VECTOR

Dyalog 2003 Conference ...

- Conference Report 44
- *Take Care* in Sweden 63
- Daintree's CD Database 70
- "Now I know" in J (Peelle) 92
- Clark on Formula Inversion 100
- Crossley's *Variable Editor* 111

The Journal of the British APL Association

ISSN 0955-1433
www.vector.org.uk
Vol.20 No.2 October 2003

A specialist Group of the British Computer Society

Contributions

All contributions to VECTOR may be sent to the Journal Editor at the address on the inside back cover. Letters and articles are welcome on any topic of interest to the APL community. These do not need to be limited to APL themes, nor must they be supportive of the language. Articles should be accompanied by as much visual material as possible (b/w or colour prints welcome). Unless otherwise specified, each item will be considered for publication as a personal statement by the author. The Editor accepts no responsibility for the contents of sustaining members' news, or advertising.

Please supply as much material as possible in machine-readable form, ideally as a simple ASCII text file on an IBM PC compatible diskette or via email. APL code can be accepted in workspaces from I-APL, APL+Win, IBM APL2/PC or Dyalog APL/W, or in documents from Windows Write (use the APL2741 TrueType font, available free from Vector Production), and MS Word (any version).

Except where indicated, items in VECTOR may be freely reprinted with appropriate acknowledgement. Please inform the Editor of your intention to re-use material from VECTOR.

Membership Rates 2003–2004

Category	Fee	Vectors	Passes
UK Private	£20	1	1
Overseas Private	£22	1	1
(Supplement for Airmail, not needed for Europe)	£4		
UK Corporate Membership	£100	5	5
Overseas Corporate	£110	5	
Sustaining	£500	10	5
Non-voting Member (Student, OAP, unemployed)	£10	1	1

The membership year normally runs from 1st May to 30th April. Applications for membership should be made to the Administrator using the form on the inside back page of VECTOR. Passes are required for entry to some association events, and for voting at the Annual General Meeting. Applications for student membership will be accepted on a recommendation from the course supervisor. Overseas membership rates cover VECTOR surface mail, and may be paid in sterling, or by Visa, Mastercard or JCB, at the prevailing exchange rate.

Corporate membership is offered to organisations where APL is in professional use. Corporate members receive 10 copies of VECTOR, and are offered group attendance at association meetings. A contact person must be identified for all communications.

Sustaining membership is offered to companies trading in APL products; this is seen as a method of promoting the growth of APL interest and activity. As well as receiving public acknowledgement for their sponsorship, sustaining members receive bulk copies of VECTOR, and are offered news listings in each issue.

Advertising

Advertisements in VECTOR should be submitted in typeset camera-ready format (A4 or A5) with a 20mm blank border after reduction. Illustrations should be photographs (b/w or colour prints) or line drawings. Rates (excl VAT) are £250 per full page, £125 for half-page or less (there is a £75 surcharge per page if spot colour is required).

Deadlines for bookings and copy are given under the Quick Reference Diary. Advertisements should be booked with, and sent to Gill Smith, Vector Production, Brook House, Gilling East, YORK YO62 4JJ. Tel: 01439-788385.

Email: apl385@compuserve.com.

Contents

		Page
Editorial: Quotable Quotes	Stefano Lanzavecchia	3
APL NEWS		
Quick Reference Diary		5
British APL Association News		
Internet Update	Bob Hoekstra	7
APL Product Guide	Gill Smith	11
The Education Vector		
Zark Newsletter Extracts	edited by Jon Sandles	26
Crossword and Puzzle		34
J-ottings 38: Decline and Rise: a Matter of Rank	Norman Thomson	37
RECENT MEETINGS		
Dyalog 2003 Conference	Adrian Smith	44
The *Take Care* System	Peter Pichler	63
ZIMMERER - Roof Design with APL	Michael Zippel	68
Using Namespaces to Manage a CD Database	John Daintree	70
TECHNICAL SECTION		
Solving the "Now I Know" Problem in J	Howard Peelle	92
Towards a General Theory of Formula Inversion	Ian A. Clark	100
Data Compression with Dfns	Veli-Matti Jantunen	107
A Variable Editor	David Crossley	111
At Play with J: The Magical Matrix	Gene McDonnell	122
Taking the RTF Challenge	Adrian Smith	127
R.net: A Reverse Polish Interpreter for .Net	Richard Smith	136
Index to Advertisers		143

Editorial: Quotable Quotes

by Stefano Lanzavecchia

What is a quotable quote? Everybody knows that a quote is a word, a sentence, a paragraph extracted from somebody's speech or writing. A quotable quote is a quote that should be used, perused and misused. Back in the days when television hadn't yet plagued our social lives, people used to meet and spend the evening mixing brilliant drinks and brilliant conversation. In those days of the quick rise and decline of somebody's popularity, the tool of survival was a good bag of quotable quotes to inflict on the gathered. The rules didn't prescribe originality nor the need to associate correctly the quote with the source. In fact, it's always been much better to leave the boundaries of original creation and opportunistic plagiarism in a confusing mist of the unsaid.

A quotable quote is something that should be remembered and, sometimes, something that should be forgotten; something that should be attributed and, sometimes, something that cannot or must not be attributed; something that inspires, amuses, something that can be used in context and, some other time, out of context. The *quoter* should never feel too ashamed about the use he is about to make of his little treasure.

With all the bright people who met at the recent Dyalog User Group Meeting, somebody well versed in the art of catching and collecting quotable quotes would certainly have made a big catch. I contented myself with writing down a few that tickled my fancy and that I am now going to report.

"Crude oil is more or less the same as wine"

Steve Levine

The beauty and the power of quoting out of context is self-evident in this pearl that could be used by both an engineer who cannot distinguish between a glass of kerosene and one of *Amarone della Valpolicella* (I am not an engineer but have the same fine taste for wines... er... liquids) and an outstanding sommelier, who studied organic chemistry at the university.

"LISP came out from people who didn't see the progression raw binary→assembler→FORTRAN"

John Scholes

LISP would be a so much better language if instead of having just semantics it had also syntax...

"I should be working in a power plant (I am no statistician; I am no programmer either: I am an engineer specialised in nuclear power plants) but Sasha's guys blew up Chernobyl"

Veli-Matti Jantunen

The Finns may be people who have a hard time expressing their emotions, especially by facial expressions, but they sure know how to entertain their audience. And, personally, I am really happy Veli-Matti became part of the APL community even if that meant changing his well-laid plan. I am also happy that his friendship with good Sasha allows him to make sapid comments on their respective and respectful jobs.

"Statistics are an important aspect of modern society: had the Iraqis produced good statistics about their weapons of mass-destruction, a certain incident could have been avoided"

V.M.J.

It's amazing how one's insight in a particular topic can change perspective on historical matters, whether it's classic history or modern one.

"In Finland there are 456 municipalities. Excel can only handle 256 columns in one spreadsheet: somebody suggested to merge municipalities to reduce their number"

V.M.J.

It's a known fact that I have a soft tooth for Microsoft and their products. It is a joy for me to see that there are people who instead of complaining (as one would) about the weak points in Excel, come out with a bold but basically good suggestion for a workaround.

"Chunking: sounds like a Chinese restaurant"
"Beware of chunking"
"I am the man who used to be Maurice Jordan"
"The SIGAPL conference is dead"

Various anonymous

These quick spots bring me to my last point: I am sure that you, dear reader, who for many and various reasons could not attend the first Dyalog User Group Meeting in sunny Horsley (believe me or not, it was sunny for real! England is no more what is used to be... how are we going to find the rainy afternoons required to do some serious hacking?) would like to know what *chunking* is and how the meeting was, silly jokes apart. I can tell you in one sentence: you should have been there! It was just that good.

Quick Reference Diary

Date	Venue	Event
2-5 Nov 2003	USA	APL2000 User's Meeting. Naples Beach Hotel, Florida.

Dates for Future Issues of VECTOR

	Vol.20 No.3	Vol.20 No.4	Vol.21 No.1
Copy date	4th Dec	5th March	
Ad booking	11th Dec	12th March	
Ad Copy	18th Dec	19th March	
Distribution	January 2004	April 2004	July 2004

Note that the dates for July 2004 will depend on the (as yet unknown) schedule for the summer conference(s).

British APL Association News: Internet Update

from Bob Hoekstra

Most regular Vector readers have probably seen, or are at least aware of, the Vector website at *http://www.vector.org.uk/*. However, there are other interesting developments that are not so well known but might be of interest to the readers.

New Webmaster and Vector Website Move

The previous issue of Vector announced that I had become the new webmaster, Ray Cannon taking a well-earned rest. We thank him for years of hard work in this area. This change coincides with the website move to a new hosting arrangement. While this should, in itself, cause little change to the average browser (that's you sitting at your PC) it has provided the BAA with (i) cost savings, (ii) total control of the *vector.org* domain and (iii) the opportunity to have multiple websites under this domain.

The Vector Wiki

The last two points mentioned above paved the way for a (semi) permanent wiki at *http://wiki.vector.org.uk/*. A wiki has been running on a trial basis, hosted on the servers at Causeway. Now the wiki is quite likely to become a permanent part of our arsenal in promoting APL and related languages to the world at large.

What *is* a Wiki?

In recent times the WikiWikiWeb has become almost a phenomenon. There is little clarity about what it *really* is. To quote from *http://c2.com/cgi/wiki* (a web page that deserves reading):

> "The concepts behind "Wiki" may seem strange at first. Wiki is a composition system, it's a discussion medium, it's a repository, it's a mail system, it's a chat room, it's a tool for collaboration. Really, we don't know quite what it is, but try it and explore its links - it's a fun way to communicate. "

On a more practical level, a wiki is a basic website that everyone can edit and add to very easily. Anybody can insert their views and insights, or alter and/or remove those of others. It sounds chaotic, yet it seems to work rather well. The

most amazing thing is that it is so *easy*. You are encouraged to take part by pointing your browser at *http://wiki.vector.org.uk/*, then select the *WelcomeVisitors* link. Some way down the page you will find links to *StyleGuide*, *WikiWord* and *WikiEdit* - glance through these before selecting the *WikiSandbox* to hone your new skills. Within ten minutes, you should know everything you need to.

So, what can this wiki thing do for you? The answer depends on how you participate. You can share information, comment on the ideas of others, learn from others and let them learn from you, or get nothing whatsoever, depending on your response to it. I urge you to become involved. If enough people join in, the wiki can become a great meeting place on the web for gathering and sharing information or thoughts. Too often APLers have been criticised as using yesterday's language. Let's show all the young whippersnappers with their Java and C# that we dinosaurs can also be at the cutting edge!

News from Sustaining Members

Dyalog Ltd

Thanks largely to the quality of the speakers and the enthusiasm of the delegates, the inaugural Dyalog User Conference was a great success and was thoroughly enjoyed by all who came. One of our objectives was to showcase the range, scale, and sheer brilliance of the many successful APL applications written by Dyalog customers, and we thank all speakers for their excellent presentations. We will definitely be running the conference again next year and we hope that it will attract even more interest. For the Dyalog development team, the feedback and informed discussions about potential future enhancements to Dyalog APL were particularly useful. As evidence that we take notice of your suggestions, the Editor enhancement, that was voted number 1 by conference delegates, was available to DSS subscribers only 2 weeks after the conference ended.

Following the conference, there is now an independent and enthusiastic Dyalog user discussion group at *http://groups.yahoo.com/group/dyalogusers* in addition to the three mail groups which we administer ourselves. See *http://www.dyalog.com* for details.

On October 3rd, we announced a new release (10.0.3) of Pocket APL which is fully compatible with Pocket PC 2003. Even better news is that on the ubiquitous XScale processor that powers nearly all current models, Pocket PC 2003 runs Pocket APL up to 3.5 times faster than Pocket PC 2002. This should open up even more APL applications to the Pocket PC market.

At the same time, we have released a 30-day free trial version of Pocket APL. The trial version is functionally identical to the $39.99 download, so Pocket PC users can evaluate the complete Pocket APL package, with absolutely nothing to pay. In addition, the product now includes a toolkit for creating packaged applications, ready to be uploaded to Handango or other on-line distributors. The toolkit may be used to produce free-trial versions as well as full commercial products. Creating and distributing APL applications for the Pocket PC could not be easier.

Causeway Graphical Systems Ltd

Along with lots of other APLers, Causeway greatly enjoyed Dyalog's first annual conference. Oddly enough, we even sold a few copies of our Desktop version of GraPL to long-time RainPro customers. I suppose I should have expected this, because even I now tend to prototype examples in GraPL and save them out as

Dyalog or +Win functions. It just saves so much typing - when we have autocomplete on property names in classes, I might go back to the APL session again! Anyway, if anyone out there wants a chart design IDE to help build RainPro charts, you can buy GraPL from our website for all of £40 and then mail Causeway for the free APLFN plugin to clipboard your designs as APL code.

Getting a solid build of Grapl.NET in C# remains the priority. We have trial versions on about 6 customer sites at present, and apart from a few minor details (like that damned style character getting mis-translated) it all just appears to work. It is strange how the market has suddenly shifted - following all the .Net hype 2 years ago, almost nothing happened in the 'real' world until this summer. Then Microsoft shipped Windows 2003 Server (with full .Net support) and almost overnight the market for COM components died, and existing customers started mailing us to ask how the .Net build was going. We need to be there with something we can trust, and so far APLScript looks the best bet. Look out for a progress report in the January Vector.

MicroAPL Ltd

Version 2.0 of APLX was previewed at APL2003 in San Diego, and will be released soon on Windows, MacOS and Linux. Amongst the enhancements will be the ability to create new APL tasks from the menu bar, with multiple session windows, each with its own workspace.

You will also be able to create new APL tasks under program control, either wth a session window, or as invisible background tasks. Signals can be used to communicate between tasks, and you can share variables between them.

Full details will be given in the next Vector.

In the mean time, we have released an update to APLX for Linux which fixes an incompatibility with some recent Linux distributions including RedHat 9.0 and Mandrake 9.1. The update can be downloaded from *http://www.microapl.co.uk/apl.*

The Vector Product Guide

compiled by Gill Smith

VECTOR's exclusive Product Guide aims to provide readers with useful information about sources of APL hardware, software and services. We welcome any comments readers may have on its usefulness and any suggestions for improvements.

We reserve the right to edit material supplied for reasons of space or to ensure a fair market coverage. The listings are not restricted to UK companies and international suppliers are welcome to take advantage of these pages.

For convenience to readers, the product list has been divided into the following groups ('poa' indicates 'price on application'):

- Complete Systems (Hardware & Software)
- APL and J Interpreters
- APL-based Packages
- Consultancy
- Other Products
- Overseas Associations
- Vendor Addresses
- World Wide Web and FTP Sites

Every effort has been made to avoid errors in these listings but no responsibility can be taken by the working group for mistakes or omissions.

We also welcome information on APL clubs and groups throughout the world.

Your listing here is absolutely free, will be updated on request, and is also carried on the Vector web site, with a hotlink to your own site. It is the most complete and most used APL address book in the world.
Please help us keep it up to date!

All contributions and updates to the Vector Product Guide should be sent to: Gill Smith, Brook House, Gilling East, York, YO62 4JJ. Tel: 01439-788385, Email: apl385@compuserve.com

APL INTERPRETERS

COMPANY	PRODUCT	PRICES(£)	DETAILS
APL Borealis Inc.	Dyalog APL	poa	Distributor of Dyalog APL products from Dyadic
	APL2000	poa	Distributor of APL2000 products
APL Systems IDC SL	APL2000 APL interpreters and toolspoa		Distributor of APL2000 products for the UK Consulting and maintenance for APL applications
Beautiful Systems	Dyalog APL/W for Windows	poa	US Distributor of Dyalog APL products from Dyadic.
	Dyalog APL for Unix	poa	See Dyadic listing for product details.
Dinosoft Oy	Dyalog APL/W for Windows	poa	Finnish distributor of Dyalog APL products.
	Dyalog APL for Unix	poa	See Dyadic's listing for product details.
Dittrich & Partner	APL+Win	poa	Cognos/APL2000 Inc products
	Dyalog APL	poa	Dyadic Systems Ltd. products
	IBM APL products	poa	
Dyalog	Dyalog APL for DOS/386	995	Second generation APL for DOS.Runs in 32-bit mode, supports very large workspaces. Unique "window-based" APL Development Environment and Screen Manager. Requires 386/486 based PC or PS/2, at least 2Mb RAM, EGA or VGA, DOS 3.3 or later.
	Dyalog APL/W for Windows	995	As above, plus object-based GUI development tools. Requires Windows 3.0 or later.
	Dyalog APL for Unix	995-12,000	Second generation APL for Unix systems. Available for Altos, Apollo, Bull, Dec, HP, IBM 6150, IBM RS/6000, Masscomp, Pyramid, NCR, Sun and Unisys machines, and for PCs and PC/2s running Xenix or AIX. Oracle interface available for IBM, Sun and Xenix versions.
DynArray	DICE for Windows	poa	Software development kit which includes an APL interpreter as a DLL and the ability to run and link existing and new APL code to non APL code such as VB, C/C++, Java and integration with various Windows software applications and database packages such as MS Office.
I-APL Ltd	I-APL/PC or clones	8	ISO conforming interpreter. Supplied only with manual (see 'Other Products' for accompanying books).
	I-APL/BBC Master	8	As above
	I-APL/Archimedes	8	As above
IBM APL Products	TryAPL2	free	APL2 for educational or demonstration use. Write, fax or Email to APL Products; specify disk size desired.
	Workstation APL2 V2 Version 2	$1500	AIX, Linux, Solaris, Windows Product 5724-B74, Part Number 45P7514
	APL2 Version 2	poa	Product No. 5688-228. Full APL2 system for S/370 and S/390
	APL2 Application Envt Vn2	poa	Product No. 5688-229. Runtime environment for APL2 packages
Insight Systems	Cognos/APL2000 Inc	poa	Leading distributor of APL2000 products in Denmark
	Dyalog Ltd.	poa	Leading distributor of Dyalog APL products in Denmark
	IBM	poa	Leading distributor of IBM APL & GraphX products in Denmark
J Austria	J	poa	Distributor for Austria and Switzerland
	Dyalog APL	poa	Distributor
	Causeway Products	poa	Distributor
	Structural Analysis Software	poa	Complete package by IG Zenkner&Handel to perform structural analysis/engineering calculations. Also suitable for dynamic problems, e.g. earthquake simulation.
JSoftware Inc.	J on the Web online registration ...		
	J Professional (online reg.)	$695	includes manual set and one year of updates
	J Standard (online reg.)	Free	Free for download only

	Books and accessories		
	J Dictionary	$50	
	J Phrases	$50	
	J Primer	$50	
	Fractals, Visualization and J	$80	
	Concrete Math	$40	
	Exploring Math	$50	
Lescasse Consulting	APL+PC	poa	Lescasse Consulting is the exclusive APL2000 distributor in France and also distributes in Switzerland and Belgium. Call for price quotes.
	APL+Unix	poa	
	APL+DOS	poa	
	APL+Win	poa	
	Dyalog APL/W	poa	French distributor for Dyalog
MasterWork Software	Manugistics Products and ISI	poa	New Zealand distributor
MicroAPL	APLX for Windows/MacOS	499	Cross-platform APL development environment with GUI programming facilities. Interpreter modelled on APL2. Available for Windows 95/98/ME/NT/2000/XP, Mac OS 9 and Mac OS X.
	APLX Server Edition	poa	For running large multi-user APL applications on x86 Linux, RS/6000 AIX, and Windows NT/2000.
	APLX for Linux (commercial)	499	Linux desktop edition of APLX, compatible with Windows and MacOS versions, with full development environment and GUI programming facilities. Runs under RedHat, Debian, Mandrake and SuSe Linux distributions.
	APLX for Linux (personal)	Free	Full version of APLX, can be downloaded free for personal use.
Oasis	Dyalog APL	poa	Dyalog Ltd
	APL*PLUS	poa	Manugistics
	APL.68000	poa	MicroAPL Ltd
	APL2	poa	IBM
Omega	Zero	poa	A "small simple and fast" alternative to APL
Optima	Dyalog APL/W	995	Dyalog's best, leading edge APL Interpreter
RE Time Tracker Oy	APL+PC (APL*PLUS/PC)	poa	Complete APL+ and Statgraphics product range and links to various 3rd party products.
	APL+DOS (APL*PLUS II)		
	APL+Win (APL*PLUS III), APL+Link		
	APL+UNIX		
	APL*PLUS Sharefile		
Soliton Associates	SHARP APL for OS/390	poa	for IBM OS/390 mainframes
	SHARP APL for UNIX	poa	for SunOS and IBM AIX
	SHARP APL for Linux	poa	for Intel Linux
Strand Software	Canada		
	All APL*PLUS products	poa	All APL*PLUS products including upgrades and educational.
	Dyalog and JSoftware products	poa	
	USA		
	Dyalog and JSoftware products	poa	

APL PACKAGES

COMPANY	PRODUCT	PRICES(£)	DETAILS
ADAPTA Software	MPS	poa	Master Production Scheduling
	FBS	poa	Forecasting and Budgeting System
	DRP	poa	Distribution Requirements Planning
Adaptable Systems	FLAIR	poa	Finite loader and interactive rescheduler. Customisable full-function scheduling system. (Available outside Australia by special arrangement only.)
Adaytum	Adaytum e.Planning	poa	Adaytum e.Planning offers a Web-based solution that combines planning, forecasting, budgeting, modelling and reporting in a single, integrated application.
APL Software\Services	APL Utilities	poa	Software: mostly .AWS for DOS, utilities for most APL interpreters. Public domain APL*Plus v10 with on-screen documentation and interactive tutorials. APL Conference Software. Books: APL user manuals for STSC, IBM, and Sharp. Request email catalog from dick.holt@juno.com.
Beautiful Systems	ASF_FILE	$399	Dyalog APL/W auxiliary processor for access to APL*PLUS/PC APL component files (*.ASF).
	NAT_FILE	$299	Dyalog APL/W auxiliary processor which emulates the APL*PLUS/PC quad-N native file subsystem for access to the DOS file system.
	DBF_FILE	$299	Dyalog APL/W auxiliary processor for efficient block mode access to dBASE format files. Designed to get large amounts of data in and out of dBASE. Not suited for random access to small amounts of data (it does not handle keys).
	SF_READ	poa	Dyalog APL/W functions to read APL*PLUS data objects of any type or structure from *.SF style component files created by APL*PLUS II or III.
Causeway	*CausewayPro* for Dyalog/W	400	Causeway application development platform for Dyalog APL/W.
	RainPro Business Graphics	250	The ultimate graphics toolkit for the APL developer. Adds 3D charting capability, Web publishing and clipboard support to the shareware product. Charts can be included in *NewLeaf* reports. Functionally compatible across Dyalog/W and APL+Win.
	NewLeaf for Dyalog and +Win	400	Frame-based reporting tool with comprehensive table-generation and text-flow support. Offers multiple master-page capability, bitmap wrap-around and on-screen preview with pan and zoom. Fully supported on Dyalog/W and APL+Win
Cinerea AB	ORCHART	250	Organization chart package for IBM APL2/PC. Full & heavily commented source code included - free integration into other applications. NB: ASCII output with line-drawing (semi-graphic) characters for boxes.
CODEWORK	3-way TANGRAM	poa	3-way TANGRAM is a DSS-OLAP product, basically a powerful and versatile handler of multi-way tables (also known as hypercubes). It entails 46 analysis modules, including computations, cube merging, sensitivity analysis, time intelligence, free format queries, HTML and LaTeX outputs. Current version is 7.0 At the time of writing, the product is available on Dyalog APL 8.3.1 for Windows 95/98/NT. Future plans include an APL+WIN version and later a LINUX version.

