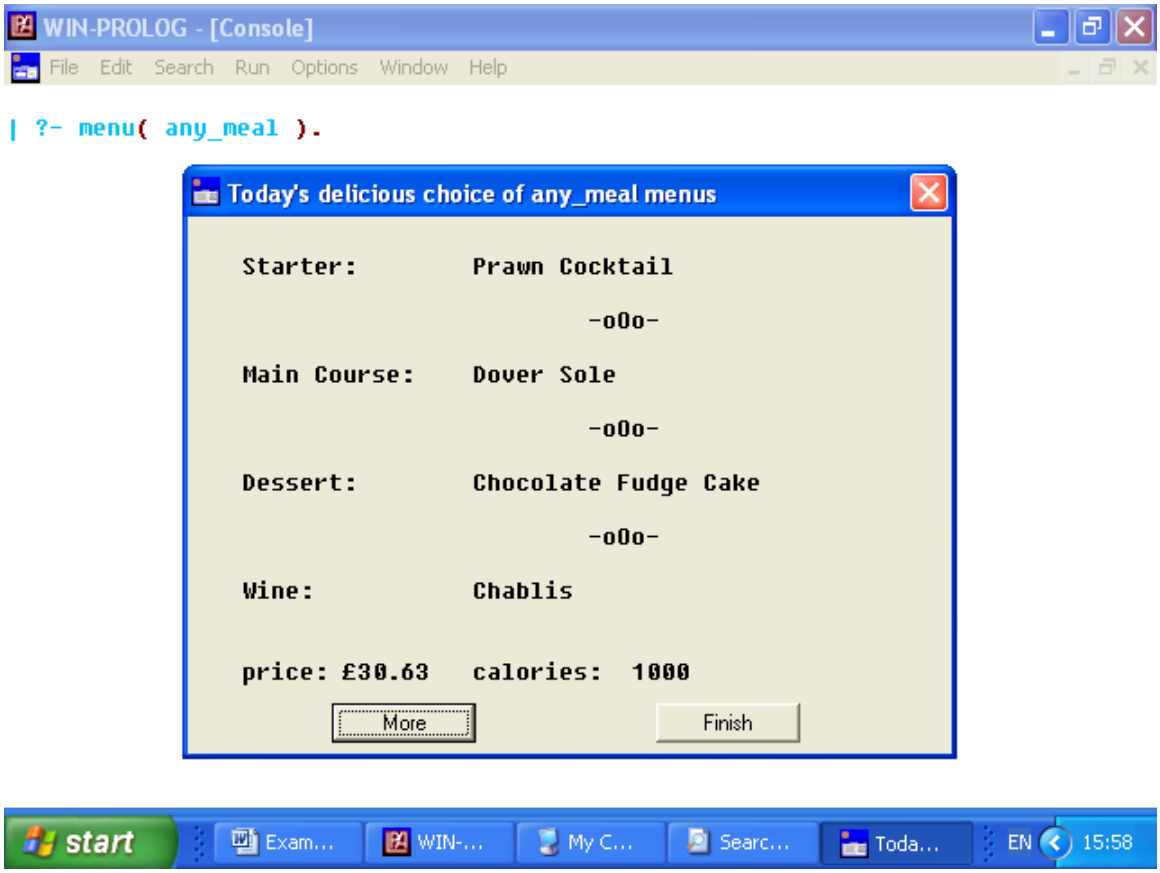
The mci/1 and mci/2 predicate are used to send command strings to the WINMM.DLL (Windows MultiMedia) API, and permits supported files to be played under Prolog control.



Examples

MEALS.PL – Meal Selection Example

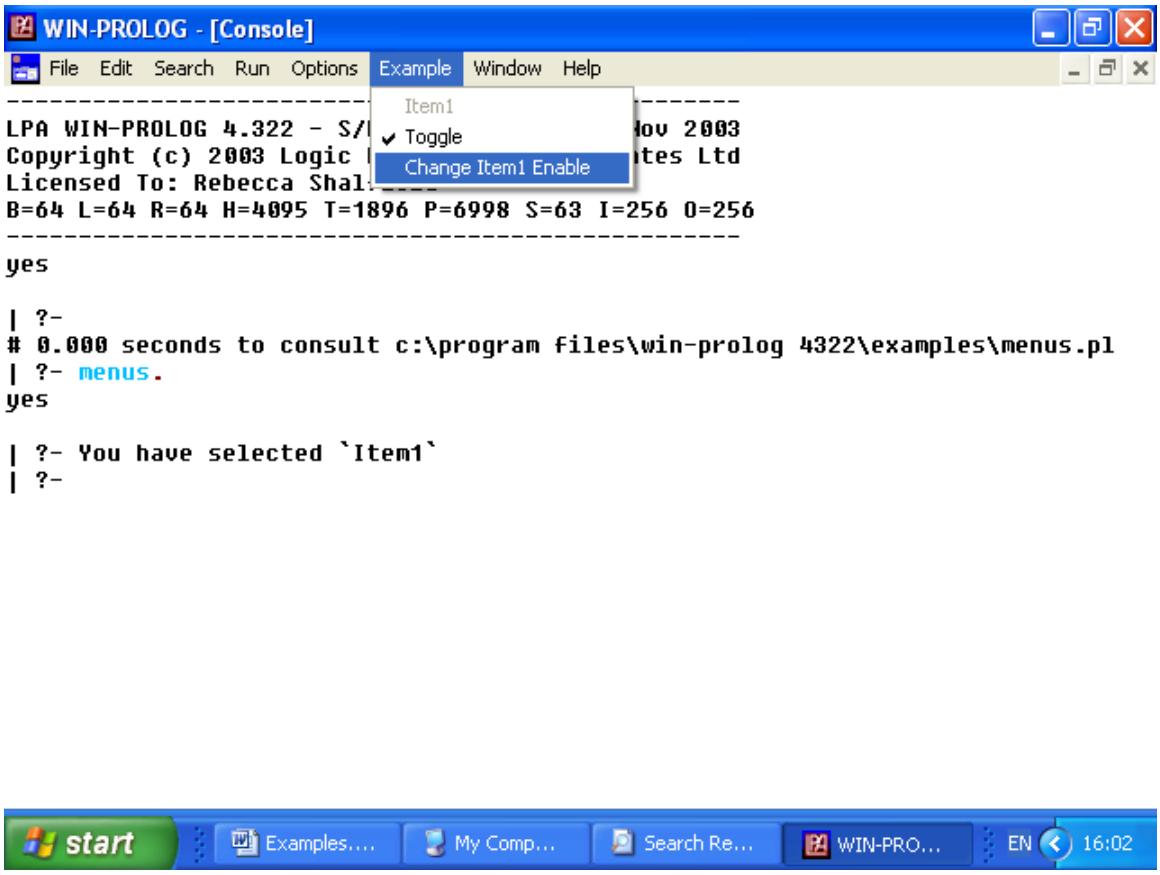
This program is designed to help you select meals of various sorts from a database of dishes.



MENUS.PL – A Simple Menu Demonstration

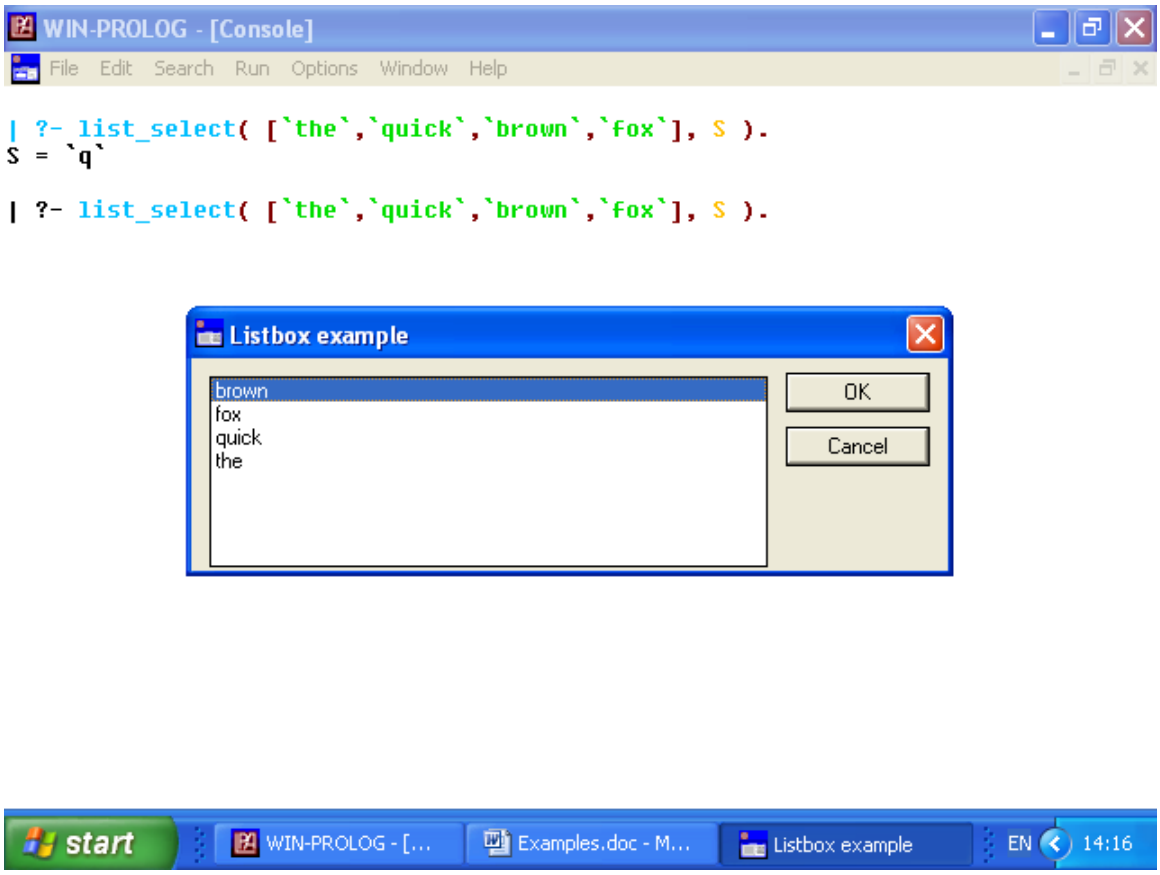
The menus/0 predicate creates and installs a simple menu example.

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



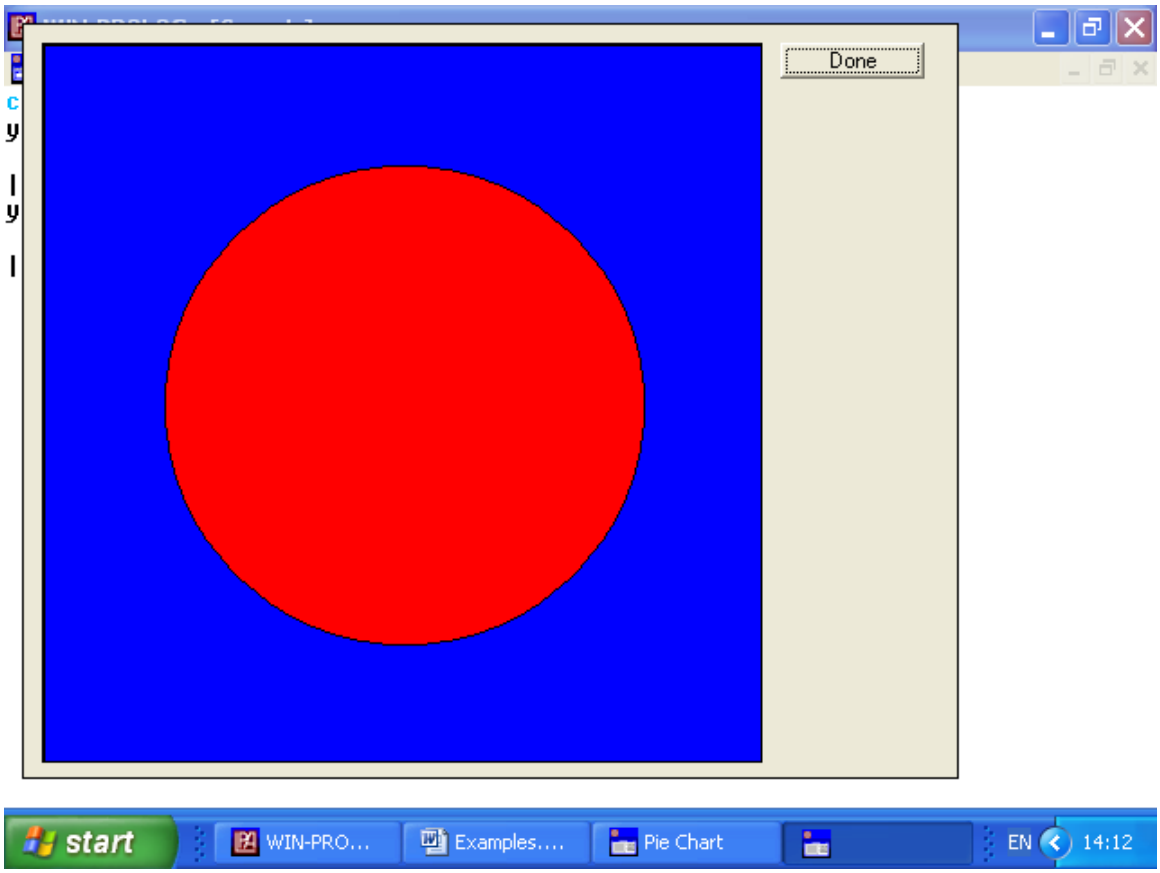
MESSAGES.PL – Listbox Example

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.