CoSy	K.CoSy	$30 % yr sub	K.CoSy is a general purpose computing and programming environment constructed, all in open code, in Arthur Whitney's very high level, yet structurally transparent Array Programming Language, K , and its tightly coupled User Interface. K.CoSy is an extremely productive environment in one of the most powerful and fastest of APL's progeny, and therefore, likewise, of all languages. K.CoSy provides a workspace-like interactive development environment previously impossible in K. Because of its unique open construction within the language itself, this environment is clearly competitive in a large domain with the APLs from the other vendors. K.CoSy notepad nature, interactivity, and open K code vocabulary make learning Arthur Whitney's K far less daunting and far more productive than its raw console, or any external scripting method. If you are a client of Kx Systems , or are investigating the possibilities, Contact us. See CoSy/K/CoSy for more information.
DynArray	DynaWeb Server	poa	A web server providing web based access to applications running on the DICE interpreter from DynArray, or on an IBM mainframe running APL2.
	DynaHarry	poa	A DSS system which offers the next generation capabilities for current APLDI, IC/E and IC/1 users. It comes with ROLAP capabilities, multisystem access to a wide variety of databases and data warehouses.
	DynaLink	poa	An ODBC client interface for DICE and IBM APL2 programs.
Entropy Software	Personal Information Management	poa	Entropy specialises in Personal Information Management Systems (PIMS) which run on a PC. Most systems are inadequte in some way. This is an attempt to fulfil all the needs of George MacLeod who has been an unhappy user of a variety of systems for many years. The product is being developed in modules (Calendar, Address List, To Do List, etc) which will be available at no charge. When all the modules are complete they will be integrated together. There will be a charge for the integrated product.
HMW	4XTRA	poa	Networked, Windows/Unix based Front End and Middle Office Foreign Exchange and Money Market Dealing System. Scalable from 1 user to 120+.
	Inca	poa	Software Change Management System. Enables the user to co-ordinate development work from several sources, resolve clashes, promote work items for testing and configure releases to a live environment.
	Maya	poa	APL code file manager. A comprehensive suite of tools giving a multi-window IDE style interface to file based APL code. Offers features such as copying from file to file, object comparison, string search, style formatting, hot-spot editor for filed objects (including variables), etc.
	Aztec	poa	System shell for APL development. Manages real-time and batch applications across multiple platforms. Offers standardised error trapping, job scheduling, task communication and recovery/restart features.
	Olmec	poa	APL GUI environment, providing menu bar, tool bar, status area, navigation sidebar (with treeviews & listviews)and client area. All are configured by simple text files and require no programming. Client area has a "tab wizard" option to provide ordered transaction processing
	Nazca	poa	fast, flexible and reliable static database and editor.
I-APL Ltd	Educational workspaces	5	PC format disks with the examples from: Thomson. Espinasse (Kits 1-4), Kromberg, Jizba & FinnAPL. All the examples to save your fingers!
IBM APL Products	A Graphical Statistical System	$250	for DOS, Product Number 5764-009
Insight Systems	Causeway	poa	Leading distributor of Causeway products in Denmark

All our old products are now either OEM'd, in the public domain, out of date, or all of the above. We'll be back!

JAD Software	JAD SMS	poa	JAD SMS is a multi-user software management system for Dyalog APL™ based on shared, hierarchical databases. JAD SMS databases let you keep historical versions of apl items as well as attributes such as timestamp, user name and documentation. The software includes a graphical user interface as well as specialized functions for inclusion in applications. No charge for single-user version; $100/user for multiple users
Lescasse Consulting	APL+Win Monthly Training	$600	Download 50+ page document about APL+ programming each month. You also get one or more workspaces full of re-usable APL code and sometimes additional files or products.
	Advanced Windows Programming	$95	200-page book plus companion disk on interfacing APL and Delphi. Contains full coverage of Delphi-2, +Win and Dyalog.
	DLL parser for APL	$250	Parse any Visual Basic DLL declaration file into a set of quadNA definitions. Turn constants and structures into APL variables. Available for APL+Win and Dyalog/W.
	Delphi Forms Translator	$195	Design forms with Delphi and turn them automatically into APL programs which recreate the same form (+Win and Dyalog/W).
	APL+Link Pro	poa	ODBC interface for APL+Win
	SQAPL Pro	poa	ODBC interface for Dyalog APL/W
	RainPro	poa	Highly customisable 2D and 3D publication graphics for APL+Win and Dyalog APL/W
	NewLeaf	poa	Page layout and printing tools for APL+Win and Dyalog
	GraphX and ChartFX	poa	High-quality business graphics for APL+Win
	Formula One and Dyalog APL	$95	100-page book + companion disk on how to use the Formula One VBX with Dyalog APL/W
Lingo Allegro	FACS	poa	EMMA-like interface to DB2 or ODBC databases
	QWIN	poa	Legacy DOS Windowing support for APL+Win
	ODBC/127	poa	IBM AP127-like ODBC Interface for APL+Win and Dyalog APL/W
Optima	ServiceLine	poa	A property management system which keeps a record of all outstanding tasks, produces an up to date list of work to-do and Scheduled reminders plus automated standard letters and basic financial control.
	TravelLine	poa	A system designed to control the workload allocation for a fleet of chauffeur vehicles plus a reminder system for fleet management and Local Authority requirements.
	BPA	poa	"Brand Performance Analysis" allows for the modelling of product/brand performance over time and comparison with competitor products.
	DBI	poa	"Database Interrogator" allows for the non technical user to ask sensible questions of a large database such as a questionaire and obtain results tables and graphs quickly, easily and accurately.
Qualedi	Qualedi	$850-$5,500	Electronic Data Interchange (EDI) translation software for the PC, with strict compliance checking.
RE Time Tracker Oy	UIT/W	poa	Comprehensive high-level Windows User Interface library for APL+Win and +II v 5.1. Comprehensive spreadsheets, replicated fields, special field types, etc. 16 and 32 bit versions available.
	AJGRAPH	poa	Graphpak-compatible 2D graphics package for +Win and +DOS. Includes multi-window support, print and metafile support. No DLLs required.
	ECCO PRO with APL	poa	Leading group and personal information management system with comprehensive customising. Supplied with sample +Win workspace to interface to ECCO databases via DDE.
	NEWT TCP/IP SDK with APL	poa	Lead TCP/IP SDK with interfaces to all protocols. Supplied on 3 CD ROMS together with a sample +Win workspace.

	DB+	poa	Database interface for APL+DOS under Windows. Allows combining character-based APL applications with ODBC-compliant databases such as Oracle and SQL-server.
Warwick University	BATS	250	Menu driven system for time series analysis and forecasting using Bayesian Dynamic modelling. Price is reduced to £35 for academic institutions.
	FAB	free	Training program for the above.
Welghahead Systems	Welghahead Windows Weighing System (3WS)	poa	Recipe Weighing System for Manufacturing Industries. Pharmaceutical, Cosmetics, Foods etc. Works without keyboard or mouse. Uses Electronic Balances, Laser scanners, bar codes and label printers.
Zark	APL Tutor (PC)	$299	APL computer-based training. Available for APL*PLUS PC & APL*PLUS II. Demo disk $10.
	APL Tutor (MF)	$5000	Mainframe version.
	Zark ACE	$99	APL continuing education. APL tutor news and hotline phone support.
	APL Advanced Techniques....	$59.95	488pp. book, (ISBN 0-9619067-07) including 2-disk set of utility functions (APL*PLUS PC format).
	Communications	$200 pc, $500 mf	Move workspaces or files between APL environments.

APL CONSULTANCY AND DEVELOPMENT

COMPANY	PRODUCT	PRICES(£)	DETAILS
Adfee	Consultancy	poa	Development, maintenance, conversion, migration, documentation, of APL products in all APL environments
Andrews	Consultancy	poa	APL programming and analysis, algorithms, tree processing and design programs for craft work.
APL Borealis Inc.	Support and Development	poa	APL Software Support and Development. Specialists since 1979 in Sharp APL, APL*Plus, APL+Win, Dyalog APL
APL Solutions Inc	Consultancy	poa	APL systems design, development, maintenance, documentation, testing and training. Providing APL solutions since 1969.
APL Systems IDC SL	Consultancy	poa	Consulting and maintenance for APL applications.
AUSCAN Software	Consultancy and Training	poa	APL software development, training
Camacho	Consultancy	poa	Manuals; feasibility reports and estimates; analysis and programming; APL and MS Windows applications; Sharp, IS! APL, APL*PLUS, APL2/PC and other APLs spoken. Fixed price systems a speciality
Ian Clark	Consultancy	poa	Interfacing APL, VB, C/C++, ActiveX/COM. Screen design and documentation. National language porting.
Ray Cannon	Consultancy	poa	APL, C, Assembler, Windows, Graphics: PC and mainframe
Causeway	Consultancy and Training	poa	On-site training for Causeway, RainPro and NewLeaf. Customisation and enhancement to meet local needs. Code review and pre-implementation check of Causeway applications.
Paul Chapman	Consultancy	250-500	24-hour programmer: APL, Smalltalk, C; Windows front end design a speciality.
CODEWORK	Consultancy	poa	Development, maintenance, migration, documentation of APL applications. Speciality: info systems for top executives, internet applications.
CoSy	Consultancy	poa	CoSy.com, Coherent Systems, provides rapid development in the K language and associated data base products, with a particular interest in quantitative (financial) modelling.
David Crossley	Consultancy	poa	Experienced in large APL system developments since 1969 for PC or mainframe.
Dinosoft Oy	Consultancy	poa	Specialised in very large databases.

Dittrich & Partner	Consultancy	poa	APL programming and analysis; APL workshops and training on the job
Dyalog	Consultancy	poa	APL and Unix system design, consultancy, programming and training.
DynArray	Consultancy	poa	DynArray offers consulting in the areas of DSS, Y2K and APL programs upgrade/conversion to modern Web enabled platforms.
Evestic AB	Consultancy	poa	Excellent track record from 15+ years of APL applications in banking, insurance, and education services. All dialects, platforms and project phases. SQL expertise.
First Derivative Analytics Ltd.	Consultancy	poa	Analysis, design, prototyping, development & testing of APL (especially financial) applications: Sharp, Dyalog APL/W.
General Software	Consultancy	from 200	Over 20 years experience with every version of APL, large mainframe systems and small PC based programmes.
Godin London Inc	Software Development	poa	We have applications in the food manufacturing field, travel agency and airline bookings field and in product lease management.
HMW	Consultancy	poa	System design consultancy, programming. HMW specialize in banking and prototyping work.
Hoekstra Systems	Consultancy	poa	APL consultancy, programming, etc. Also UNIX system administration
Michael Hughes	Consultancy	poa	APL consultant with 20 years experience with all versions of APL. I can create your dynamic Web sites using the full power of APL working with Microsoft IIS (Internet Information Service) on Windows NT or 2000. I also undertake System design, Programming and Maintenance on all platforms, particularly MS Windows.
INFOSTROY	Consultancy	poa, competitive	Broad experience in various APL platforms. Special skills and knowledge in developing complex applications for investment, financial and construction markets. Implementation of hybrid solutions based on APL, Delphi, C++, VBA, SQL servers.
Insight Systems	Consultancy	poa	We have experience with just about every APL system and platform in common use during the last 20 years, from SHARP APL under MVS or Linux to APL+Win and in particular Dyalog APL under Windows 9x, NT or 2000. If you have decisions to take about adapting your APL application to take advantage of emerging technologies, or would like your strategy reviewed, give us a call. We have extensive experience in all areas of APL development, from legacy systems, up, down and sideways migrations, to the development and support of shrink wrapped solutions based on APL. Even if we don't have time to do the work ourselves, we will know where to find someone who is an expert in your version of APL and your application area, on your continent.
JAD Software	Consultancy	poa	Systems design and development, project management, technical manuals, financial and actuarial expertise in APL.
KJK	Consultancy and software development	poa	APL-based data management: conversions, ad hoc-analyzing tools, well-interfaced methods for defining, processing and browsing of multi-dimentional reports. Rapid custom software development based on proven modular toolset approach.
Lambent Technolgy	Consultancy	poa	APL programming, consulting & training; web design and construction.
Phil Last	Consultancy	poa	APL consultancy, modelling and programming.
Lescasse Consulting	Consultancy	poa	A range of consultants, experts in Windows programming, with APL+Win and Dyalog APL/W. More than 100 major APL applications already developed. We all have additional expertise in Formula One and Delphi.
Lingo Allegro	Consultancy	poa	General APL consulting, internet website development, migration and downsizing, performance tuning, education and training.
Lucas Solutions	Consultancy	poa	Rates depend on task and location.

Mackay Kinloch Ltd	Consultancy	poa	Design, analysis and programming for banking, insurance and pensions, financial planning and modelling, corporate performance and legal reporting
MasterWork Software	Consultancy	poa	Consulting and J programming for econometrics and statistics in public policy, health and food industries.
Milinta Inc	Consultancy	poa	Design, development, maintenance, conversion, documentation in all APLs, most APs and some specific Sharp products (LOGOS, ViewPoint, Retrieve). Experience in multi-user, multi-task systems, databases, Windows programming.
Ellis Morgan	Consultancy	250-500	Business Forecasting & APL Systems.
Oasis	Consultancy	poa	Expertise in APL system design, Project management, conversion, migration, tuning; for all APL versions (10+ years experience)
Omega Computing	Consultancy	poa	APL consultancy, programming, etc.
Optima	Consultancy, Training & Development	poa	We have been in business since 1990 and have a team of APL professionals with many years experience in pharmaceutical, industrial and financial systems on both PC and Mainframe platforms. In addition to pure APL we can also offer assistance with network integration, ACT! customisation and issues concerning the internet and/or web design.
RadSys Technologies	Consultancy	poa	Areas of expertise: financial systems, risk analysis systems, healthcare systems.
Resources & Results	Consultancy	poa	Knowledge management company builds decision support, data warehouse, data mining and strategic planning systems for CFO's, CEO's, and senior management, using our Rapid Application Development (RAD) methods and tools. Extensive experience in large-scale system development and ad hoc executive support for Fortune 500 clients.
RE Time Tracker Oy	Consultancy	poa	APL application conversions, APL Windows interfaces, APL to API-level interfacing to any system under Windows, TCP/IP network and database connectivity.
Rex Swain	Consultancy	poa	Independent consultant, 25 years experience. Custom software development, PC and/or mainframe.
Graeme Robertson	Consultancy & Training	250-500	Mathematical, statistical and financial APL runtime systems designed, written from scratch and documented. Legacy systems upgraded. 20 years experience covering most APLs (esp. Dyalog) on most platforms (esp. Windows). Particularly interested in dynamic-web or Mathematica applications.
Rochester Group	Consultancy	poa	Specialise in MIS using Sharp APL
Shepp & Associates	Consultancy	poa	APL applications development and consulting, especially in the travel industry, especially on small computers. 25 years experience in APL programming.
Snake Island Research Inc	Consultancy	poa	APL interpreter and compiler enhancements, intrinsic functions, performance consulting. APL parallel compiler APEX is giving very good initial performance tests with convolution somewhat faster than FORTRAN.
SovAPL	Consultancy	poa	Offshore APL development service.
Strand Software	Consultancy	poa	Advice on migrating to and from all flavours of APL and hardware platforms. Full-screen interface implementation, APL utilities, benchmarking, efficiency analysis, actuarial software, system development tools, valuation, pricing and modelling systems.
Sykes Systems Inc	Consultancy	poa	Complete APL services specialising in audit, optimisation and conversion of APL systems. Excellent design skills. All dialects and platforms. 17-23 years experience.
Weighahead Systems	Consultancy	poa	Specialising in industrial systems. Links to PLCs, laser scanners, bar codes, weigh scales, label printes etc. Also programmable hand held scanners.
Stephen Wynn	Consultancy	poa	Most experience of financial planning, and mathematical areas: operational research, quality control, experimental design.

OTHER PRODUCTS

COMPANY	PRODUCT	PRICES(£)	DETAILS
Adfee	Employment	poa	Contractors and permanent employees
APL-385	Typefaces	poa	Variants of the APL2741 typeface available to specification.
APL Borealis Inc.	APL Training	poa	Hands-on courses in Introductory, Intermediate, Advanced and Windows APL. Courses are customized and flexible, and may be delivered on-site, with strong emphasis on methods for efficient and maintainable APL systems development.
Atomic Vector	RELCELL	poa	Details: Report Writer, Data Mining, Database Visualization, and Export to Excel from relational databases. Offered in English, Spanish, Italian, Portugese, French and German. With this product, you don't have to know English in order to use SQL. Point and click with easy to use Wizards.
	APL Games Volume 1	$15	Details: Volume 1 - A collection of four APL games on cd, saved from past yearly issues of the APL Planet. Great way to demonstrate the capabilities of APL, and to entice the younger generations into APL. Written entirely in Dyalog APL.
ComLog	Comic-Logger	$25.95+p&p	APL*PLUS II comic-book inventory system. Shareware version available on America OnLine.
HMW	Employment	poa	Contractors and permanent employees placed.
I-APL Ltd	Books	poa	I-APL stocks books written to go with the I-APL interpreter and some APL Press books. For a list write to 11 Auburn Road, Bristol BS6 6LS, ring 0117 973 0036 or email acam@blueyonder.co.uk.
Oasis	Training	poa	Introductory courses in APL Advanced courses for different APL versions
Optima	IBM Compatible	poa	Complete business solutions including hardware, software and support plus additional network installation and maintenance if required.
Renaissance Data Systems	Booksellers		The widest range of APL books available anywhere. See Vector advertisements.
Right Seat Software	Vox Proxy	$199 (comm) $69.95 (edu)	Vox Proxy is authorware for PowerPoint(r) 2000 or 2002 which allows the use of Microsoft Agent Technology (3D talking animated characters) within slide shows. VP appears on PowerPoint's main menu and provides editing side-by-side with slides. Automated script-writing provides control of PowerPoint, allowing the use of characters for live presentations or fully-automated tutorials, demos, or training programs. Optional CD Prep program allows the user to create auto-starting CD's that will play on any version of PowerPoint or without PowerPoint.

OVERSEAS ASSOCIATIONS

GROUP	LOCATION	JOURNAL	OTHER SERVICES	Ann.Sub.
ACM SigAPL	International	APL QuoteQuad	Conferences; APL white pages; web site	$30
APL Bay Area	USA N. California	APLBUG	Monthly Meetings (2nd Monday)	$20
APL Club Austria	Austria	-	Quarterly Meetings	200AS(indiv), 1000AS(corp)
APL Germany e.V.	Germany	APL Journal	Semi-annual meetings	DM60
Ass. Francophone pour la promotion d'APL	France	Les Nouvelles d'APL		FF350 (private) FF2800 (Company)
BACUS	Belgium	APL-CAM	Conferences & Seminars	£18 ($30)
Danish SIG	Denmark			
Dutch APL Assoc.	Holland	Vector provided	Mini-congress, APL ShareWare Initiative	
FinnAPL	Helsinki, Finland	FinnAPL Newsletter	Seminars on APL	100FIM(private), 30(student), 1000 (Co)
Japan APL Assoc	Tokyo	APL Journal	Monthly meetings (4th Sat)	10,000yen to join
NY SigAPL	New York, USA	Big Apple APL	Monthly meetings	$35/$25(ACM)

Rome/Italy SIG	Roma, Italy			
SE APL Users Grp	Atlanta, Georgia	The APL Planet	Semi Annual meetings	$13 (US), $25 (non-US)
SovAPL	Moscow, Russia	-	Seminars and Annual Meeting	
SwedAPL	Sweden	SwedAPL Nytt	Semi-annual meetings, seminars	SEK 75
SWAPL	Texas, USA	SWAPL		$18
Swiss APL (SAUG)	Bern	Part of Qtly SI-Info		SF60 (SI) + SF20 (SAUG)
Toronto SIG	Toronto, Canada		Occasional meetings, APL Skills Database, Toronto Toolkit	

ADDRESSES

ORGANISATION	CONTACT	ADDRESS, TELEPHONE, FAX, EMAIL etc.
ACM SigAPL	David Siegel	ACM, 1515 Broadway, 17th Floor, New York, NY 10036, USA (Subs only)
ADAPTA Software GmbH	Michael Baas	Stellinger Weg 19, 20255 Hamburg, Germany. Tel: +49 40 40170951 Fax: +49 40 40170952. Email: info@adapta.de
Adaptable Systems	Lois & Richard Hill	49 First Street, Black Rock 3193, Australia. Tel: +61 3 9589 5578 Fax: +61 3 9589 3220 Email: hillrj@melbpc.org.au
Adfee	Bernard Smoor	Dorpsstraat 50, 4128 BZ Lexmond, Netherlands. Tel +31 347 342 337 Fax: +31 347 342 342 Email: adfee@concepts.nl
Andrews	Dr Anne D Wilson	12 Thorny Hills, Kendal, Cumbria LA9 7AL, UK. Tel: 01539-731205 Email: ADWilson@kencomp.net
APL-385	Adrian Smith	Brook House, Gilling East, York YO62 4JJ, UK. Tel: 01439-788385 Email: apl385@compuserve.com
APL2000 (Europe)	Fred Honea	see APL Systems IDC SL.
APL Bay Area APLBUG	Curtis Jones (Sec)	228 South 15th Street, San Jose, CA 95112-2150, USA Tel: +1 (408) 292-4060 Email: jonesca@vnet.ibm.com
APL Borealis Inc.	Richard Procter	381 Manor Road East, Toronto, Ontario M4S 1S7, Canada. Tel: (416) 457-7828. Fax: (416) 482-6582 Email: info@aplborealis.com
APL Club Austria	Harald F. Nelson	c/o N-TECH, Siebenbrunnenfeldg. 4-6, A-1050 Wien, Austria. Tel: +43 1 5458063 Fax: +43 1 5458063-17
APL Germany e.V.	Dieter Lattermann	Rheinstraße 23, D-69190 Walldorf, Germany. Tel: +49 6227-63469. Email: Dieter_Lattermann@t-online.de
APL Software\Services	Dick Holt	3802 N Richmond St, Suite 271, Arlington, VA 22207 USA Tel: +1 (703) 528-7624; Fax: +1 (703) 528-7617; Email: dick.holt@juno.org
APL Solutions Inc	Eric Landau	1107 Dale Drive, Silver Spring, MD 20910-1607 USA Tel: +1 (301) 589-4621 Fax: +1 (301) 589-4618 Email: aplsi@starpower.net
APL Systems IDC SL	Fred Honea	Alfredo Marquerie, 12 - 2 F, 28034 Madrid, Spain. Tel: +34 91 730 7008 (Office) +34 60 680 5949 (Mobile). Fax: +1 775 743 6131. Email: uksales@apl2000.net
Association Francophone pour la promotion d'APL	Ludmila Lemagnen	174 Boulevard de Charonne, F-75020 Paris, FRANCE Email: lemagnen@aol.com
AtomicVector Productions Inc.	John Manges	413 Comanche Trail, Lawrenceville, GA 30044, USA. Tel: +1 (770) 972-3755 Email: E-mail: mailto:seapldoc@aol.com
AUSCAN Software Ltd	Richard Procter	PO Box 39, Mansfield, Ontario L0N 1M0 Canada Tel: +1-705-434-1239 Email: rjp@ca.inter.net
BACUS	Joseph De Kerf	Rooinberg 72, B-2570 Duffel, Belgium. Tel: +32 15 31 47 24
Beautiful Systems, Inc.	Jim Goff	308 Old York Road, Suite 5, Jenkintown, PA 19046, USA Tel: +1 (215) 886-2636; Fax: +1 (215) 886-4888 Email: BeautifulSystems@goffs.net
Camacho	Anthony Camacho	11 Auburn Road, Redland, Bristol BS6 6LS, UK. Tel: 0117-973 0036. email: acam@blueyonder.co.uk
Ray Cannon		21 Woodbridge Rd, Blackwater, Camberley, Surrey GU17 0BS, UK. Tel: 01252-874697 Email: ray_cannon@compuserve.com
Causeway Graphical Systems Ltd	Adrian Smith	The Maltings, Castlegate, MALTON, North Yorks YO17 7DP, UK. Tel: 01653-696760 Fax: 01653-697719 Email: adrian@causeway.co.uk