META.PL - Creating a Metafile

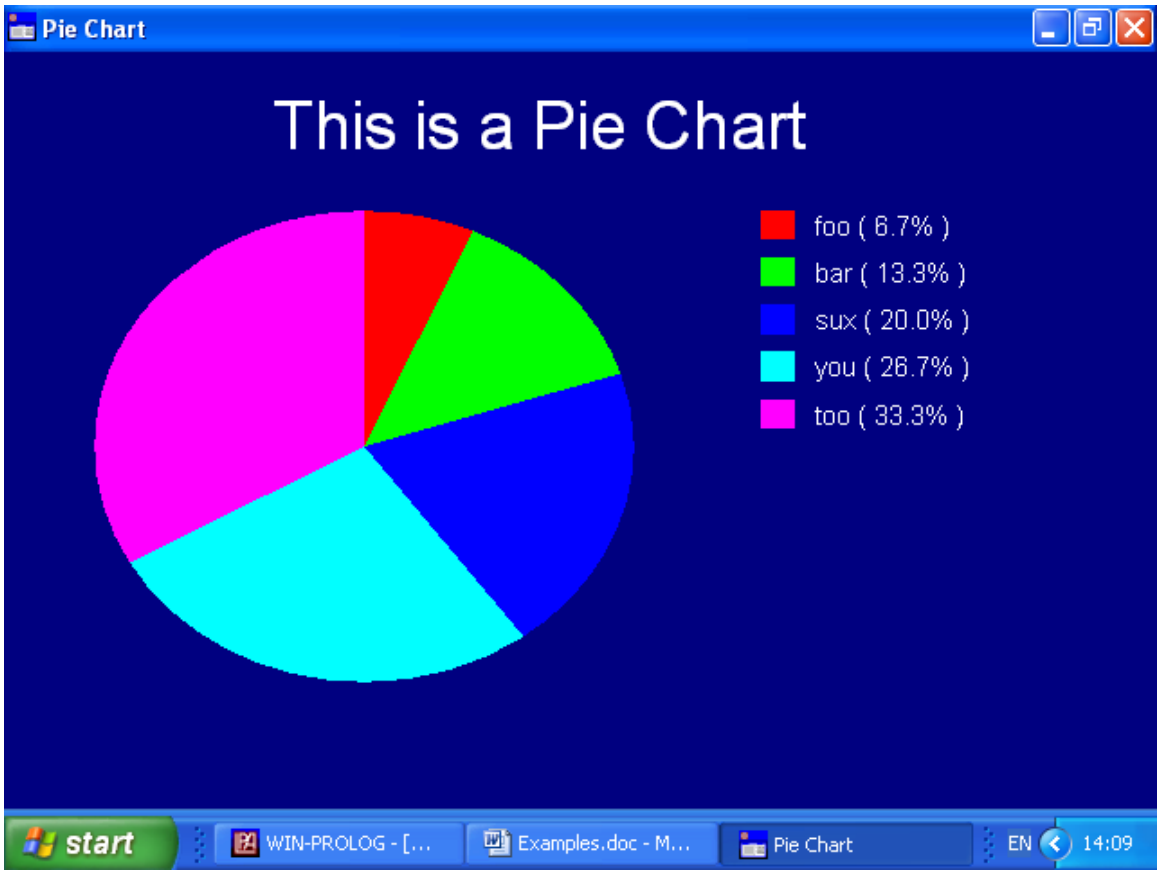
This program demonstrates how to create a Windows "enhanced metafile" (.wmf file) using "gfx" style graphics. Use is made of winapi/4 to create, close, save and delete the metafile, but all other graphics are handled by the gfx/n predicates. Please note that as create_metafile/0 has called the "CloseEnhMetaFile"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



Examples

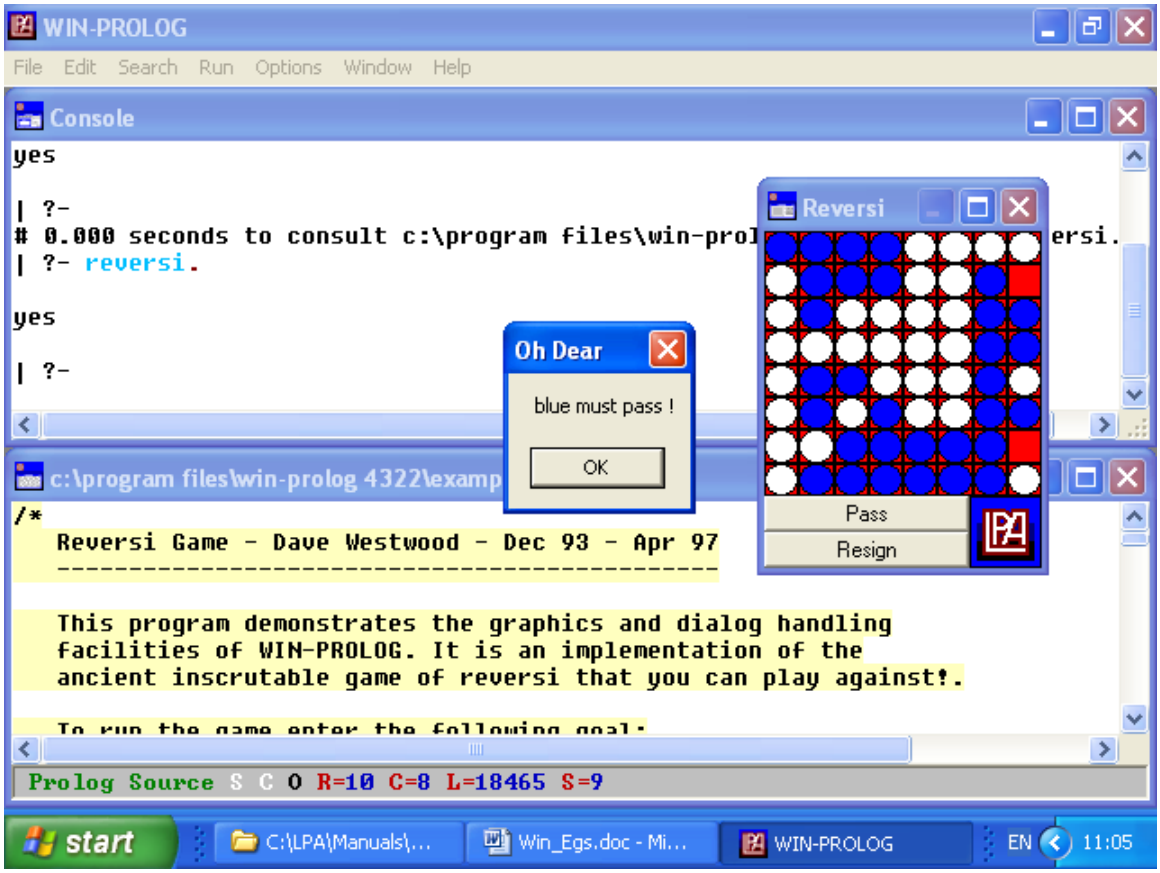
PIECHART.PL - Pie-Chart Display

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.