Paul Chapman		51B Lambs Conduit Street, London WC1N 3NB, UK. Tel: 020 7404 5401. Compuserve: 100343,3210
Cinerea AB	Rolf Kornemark	Box 61, S-193 00 Sigtuna, Sweden. Tel/Fax: +46 859 255 421 Email rolf@cinerea.se
Ian Clark	Ian Clark	1205 Deer Creek Drive, Plainsboro, NJ 08536, USA. Tel: +1 609 716 8832 Email: earthspot2000@hotmail.com
CODEWORK	Mauro Guazzo	Corso Lanza 105, 10133 Torino, Italy. Tel: +39 11 6601560 Fax: +39 11 6601560 Email: codework@codework-it.com
ComLog Software	Jeff Pedneau	18728 Bloomfield Road, Olney, MD 20832 USA Tel: +1 (301) 260-1435 Email: jeff@softmed.com
CoSy.com	Bob Armstrong	42 Peck Slip #4B, New York, NY 10038-1725, USA. Tel: +1 212-285-1864 Fax: +1 212-285-1864. E-mail: bob@CoSy.com.
David Crossley	David Crossley	187 Le Tour du Pont, 84210 ST DIDIER, France. Tel: +33.4.90.66.08.87 Email: crossley@au-village.com
Danish User Group	Helene Boesen	c/o Insight Systems ApS, Nordre Strandvej 119G, Hellebæk, Denmark
Dinosoft Oy	Pertti Kalliojärvi	Lönnrotinkatu 21C, 00120 Helsinki, FINLAND. Tel: +358 9 70028820 Fax: +358 9 70028824 Email: dinosoft@dinosoft.fi
Dittrich & Partner Consulting GmbH	Axel Holzmüller	Kieler Strasse 17, D-42697 Solingen, Germany. Tel: +49 212-260 660 Fax: +49 212-260 6666; Email: info@dpc.de
Dutch APL Association	Bernard Smoor (Sec)	Postbus 1341, 3430BH Nieuwegein, Netherlands. Tel: +31 347 342 337 Fax: +31 347 342 342
Dyalog Ltd.	Peter Donnelly	Grove House, Lutyens Close, Chineham Court, Basingstoke, Hampshire RG24 8AG, UK. Tel: 01256-338461 Fax: 01256-316559 Email: sales@dyalog.com; support@dyalog.com
DynArray Corporation	Dr James Brown	16360 Monterey Rd. Suite 260, Morgan Hill, CA 95037, USA Tel: +1 (408)-782-6648 Fax: +1 (408)-782-6627 Email:info@DynArray.com
Entropy Software Limited	George MacLeod	Calle Viena 13, Residencia Larrosa, Guardamar del Segura, 03140 Alicante, Spain. Tel: +34 61 00 41 940 Email: george.macleod2003@yahoo.co.uk
Evestic AB	Olle Evero	Berteliusvagen 12A, S-146 38 Tullinge, Sweden Tel & Fax: +46 778 4410 Email: olle.evero@mail.com
FinnAPL	Olli Paavola	Suomen APL-Yhdistys RY, FinnAPL RF, PL 1005, 00101 Helsinki 10, Finland Email: olli.paavola@pyr.fi
First Derivative Analytics Ltd.	Ken Chakahwata	114 Lemsford Lane, Welwyn Garden City, Herts AL8 6YP, UK Tel/Fax: 01707-339620. Email: KenChakahwata@compuserve.com
General Software Ltd	M.E. Martin	Little Wester House, Westerhill Road, LINTON, Kent ME17 4BS Tel: 01622 832463 E-mail: martin@gsoft.plus.com
Godin London Incorporated	Gaëtan Godin	12 Gerrard St., London, Ontario, Canada N6C 4C5 Tel: +1 (519) 679-8290 Fax: +1 (519) 438-6381 Email: info@godin.on.ca
HMW Computing	Chris Hogan	Hamilton House, 1 Temple Avenue, London EC4Y 0HA, UK. Tel: 0870-1010-469; Email:HMW@4xtra.com
Hoekstra Systems Ltd	Bob Hoekstra	Dominique, Salisbury Road, Woking, Surrey, GU22 7UR, UK. Tel: 01483-771028. Fax: 01483-837324 Email: Bob.Hoekstra@HoekstraSystems.ltd.uk
Michael Hughes		28 Rushton Road, Wilbarston, Market Harborough, Leics. LE16 8QL, UK. Tel: 01536-770998 Email: Michael@Hughes.uk.com
I-APL Ltd	Anthony Camacho	11 Auburn Road, Redland, Bristol BS6 6LS, UK. Tel: 0117-973 0036. Email: acam@blueyonder.co.uk
IBM APL Products	Nancy Wheeler	APL Products, IBM Silicon Valley Lab, Dept H36/F40, 555 Bailey Avenue, San Jose CA 95141, USA. Tel: +1 (408) 463-APL2 [+1 (408) 463-2752] Fax: +1 (408) 463-4488 Email: APL2@vnet.ibm.com
INFOSTROY	Alexey Miroshnikov	3 S. Tulenin Lane, St. Petersburg 191186 Russia. Tel:+7 812 325-9797 Fax:+7 812 311-2184 Email:alm@infostroy.ru
Insight Systems ApS	Helene Boesen	Nordre Strandvej 119G, DK-3150 Hellebæk, Denmark Tel:+45 70 26 13 26 Fax: +45 70 26 13 25 Email: info@insight.dk
JAD Software	David Crossley	175 East 96th St., Apt. 17G, New York, NY 10128 Country: USA Tel: +1 (212) 369-6713 Fax: +1 (212) 761-0124 Email: jadsms@usa.net
Japan APL Assoc	Toshio Nishikawa	1-8-13 Masujima Buld.6F Higashi Gotanda Shinagawa-ku, Tokyo Japan 141-0022. Tel: +81 (03) 3280-0411 Fax: +81 (03) 3280-0418 Email: KYY00361@niftyserve.or.jp

J Austria	Joachim Hoffmann	Hochsteingasse 13/26, 8010 Graz, Austria. Tel:0043 (0)316 91 42 51 Mobile: 0043 (0)699 1 91 42 51 2. Email: joachim.hoffmann@gmx.at
JSoftware Inc.	Eric Iverson	33 Major Street, Toronto, Ontario, Canada M5S 2K9. Tel: +1 (952) 470-7345 Fax: +1 (952) 470-9202 Email: info@jsoftware.com
KJK-tieto Oy	Kimmo Kekäläinen	Merikasarminkatu 10 B 56,00160 Helsinki, Finland. Tel: +358 50 55 27 207; Email: Kimmo.Kekalainen@pp.htv.fi
Lambent Technology Ltd	Stephen Taylor	81 South Hill Park, London NW3 2SS, UK. Tel: +44(0)20 7813 3786. Email: sjt@lambenttechnology.com
Phil Last Ltd	Phil Last	146 Crossbrook Street, Cheshunt, Herts, EN8 8JY, UK. Tel: 01992-633807 Fax: 0121-359 0375 Email: phil.last@net.ntl.com
Lescasse Consulting	Eric Lescasse	18 rue de la Belle Feuille, 92100 Boulogne, France. Tel: +33.1.46.05.10.76 Fax: +33.1.46.04.60.23 Email: eric@lescasse.com
Lingo Allegro USA, Inc	Walter G. Fil	203 N. LaSalle Street, Suite 2100, Chicago, Illinois 60601, USA. Tel:+1 (800) 546 4621 E-mail: lingo-allegro@visto.com
Lucas Solutions	Jim Lucas	Stubbedamsvej 9C, 3.tv., 3000 Helsingør, Denmark Tel: +45 49 26 52 42. Email: jel@danbbs.dk
Mackay Kinloch Ltd	Alastair Kinloch	519 Webster's Land, Edinburgh EH1 2RX, Scotland, UK. Tel: +44 (0)131 228 5235 Email: alastair.kinloch@btinternet.com Voicemail/Pager 07626 98 3858
MasterWork Software Ltd	Fraser Jackson	PO Box 56-036, Tawa, Wellington, New Zealand. Tel: +64 (4) 232-4440 Fax: +64 (4) 232-4452 Email: fraser.jackson@xtra.co.nz
Mercia Software Ltd.	Gareth Brentnall	Mercia House, Ashted Lock, Aston Science Park, Birmingham, B7 4AZ, UK. Tel: 0121-359 5096. Fax: 0121-359 0375
MicroAPL Ltd.	Richard Nabavi	The Roller Mill, Mill Lane, Uckfield, E.Sussex TN22 5AA Tel: 01825 768050. Fax: 01825 749472 Email: MicroAPL@microapl.demon.co.uk
Milinta Inc.	Dan Baronet	Contact Dan Baronet at danb@milinta.com
Ellis Morgan	Ellis Morgan	Myrtle Farm, Winchester Road, Stroud, Petersfield, Hants GU32 3PE, UK. Tel: 01730-263843 Email: Ellis@mrtifrm.demon.co.uk
NY Sig	David Siegel	PO Box 2697; New York, NY10163-2697, USA. Email: NYSIGAPL@ACM.ORG
Oasis b.v.	Louis Rijkse	Lekstraat 4, 3433 ZB Nieuwegein, Holland. Tel: +31 30 60 65 336 Fax: +31 30 60 65 844 Email: rijkse@oasis.nl
Omega Computing Inc	Alan Graham, Andrew Chou	3 Columbus Avenue, Edison, NJ 08817, USA. Tel: +1 (732) 985 9519 Email: alangraham@mindspring.com
Optima Systems Ltd	Paul Grosvenor	Optima House, Mill Court, Spindle Way, Crawley, West Sussex, RH10 1TT, UK. Tel: 01293 562700 Fax: 01293 562699 Email:paul@optima-systems.co.uk
Qualedi Inc.	Nicole Schless Georges Brigham	121 West Main Street, Milford, CT 06460, USA. Tel: +1 (203) 874-4334 Fax: +1 (203) 876-9083. Email: sales@qualedi.com; info@qualedi.com
RadSys Technologies AB	Randolph Schrab	Lovsangarv. 18, S-756 52 Uppsala, Sweden. Tel: +46 18 32 41 53 Fax: +46 708 1996 11 Email: randolph.schrab@radsys.se
Renaissance Data Systems	Ed Shaw	P.O. Box 511, Botsford, CT 06404, USA. Tel: +1 (203) 270-9729 Email: apibooks@earthlink.com
Resources & Results	Frank Rhodes	342 Glen Oaks Blvd, Dallas, TX 75232, USA. Tel: +1 (817) 501-9825 Email: frank21@decision-support.net
RE Time Tracker Oy	Richard Eller	Mikonkatu 8 A, 2.krs, PL 363, 00101 Helsinki, Finland. Tel: +358 9-621 3300 Fax: +358 9-621 3378 Email: re@rett.fi
Right Seat Software, Inc.	Tom Atkins	1110 12th Street, Golden, CO 80401, USA. Tel:+1 303 278 2244. Fax: +1 303 278 6967. Email: info@voxproxy.com
Graeme Robertson	Graeme Robertson	15 Little Basing, Basingstoke, Hants RG24 8AX, UK. Tel: +44 0 1256-364071 Email: GraemeDR@aol.com
The Rochester Group Inc.	Robert Miller	600 Park Avenue, Rochester, NY 14607-2926, USA. Tel: +1 (716) 271-1110. Fax: +1 (716) 271-1230
Rome/Italy SIG	Mario Sacco	Casella Postale 14343, 00149-Roma Trullo, Italy Email: mario.sacco@tin.it
SE APL Users Group	John Manges	413 Comanche Trail, Lawrenceville, GA 30044, USA Tel: +1 (770) 972-3755 Email: seapldoc@aol.com
Shepp & Associates LLC	Andrew Shepp	1312 Washington Avenue, 6th Floor St. Louis MO 63103, USA Tel: +1 (314) 621-3272 Fax: +1 (314) 621-4267 Email: ashepp@compuserve.com

Snake Island Research Inc	Bob Bernecky	18 Fifth Street, Ward's Island, Toronto, Ontario M5J 2B9 Canada Tel: +1 (416) 203-0854 Fax: +1 (416) 203-6999 Email: bernecky@interlog.com
SOCAL (South California)	Roy Sykes Jr	Sykes Systems Inc, 4649 Willens Ave, Woodland Hills, CA 91364-3812 USA. Tel: +1 (818) 222-2759 Fax: +1 (818) 222-9250
Soliton Associates	Benoit Paquin	Soliton Denmark, Havsgårdsvej 4, 2900 Hellerup, Denmark Email: sales@soliton.com
SovAPL Russian Chapter of SIGAPL	Alexander Skomorokhov	PO Box 5061, Obninsk-5, Kaluga Region 249020, Russia Tel: +7(08439)47109 Fax: +1 (530) 6885510 Email:askom@obninsk.com
Strand Software Inc	Anne Faust	P.O. Box 330, Excelsior, MN 55331 USA Tel: +1 (952) 470-7345 Fax: +1 (952) 470-9202 Email: info@strandsoft.com
Rex Swain	Rex Swain	8 South Street, Washington, CT 06793 USA. Tel: +1 (860) 868-0131 Fax: +1 (860) 868-9970 Email: rex@rexswain.com
SwedAPL	Christer Ulfhielm	Novator Consulting Group AB, Svärdvägen 11C, S-182 33 Danderyd Sweden. Tel: +46 8 622 63 50 Fax: +46 8 622 63 51 CServe: 100341,404
Swiss APL User Group		Swiss APL User Group, CH-3001, Bern 1, Switzerland Email: si@ifi.unizh.ch
Sykes Systems Inc	Roy Sykes Jr	4649 Willens Ave., Woodland Hills, CA 91364, USA Tel: +1 (818) 222-2759 Fax: +1 (818) 222-9250
Toronto SIG	Richard Procter	PO Box 55, Adelaide St. Post Office, Toronto Ontario M5C 2H8, Canada Email: info@torontoapl.org
Weighahead Systems	Philip Bulmer	Camberley House, 1 Portesbery Road, Camberley, GU15 3RB, UK. Tel +44 1276 20789 Email: sales@weighahead.com
Stephen Wynn		8 Clarence Gardens, Brighton, Sussex BN1 2EG, UK. Tel: 01273-327238 Email: centre@cwcom.net
Zark Incorporated	Gary A. Bergquist	23 Ketchbrook Lane, Ellington CT 06029, USA. Tel: +1 (860) 872-7806

FTP SITES

IBM APL2	ftp.software.ibm.com/ps/products/apl2
Waterloo Archive	archive.uwaterloo.ca/ftparch/languages/apl
APL-to-ASCII	archive.uwaterloo.ca/languages/apl/workspaces/aplascii

WORLD WIDE WEB SITES

ACM SigAPL	www.acm.org/sigapl/
Adapta Software	www.adapta.de/
Adaptable Systems	www.assuredsystems.com.au/
AFAPL	www.afapl.asso.fr/ (Journal available on line)
APL2000	www.APL2000.com/
APL-385	www.demon.co.uk/apl385/
APL Forum	www.aplforum.com
APL Journal, Germany	www.rhombos.de/apljourn.htm
APL Borealis Inc.	www.aplborealis.com
APL Systems IDC SL	www.apl2000.net
AUSCAN	http://home.ca.inter.net/rjp/auscan/
Eke van Batenburg	wwwbio.LeidenUniv.nl/~Batenburg/index.html
Causeway	www.causeway.co.uk/
CODEWORK	www.codework-it.com/tangram/eng/
CoSy (Bob Armstrong)	CoSy.com/
Dinosoft Oy	www.dinosoft.fi/
Dittrich & Partner	www.dpc.de; www.apl-online.de
DMOZ - Open Directory	http://dmoz.org/Computers/Programming/Languages/APL/
Dyalog Ltd	www.dyalog.com/
DynArray	www.dynarray.com/

FinnAPL	www.pyr.fi/apl/
Godin London Inc	www.godin.com/
Houben (IQL)	www.apl.olap.club.tip.nl
Hoekstra Systems	www.HoekstraSystems.ltd.uk/
IBM APL2	www.ibm.com/software/ad/apl
Infostroy	www.infostroy.ru
Insight Systems ApS	www.insight.dk/
Japan APL Association	www.naska.co.jp/JAPLA/
JSoftware Inc	www.jsoftware.com/
KJK-tieto Oy	www.kjk-tieto.com
Lambent Technology	www.lambenttechnology.com
Lescasse Consulting	www.lescasse.com/
Lingo Allegro USA Inc	www.lingo.com/
Mackay Kinloch	http://mackaykinloch.ltd.uk/
MicroAPL Ltd	www.microapl.co.uk/apl
Milinta Inc	www.milinta.com
Oasis b.v.	www.oasis.nl/
Optima Systems Ltd	www.optima-systems.co.uk
Qualedi, Inc.	www.qualedi.com
Renaissance Data	www.aplbooks.com/
Resources & Results	www.decision-support.net
RE Time Tracker Oy	www.rett.fi/
Right Seat Software, Inc.	www.voxproxy.com
The Rochester Group Inc.	www.rochgrp.com/
Shepp & Associates	www.digitravel.com/
SigAPL	www.acm.org/sigapl/
Soliton	www.soliton.com/
Snake Island Research Inc.	www.snakeisland.com
Strand Software Inc.	www.strandsoft.com/
Rex Swain	www.rexswain.com/
Toronto SIG (for Toolkit)	www.torontoapl.org/
Jim Weigang	www.chilton.com/~jimw/
Weighahead Systems	www.weighahead.com

Zark Newsletter Extracts

Introduced by Jon Sandles

Here is another Zark newsletter extract. As always, don't hesitate to send in any solutions you may have that you may be inspired to investigate. There may be new more elegant ways of solving an old problem, which may also turn out to be faster. Please don't be afraid to try and shed "new" light on the "old" problems. Once again, thanks to Jonathan Barman for helping typeset this column. We haven't had much feedback of late - I hope you are still finding these enjoyable!

Utility Corner: Numbers to Words

(The purpose of this column is to make you more productive by introducing you to utility functions. Think of utility functions as APL functions that have names instead of symbols. By expanding your function vocabulary, you'll be able to write APL code that's more concise, more efficient, and more readable.)

In last issue's *Limbering Up* column, you were asked to define a utility function that compares two similar functions.

When maintaining a large APL application, you will occasionally find yourself face-to-face with two copies of the same function. Yet the copies are not the same. Perhaps two different programmers enhanced the same function in different ways. You would like to know how the functions are different.

The first objective is to bring the two functions together so they can be compared. Yet, if you copy both functions into the workspace, only one will survive (since they have the same name!) To compare the functions you must convert them to character array representations via ⎕CR (canonical representation) or ⎕VR (visual representation). For example:

```
      )COPY WS1 MODEL
SAVED...
      CR1←⎕CR 'MODEL'
      )COPY WS2 MODEL
SAVED...
      CR2←⎕CR 'MODEL'
```

Now you have two character matrices that "look like" the function. The task was to write a function CRCOMPARE (or VRCOMPARE if you are using ⎕VR) that returns/displays the differences between the functions. For example, given these two functions,

```
     ∇ CALC
[1]    A←B←C
[2]    D←3
[3]    E←D-A+B+C
[4]    F←D+E
[5]    A←0 0⍴0
[6]    B←0 0⍴0
[7]    G←F×0.5
[8]    'DONE'
[9]    Z←0.01×D
     ∇

     ∇ CALC
[1]    A←B←C ⋄ D←3
[2]    E←D-A+B+C
[3]    F←D+E
[4]    G←F×0.5
[5]    A←B←0 0⍴0
[6]    'DONE'
[7]    Z×D÷100
     ∇
```

the comparison looks like:

```
      CR1 CRCOMPARE CR2

[1]  A←B←C
[2]  D←3
---- replaced by ----
[1]  A←B←C ⋄ D←3

[6]  A←0 0 ⍴0
[7]  B←0 0 ⍴0
---- deleted ----

[5]  A←B←0 0 ⍴0
---- added ----

[9]  Z←0.0×D
---- replaced by ----
[7]  Z←D÷100
```

A straightforward approach to this problem is to place the representation of the old function on the left side of your desk, and the new one on the right. Move your hands down both functions until you come to lines that don't match. Keeping your left hand on the first mismatching line, move your right hand down until you find a matching line. If you don't, more your left hand down to the second line and again scan through the remaining lines of the function to the fight until you find a match. Keep this up until you find a match or run out of lines. If you run out of lines, report all remaining lines in the left function as "replaced" by the remaining lines in the right function.

If there *is* a match, and if your left hand is still pointing at the furst remaining line, report the lines of the right function up to the matching line as "added" lines. If your right hand is pointing at the first remaining line, report the lines of the left function up to the matching line as "deleted" lines. Otherwise, report the lines of the left function up to the matching line as "replaced" by the lines of the right function up to the matching line. Then repeat this process, starting at the matching lines.