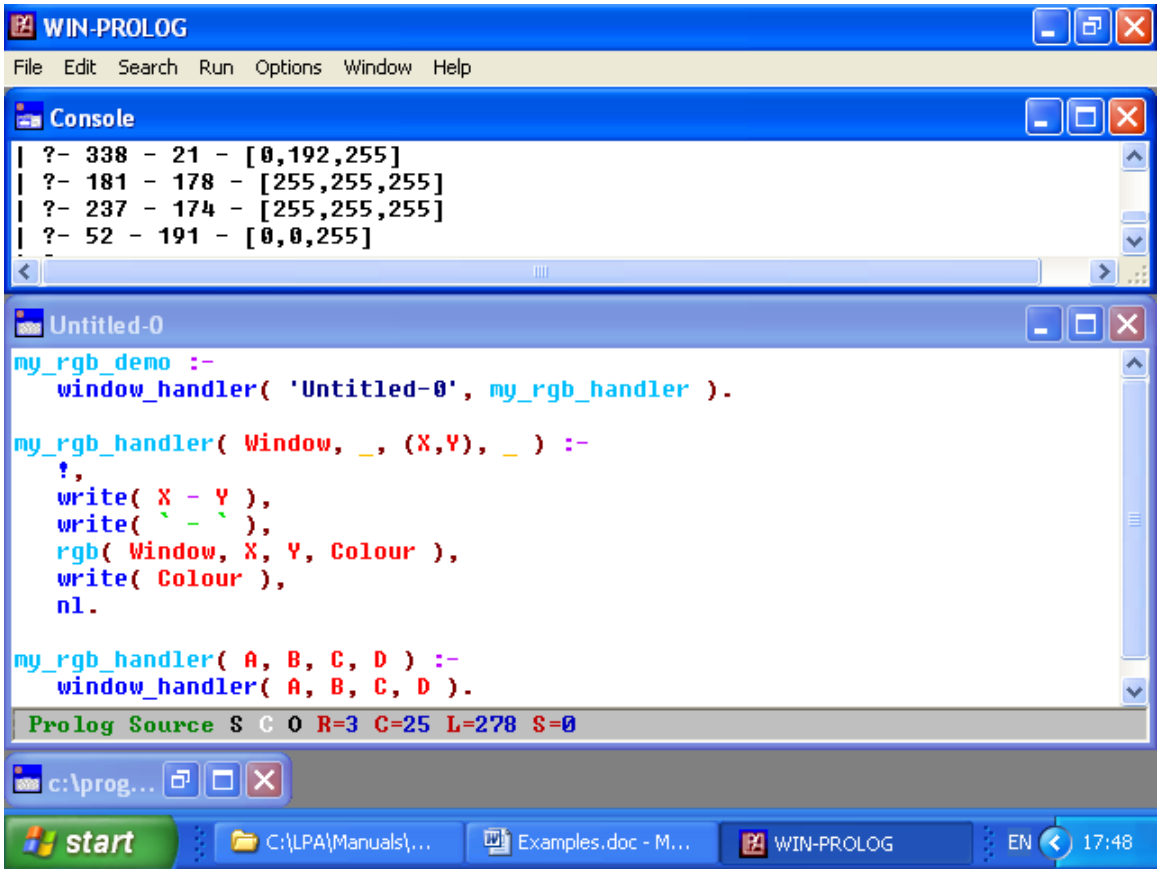
REVERSI.PL - Reversi Game

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!.



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

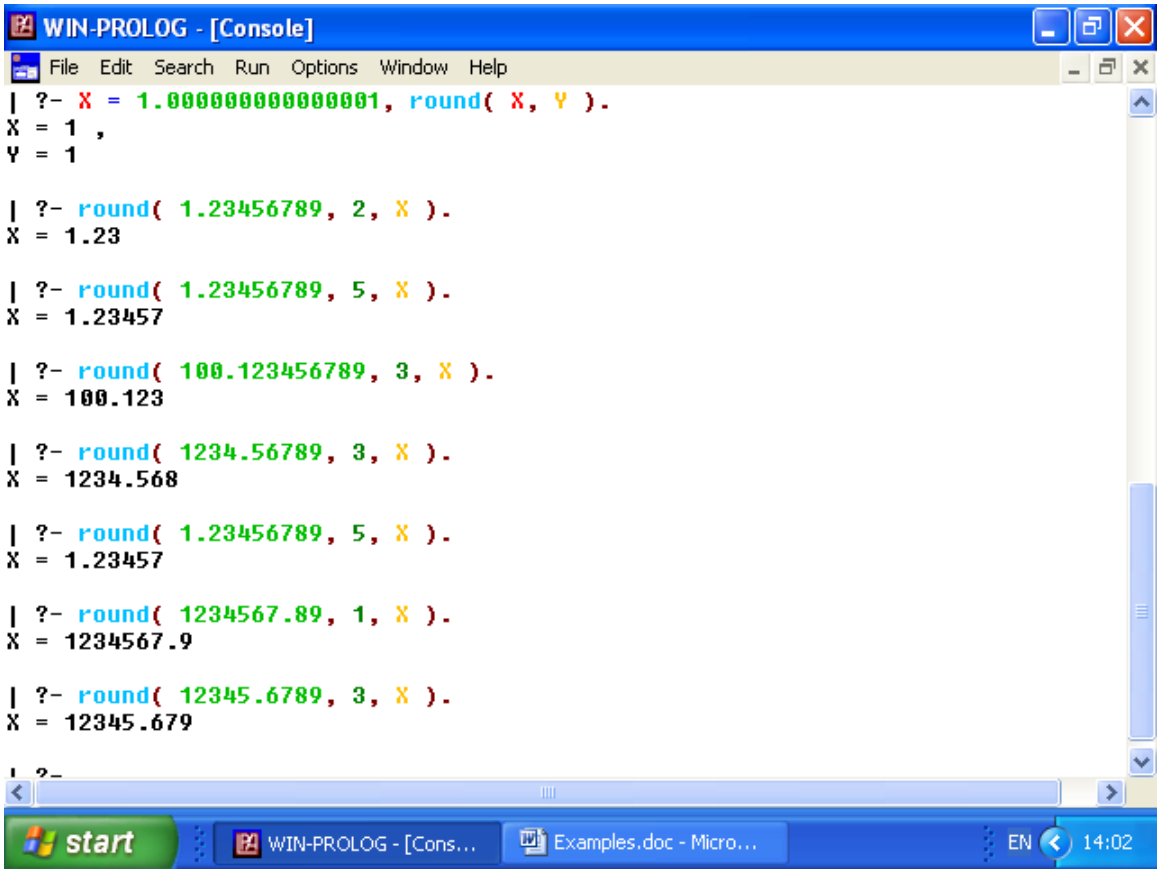
This program uses the Windows API, "GetPixel", to obtain the Red, Green and Blue (RGB) values associated with a single pixel at the given (X,Y) offset within a given Window. If the Window is not visible, or the offset outside of its clipping region, an empty list, "[]", is returned; otherwise, a list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



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 exponentional notation (eg 1.23456e78).