The trickiest part of the CRCOMPARE function is formatting the lines so they have line numbers and so the comment and label lines are exdented (lines [86] to [106]):

```
     ∇ R←CR1 CRCOMPARE CR2;A;C;D;F;I;I1;I2;L;M;N1;N2;NL;S;T;U;Z;⎕IO
[1]   ⍝ Compares the canonical representa-
[2]   ⍝ tions (⎕CR) of two similar fns CR1
[3]   ⍝ and CR2 and returns a character
[4]   ⍝ vector (with embedded newlines) of
[5]   ⍝ the differences between the two.
[6]   ⍝
[7]    ⎕IO←0 ⍝ Origin 0 simplifies logic
[8]    R←'' ⍝ Initialise the result
[9]    →(CR1≡CR2)/0 ⍝ Exit if the same
[10]   NL←⎕TC[1] ⍝ Newline (⎕RCNL)
[11]  ⍝ Pad both to the same no. of cols:
[12]   C←0,(1↓⍴CR1)⌈1↓⍴CR2
[13]   N1←1↑⍴CR1←(C⌈⍴CR1)↑CR1
[14]   N2←1↑⍴CR2←(C⌈⍴CR2)↑CR2
[15]   I1←0 ⍝ Row index into CR1
[16]   I2←0 ⍝ Row index into CR2
[17]  ⍝
[18]  ⍝ Grab as many of each as possible:
[19]  LOOP1:I←(N1-I1)⌊N2-I2
[20]  ⍝ Where is next mismatch line?
[21]   I←(∨/CR1[I1+⍳I;]≠CR2[I2+⍳I;])⍳1
[22]   I1←I1+I
[23]   I2←I2+I
[24]  ⍝ Branch if more in CR1:
[25]   →(I1<N1)/L1
[26]  ⍝ Exit if none left in either:
[27]   →(I2=N2)/0
[28]  ⍝ Else remainder in CR2 added:
[29]   L←I2+⍳N2-I2 ⍝ Line numbers
[30]   T←CR2[L;] ⍝ Text of lines
[31]   C←0 ⍝ Class of format (0=add)
[32]   Z←0 ⍝ Return label (exit)
[33]   →FMT ⍝ Branch to format lines
[34]  ⍝
[35]  ⍝ More left in CR1; branch if more
[36]  ⍝ left in CR2:
[37]  L1:→(I2≠N2)/L2
[38]  ⍝ Else remainder in CR1 deleted:
[39]   L←I1+⍳N1[I1 ⍝ Line numbers
[40]   T←CR1[L;] ⍝ Text of lines
[41]   C←1 ⍝ Class of format (1=delete)
[42]   Z←0 ⍝ Return label (exit)
```

```
[43]   →FMT ⍝ Branch to format lines
[44]   ⍝
[45]   ⍝ More left in both; 1st rows sof
[46]   ⍝ each are different; search CR2 for
[47]   ⍝ the next line of CR1:
[48]   L2:S←I1 ⍝ This is the starting point
[49]    Z←LOOP1 ⍝ Return label
[50]   LOOP2:I←(((I2,0)↓CR2)∧.=CR1[I1;])⍳1
[51]   ⍝ Branch if found
[52]    →(N2>I2←I)/L3
[53]   ⍝ Iterate if some rows left in CR1:
[54]    →(N1>I1←I1+1)/LOOP2
[55]   ⍝
[56]   ⍝ Branch unless 1st CR1 row matches:
[57]   L3:→(S≠I1)ρL4
[58]   ⍝ CR2 lines have been added:
[59]    L←I2+⍳I ⍝ Line numbers
[60]    T←CR2[L;] ⍝ Text of lines
[61]    C←0 ⍝ Class of format (0=add)
[62]   ⍝ Increment counters for return:
[63]    I1←I1+1
[64]    I2←I2+I+1
[65]    →FMT ⍝ Branch to format lines
[66]   ⍝
[67]   ⍝ CR1 lines have been deleted:
[68]   L4:L←S+⍳I1-S ⍝ Line numbers
[69]    T←CR1[L;] ⍝ Text of lines
[70]   ⍝ Branch unless 1st CR2 row matches:
[71]    →(I≠0)/L5
[72]    C←1 ⍝ Class of format (1=delete)
[73]   ⍝ Increment counters for return:
[74]    I1←I1+1
[75]    I2←I2+I+1
[76]    →FMT ⍝ Branch to format lines
[77]   ⍝
[78]   ⍝ A later line in CR1 matches a
[79]   ⍝ later line in CR2 (or no match):
[80]   L5:C←2 ⍝ Class of format (2=replace)
[81]    M←I2+⍳I ⍝ Line numbers in CR2
[82]    U←CR2[M;] ⍝ Text of lines
[83]   ⍝ Increment counter for return:
[84]    I2←I2+I
[85]   ⍝
[86]   ⍝ Format here, line numbers first,
[87]   ⍝ right justified w/ right brackets:
[88]   FMT:L←(((3⌈ρ⍕⌈/L),0)⍕((ρL),1)ρL),']'
[89]   ⍝ Left justified, with both brackets
[90]   ⍝ and newline:
[91]    L←NL,'[',(L+.=' ')⌽L
[92]   ⍝ Characters that may begin labels:
[93]    A←'ABCDEFGHIJKLMNOPQRSTUVWXYZΔabcdefghijklmnopqrstuvwxyzΔ'
[94]   ⍝ Other characters within labels:
[95]    D←'0123456789_'
[96]   ⍝ First character in each line:
[97]    F←T[;0]
[98]   ⍝ Flag comment of label lines:
[99]    D←(F='⍝')∨(F∈A)∧∨/(T=':')∧<\~T∈A,D
[100]  ⍝ De-indent comment or label lines
[101]  ⍝ and include line numbers:
[102]   T←L,D⌽' ',T
```

```
[103] ⍝ Squeeze out trailing blanks
[104]  T←(,¢⍤\' '≠¢T)/,T
[105] ⍝ Brance by class of format:
[106]  →(L6,L7,L8,L9)[C]
[107] ⍝
[108] ⍝ Add (C=0):
[109] L6:R←R,T,NL,'---- added ----',NL
[110]  →Z
[111] ⍝ Delete (C=1):
[112] L7:R←R,T,NL,'---- deleted ----',NL
[113]  →Z
[114] ⍝ Replace (C=2):
[115] L8:R←R,T,NL,'---- replaced by ----'
[116]  L←M ⍝ Line no.s in CR2
[117]  T←U ⍝ Text of lines
[118]  C←3 ⍝ Class of format (end of repl)
[119]  →FMT
[120] ⍝ End of replace (C=3):
[121] L9:R←R,T,NL
[122]  →Z
      ∇
```

The trickiest part of the VRCOMPARE function is, well, all of it:

```
      ∇ R←VR1 VRCOMPARE VR2;I;I1;I2;E;EL1;EL2;J;L;N;N1;N2;NB1;NB2;NL;NL1;NL2;PL1;PL2;PL
USIOTA;S;T;U;Z;⎕IO
[1]   ⍝ Compares the visual representa-
[2]   ⍝ tions (⎕VR) of two similar fns VR1
[3]   ⍝ and VR2 and returns a character
[4]   ⍝ vector (with embedded newlines) of
[5]   ⍝ the differences between the two.
[6]   ⍝
[7]    ⎕IO←0 ⍝ Origin 0 simplifies logic
[8]    R←'' ⍝ Initialise the result
[9]    →(VR1≡VR2)/0 ⍝ Exit if the same
[10]   NL←⎕TC[1] ⍝ Newline (⎕TCNL)
[11]  ⍝ Define local function A PLUSIOTA L
[12]  ⍝ that returns (A[1]+⍳L[1]),
[13]  ⍝ (A[2]+⍳L[2]), and so on:
[14]   T←⎕FX 3 15⍴'R←A PLUSIOTA L
                   R←L/A-¯1↓0,+\L
                   R←R+⍳⍴R          '
[15]  ⍝
[16]  ⍝ Find where lines begin and end;
[17]  ⍝ consider just the important chars
[18]  ⍝ (newlines, or nonblanks after a
[19]  ⍝ blank or right bracket):
[20]   NL1←VR1=NL
[21]   I←NL1∨(VR1∈'∇ ')<0,¯1↓VR1∈'∇ ]'
[22]   I←I/⍳⍴I ⍝ Bits to indices
[23]   NL1←NL1[I] ⍝ Which are newlines?
[24]  ⍝ Flag NLs starting nonempty lines:
[25]   N←NL1>1↓NL1,1
[26]  ⍝ ... & of NLs that end those lines
[27]  ⍝ (including the end of line [0]):
[28]   E←(NL1\¯1↓1,NL1/N)/I
[29]  ⍝ Inds(+1) of NLs that start lines:
[30]   NL1←0,1+N/I ⍝ 0 to begin line [0]
[31]  ⍝ Inds of 1st nonblanks per line:
```

```
[32]   NB1←(1,¯1↓N)/I
[33]  ⍝ Lengths of entire lines (with
[34]  ⍝ trailing newline):
[35]   EL1←1↓E-NL1
[36]  ⍝ Lenghts of lines without line no.s
[37]  ⍝ (and without newlines):
[38]   PL1←E-NB1
[39]  ⍝
[40]  ⍝ Do the same for VR2:
[41]   NL2←VR2=NL
[42]   I←NL2∨(VR2∈'∇ ')<0,¯1↓VR2∈'∇ ]'
[43]   I←I/⍳⍴I
[44]   NL2←NL2[I]
[45]   N←NL2>1↓NL2,1
[46]   E←(NL2\¯1↓1,NL2/N)/I
[47]   NL2←0,1+N/I
[48]   NB2←(1,¯1↓N)/I
[49]   EL2←1+E-NL2
[50]   PL2←E-NB2
[51]  ⍝
[52]   N1←⍴NL1 ⍝ Number of lines of each
[53]   N2←⍴NL2
[54]   I1←0 ⍝ Line index into each
[55]   I2←0
[56]  ⍝
[57]  ⍝ Grab as many of each as possible:
[58]  LOOP1:N←(N1-I1)⌊N2-I2
[59]  ⍝ But only if they have same length:
[60]   →(0=N←+/∧\PL1[I1+⍳N]=PL2[I2+⍳N])/L2
[61]  ⍝ Compare next N lines:
[62]   L←PL1[I1+⍳N] ⍝ Their lengths
[63]   I←VR1[NB1[I1+⍳N]PLUSIOTA L]≠VR2[NB2[I2+⍳N]PLUSIOTA L]
[64]   →(∨/I)↓L1 ⍝ Branch if all the same
[65]  ⍝ Where is the next mismatch line?
[66]   N←((I⍳1)<+\L)⍳1
[67]  L1:I1←I1+N
[68]   I2←I2+N
[69]  ⍝ Branch if more in VR1:
[70]  L2:→(I1<N1)/L3
[71]  ⍝ Exit if none left in either:
[72]   →(I2=N2)/0
[73]  ⍝ Else remainder in VR2 added:
[74]   I←I2+⍳N2-I2 ⍝ Line indices
[75]   L←EL2[I] ⍝ ...and line lengths
[76]  ⍝ Text of lines:
[77]   T←VR2[NL2[I]PLUSIOTA L]
[78]   Z←0 ⍝ Return label (exit)
[79]   →ADD
[80]  ⍝
[81]  ⍝ More left in VR1; branch if more
[82]  ⍝ left in VR2:
[83]  L3:→(I2≠N2)/L4
[84]  ⍝ Else remainder in VR1 deleted:
[85]   I←I1+⍳N1-I1 ⍝ Line indices
[86]   L←EL1[I] ⍝ ...and line indices
[87]  ⍝ Text of lines
[88]   T←VR1[NL1[I]PLUSIOTA L]
[89]   Z←0 ⍝ Return label (exit)
[90]   →DEL
[91]  ⍝
```

```
[92]  ⍝ More left in both; 1st lines of
[93]  ⍝ each are different; search VR2 for
[94]  ⍝ the next line of VR1.
[95]  L4:S←I1 ⍝ This is the starting point
[96]   Z←LOOP1 ⍝ Return label
[97]  ⍝ Compare just lines of same length
[98]  LOOP2:L←PL1[I1]
[99]   E←L=I2↓PL2
[100]  I←(E\VR2[NB2[I2+E/⍳⍴E]∘.+⍳L]∧.=VR1[NB1[I1]+⍳L])⍳1
[101] ⍝ Branch if found
[102]  →(N2>I2+I)/L5
[103] ⍝ Iterate if some lines left in VR1:
[104]  →(N1>I1←I1+1)/LOOP2
[105] ⍝
[106] ⍝ Branch unless 1st VR1 row matches:
[107] L5:→(S≠I1)pL6
[108] ⍝ VR2 lines have been added:
[109]  J←I2+⍳I ⍝ Line indices
[110]  L←EL2[J] ⍝ ...and line lengths
[111] ⍝ Text of lines:
[112]  T←VR2[NL2[J]PLUSIOTA L]
[113] ⍝ Increment counters for return:
[114]  I1←I1+1
[115]  I2←I2+I+1
[116]  →ADD
[117] ⍝
[118] ⍝ VR1 lines have been deleted:
[119] L6:J←S+⍳I1-S ⍝ Line indices
[120]  L←EL1[J] ⍝ ..and line lengths
[121] ⍝ Text of lines:
[122]  T←VR1[NL1[J]PLUSIOTA L]
[123] ⍝ Branch unless 1st VR1 row matches:
[124]  →(I≠0)pL7
[125] ⍝ Increment counters for return
[126]  I1←I1+1
[127]  I2←I2+I+1
[128]  →DEL
[129] ⍝
[130] ⍝ A later line in VR1 matches a
[131] ⍝ later line in VR2 (or no match):
[132] L7:J←I2+⍳I ⍝ Line indices in VR2
[133]  L←EL2[J] ⍝ ...and line lengths
[134] ⍝ Text of lines:
[135]  U←VR2[NL2[J]PLUSIOTA L]
[136] ⍝ Increment counter for return:
[137]  I2←I2+1
[138] ⍝
[139]  R←R,NL,T,'---- replaced by ----'
[140]  R←R,NL,U
[141]  →Z
[142] ⍝
[143] ⍝ Add:
[144] ADD:R←R,NL,T,'---- added ----',NL
[145]  →Z
[146] ⍝ Delete (C=1):
[147] DEL:R←R,NL,T,'---- deleted ----',NL
[148]  →Z
      ∇
```

Jim Weigang pointed out several potential problems with the (relatively) straightforward approach presented above. To illustrate one such problem, try moving a line from `[20]` to `[10]`. What do you get? The try moving it from `[10]` to `[20]`. Now what? Jim offered algorithms and code for overcoming some of these problems, but the solutions are beyond the scope of this quarter's newsletter.

Limbering Up: Comparing Functions

(The purpose of this column is to work some flab off you APL midsection. Like muscles, your APL skills can atrophy if not exercised with adequate frequency and variety. This column presents a task for you to perform. Set aside a few minutes from your busy schedule and work the task. Mail in your solution and stay tuned for the results.)

The symbols for format (⍕) and execute (⍎) were chosen carefully by the implementers of APL to imply an inverse relationship. After all, ⍕ turns the numbers `3 4 5` into the character vector `'3 4 5'`. However, ⍕ and ⍎ are not perfect inverses. For example, ⍕ converts a numeric matrix into a character matrix, but ⍎ does not do the reverse.

Your task is to write a monadic function `EI` (Execute Inverse) that converts any array `A` to a character vector such that `A≡⍎EI A`. There are a number of useful applications for a function that translates an APL array into a character vector. We'll discuss them next issue.

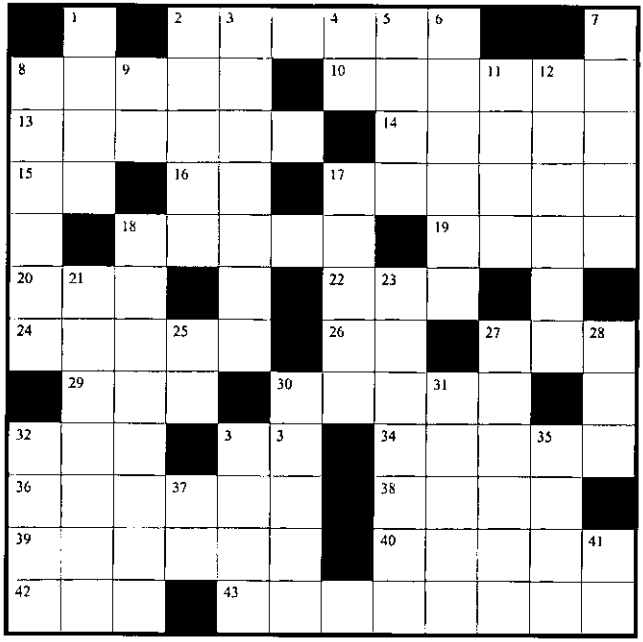
Send your solution to:

Vector Production
Brook House
Gilling East
York YO62 4JJ
UK email apl385@compuserve.com

The notable functions and their authors' names will be printed in the next issue of *Vector*. Good luck and happy limbering.

Reprinted with kind permission from Zark APL Tutor News, a quarterly publication of Zark Incorporated, 23 Ketchbrook Lane, Ellington, CT06029, USA

New Crossword Puzzle and 20.1 Solution



Across

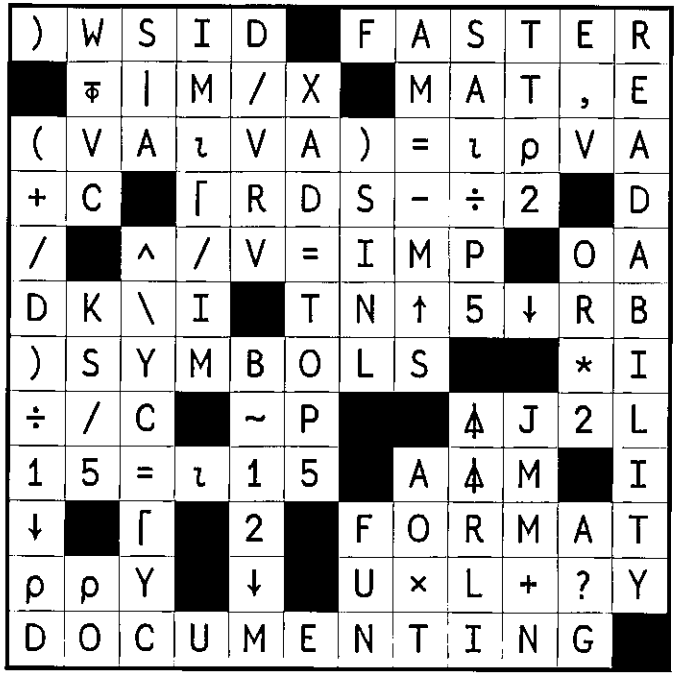
2. Start with a blank slate_____
8. Preserve your work
10. (|K)×(|RL), expressed as words
13. The location in vector E of the elements in K[⍳VA]
14. The magnitude of the elements of the cumulative sum of the reciprocal of S
15. 2↑⍕'''R5↓3'''
16. □__, the current "seed" of the random number generator
17. Release storage from suspensions
18. List of all suspended identifiers
19. 1÷VEC
20. (⍴S)⍴1
22.)___ when done

24. (+/E)ρYPV for positive integer vector E, scalar YPV
26. □__, for timing algorithms
27. List of all suspended functions
29. MP+1|-MP
30. Delete a stored workspace
32. Branch to line labelled MB
33. One random element of ιW
34. (VιAC)=□IO+ρV
36. (LTO↓JS),LTO↑S for positive scalar LTO
38. □IO++/^\L≠OV for vector L, scalar OV
39. 0 if RLM=I, 1 if RLM>I, ¯1 if RLM<I
40. ((POP,(ρN)-POP)/1 0)/N
42. 2⊛×\XSρ2 in origin 1
43. ∨/BD∘.=PAYRL for vector PAYRL

Down

1. +\(1++/^\S≠5)ρ1 for vector S, □IO=1
2. Names of arrays in the workspace
3. V[LI+ιCE] for scalar LI
4. Format to E decimal places ...
5. A-A×⌊(|R)÷A
6. (E+F×RK)÷F
7. (φιρρLSTC)⍉LSTC
8. Delete from active workspace
9. □__, all available characters
11. Cumulative product of SV
12. Divide into R the lesser of EE or S
17. Replace active workspace
18. Number of identifiers tracked in the active workspace
21. (⌈MTRX[;1])=(⌈MTRX[;2]) for two-column MTRX
23. (~FIR∈LPP)/FIR
25. □__, maximum number of significant digits to display
27. Retrieve objects without erasing or over-writing current objects
28. V+.×φI*(ιρV)-□IO for scalar I
30. Name of active workspace
31. (ιρOA)=_____ to flag first of each distinct value in vector OA
32. Branch to line labelled L if ...
33. A random number from 1 to J-B, in origin 1
35. The elements (simple vector) of the first V items of nested vector R in APL2 notation
37. M[(1+ρM)-ιρM] in origin 1
41. □__, the names of workspace objects

Solution to Crossword in 20.1



J-ottings 38: Decline and Rise: a Matter of Rank

by Norman Thomson

Rank is of enormous importance in J. There are two types of rank, noun rank and verb rank. The former can be applied to any object in the J universe, and is the "tally-of-the-shape" or in J :

```
nrank=.#@$
```

Verb rank is somewhat more subtle, and is of even greater importance in understanding J language constructs. Henry Rich put things with admirable conciseness in "J for C Programmers" :

"If you don't know the rank of a verb, you don't know the verb!"

It is not just primitive verbs which possess rank, user defined verbs also possess it, and so do compound verbs, that is verbs formed by combining or adapting primitive verbs using appropriate conjunctions and adverbs. Moreover, there is no excuse for J programmers not knowing any relevant verb ranks, since these are immediately obtainable for any verb by using the `b.` adverb with right argument 0 :

```
   %. b. 0                    NB. primitive
2 _ 2
   +/ b. 0                    NB. compound (adverbial)
_ _ _
   +.@*: b. 0                 NB. compound (using conjunction)
0 0 0
   mean=.+/ % #
   mean b. 0                  NB. user defined
_ _ _
```

Within the J literature I am not aware of any explicit categorisation of verbs by rank, which is why much of this article consists of a lengthy appendix which does just this. I hope that some readers may find this appendix helpful, perhaps long after this preamble has been forgotten.

As postings on the J conference testify, the distinction between `@` and `@:` is a stumbling block in the path of almost every new learner; what I hope to show here is how this block can be rapidly and effectively smoothed by a conscious endeavour to understand verb rank.

In general verbs have three unforced ranks, which are, in the order given in the dictionary definitions:

monadic rank, left rank, right rank

The word "unforced" acknowledges that verbs can have their basic ranks changed to an explicit value by using the rank conjunction " .