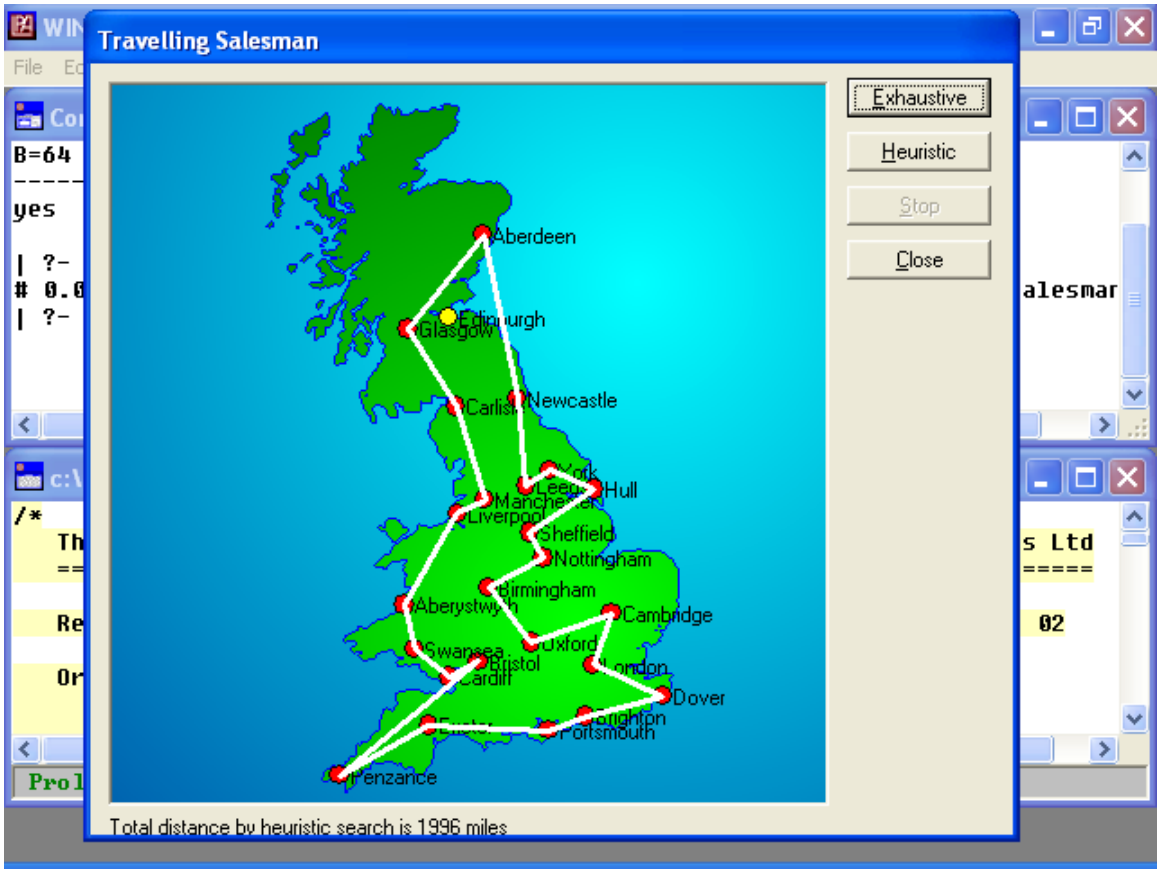


Examples

SALESMAN.PL - The Travelling Salesman

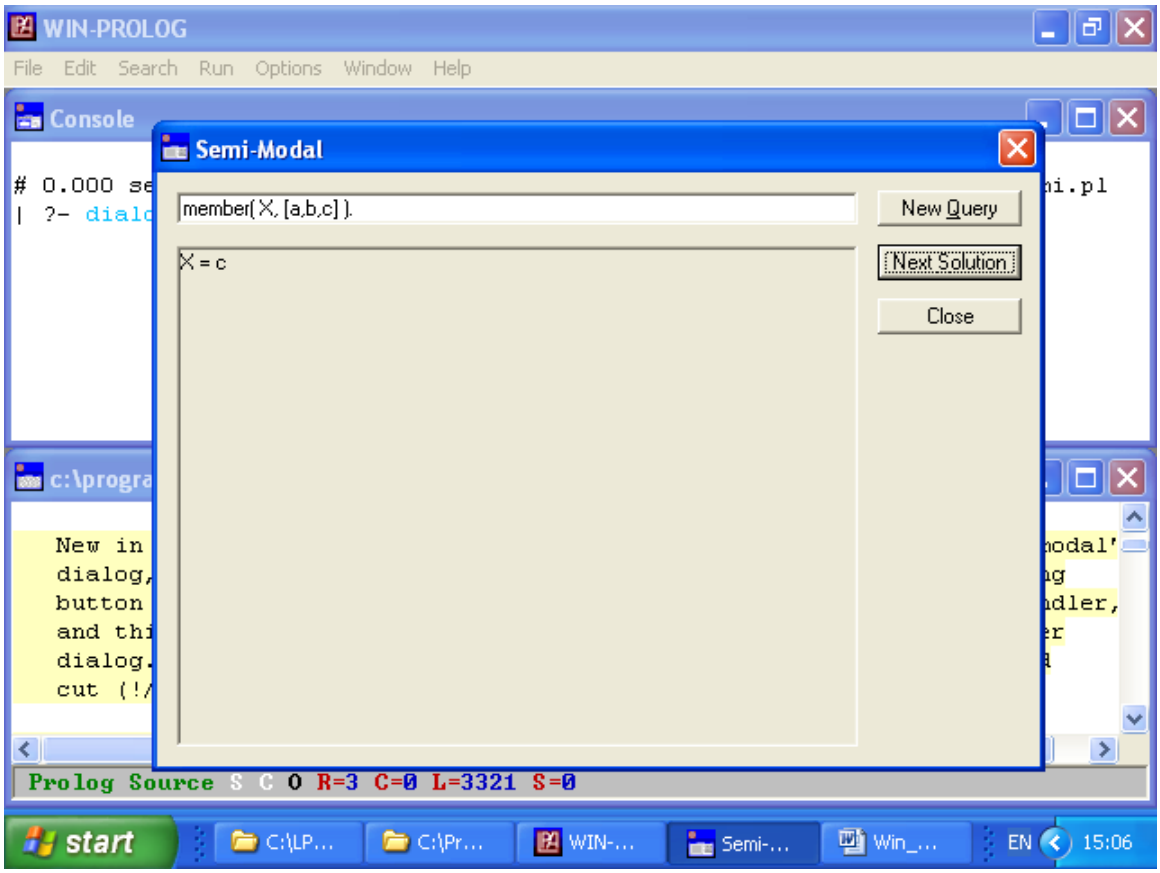
This program displays a map of the mainland UK, showing a number of towns. These can be selected using the mouse, and then the shortest route found between them.

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.