Classification by rank has some affinity with verb declensions in the classical languages, and although this analogy should not be pushed too far, it seems reasonable to think of verbs as being grouped by declensions along lines such as the following :

1. **Pure scalar verbs, all three rank vector items = 0**
2. **Pure list verbs, all rank vector items = 1**
3. **Verbs with hybrid ranks, mostly specialised verbs**
4. **Pure structural verbs i.e. all rank vector items = infinite**

4a. **'Pure structural plus' verbs, ranks are _ 1 _**

It should be stressed that a verb rank of, say, 1 should not be read as meaning that only objects of noun rank 1 are acceptable as arguments, but rather that all arguments will be *processed* as assemblages of rank 1 objects. There is an analogy with the operation of a clinic which has a large and motley assembly in the waiting room, from which patients may either be called one by one (processing at rank 0), or by families (that is lists, rank = 1) or the whole lot may be taken together as a single block of humanity (rank infinite). The analogy goes further than this, because the method of calling is independent of what happens once the surgery is entered, in other words, rank comes first, semantics later. Immediately following the above quotation in Henry's book is an explanation in detail of how rank is applied in particular cases, that is how rank and semantics are married — together in terms of cell and frame selection. However, focussing on ranks shows that, apart from the "special algorithm" verbs, J verbs can be grouped into quite a small number of categories.

A general problem in categorising verbs by rank is that of deciding when the monadic and dyadic forms are sufficiently related to each other to justify retaining this association even if they possess different rank vectors. At one extreme monadic > (open) and dyadic > (greater than) are completely unrelated in meaning, even although all their rank vector items are zero. On the other hand verbs such as monadic #: (= antibase 2) and dyadic #: (= antibase) have a

strong semantic association although formally they belong to different declensions.

The first declension is in many ways the most familiar, consisting of pure scalar verbs which fall into three broad categories: logical, arithmetic and algorithmic.

Declension 2 is the next easiest to review, since it contains verbs which make no sense other than when applied to lists, for example integers `i.` (= integers) and `;:` (= word formation).

Declension 4 consists of the pure structural verbs, all of whose ranks are infinite, meaning that their arguments are processed in their entirety as single objects.

Declension 4a is sufficiently similar to the previous set to merit sub-classifying rather than a new declension. In each case the primary monadic form has infinite rank, but there is also a matching dyadic form which applies left arguments at the list level (i.e. left rank = 1).

Finally declension 3 consists of hybrids. Three of these are general, `#:` (monadic and dyadic) and `{` (from), the other five are specialised algorithmic verbs such as `%.` (= matrix inverse / divide), which are of great value to a minority of J users, and of little or no interest to the rest.

Finally, compound verbs possess rank vectors in exactly in the same way as primitive verbs, and Henry's maxim might be extended :

> **"If you don't know the rank of a compound verb, you don't know the conjunction/adverb!"**

The dictionary descriptions may seem somewhat bewildering at first, with references to mv, lv, etc. Not so! The present appendix contains a list of conjunction rank rules, together with the permissible noun-verb combinations. For each vv combination the conjunction has the form "u conjunction v", and m=monadic, l=left, r=right.

atop (`@`) has rank vector mv lv rv, which means that in all three cases the rank of the compound verb is identical to that of the right verb in the compound. For *at* (`@:`), however, all ranks are infinite regardless of the composed verbs. So for

```
   2 3(+/@%)4 5          NB. rank of verb v(%) is zero
```

using the clinic analogy, the "patients" are called in matched pairs, and divided to give the result `0.5 0.6`. Then `+/` at rank 0 leaves items unchanged for a final result `0.5 0.6`. However, for

```
2 3(+/@:%)4 5
```

the patients are called in two blocks (left and right), which are divided as blocks to give `0.5 0.6`, and then "plus-insert" is applied to this single block to give `1.1`.

Informally the difference between atop and at is that u and v are more closely bound in the former. The name "atop" is very apt as it gives a picture of two creatures, one piggy-backing on the other, and thereby fusing to make a tight compound before any considerations of data come into play. "at" conveys less well the way in which infinite rank produces verb sequencing, that is "u following v", is a more meaningful phrase than "u at v". Here are some further examples of atop and at :

Compare `({:@*:)i.2 3` and `({:@:*:)i.2 3`. As above, the only rank which matters in the first case is that of `*:` (= square) which is 0. The semantic rule which extends tail when applied to scalars (i.e. at rank 0) is "no change" and so the final result is

```
0  1  4
9 16 25
```

In the second case, tail with infinite rank means the tail of the list of lists arising from squaring, which gives the result list `9 16 25`.

Compare `({:@#.)2 3$1 0 1 1 1 0` and `({:@:#.)2 3$1 0 1 1 1 0`

Here the rank of the rightmost verb is 1, and so in the first case tail is applied to *each* of the antibase 2's of the two three-lists which make up the right argument, giving a final result `5 6`. In the second case, tail applies to the entire result of `#.` which is a two-list, so that the final result is `6`.

Compare `2 3 4(+/@%.)m=.?2 3 3$10` and `2 3 4(+/@:%.)m`

This is a dyadic example, in which the rank of the rightmost verb is 2. `m` represents two sets of 3 by 3 linear equations with the same right hand side `2 3 4`. In the first case the result is the sums of *each* of the three solution sets ($x_1+ y_1+ z_1$, $x_2+ y_2+ z_2$) . In the second case summation applies to the two-list of solution values, so that the final result is ($x_1+ x_2$, $y_1+ y_2$, z_1+z_2) .

... and so I could continue. However, space is not unlimited, and in any case, if I have succeeded in explaining the general principles, then the reader will already have reached a point where he needs no further assistance from me!

Appendix: Verb Ranks

Declension 1, rank vector = 0 0 0

Logicals (all dyadic)

= Equal	~: Not-Equals
< Less Than	<: Less than or Equal
> Larger Than	>: Larger or Equal

Arithmetics

	Monadic	Dyadic		Monadic	Dyadic		Monadic	Dyadic
+	Conjugate	Plus	+.	Real/Imag	GCD(Or)	+:	Double	Not-Or
-	Negate	Minus	-.	Not		-:	Halve	
*	Signum	Times	*.	Len/Angle	LCM(And)	*:	Square	Not-And
%	Reciprocal	Divided by				%:	Sq Root	Root
^	Exponential	Power	^.	Natural Log	Logarithm			
<.		Lesser of	>.		Larger of			
|	Magnitude	Residue						
!	Factorial	Out Of						

Algorithmics

	Monadic	Dyadic		Monadic	Dyadic
?	Roll	Deal	p:	Primes	
j.	Imaginary	Complex	q:	Prime Factors	Prime Exponents
o.	Pi Times	Circle Functions	r.	Angle	Polar

Declension 2, all rank vector items = 1

	Monadic		Dyadic	
i.	Integers	1		
{	Catalogue	1		
;:	Word formation	1		
#.	Base 2	1	Base	1 1
".	Do	1		

Declension 3, hybrid ranks

	Monadic		Dyadic		
#:	Antibase 2	_	Antibase	1	0
{			From	0	_
%.	Matrix Inverse	2	Matrix Divide	_	2
A.	Atomic Permute	1	Atomic Permute	0	_
c.	Characteristic	2	Eigenvalues	0	2
C.	Cycle	1	Permute	1	_
p.	Roots	1	Polynomial	1	0

Declension 4, all ranks = infinite

Monadic	Dyadic
= Self-Classify	-. Less
< Box	-: Match
~. Nub	i. Index Of
~: Nub Sieve	E. Member of Interval
{: Tail	". Numbers
}: Curtail	
L. Level Of	

	Monadic	Dyadic		Monadic	Dyadic		Monadic	Dyadic
,	Ravel	Append	,.	Ravel Items	Stitch	,:	Itemize	Laminate
/:	Grade Up	Sort Up	\:	Grade Down	Sort Down			
;	Raze	Link	e.	Raze In	Member In			
$.	Sparse	Sparse	$:	Self-Reference	Self-Reference			
[	Same	Left	]	Same	Right			
{::	Map	Fetch						

Monadic and Dyadic

s: Symbol
u: Unicode
x: Extended Precision

Constant Functions, that is 9:, _8:,..., 0:, 1:,2:, ... 9: and also _: (infinity)

Declension 4a, rank vector = inf 1 inf

	Monadic		Dyadic		
$	Shape Of	_	Shape	1	_
|.	Reverse	_	Rotate (Shift)	1	_
|:	Default Transpose	_	Transpose	1	_
#	Tally	_	Copy	1	_
{.	Head	_	Take	1	_
}.	Behead	_	Drop	1	_
":	Default Format	_	Format	1	_

Conjunctions

	nn	nv	vn	vv		Rank	Vector	
&			x	x	bond, compose	mv	mv	mv
&.			x		under	mv	mv	mv
&:			x		appose	_	_	_
@				x	atop	mv	lv	rv
@.		x			agenda (with gerund)	mv	lv	rv
@:				x	at	_	_	_
.				x	dot	2	_	_
:.					obverse	mu	lu	ru
::					adverse	_	_	_
^:			x		power	_	_	_
`:		x			evoke gerund	_	_	_
;.			x		cut	_	1/2	_ *
d.			x		derivative	0		
D.			x		derivative	mu		
H.			x		hypergeometric	0	0	0

* 1/2 is appropriate because cut is the J mechanism for partitions, for which there is a wide range of options, some of which process the left argument on a rank 1 basis, others on a rank 2 basis.

Adverbs

	Monadic		Dyadic		
/	Insert	_	Table	_	_
/.	Oblique	_	Key	_	_
\	Prefix	_	Infix	0	_
\.	Suffix	_	Outfix	0	_
~	Reflexive	_	Passive	ru	lu
{.	Item Amend	_	Amend	_	_
b.	Basic characteristics	_	Boolean	0	0

Dyalog APL 2003 Conference

reported by Adrian Smith

Background

It is really only out of a spirit of loyalty to the old SigAPL conferences that Dyalog have been resisting pressure for an APL2000-style users' meeting. Now that the old *APLnn* annual conference has finally died, they can go ahead with a clear conscience and run a simple, focussed, commercially-oriented gathering for their own community.

As you can see, it worked! Lots of people came, from lots of countries. Everyone had a really good time, there was some decent technical content, and a broad selection of 'how we used APL to save/make $xxx' papers from users. The flat fee of £600 was actually quite good value, as it covered accommodation (full board) as well as the conference itself.

Location

I still have no real idea where the conference was. My geography of anything South and West of London was already hazy, but the experience of driving back, turning onto the A3 towards London (heading just North of East) then choosing *M25 West* left me well confused. Anyway, it was in a nice (mostly Victorian fake) fantasy castle somewhere in deepest Surrey. If you have heard of Occam's Razor, then you would have relished the association with the nearby village of Ockham,

and students of computing history will have been delighted to tuck in to the conference banquet in a hall decorated with the arms of Byron and Lovelace:

Argent 3 bendlets enhanced gules (Byron)

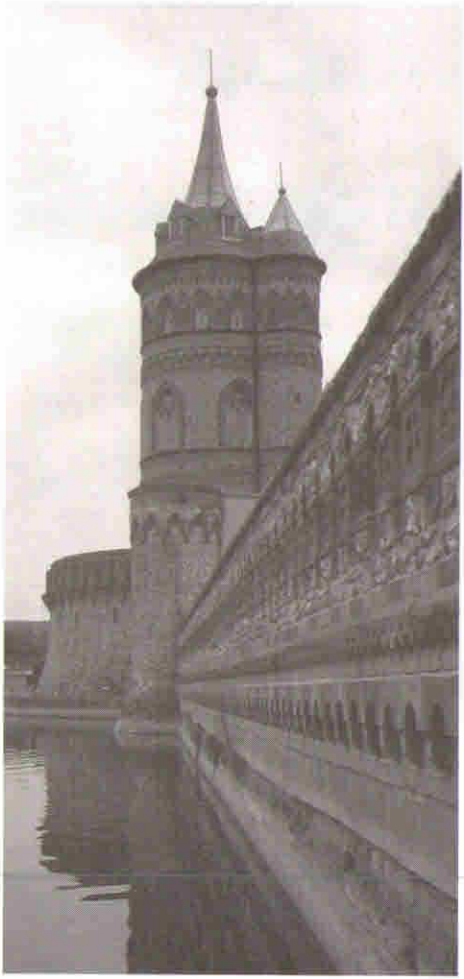
Gules on a chief indented sable, 3 martlets argent (Lovelace)

Lovers of formal languages and obscure notations might also be interested in the blazons (heraldic descriptions) of the two most interesting coats of arms. Was the Royal College of Heralds the true originator of object-oriented design?

The receptions, barbecue and banquet were held in and around the 'old' part of the building; the modern purpose-built conference centre was nicely hidden away in the stable block, and provided most of the rooms, the restaurant and plenty of comfortable sitting-out and relaxing areas. It did have some pretty funny tunnels (looked like the sort of place you might find a total perspective vortex if you took a wrong turning ...) and the usual lack of signposting. However I think everyone found the rooms in time, and once there, the chairs were comfortable, the pens worked, the aircon was only a bit too noisy, and the PA could be turned off at the first hint of howlround.

As always with these events, there was too much food and plenty of wine and beer (even Pimms on the first night), most of which was free courtesy of someone or other.

Conference Highlights



So what did I get out of it? I think the major benefit was meeting quite a surprising number of RainPro and NewLeaf users who are in my card-index, but are otherwise just names and licences. Now that they know who I am (just another harmless APL nut, like the rest of them) they will be much more likely to phone up and complain when something doesn't work quite right.

The average age of the attendees was encouragingly low, and it was the young folk who were starting to agitate for some serious object-oriented capabilities 'native in the interpreter'. This is good - control-structures finally kicked their way into APL due to pressure from 'modern' programmers and objects can do the same.

The talk from Peter Pichler and Svante Lewald was the most wonderful foil to Stephen Taylor's masterly exposition of the principles of XP. Here we had a classic example of rapid APL development, using all the good habits APLers have acquired over the years, followed by news that at least some of the programming community at large is beginning to get the message. The audience gave Stephen some very serious attention, and I can see why. If we can re-badge ourselves as long-time XP practitioners (instead of the followers of an obscure greek language that executes backwards) then the road to mainstream acceptance suddenly opens up. A long-shot but worth a try.

Conference Programme

The basic shape of the conference worked very well. On the first day we simply had end-to-end plenary sessions, with substantial user-presentations (each user had 90 minutes to describe and demo their work) and some light relief from John Daintree at the end of the day. Days 2 and 3 were split between a cycle of workshops (each repeated to keep numbers sensible) and shorter technical presentations.

One excellent idea (APL2000 please copy) was a web-based submission of up to 3 'favourite enhancement ideas' per delegate. These were printed and distributed with the conference packs, each given a 'developer cost' on a scale of 1 (trivial) to 100 (serious work for someone). Delegates were allowed to vote for as many as they liked, providing that the total vote did not exceed 100. We all came together for the wrap-up at the end, where the votes were revealed and the developers could add their own comments. It all made for a very interactive and enjoyable final session.

Day-1 - September 3

Welcome and Introduction from Peter Donnelly

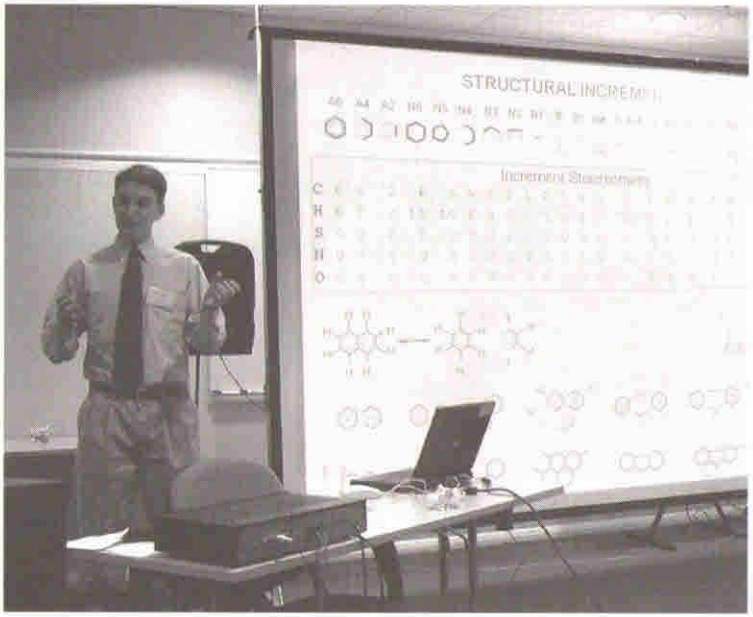
Pete explained why they had resisted calls for an independent users' meeting, until at last it had become clear that the annual SigAPL event was no longer supporting "our" user community. He also explained the change of name from *Dyadic* to *Dyalog Ltd.*

> "Dyadic/Dyalog have been in the APL business 'for ever' and have been selling hardware for 10 years or so, initially simply as a convenient way to ship our Unix APL systems. The APL world may be unaware of it, but Dyadic has become one of the leading suppliers of IBM RISC hardware in the UK. For various reasons, it became increasingly difficult to manage both parts of the business effectively, and we decided to split Dyadic into two separate entities, transferring the Dyalog APL business to a new company, Dyalog Limited. As a result, John Scholes and I are now able to devote 100% of our time to Dyalog APL"

Clearly, Pete was delighted to see such strong support from his users, and could take away a very positive feeling from the 'kick off' of the new company. We even got a souvenir Dyalog mug (sadly duck-free) to mark the event.

APL in Research at ExxonMobil by Steve Levine

Organic chemistry was never my strong subject, but I think that if someone had taught it with Mobil Research's brilliant matrix model of reaction chains, I would have understood an awful lot more.



This was one of those talks we will never see in detail in Vector – much of the more proprietary technical material was understandably excluded from the talk. However, from an APLer's point of view, one of the most hopeful signs was the emergence of a strengthened APL group from the merger between Exxon and Mobil.

From my own rather sad memories of the Nestlé absorption of Rowntree (and the slow death of the Rowntree APL group that followed) I felt a genuine concern for the APL community in Mobil Research when I heard of the ExxonMobil merger. Actually the opposite seems to have happened – Steve came from the Exxon side (he is a chemical engineer first, and an APLer second) and had previous experience of Fortran modelling, and LP optimisation.

Interestingly, Steve initially saw APL as a step back into the 60s, even from Fortran. Now he is an APL enthusiast, and is happy to carry the flag for a refinery modelling tool which already saves ExxonMobil many hundreds of millions of

dollars every year in crude oil costs, and can be expanded to work within the far larger asset-base of the joint company. The basic problem is down to finding the right mix of crude to allow a refinery to 'crack' out what the market demands in gasoline, kerosene and so on. The way to do this is to be able to associate the parameters required by the final product (expressed as measurables like viscosity, octane rating, lo-sulfur rating) with the chemistry of the cracker output.

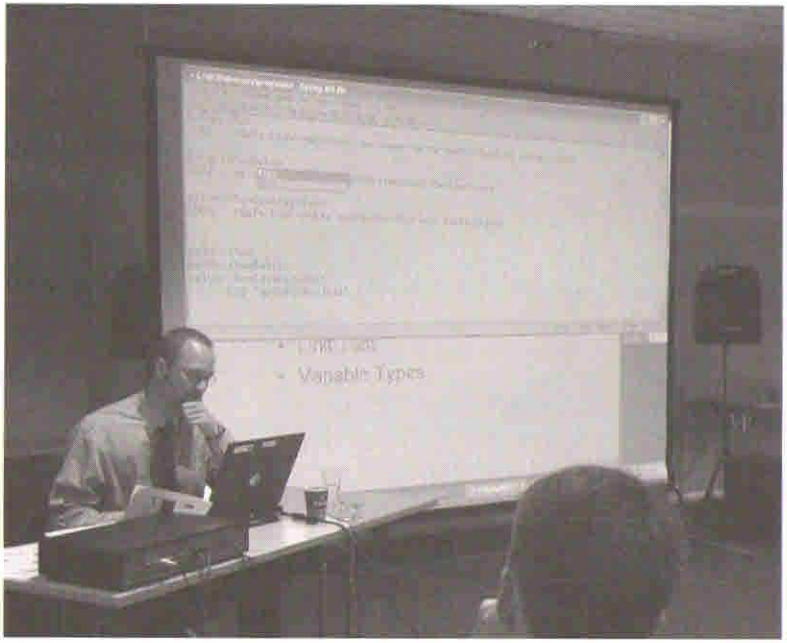
Then you need to model as exactly as you can the sequence of reactions that split up those naphthas, open up those rings, snip apart the paraffin chains to get from what you have to what you need. Finally you must associate the measurables of your various crude supplies with the required chemistry of your inputs and (hey presto) you can choose a selection of crudes which will let you operate the refinery with less waste, less energy and less catalyst. It all relies on a wonderfully neat APL reaction model, some APL-generated Fortran code which does the heavy lifting, and a brilliant user-interface designed as a sequence of plant diagrams with everything in terms the engineers understand.

As the oil gets scarcer and crude prices keep rising, the APL code in this model is going to get ever more business-critical. This system is well-conceived and very well supported for the simple reason that it delivers value that nothing else could. It should be around for a long time to come. As you can tell, I was mightily impressed.

Simcorp Dimension by Frederik Jensen

Simcorp were actually Dyadic's first big customer. Back in 1984 Frede Hansen saw Pete Donnelly at Helsinki and commented "interesting product – if you're in business next year we might buy it". The next year he said much the same, and in 1986 he bought it. Now *Simcorp Dimension* is used by over 100 investment companies across the world, and a team of 70 Dyalog programmers maintain nearly 700,000 lines of code.

Frederik started with the inevitable corporate PowerPoint, briefly showed us around the finished application and then focussed on some specific technical stuff of real interest to his audience.

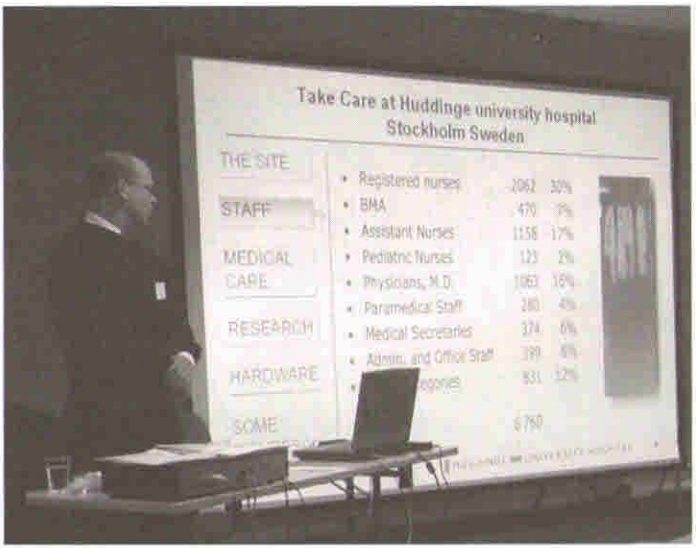


One immediately obvious thing was the huge benefit of the autocomplete capability introduced with Dyalog10. The SimCorp codebase predates namespaces by many years, so everything is carefully organised by prefix. Even with some thousands of functions in the root namespace, it was quick to find the right one to edit with just a few characters of typing and a click on the drop-list offered by the session.

The source code management is simple and powerful (master copies are held on component file, as always) but seemed to me to actually help the programmer in finding and fixing code, rather than getting in the way, as so many similar systems seem to.