SEMI.PL - Controlling Backtracking with Semi-Modal Dialogs

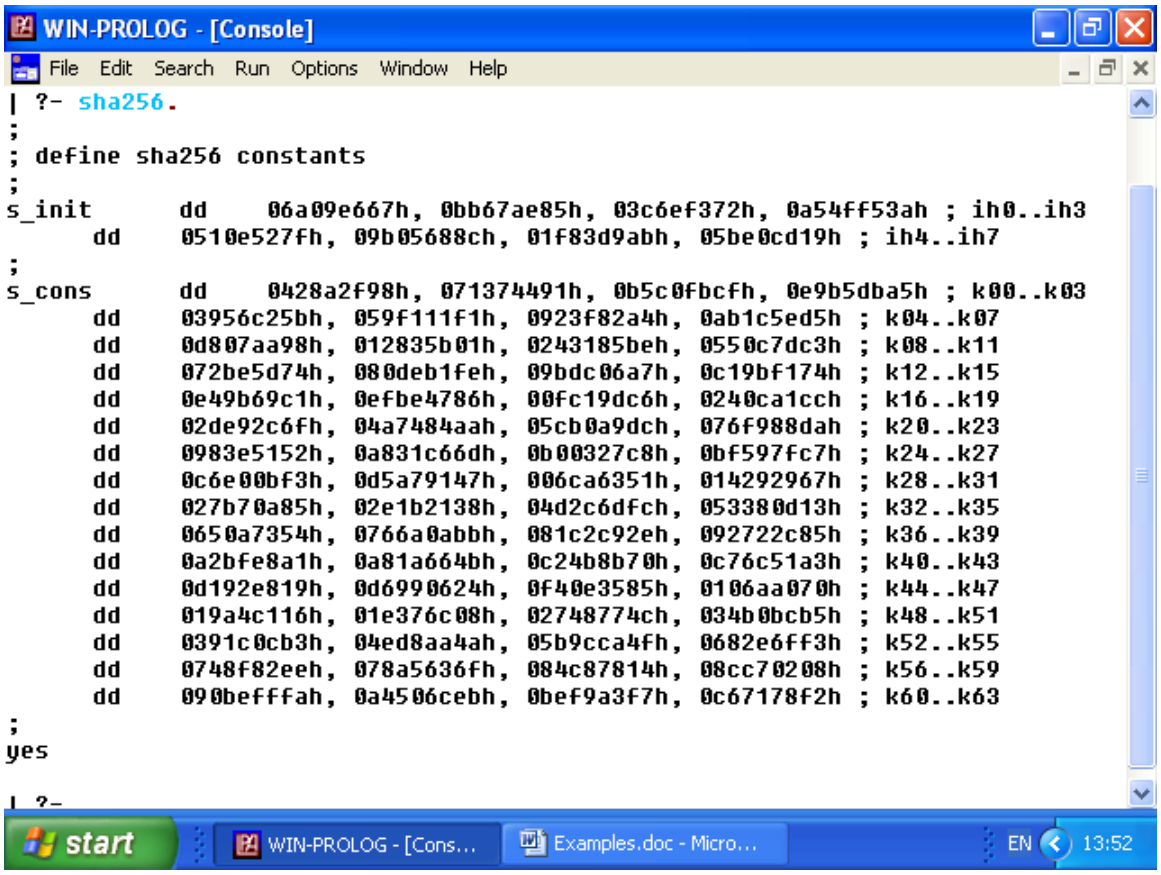
New in version 4.400 of ***WIN-PROLOG*** is the ability to define a "semi-modal" dialog, in which it is easy to control backtracking of a program using button events. This can be acheived without even writing a dialog handler, and this example shows how to do this, using a simple query-and-answer dialog. 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 Erasthatones 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.

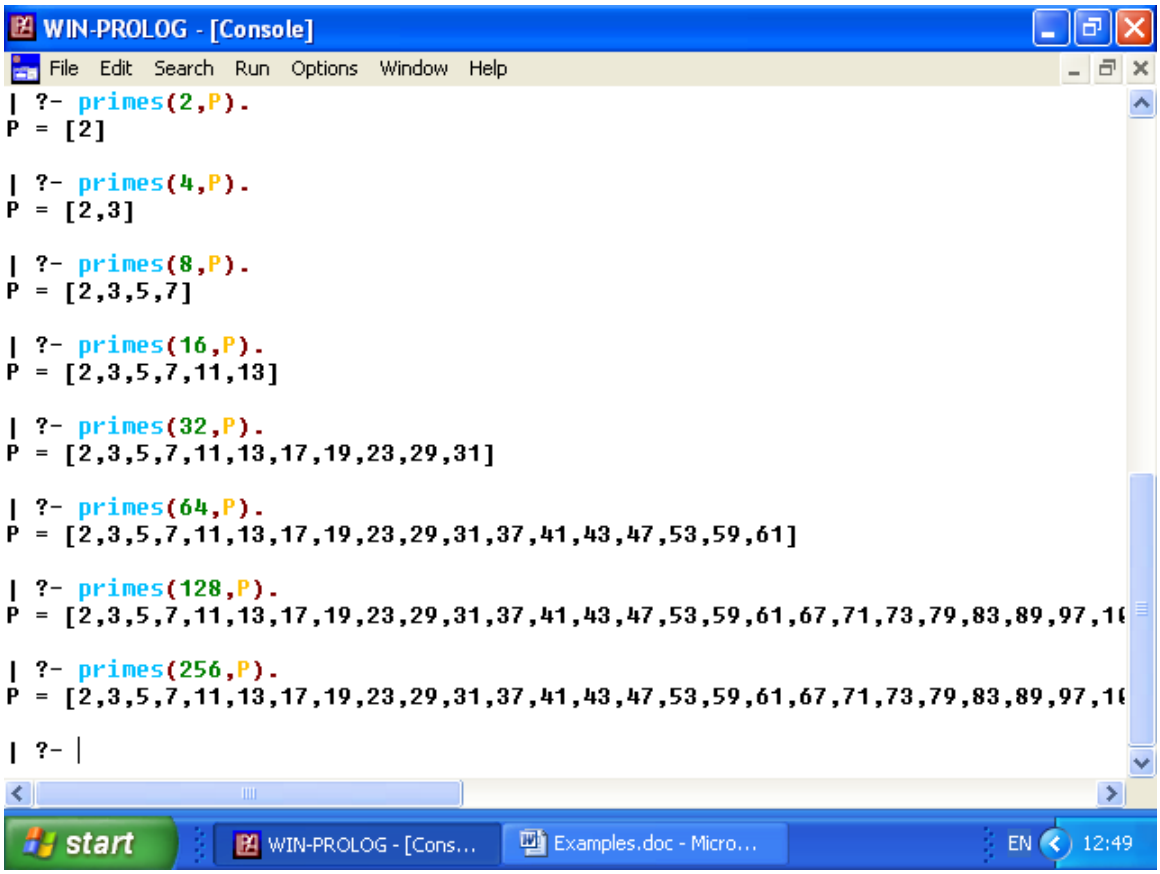


SIEVE.PL - Sieve of Erasthatones

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.

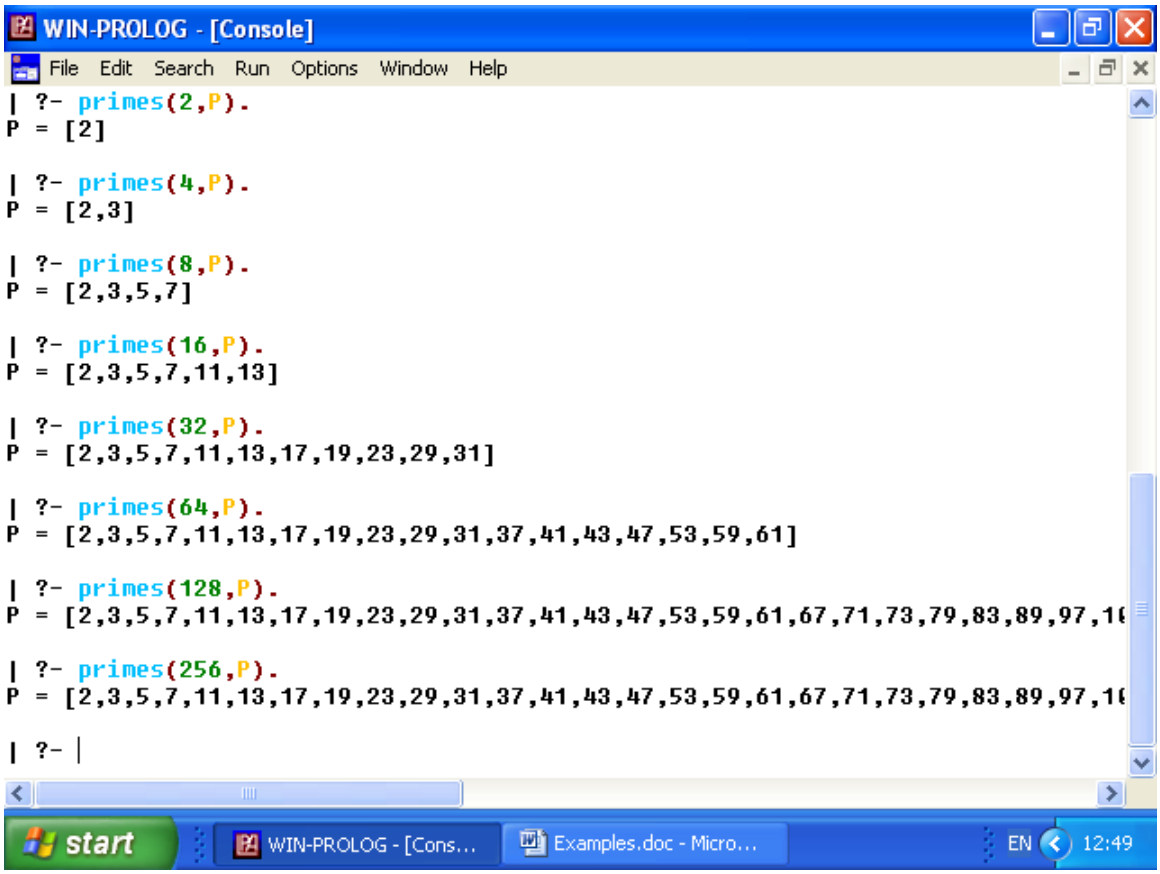
A shorter variant of this algorithm can be found in the file SIEVE2.PL.



SIEVE2.PL - Sieve of Erasthatones

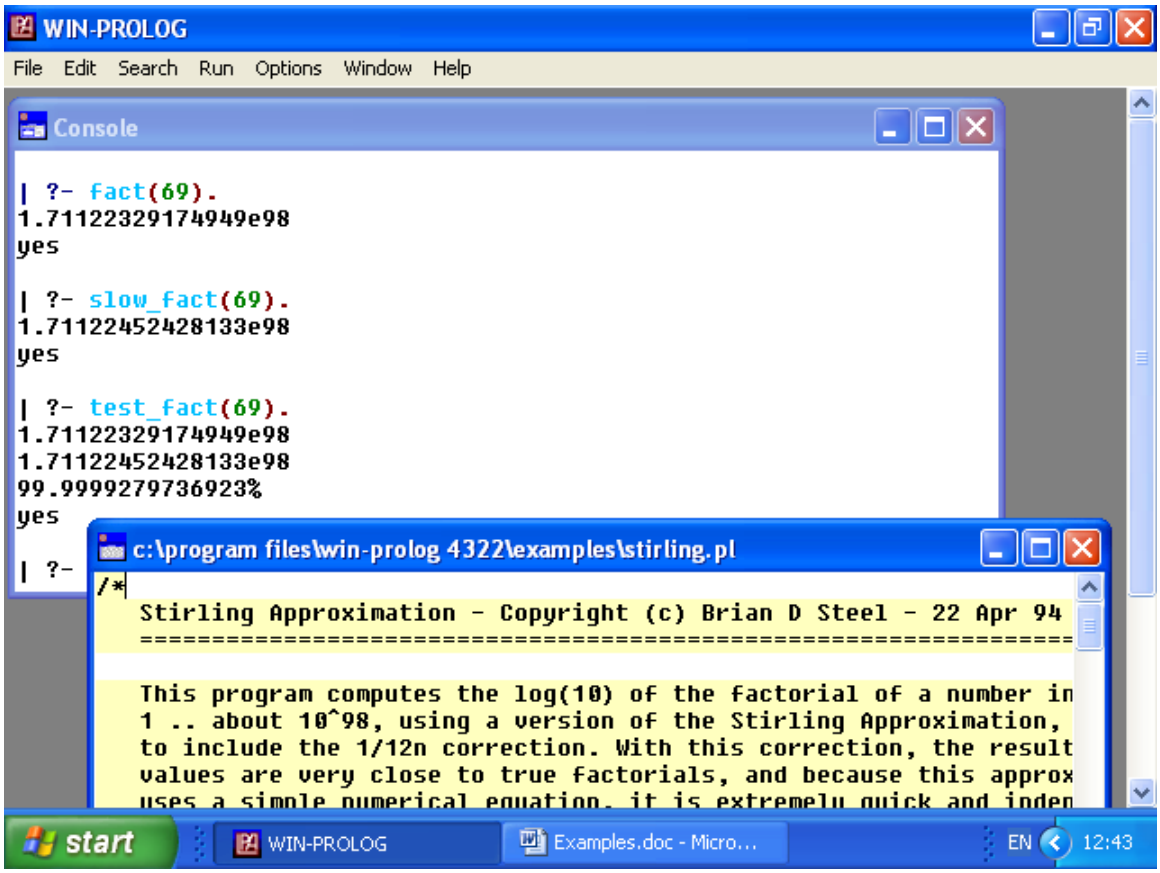
In this classic algorithm, a list of prime numbers is generated by sifting a list of all integers between 2 and the given limit, and discarding those which are directly divisible by a known prime number.

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.