The *TakeCare* system for hospital administration
by Peter Pichler and Svante Lewald

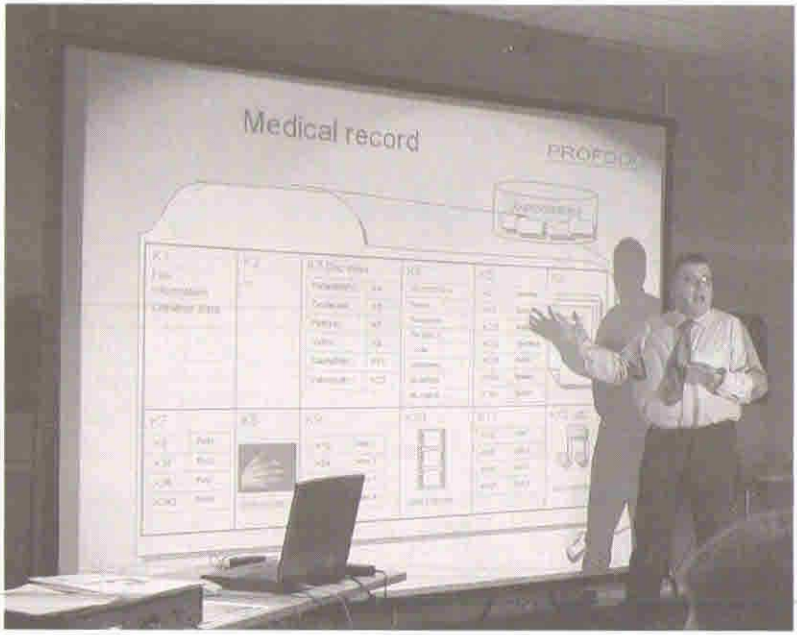
The conference programme says (with masterly understatement) "the authors will discuss the system architecture and describe the unique features provided by the system". 10 years ago, Peter dreamt of a unified healthcare administration system for Sweden. As you will see, the *TakeCare* system he has devised is well on track to become just that.



As it stands today, *TakeCare* supports the entire hospital administration of a substantial healthcare operation (pharmacies, doctors and nursing care, as well as the main hospital group) in the Stockholm area, with several hundred concurrent users running Dyalog APL desktop clients supported by a major Unix installation communicating via TCP/IP. Here are a few stats:

- 8,500 registered users
- 1,700 socket connections
- 25 server calls per second
- 700,000 medical records
- 100,000 EDI messages processed per month
- 4 million financial transactions per year
- better than 1 sec response time

The key to success has been a ruthless pursuit of simplicity, and a strictly user-centred approach to implementation. They started with something simple, implemented it well, and gradually increased coverage without ever outgrowing the fundamental architecture. Now they are well in sight of the original dream of creating a standard healthcare system for Sweden, simply by being the best and showing the way. The way they organise the 'database' is typical of their approach. In Sweden everyone has a unique serial number, for example I would be identified by the magic code 19530321*nnn* being the *nnn*th birth on this day. In *TakeCare* the patient record is a simple component file, with one file per patient, the file named from the patient number.



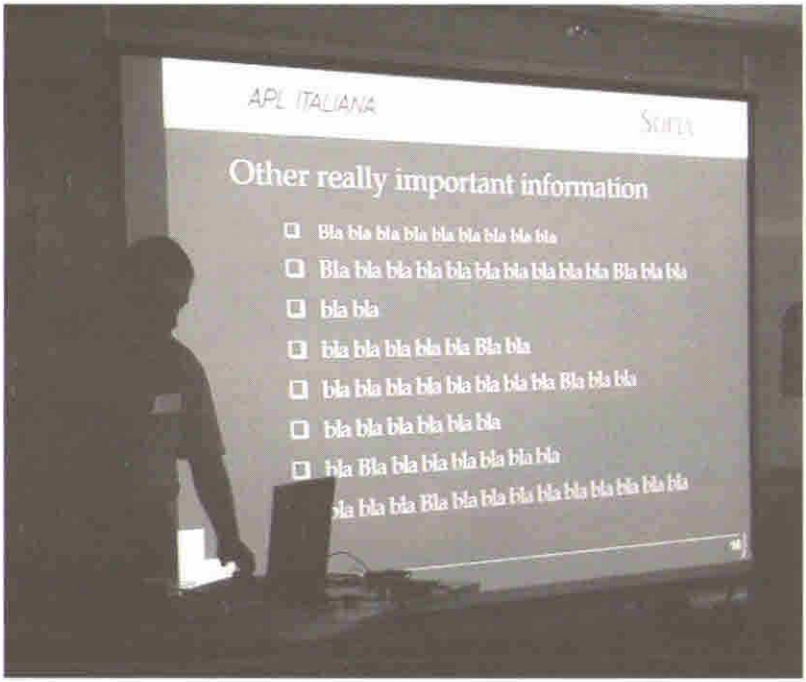
Extending the system to accept new information is simple, and the whole database scales effortlessly. When asked about deletion of old records, Peter shrugged and pointed to the way disk size is doubling every year. Just let it grow!

Note that component K2 is unused. Why? Who knows, and who cares? If it does no harm, just let it go and move on – if there was ever a paradigm for eXtreme programming in action, *TakeCare* is it. I wish we had a video of Peter's lovely hand gesture to illustrate ".... we just typed ...." as he illustrated their response to yet another outrageous request from a user. It spawned numerous imitators during

the conference, but no-one did it half as well. Oh (readers beware, Causeway plug coming up) they rely totally on *NewLeaf* for their reporting.

Sofia shown by Carlo Spinnici and Stefano Lanzavecchia

Sorry, Carlo – I really cannot resist this one ...



Carlo very sensibly dived through the history quite quickly, and got on with showing us the system itself. To summarize very briefly, *Sofia* has grown from a small back-office system in a small insurance company to the point where they now have 80% of the Italian market, and are closing fast on that last 20%. The system has grown through the DOS phase, moving to Dyalog, but (as ever) retaining huge tracts of legacy code and DOS-style interface.

The first sign of success – when Carlo noticed a job advert for an insurance administrator with "Experience of Sofia" in big letters in the middle. As he put it "To be a *Sophist* is now a job in Italy".

Stefano took us through the technical side of the way APL Italiana have exploited Dyalog's multi-threading capability to make a responsive server for their recent move to a distributed architecture.

It's a Mad Mad Mad database by John Daintree

This one needs the author to write it up himself, so please read John's notes on page 70. Let's use references to arrays of unnamed namespaces to store a vast catalogue of CD information and just treat)SAVE as our file system. It has a lot going for it – I want to see some of that code again when I am awake!

Reception and Banquet

Plenty of late-evening sunshine, plenty of Pimms, and a decent meal. And time for our first caption contest ...

Just what is John Scholes describing here? Paul Mansour obviously knows something, but I am sure our readers will have some ideas.

Day-2 - September 4

Electricity supply and quality - *e-Line* described by Tomas Gustafsson

Tomas works with Anssi Seppälä on a collection of niche-market applications for the electricity supply industry in Finland. He concentrated on two particular technical aspects of the system he showed. The first was an infinitely scrollable graph – a timeseries of power quality on a timescale from 10 years down to 1 hours, and maybe seconds/milliseconds in the future. The scrolling is driven by the mouse position – hold the button down and move a little way left or right and the chart pans at a velocity proportional to the mouse displacement. Move it a reasonable way off centre and the chart just flies by, and the key thing is that it flies by *smoothly*. Tomas gets a frame-rate of nearly 40 frames per second on a reasonable graphics card, using raw Windows calls to scroll the window and refresh only as much as is needed. Tomas comments by email:

> "I did some frame rate tests on the 2D scrolling chart, and it's indeed around 40 fps when I have 7 charts visible. Given that each chart reacts on 40 Expose events during one second, and each event is handled by almost 300 lines of APL code, we get a throughput of around 80000 lines of APL code per second. Taking conditionally excluded lines into account, there should still be around 65000 lines per second of float crunching remaining to execute (inclues all WinAPI & GDI). I find myself sitting here amazed by that fact again and again, even though this laptop is a highend one with a 3.06 GHz CPU. Minimizing the data really pays off!"

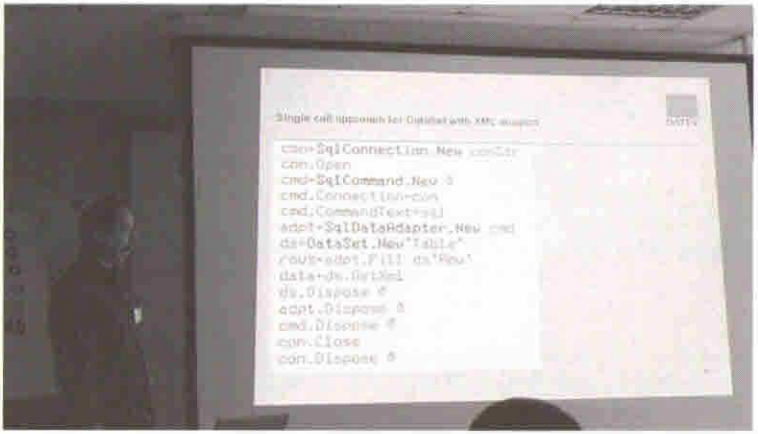
He also showed an experimental interface to a freely available 3D game engine (Revolution3D) which is a powerful DLL covering DirectX, where all the calls are available from APL (unlike many such engines) and "It's a real joy to get into its more advanced features". Again this looked really smooth in operation, rendered as nicely as OpenGL and seemed much easier to drive. Hopefully Tomas will document the code for us in detail one day.

Inquisitor by Paul Grosvenor (Optima)

Hands up every APLer who has written something like this in the past! Kimmo has one called *Tabla*, my old Rowntree mainframe stuff was called *Tabman*, and so on. This felt like the classic data enquiry and presentation tool, written for a particular client (pharmaceutical in this case) and desperately trying to become less industry-specific and more approachable by users unfamiliar with its interface quirks. I hope it makes it, but (like all the others) it has a tough challenge on its hands. Let's see if it's back next year with a nice customer-list at its back.

APL at DATEV by Dr Gunter Roche

DATEV is Germany's #1 supplier of software to tax consultants, and they have a huge APL operation. Dyalog is mostly used in research projects, and Dr Roche concentrated on some interesting discoveries with Dyalog.NET and external data.



He has been working with XML data, and with various strategies for parsing or generating XML documents from APL. He has also made some timing experiments with ADO.net, and showed how much speed-up you can get by sharing the workload between APL and a few simple fragments of C#.

The typical problem is that most of the ADO calls (such as fetching data from tables) work row by row. Basically they are designed to be run from compiled code, and reading a big table from APL is extremely simple and extremely slow. The long-term answer may be to wait for Dyalog to include native database support in the language – the short-term answer is to write a simple bridging DLL in C# which can hammer round the loop and hand back a decent block of data to APL in one call.

The timing differences were spectacular, and I hope that when DATEV get this code up to production quality, Dr Roche will be willing to share it with the rest of us (see Morten's Library proposals a little further down these notes).

Reporting Solutions with *RainPro* and *NewLeaf* by Adrian Smith

This was a pretty unstructured ramble through the possibilities of SVG (gradient fills, animations) and PDF (multi-page reports with outline trees and internal

links). Mostly designed to trigger ideas for more imaginative use of charts and reports on the web. Someone else may want to write this one up for me!

Mapped Files in Practice by Veli-Matti Jantunen

This was a bit of a disappointment, mainly due to unfortunate scheduling. Veli-Matti was (for some reason) given a full 90-minute session to himself. He correctly realised that he could not talk about mapped files for an hour and a bit, so added a bit of padding to the front (a history of Statistics Finland, and so on), followed with a description of a nice piece of in-house software call PXEdit (which I would like to know more about) and let this overrun into the time he should have left for the *really* interesting stuff about mapped files.

Pity – I really want to know about file mapping and what you can do with it. Someone should write Kdb in Dyalog, just to see if they can! Maybe Veli-Matti can make me feel a little less frustrated by sending us some notes for the next Vector?

Day-3 – September 5

APL and eXtreme Programming
APL and OOPS
Panel with Stephen Taylor, Maria Wells, Paul Mansour

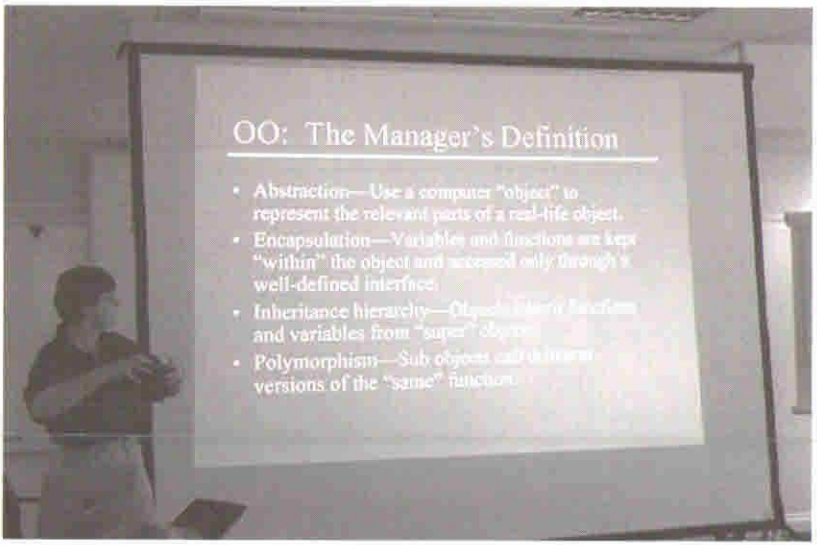
Snapped over lunch - Stephen is the one in the middle with the halo ...

This was a session where the audience listened hard, and contributed more or less continuously with relevant and supportive comments. Stephen basically summarised his XP article in Vector 19.4, with a reference to Kent Beck's excellent

"eXtreme Programming Explained" (the thinnest book that could possibly work) and a selection of well-chosen foils. I particularly liked the comment from Georges Brigham (of QualEDI) that when you ordered a sailboat you could choose any two from *Speed*, *Comfort* and *Safety*. XP recognises that you can attempt to set the *Budget*, the *Timescale* and the *Features*, but in practice you will never control more than two from the three.

Put simply, most APLers have been doing most of this stuff for years. You will recognise quite a lot of it in my Design Handbook (1982) but none of us have been doing *all* of it *all* the time. Let's take what we do well, turn all the dials up to 10, and be proud of it. This is a movement we need to be part of, so go and read the books and give it a go.

The OOPS part of the talk was just a little harder to get to grips with, partly because OOPS is full of daft words like *Polymorphism* to describe something most APLers do in their sleep.



I don't have any problem with the OOPS principles, or the excellent *Design Patterns* (Gamma, Helm *et al*) book – in fact CausewayPro is exactly the *Observer Pattern* from somewhere near the middle. I just wonder if it makes any sense to graft this stuff onto APL? If I want to write serious OOPS code, I reckon I should follow Richard and write in Java or (better) C# which are built from the little rubber feet upwards with OOPS in mind. My objects could have methods which need to do some serious array stuff – so let's do that in a pure functional array

language which I can call from C#. Now where can I get my hands on one of those (.... thinks).

Anyway, Paul Mansour's stuff is really pretty ("The code is so neat is would be a shame to bury it in the interpreter") and I want time to learn more about it. Thinking purely selfishly, what do I need to make *RainPro* behave like a class instead of a namespace? Some of it is pretty obvious:

- **Encapsulation.** Dyalog have accidentally taken a big step towards OOPS by adding autocomplete to the session. However if the developer creates an object called `mych` and types `mych.B__` I want them offered `Bar` and `Boxplot` and a few other bits, but not the entire contents normally reported by `'B' ⎕nl 3`. They already built most of this stuff to support COM export, so to make the small step of turning a *Namespace* into a *Class* can't be too hard.

- **Properties.** In C# you can say `ch.Heading = "My Chart"`. I want to be able to make a property, either out of a variable or any arbitrary pair of get/set functions or expressions. A little harder, maybe but not impossible. Then I can do the same in APL, and my RainPro users can benefit from Autocomplete just like my C# users.

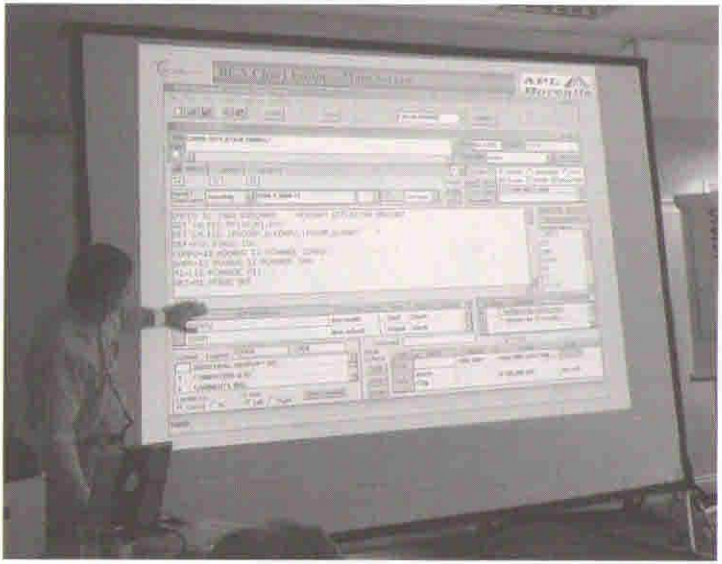
- **Inheritance.** OK, you can kinda fake this with `⎕PATH`, but it is far too dangerous and has too many funny side effects. A namespace needs a clear list of other namespaces from which to inherit behaviour. This should apply to both methods and properties, so that when you do `ns1.foo` and the interpreter trips VALUE ERROR you are completely in control of where it looks. Another crucial difference from `⎕PATH` is that it must execute the inherited code in the original object, not in the namespace where it managed to turn it up. In fact if you)cs into an object, I reckon that)fns should show the full inherited function set from the classes in the inheritance tree, even though none of them are 'really' there at all. Hmmm etc. Are they editable, or just shadows?

I am also becoming increasingly aware of the importance of the *New* keyword in C# and other such languages. I don't think assignment does enough for us here – perhaps we will need to bring in another symbol like the APLX {QuadArrow} to signify the instancing of an *object* from a *namespace*, or was that a *namespace* from a *class*, I hear you ask?

All of which is probably controversial, so let's have a good barney about it before Dyalog do something really screwy in the attempt to fix this. Or maybe we just work with Paul's stuff for a year or two to get a feel for how it works with APL's oddball scoping rules and other snags we don't know about yet.

The BCA Chart Editor by Richard Proctor

Long ago and far away, I met a materials management system called Logol, written by Bob Brown. It was DOS-based, and an APL development interpreter shipped with every product. The user-interface was simple command line, with some prompted stuff here and there, designed to facilitate 'enter-bashing' by the experienced user (which most users were).

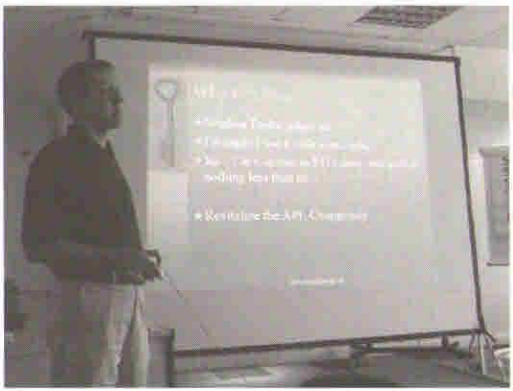


Richard has been attempting to take a workspace full of analytical routines and charting commands (based on the excellent IPSharp 'SuperPlot' functions) and build a Gui to replace what the experienced users currently do by typing at the session. As the slide shows, this is a very tough call, and Richard is well aware that not all his users are convinced yet. I have the same problem with GraPL (desktop) – is it really that much easier to build charts in a Gui environment than in a decent code-editor which autocompletes for me?

I think my answer is 'yes' – just. I now prefer to work in GraPL, rather than at the 6-space prompt in the RainPro workspace. If I had just a little more control over autocomplete (see previous notes) I might well swing back. For expert users, Notepad is a pretty good development environment for almost anything.

The BAA Utility Library by Morten Kromberg

Again, most of Morten's excellent talk was outlined in the last Vector; what was new was a firm emphasis on collaboration, and specifically online collaboration:



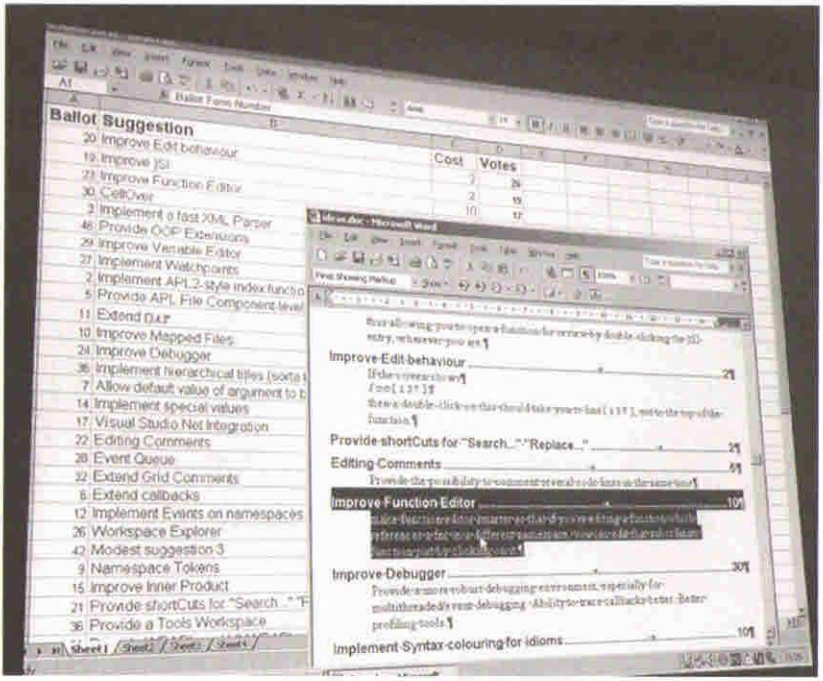
I think that what emerged from the discussion was that to be effective, a library needs to grow organically, and not simply be a bunch of useful code assembled by one author, however well-intentioned.

The internet gives us a chance, and we can start by continuing these discussions on the Vector wiki (accessed from the Vector front page) and the Yahoo discussion group set up by Stephen Taylor. It is in everyone's interest to move this along – the first thing any project manager will check when contemplating hiring an APLer or an APL team is "what is the online support like?". Let's make sure there is something out there for him to find.

Enhancements and Future Plans by the Dyalog team

Time for another caption contest (sorry John) ...

This was where the votes were revealed, and the top few enhancement requests chewed over and spat out by the developers.



The top two were really low-cost (improving `)SI` simply involves adding `#.` to the front of what it currently shows) and the change to the function editor would be voted for by anyone with a *WSLoc* utility which searches out calling lines.

The highlighted entry needs explanation – if you rely on `⎕PATH` to locate your utility set, then it is frustrating in the extreme that you can't edit a utility function with a simple double-click in the caller. Thinking back, this was one of the reasons I decided to avoid `⎕PATH` altogether in my code. Of course this is non-simple as the current path could be localised, so where to go may depend on the calling stack.

The request for a *CellOver* event on the grid was another low-cost improvement, but the request for a fast XML parser rapidly evolved into a vigorous discussion, which clearly means someone needs to do some experiments. OOP extensions were the same only more so. *Watchpoints* got another strong discussion, as did component-level locking. Adding the APL2 `⌷` indexing was readily accepted by everyone, which is about as far as we got!

The *Take Care* System

additional notes from Peter Pichler (peter.pichler@profdoc.se)

Take Care, an infrastructure application for patient-related information, written in APL, is the largest health-care system in operation in Sweden. In the summer of 2003 there are

- 700,000 patient record files accessed from
- 4,000 clients all over Stockholm at
- 25 server calls per second from
- 1,700 concurrent users on open socket connections;
- 8,500 registered users have online access to the patient records stored in a central database all based on APL component files.

The server is a mirrored UNIX machine running 30 workspaces on 30 different ports, listening for incoming traffic, plus some extra workspaces for communication with other systems.

1993

Let's start from the beginning, ten years ago. By coincidence I met a nurse who introduced me to the world of hospitals. By that time, computers were common in primary care, but there were few in hospitals, for many reasons. The systems there were lab systems and cashiers' financial records.

So we sat down and asked ourself "what would it take to make an administration journal system for the whole of Sweden (9 million people)?" We were allowed to visit the journal archive at a medium-sized hospital in mid-Sweden, and could study the paper files there. Some were small and some were huge.

As a result of that study, we found that APL component files were the perfect data storage. It felt almost as if component files were made for patient records. So one file per patient was a natural thing to select. In Sweden each person has a unique personal number that's used all his life. A small formula translates that personal number into a file name and we had easy access to the files.

Having seen the lack of support systems for hospital care, this was a challenge and at the same time a real opportunity: to be able to build a system, starting with CLEAR WS.

All data was to be stored in the central database. The data recorded in the files would never be erased or changed. We studied the laws, and soon learned that no recorded information may be changed or updated. Any changes had to be traceable. So we simply stored any new version of the same document in what we call a new instance. That is a three-dimensional structure, where each version of the stored document carries information about who, where, when, etc.

Simplicity

Not even capable of programming my video, I knew at that time the importance of simplicity. The user interface must be clear, easy to understand and intuitive. Doctors and nurses do not work with computers, so a fast access to patient information must be very intuitive. The need for training must be minimal.

Everything we did focused on finding the important parts of a specific problem, and then making that as simple as possible. The benefits of this approach showed later on: users liked the system. As a very important effect, we also got very fast response times, almost everything subsecond. And that is true even when there are 1,700 concurrent users on the system.

Terminology database

Having a background in statistics, I also knew the need for "speaking the same language"; so one of the first things we did was to add a terminology database to the system. Definition of terminology used is available online.

We continued to design the server parts as well. The system was to become a central system, with access to all data from everywhere within the local network. In Stockholm there was already a large network in place. No medical information is stored on the clients.

1995

After a year or so we were looking for the implementation guide for EDIFACT. And by coincidence, there was Svante Lewald at the other end of the telephone, wondering what we were doing. So we told him of this fantastic new system, how easy it would be to add new functionality and so on.

He was at the time responsible for the clinical chemistry laboratory at Huddinge Hospital.

> "We are ready to receive electronic lab orders from journal systems, but so far no vendor can do that."

Our reaction was, "Great, let's do that!"

So we started designing the form for ordering lab tests. There was a well-established paper based system in place, so we had to make Take Care better than that. No one was going to make doctors use Take Care instead of paper.

Our first test started at the lung clinic, and after a few weeks we found doctors ordering lab tests happily with Take Care. This inspired us to develop further.

So far we had no contracts whatsoever. All our work had been done just to prove that it was possible to develop. In 1996 we got our first contract with Huddinge Hospital.

The lab order to the clinical chemical lab was very successful: electronic orders went straight into the lab system, and blood tests could start earlier, because no more registration was needed: just read the barcodes of the already-registered order. Test results were recorded in Take Care and stored in the journal files. The doctors had easy access to the complete history of test results for a patient.

The next laboratory to join Take Care was clinical bacteriology. The order form was programmed to have some intelligent guidelines making the quality of the orders better.

Version handling

We continued to add functionality, documenting journal texts, all based on the terminology database we had included at an early stage. The work defining terms really took off.

We also knew that we would have a lot of users running the system in the future, so the system updates had to be stable and automatic. Code tables and workspaces and namespaces are distributed automatically from the server to the

Windows PCs as required. This means any new release is mostly a matter of replacing a handful files on the server. (So far we have had 408 releases since December 1996. Every one of the today's approx 4,000 clients runs the same version of the system.)

Error control

We also built in automatic error control. If and when an error occurs on the client PC, we can catch it with `⎕TRAP` and send that on to the server. So we see every domain error, etc., that occurs in the system.

Shipping new releases frequently makes things easy for us this way too. Should we introduce errors with the new code, it will take a fraction of a second before we can see it in the error log. (And there will be a lot of them too.) This has happened a few times, and we could act long before users reported the error. If no new errors pop up in the first hour or so, we can happily go on holiday.

Y2000

Then the year 2000 was approaching. Some of the older systems used for cashiers and finance were not going to work after 2000. In November 1998 we were asked if we could develop a new cashier, and economic and diagnosis reports. There was very little time, and we had no idea what we were going into, but we said Yes and had a seven months extreme period, calling in two more APLers to help us.

We had an intensive period with users around us all the time so we could construct what was needed as we went. In March 1999 we delivered the first parts, and the whole hospital managed to train their personnel and everything worked well as 2000 arrived.

Onsite customers

After that we continued to add functions in close cooperation with the users: meeting them once a week, discussing solutions and implementing them once we understood what we were supposed to do.

Nobody knows, the customer at least of all. Mostly they have only an idea—a good start. The first screens are taken back to the users, and we continue from there, testing, and detecting missing and unnecessary things.

While we work, we always focus on simplicity, removing every button or field we possibly can. Then comes a point when we look at the newly developed

functionality and wonder "why did we not find this solution from the start—is this all?" And when that happens, we know we have made it.

Here are some parts of the system:

- Activity Organiser
- Prescriptions
- Intelligent Remittals
- 10 Laboratories
- Patient Registration (e.g. for operations)
- Consultation Remittals (to and from the hospital)
- Resource Planning
- Scanning
- Images import
- Emergency Department, where they all use digital dictation: a Philips Speech mike integrated with Take Care

The Future

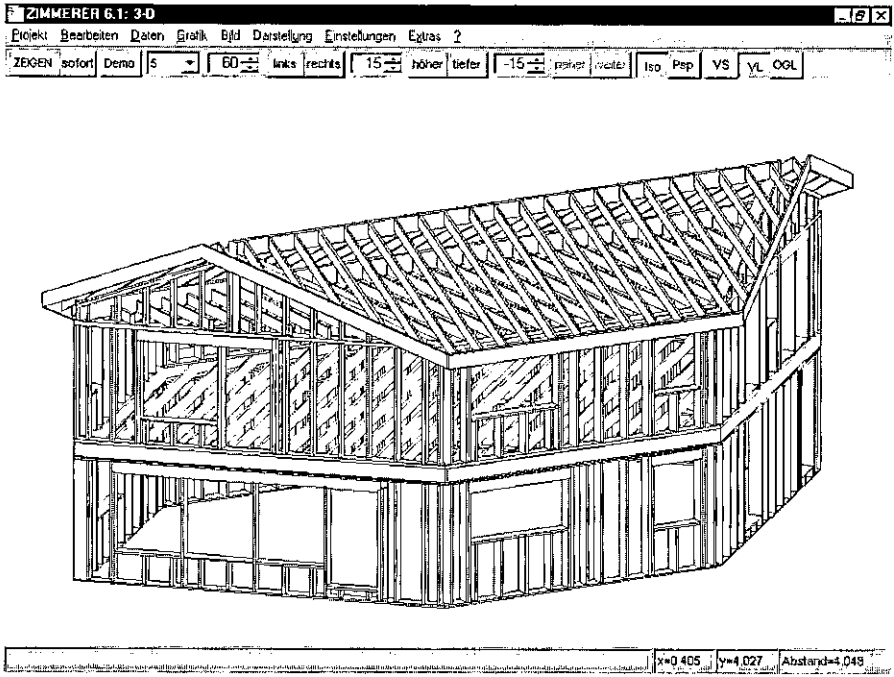
Today more functionality is being added, and new customers are invited. Having joined Profdoc, Scandinavia's largest vendor in the health-care arena, we find our future looks very exciting. Now we have access to well-defined data that just await analysis. An hypothesis test function is waiting to become reality and, who knows, we might be able to help the clever people out there make new medical discoveries.

The development continues...

ZIMMERER – Roof Design with APL

presented by Michael Zippel (www.zimmerer.de)

ZIMMERER is tailored for craftsmen (they are called "Zimmermann" or "Zimmerer" in German) building timber frame houses and roofs. The first version was sold in 1988: MicroAPL-based running on Atari Computers (at 8 Mhz!). A Windows Version (DyalogAPL) was introduced in 1993.



ZIMMERER attempts to satisfy the software needs of a typical small business, all integrated in one piece of software.

Tender

- rapid entry of "in-about" construction
- lists of all items required for bidding
- formal printouts

Construction

- Starting with macros
- "visual" elaboration of details in CAD-style:

  Each individual beam can be modified

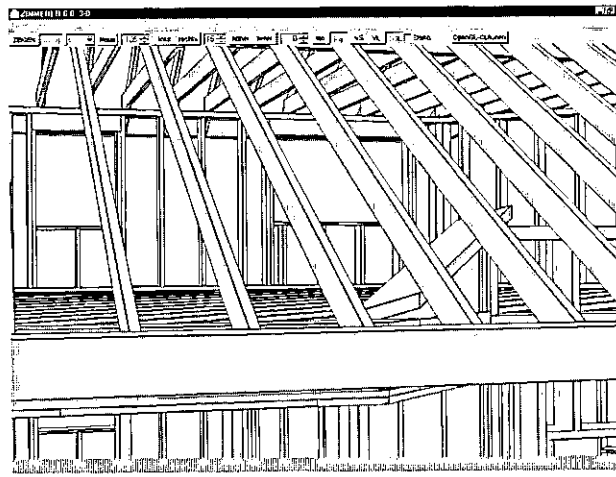
  Groups of beams can be modified in a uniform way

  Openings (e.g. for chimneys or windows) can be set in one pass

  Connections between different types of beams and/or different parts of the roof (Lap Joints, Tenon, Mortise, Drillings) can be set

  Colliding beams can be fitted

  The overall Contour can be modified and and beams fit into it
- Produces 10 different technical plans with all required measurements
- 3D-graphics (choice of vector graphics with hidden line detection - ideal for high res printing; or pixel graphics using openGL - ideal for visualisation



- Allows interfacing with other systems, including various brands of CNC-Sawing Machinery (via DXF and custom formats)

Billing

- provides lists needed for billing: lumber cut and other material
- standard price tables and invoice writing (using RichEdit with RTF-templates)

ZIMMERER has a live-update feature included, to download a patch-workspace.

Using Namespaces to Manage a CD Database

by John Daintree
Senior Programmer, Dyalog Ltd (johnd@dyalog.com)

Abstract

Any collector of music (or for that matter of anything) will recognise the need for a mechanism for keeping track of the items in a collection. Namespaces are an obvious choice for representing uniform, structured data, such as a collection of music CDs. This paper describes software that make extensive use of namespaces, namespace refs, and namespace reference arrays. A small number of functions provide simple but extensive query facilities which allow convenient browsing of the catalogue.

Overview

For some time I have been aware of the need to have a complete and up-to-date list of the CDs in my modest collection (1565 discs to date). There are two main reasons for this database. Firstly, in the event of theft or the house burning down and the entire collection needing to be replaced. Secondly, I'm not the most astute shopper and have, on more occasions than I would like to admit, purchased a CD that I already own.

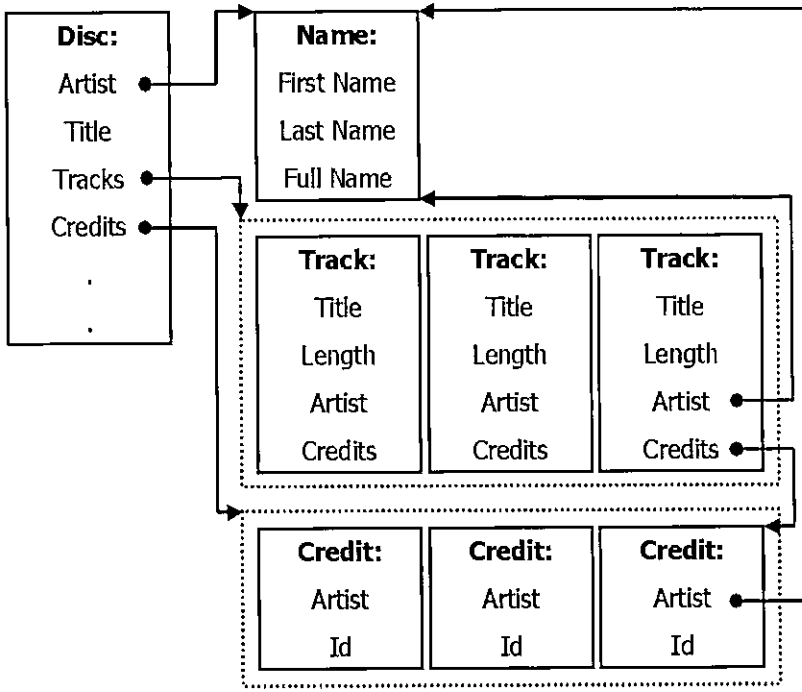
It would be madness to attempt to manually enter all the information on 1565 CDs. Fortunately there are a number of on-line resources which can be used to retrieve catalogue information for CDs, be they currently available, deleted or even bootlegs.

One such resource is the CDDB database from Gracenote Corp. (www.gracenote.com). Gracenote provide an ActiveX Control that can be incorporated into any application. This control manages all the CD drive and TCP/IP internet access that is required to retrieve information about a CD. In practice, one can insert a CD into the drive on the computer, make the appropriate calls to the control and have artist, track, musician information, recording date and location etc. retrieved automatically from the online database. The Gracenote database currently contains extensive information for over 2 million CDs.

Having identified our information source and our retrieval mechanism it becomes necessary to design the layout of our local database.

Data Structure

One can consider the catalogue information of a CD as the following diagram:



Each *Disc* has been recorded by a specific *Artist*. The artist may have a first name, last name, and a full name. Separating the artist's name into first and last parts allows for efficient alphabetic sorting. In the case of a group, e.g. "The Tubes", the first and/or last name fields may not be used.

A *Disc*, of course, also has a *Title*, a list of *Tracks* and potentially a number of *Credits*, i.e. a list of which musicians performed which role in the recording of the disc.

A *Track* on a *Disc* has a *Title*, a *Length* (its duration), and a further list of *Credits*, which may override those *Credits* on the main disc. Given the existence of

compilation discs, where each track may be recorded by a different artist, it is necessary for each track to refer to an *Artist*. In non-compilation discs this refers to the same information as the *Artist* information of the *Disc*.

A *Credit* is simply an association of an artist with a particular role, for example Bass Guitarist, Composer, or Recording Location.

Given the uniformity of the data, every *Disc* has the same fields, every track has the same fields etc, it is simple to think of a *Disc* as a namespace, and the *Tracks* on a discs as an vector of namespaces. A CD collection therefore, is a vector of *Discs*.

Notes on Implementation

There are a number of points that I think are worth making:

1) I intentionally implemented this software with no GUI. The nature of the queries and the results returned is such that the Dyalog APL session is a fine input and output device. Especially given a large ⎕PW and the session's horizontal scrollbar which is available in version 10.

2) I have used dynamic functions as much as possible. This was primarily to extend my own understanding of programming in "D".

3) I typically run with ⎕IO←0 and ⎕ML←0, so ⊃ is "first" and ↑ is "mix".

4) As we are retrieving all our data from an external database any errors or omissions in the database will show up in our data. For example, misspelled words, incomplete listings, missing credits and so on.

Implementation

The cddb workspace contains a vector named Discs. This is (to date) a 1565 element vector.

```
      )LOAD cddb
      ⍴Discs
1565
```

Let's look at the first few elements of Discs

```
      four←4↑Discs
      four
#.Disc  #.Disc  #.Disc  #.Disc
```

Each namespace in the `Discs` vector displays as `#.Disc`. The display form of a namespace is obtained from the name used when the namespace was created

```
      'x' ⎕ns ''
      x
#.x
```

if we take a *reference*, let's call it `ref`, to this namespace, the display form of the ref is identical (in fact the interpreter cannot distinguish between the reference and the original name, `x`).

```
      ref←x
      ref
#.x
```

if we discard `x`, the reference, `ref`, is sufficient to keep the namespace "alive" in the workspace, but it remembers its display form.

```
      ⎕ex 'x'
      ref
#.x
```

We can use this to create namespaces with a particular display form:

```
      new←{
           α←#
           α.{
                r←⍵ ⎕NS''
                ns←⍎⍵
                ex←⎕EX ⍵
                ns
           }⍵
       }
```

```
      new 'Disc'
#.Disc
      new ¨4⍴⊂'Disc'
#.Disc  #.Disc  #.Disc  #.Disc
```

As we will see later it is convenient to specify that the parent of each *Track* on a *Disc* is the *Disc* itself. The new function above, allows us to specify that our namespace should be created as a child of any namespace provided in the left argument.

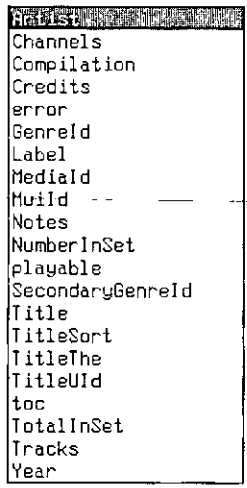
```
discs←new ¨4⍴⊂'Disc'
      discs new ¨⊂'Track'
#.Disc.Track  #.Disc.Track  #.Disc.Track  #.Disc.Track
```

And the Artist for each of the discs

```
      discs.Artist←new ¨4⍴⊂'Name'
      discs.Artist
#.Name  #.Name  #.Name  #.Name
```

For each disc that is added to the collection we create a new Disc namespace. This namespace is then populated with data obtained from the Gracenote Corp. ActiveX control.

```
Discs.
```



The AutoComplete feature of version 10 is especially useful when programming with namespaces, and namespace arrays. For example, we can see the contents of each Disc by typing "Discs." (assuming AutoComplete is set up appropriately) into the session:

So, let's examine a specific disc, the first one in the collection.

```
     (⊃Discs).Artist
#.Name
     (⊃Discs).Artist.FullName
Donald Fagan
      (⊃Discs).Tracks
#.Disc.Track  #.Disc.Track  #.Disc.Track  #.Disc.Track
#.Disc.Track  #.Disc.Track  #.Disc.Track  #.Disc.Track
     ⍴(⊃Discs).Tracks
8
     (⊃Discs).Title
The Nightfly
```

Dyalog APL supports "reference arrays" i.e. arrays of references to namespaces We can then use intuitive '.' notation to access names from each element within an array of namespace references, so

```
     ⍴(⊃Discs).Tracks
8
     ↑(⊃Discs).Tracks.Title
I.G.Y
Green Flower Street
Ruby Baby
Maxine
New Frontier
The Nightfly
The Goodbye Look
Walk Between The Raindrops
```

And of course, this applies to elements in the `Discs` vector too.

```
      (2↑Discs).Title
The Nightfly  Images And Words
      (2↑Discs).Artist.FullName
Donald Fagen  Dream Theater
      (2↑Discs).(Artist.FullName,'\',Title)
Donald Fagen\The Nightfly  Dream Theater\Images And Words
```

Reference arrays are very powerful, not only can we access variables from each element in an array of namespace references but we can also execute functions on an array of namespace references. E.g.

```
      r←Discs.⍎⍵⊂'Artist.FullName≡''Lisa Loeb'''
```

r is a Boolean vector containing a 1 for each disc recorded by Lisa Loeb and a 0 elsewhere. This leads us to our first simple query function, and also to our first problem.

```
where←{
      16::'no match'
      ⍺/⍨⍺.⍎⍵⊂⍵
 }
```

Should the boolean expression ⍺.⍎⍵⊂⍵ result in an array containing only zeroes the interpreter will generate a NONCE ERROR. Currently the interpreter cannot create an "prototypical" namespace, any attempt to do so will generate NONCE ERROR. The `where` function traps this error and returns `'no match'`. All of the functions illustrated here are intended to be used from the session, indeed there is no GUI in the workspace. For this reason a `'no match'` return is acceptable.

```
      Discs where 'Artist.FullName≡''nonexistant band'''
no match

      Discs where 'Artist.FullName≡''Lisa Loeb'''
#.Disc  #.Disc  #.Disc  #.Disc
```

So there are 4 discs in the collection recorded by Lisa Loeb. This is such a useful query that it warrants a function all of its own, which we can rewrite without the use of execute.

```
      by←{
            msk←⍺.{Artist.FullName≡⍵}⊂⍵
            msk/⍺
       }

      Discs by 'Lisa Loeb'
#.Disc  #.Disc  #.Disc  #.Disc
```

```
      ↑(Discs by 'Lisa Loeb').Title
Tails
Firecracker
Truthfully
Cake And Pie
```

The extra parentheses in the above statement are a little unwieldy, however the addition of a small function of provides great improvement.

```
of←{
      ⍵.⊃⊂⍺
}

      ↑'Title' of Discs by 'Lisa Loeb'
Tails
Firecracker
Truthfully
Cake And Pie
```

It should by now be apparent that if we design our data layout well we can make powerful queries with really tiny APL expressions.

It is at this point that I realized what most APL programmers will have known for years, APL allows us to write program statements that look like English language phrases.

A useful extension of the by function would allow partial matches of names. This is achieved with the operator find.

```
find←{16::'no match'
      ⍺←Discs
      ''≢0⍴⍵:⊃,/(⊂⍺)∇¨⍵
      n←⊂toupper ⍵↓⍨exact←'|'=⊃⍵
      exact:⍺/⍨n≡∘toupper¨⍺⍺ ⍺
      ⍺/⍨1⍷¨↓(⊃n)⍷∘toupper↑⍺⍺ ⍺
 }
```

find attempts to locate (in a case-independent match) the right argument in the data returned by the left function operand, so our by function becomes:

```
by←{⍵.Artist.FullName} find
```

so

```
     ⍴Discs by 'Lisa Loeb'
7
     ↑'Artist.FullName' of Discs by 'Lisa Loeb'
Lisa Loeb & Nine Stories
Lisa Loeb
Lisa Loeb
Lisa Loeb
Lisa Loeb & Nine Stories
Lisa Loeb & Nine Stories
Lisa Loeb
```

Or even,

```
     ↑'Artist.FullName Title' of Discs by 'Lisa Loeb'
Lisa Loeb & Nine Stories  Waiting for Wednesday
Lisa Loeb                 Tails
Lisa Loeb                 Firecracker
Lisa Loeb                 Truthfully
Lisa Loeb & Nine Stories  Do You Sleep?
Lisa Loeb & Nine Stories  Stay (I Missed You)
Lisa Loeb                 Cake And Pie
```

So in many cases the more flexible find function is more useful.