STIRLING.PL - Stirling Approximation

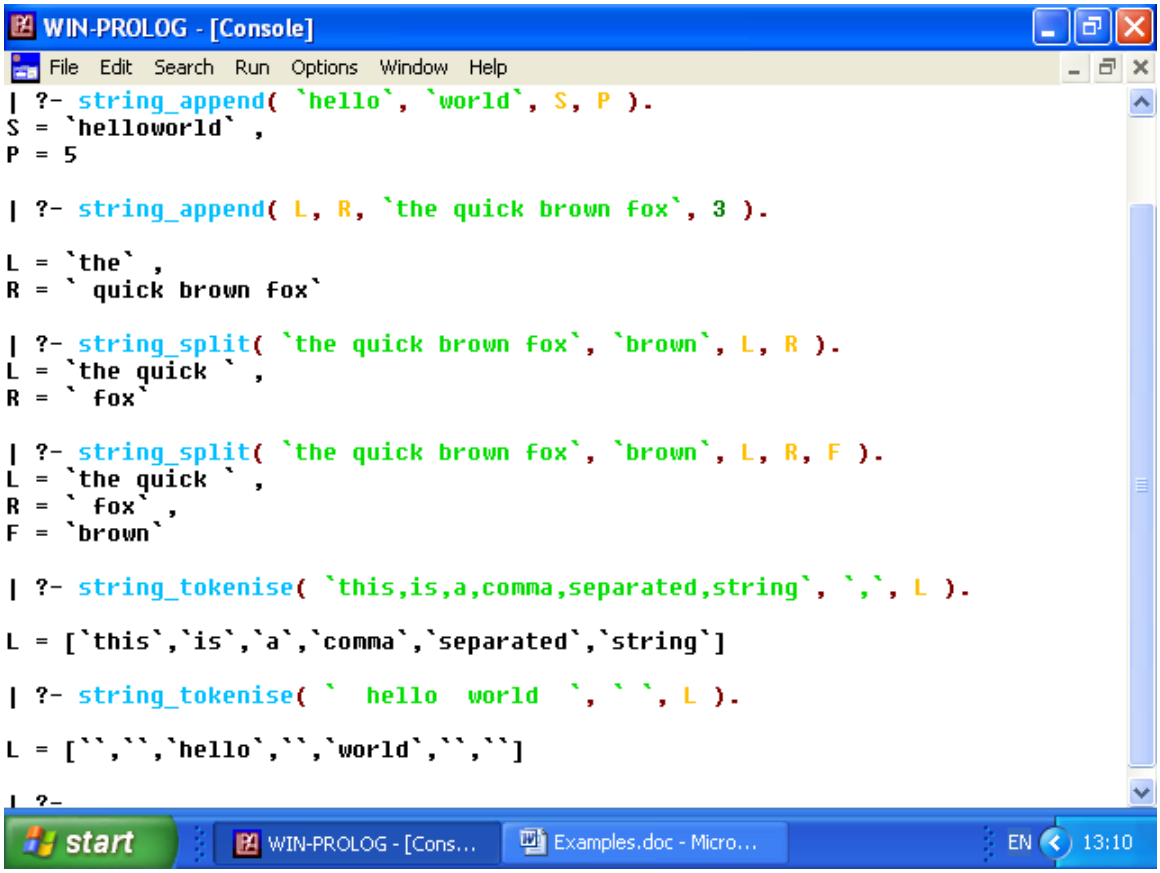
This program computes the log(10) of the factorial of a number in the range 1 .. about 10^98, using a version of the Stirling Approximation, modified to include the 1/12n correction. With this correction, the resulting values are very close to true factorials, and because this approximation uses a simple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



Examples

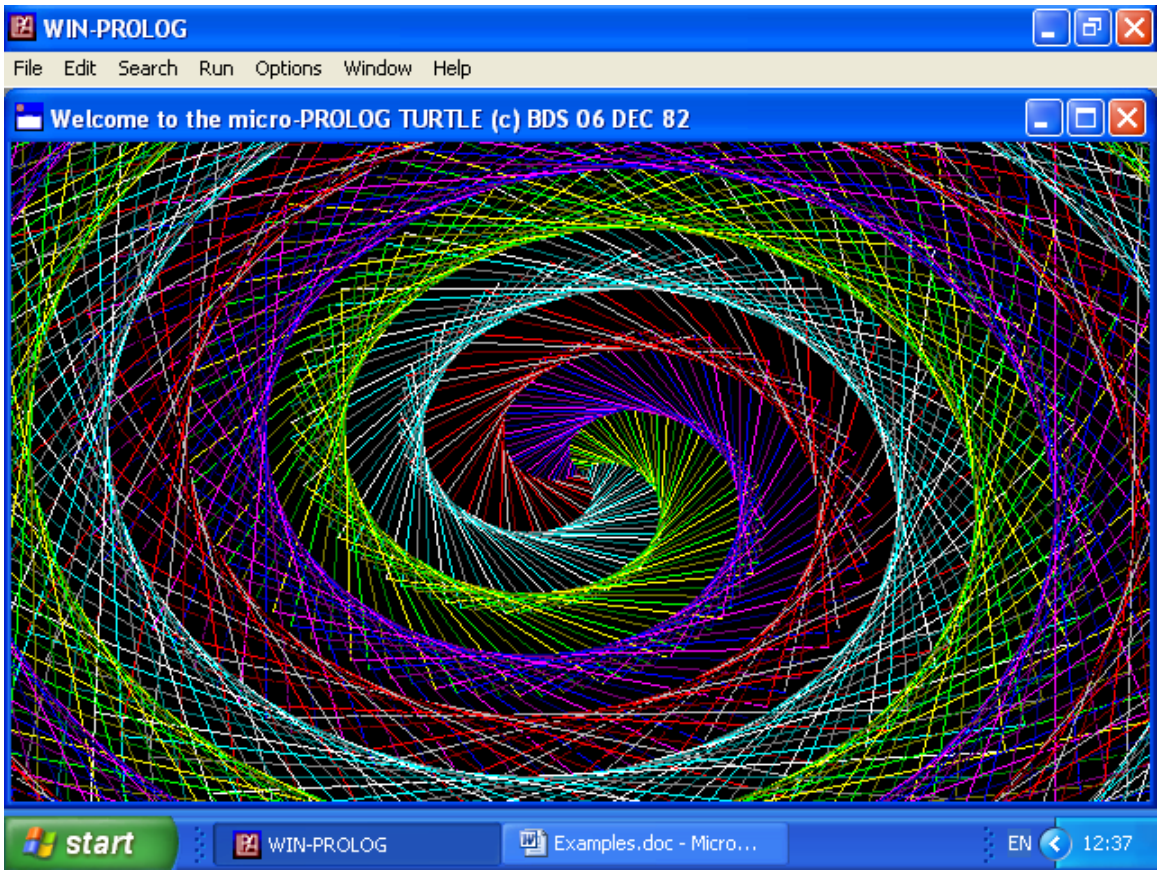
STRINGS.PL - String Examples

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.



TURTLE.PL - Turtle Graphics

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



Examples

INDEX