You may have noticed the use of '|' in `find` to force a exact match to the argument.

```
     ⍴Discs by 'Lisa Loeb'
7
     ⍴Discs by '|Lisa Loeb'
4
```

This is the only "special character" used by the `find` operator. In an attempt to maintain the use of English phrases in the search strings we have a cover function to do these exact matches.

```
exactly←{'|',⍵}

     ⍴Discs by exactly 'Lisa Loeb'
4
```

The find operator is pretty generic, we can slot in alternate function operands to search in other fields of the Disc, for example

```
     called←{⍵.Title} find

     ⍴Discs called 'Greatest Hits'
18

↑'Artist.FullName Title' of 5↑Discs called 'Greatest Hits'
Styx                       Greatest Hits Part 2
Queen                      Greatest Hits I
Queen                      Queen Greatest Hits II
Richard Marx               Richard Marx - Greatest Hits
Styx                       Styx - Greatest Hits
```

Also find can accept a nested argument

```
     ⍴Discs called 'Best Of'
37
     ⍴Discs called 'Best Of' 'Greatest Hits'
55
```

Again, to maintain the use of "English Phrases" throughout we can have an or function

```
     or←{(⊂⍺),{''≡0⍴⍵:⊂⍵ ⋄ ⍵}⍵}

     ⍴Discs called 'Best Of' or 'Greatest Hits'
55
     ⍴Discs by 'Toto' or 'Styx' or 'Journey'
106
```

So far, we have pretty much been interested only in searching through the vector of Discs. Let's look at retrieving information from individual tracks.

Each Disc has a vector of Tracks.

```
     ⍴¨(2↑Discs).Tracks
 8 8
```

```
    t¨(2↑Discs).Tracks.Title
I.G.Y                        Pull Me Under
Green Flower Street          Another Day
Ruby Baby                    Take The Time
Maxine                       Surrounded
New Frontier                 Metropolis, Pt. 1
The Nightfly                 Under A Glass Moon
The Goodbye Look             Wait For Sleep
Walk Between The Raindrops   Learning To Live

    t¨(2↑Discs by 'Lisa Loeb').Tracks.Title
Waiting for Wednesday      It's Over
Stay (live)                Snow Day
Alone (live)               Taffy
Hurricane (live)           When All The Stars Were Falling
Garden of Delights (live)  Do You Sleep?
Taffy (live)               Hurricane
                           Rose-Colored Times
                           Sandalwood
                           Alone
                           Waiting For Wednesday
                           Lisa Listen
                           Garden Of Delights
                           Stay
```

Remember our of function?

```
    of←{
        ⍵.⍷⊂⍺
    }
```

The problem with this function is that if ⍵ is a nested vector we will get a LENGTH ERROR, as the enclosed ⍺ is distributed across ⍵:

```
    'Title' of 'Tracks' of 2↑Discs by 'Lisa Loeb'
LENGTH ERROR
of[1] ⍵.⍷⊂⍺
```

We need a more generic of function:

```
of←{
     ⍵.⍺{
          0=⍵:⍺
          (⊂⍺)∇ ⍵-1
     }⍣⍵
}
```

This of makes sure that the argument ('Title' or 'Tracks') is enclosed deep enough to be distributed across the array of namespaces.

```
      ↑¨'Title' of 'Tracks' of 2↑Discs by 'Lisa Loeb'
```

```
Waiting for Wednesday       It's Over
Stay (live)                 Snow Day
Alone (live)                Taffy
Hurricane (live)            When All The Stars Were Falling
Garden of Delights (live)   Do You Sleep?
Taffy (live)                Hurricane
                            Rose-Colored Times
                            Sandalwood
                            Alone
                            Waiting For Wednesday
                            Lisa Listen
                            Garden Of Delights
                            Stay
```

Which (almost) gives us our "English Phrases" back. Even better can be achieved with a little function assignment

```
      on←of
      r←'Title' of 'Tracks' on Discs by 'Lisa Loeb'
```

A useful function is tracks

```
      ∇ r←tracks
[1]     r←⊃,/Discs.Tracks
      ∇
```

`tracks` is one of the few non-dfn functions in the workspace. It merely returns a vector of all the tracks in the database. It is niladic, which prevents it being implemented as a dfn.

It is worth noting that ⊃,/ is a recognised idiom in version 10 of Dyalog APL. As such, `tracks` runs several thousand times faster than it does on version 9.

```
      ⍴⍤⊃Discs tracks

1565 16246

      ⍴tracks called 'Animal'

10
```

Because each Track has an Artist field, which refers to the appropriate recording artist we can do

```
      ↑'Artist.FullName' of tracks called 'Animal'
Glenn Walters
Toto
Planet X
Steve Vai
Toto
Toto LIVE & ALIVE
Def Leppard
Gowan
Gowan
Carmen Grillo
```

We can use `tracks` to get some fairly scary statistics

```
      ⍴tracks by 'Styx'
389
      ⍴tracks by 'Celine Dion'
64
      longest←⊃tracks[⍒tracks.Length]

↑'Artist.FullName  Title (⍕Length)' of longest
Robbie Williams
These Dreams
1884.306667
```

So the longest track in the database is 1884 seconds (about 35 minutes) long, is called "These Dreams" and is by Robbie Williams.

Now, those of you with any knowledge of Mr Williams' body of work will be aware that he's never recorded a 35 minute track in his life. Chances are that the last track on this album ends with 25 minutes of silence and then a "hidden" song at the end. Lets look at the next longest track.

```
      longest←2⊃tracks[⍒tracks.Length]

      ↑'Artist.FullName Title (⍕Length)' of longest
Transatlantic
All of the Above
1859.226667
```

About 31 minutes. Transatlantic is a progressive rock supergroup (comprised of members of Spock's Beard, Dream Theater, Marillion and The Flower Kings). Knowing these boys, it's quite likely that they have a track that is 31 minutes long.

I mentioned earlier that a useful design point is to have the parent of a *Track* be the *Disc* on which it is recorded. This allows us to use '##' to retrieve a *Disc* from a *Track*.

```
      animals←tracks called 'Animal'

      ↑'Artist.FullName Title' of '##' of animals
Glenn Walters       The King Of Retro Cool
Toto                Past To Present 1977-1990
Planet X            Live From Oz
Steve Vai           Passion and Warfare
Toto                Hold Me Back (Live)
Toto                Toto Live & Alive
Def Leppard         Vault: Def Leppard's Greatest Hits
Gowan               Strange Animal
Gowan               No Kilt Tonight (Solo Live)
Carmen Grillo       Both Sides Of The Coin
```

Every *Disc* and *Track* has a number of *Credits* associated with it. Each *Credit* includes information such as who played bass guitar on the track, or who composed a track, or where a track was recorded and so on.

I was really taken by this idea of using English phrases for the queries, so I wanted to be able to write expressions like

```
Discs where 'Steve Lukather' played 'Guitar', or
Discs where 'Bob Dylan' played
```

The function `where` takes a character vector which is executed in each element of its left argument. The problem then was to write the function `played` so that it returns an appropriate argument for `where`. This proved to be difficult, so I compromised and wrote a function `_played` which does all the work, and `played` that generates a simple a character vector that invokes `_played`

```
      played←{
          qte←{' ''',⍵,''' '}
          ths←'(⎕ns '''').## '
          ths,'#._played',(qte ⍺),(qte ⍵)
      }

      'Lukather'played'Guitar'
(⎕ns '').## #._played 'Lukather' 'Guitar'
```

Note the use of `(⎕ns '').##` to return a reference to the current space.

```
      _played←{
          0::0
          creditsof←{
              ⊃,/⍵.(Credits,⊃,/Tracks.Credits)
          }
          name role←⍵
          c←(creditsof ⍺)by name
          r←(∨/¨(⊂toupper role)∘.⍷toupper¨Roles)/Roles
          ⊃c.Ider
      }
```

```
     lukather←Discs where 'Lukather' played 'guitar'
     plukather
12
     ↑'Artist.FullName Title' of Lukather
Richard Marx                          Rush Street
Toto                                  Toto IV
Larry Carlton and Steve Lukather      No Substitutions
Toto                                  Tambu
Wilson Phillips                       Wilson Phillips
Joseph Williams                       3
Toto                                  Toto
Various Artists                       Stone Cold Queen
Dune                                  Dune
Asia                                  Then & Now
Lara Fabian                           Adagio
Toto                                  Through The Looking Glass
```

Because `played` is a function with an argument the statement

```
Discs where 'Bob Dylan' played
```

is syntactically incorrect (`played` is called with no argument). However because of how `_played` is written we can do

```
anything←''
```

and

```
Discs where 'Bob Dylan' played anything
```

which is OK.

```
     ρDiscs where 'Bob Dylan' played anything
2
```

OK, so I'm not a big Bob Dylan fan.

An extremely useful addition to the suite of functions is "display":

```
      display ⊃Discs
```

```
Donald Fagen (1982)
The Nightfly

1. I.G.Y                          6:03
2. Green Flower Street            3:42
3. Ruby Baby                      5:39
4. Maxine                         3:49
5. New Frontier                   6:21
6. The Nightfly                   5:47
7. The Goodbye Look               4:50
8. Walk Between The Raindrops     2:38
```

The `display` function can provide various levels of information about a disc, indicated by its left argument. For example, passing a 3 as the argument to `display` lists all the credits for the disc, and for each of its tracks. So displaying the result of the "Dylan Query" shows that Bob has a writing credit for the last track on the disc.

```
      3 display  ⊃Discs where 'Bob Dylan' played anything
```

```
Billy Joel (1987)
КОНЦЕРТ (Live in Leningrad)
Columbia/ 7464-40996-2
Billy Joel

Soft Rock

Guitar-Electric      : Billy Joel
Keyboards-Various    : Billy Joel
Piano                : Billy Joel
Vocals               : Billy Joel
Engineer             : Bradshaw Leigh
Mixing               : Bradshaw Leigh
Mixing               : Brian Ruggles
Producer             : Brian Ruggles
Keyboards-Various    : Dave Lebolt
Bass Guitar          : Doug Stegmeyer
Recording Location   : Fleetwood Mobile Recording Unit
Vocals-Backing       : George Simms
Engineer             : Jim Boyer
Mixing               : Jim Boyer
Producer             : Jim Boyer
Guitar-Electric      : Kevin Dukes
Drums and Percussion : Liberty DeVitto
Saxophone            : Mark Rivera
Vocals-Backing       : Peter Hewlett
Acoustic Guitar      : Russell Javors
Guitar-Electric      : Russell Javors
Mastering Location   : Sterling Sound, New York
Mastering            : Ted Jensen
Mixing Location      : The Hit Factory, New York City
Assistant Engineer   : Tim Leitner
```

1.	Odoya	1:17
	Vocals-Backing : David Abezadze	
	Vocals-Backing : Georgi Dylidze	
	Vocals-Backing : Giya Chirakadze	
	Vocals-Backing : Nugree Gamtjebeli	
	Vocals-Backing : Timur Chkuatzeli	
	Vocals-Backing : Tomaz Eliava	
	Vocals-Backing : Yosif Zurabauli	
	Vocals-Backing : Zurab Loladze	
	Songwriter (Music & Lyrics) : <Traditional>	
2.	Angry Young Man	5:24
	Songwriter (Music & Lyrics) : Billy Joel	
3.	Honesty	3:58
	Songwriter (Music & Lyrics) : Billy Joel	
4.	Goodnight Saigon	7:20
	Songwriter (Music & Lyrics) : Billy Joel	
5.	Stiletto	5:08
	Songwriter (Music & Lyrics) : Billy Joel	
6.	Big Man On Mulberry Street	7:17
	Songwriter (Music & Lyrics) : Billy Joel	
7.	Baby Grand	6:09
	Songwriter (Music & Lyrics) : Billy Joel	
8.	An Innocent Man	6:08
	Songwriter (Music & Lyrics) : Billy Joel	
9.	Allentown	4:22
	Songwriter (Music & Lyrics) : Billy Joel	
10.	A Matter Of Trust	5:08
	Songwriter (Music & Lyrics) : Billy Joel	
11.	Only The Good Die Young	3:32
	Songwriter (Music & Lyrics) : Billy Joel	
12.	Sometimes a Fantasy	3:38
	Songwriter (Music & Lyrics) : Billy Joel	
13.	Uptown Girl	3:08
	Songwriter (Music & Lyrics) : Billy Joel	
14.	Big Shot	4:45
	Songwriter (Music & Lyrics) : Billy Joel	
15.	Back In The U.S.S.R.	2:44
	Songwriter (Music & Lyrics) : John Lennon	
	Songwriter (Music & Lyrics) : Paul McCartney	
16.	The Times They Are A Changin'	2:57
	Songwriter (Music & Lyrics) : Bob Dylan	

And Finally...

It works for DVDs too.

```
      ρDVDs
321
```

Oh, and since I started writing this

```
      ρDiscs
1575
```

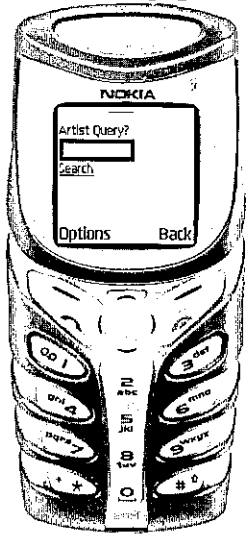
And Finally Finally...

One of the objectives of the exercise was to prevent the accidental purchase of duplicate copies of the same CD. This requires some sort of remote access to the database. The following piece of APLScript implements a Mobile Web Page (i.e. one that can be accessed by mobile phones). When on one of my many trips to the local record shop I can 'phone the server before I buy anything to see if I already have it.

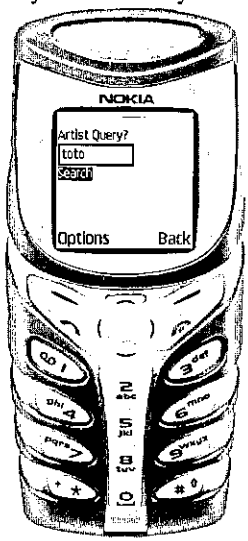
```
<%@ Page Inherits="System.Web.UI.MobileControls.MobilePage"
Language="apl" %>
<%@ Register TagPrefix="Mobile"
Namespace="System.Web.UI.MobileControls"
assembly="System.Web.Mobile"%>

<script language="apl" runat=server>
∇ListTracks args
disc←⊃##.{Discs called ⍵} (⊃⍙args).ListItem.Text
tracks←disc.Tracks
TrackList.Items.Add¨tracks.Title
TrackList.Items.Add¨disc.Notes,##.display disc.Credits
ActiveForm←ShowTracks
∇

∇ListDiscs  args
discs←##.{Discs by ⍵} (⊃⍙args).ListItem.Text
DiscList.Items.Add ¨discs.Title
ActiveForm←ShowDiscs
∇
```

```
∇ListArtists args
→(0≠⎕nc '##.Discs')/⎕lc+1◇##.⎕cy 'c:\cddb\cddb'
discs←##.{Discs by ⍵} Query.Text
⍝('no match'≡discs)/'Status.Text←discs ◇ →0'
ArtistList.Items.Add ¨ ↓{⍵[⍋⍵;]}↑⍪discs.Artist.FullName
ActiveForm←ShowArtists
∇
</script>

<Mobile:Form runat="server" id=Start>
 <Mobile:LiteralText runat="server" Text="Artist Query?"/>
 <Mobile:TextBox runat="server" id=Query/>
 <Mobile:Command runat="server" Text = Search id=Search
OnClick=ListArtists/>
 <Mobile:LiteralText runat="server" id=Status/>
</Mobile:Form>

<Mobile:Form runat="server" id=ShowArtists Paginate=true>
 <Mobile:List runat="server" id=ArtistList
OnItemCommand=ListDiscs/>
</Mobile:Form>

<Mobile:Form runat="server" id=ShowDiscs Paginate=true>
 <Mobile:List runat="server" id=DiscList
OnItemCommand=ListTracks/>
</Mobile:Form>

<Mobile:Form runat="server" id=ShowTracks Paginate=true>
 <Mobile:List runat="server" id=TrackList/>
</Mobile:Form>
```

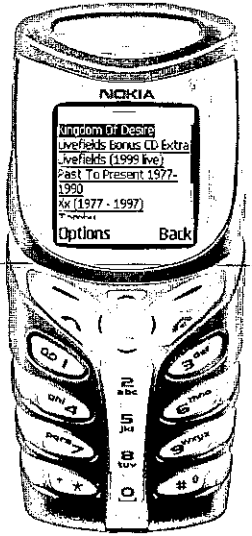
The first page



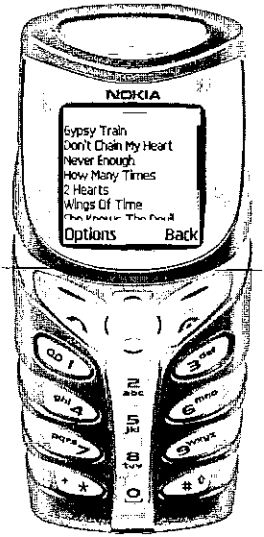
Query for 'Discs by "Toto"'



Select a Toto disc



Get the track list



(if it's in this list, don't buy it again!)

TECHNICAL SECTION

This section of VECTOR is aimed principally at those of our readers who already know APL. It will contain items to interest people with differing degrees of fluency in APL.

Contents

Solving the "Now I Know" Problem in J	Howard Peelle	92
Towards a General Theory of Formula Inversion	Ian A. Clark	100
Data Compression with Dfns	Veli-Matti Jantunen	107
A Variable Editor	David Crossley	111
At Play with J: The Magical Matrix	Gene McDonnell	122
Taking the RTF Challenge	Adrian Smith	127
R.net: A Reverse Polish Interpreter for .Net	Richard Smith	136

Solving the "Now I Know" Problem in J

by Howard A. Peelle (hapeelle@educ.umass.edu)

A problem presented in the previous issue, Vector 20.1, is analysed here using J.

Problem

"Now I Know"

> There are two positive integers, each greater than 1 and less than 100. Pete knows their product. Serena knows their sum. First, Pete says: "I don't know what they are." Then Serena says: "I know you don't know." Then Pete says: "Now I know." And Serena says: "So do I." What are the two integers?

(The allusion is to Pete Sampras and Serena Williams, tennis champions who both won the U.S. Open in 2002.)

Analysis

Imagine the following thinking by Pete (P) and Serena (S) using J:

P: I'll start with a list of possible integers:

```
n =: 2 + i.98                          n is 2 plus integers 0 to 97
n                                      Show n
  2 3 4 5 6 7 8 9 ... 99               List of 98 integers
```

Serena's sum could be any integer in n plus any integer in n. Here is a table of all possible sums:

```
s =: n +/ n                            s is each n plus each n
s                                      Show s (a 98 by 98 table)
  4   5   6   7   8   9  10  11 ... 101
  5   6   7   8   9  10  11  12 ... 102
  6   7   8   9  10  11  12  13 ... 103
  7   8   9  10  11  12  13  14 ... 104
  8   9  10  11  12  13  14  15 ... 105
  9  10  11  12  13  14  15  16 ... 106
 10  11  12  13  14  15  16  17 ... 107
 11  12  13  14  15  16  17  18 ... 108
  .                                    .
  .                                    .
  .                                    .
101 102 103 104 105 106 107 108 ... 198
```

I know Serena's sum is one of these sums.

> Note: Table s is symmetrical about the main diagonal due to the commutativity of addition, so there are duplicate sums in the bottom left half of the table. Repeats appear along reverse diagonals.

S: Pete's product could be any integer in n times any integer in n. Here is a table of all possible products:

```
p =: n */ n                                    p is each n times each n
p                                              Show p (a 98 by 98 table)
   4   6   8  10  12  14  16  18 ... 198
   6   9  12  15  18  21  24  27 ... 297
   8  12  16  20  24  28  32  36 ... 396
  10  15  20  25  30  35  40  45 ... 495
  12  18  24  30  36  42  48  54 ... 594
  14  21  28  35  42  49  56  63 ... 693
  16  24  32  40  48  56  64  72 ... 792
  18  27  36  45  54  63  72  81 ... 891
   .                                   .
   .                                   .
   .                                   .
 198 297 396 495 594 693 792 891 ...9801
```

I know Pete's product is one of these products.

> Note: Table p is symmetrical about the main diagonal due to the commutativity of multiplication, so there are duplicate products in reverse diagonals.

S: My goal is to eliminate products in p corresponding to repeats of my sum (which appear on a certain diagonal of s), hopefully leaving only one possible product (and its duplicate). Then, once I know Pete's product, I can determine the two addends of my sum.

P: My goal is to eliminate sums in s corresponding to repeats of my product which appear in different diagonals in p, hopefully leaving only one possible sum (and its duplicate). Then, once I know Serena's sum, I can determine the two factors of my product.

P: Unfortunately, I can't announce the answers right away because I know my product cannot be factored into two unique factors. Indeed, it appears repeatedly in p. Since I can't determine Serena's sum now, I must say "I don't know".

S: If Pete doesn't know the answers, his product must have three or more prime factors. So I can eliminate from p all possible products of two primes:

```
primes =: p: i.25                    List of primes < 100
primes
  2 3 5 7 11 13 17 19 23 29 31 ... 97

pp =: primes */ primes               Table of products of primes
pp                                   (25 by 25 table)
    4   6  10  14  22  26 ...  194
    6   9  15  21  33  39 ...  291
   10  15  25  35  55  65 ...  485
   14  21  35  49  77  91 ...  679
   22  33  55  77 121 143 ... 1067
   26  39  65  91 143 169 ... 1261
    .                    .
    .                    .
    .                    .
 194 291 485 679 1067 1261 ...9409

zero =: -. p e. ,pp                  0 where p in pp, else 1
zero                                 (98 by 98 table)
   0   0   1   0   1   0   1   1 ... 1
   0   0   1   0   1   0   1   1 ... 1
   1   1   1   1   1   1   1   1 ... 1
   0   0   1   0   1   0   1   1 ... 1
   1   1   1   1   1   1   1   1 ... 1
   0   0   1   0   1   0   1   1 ... 1
   1   1   1   1   1   1   1   1 ... 1
   1   1   1   1   1   1   1   1 ... 1
   .                                .
   .                                .
   .                                .
   1   1   1   1   1   1   1   1 ... 1
```

Alternatively: `zero =: 2 < #@q: p`

```
p =: p * zero                        p times zero (item by item)
p                                    Updated p
   0   0   8   0  12   0  16  18 ... 198
   0   0  12   0  18   0  24  27 ... 297
   8  12  16  20  24  28  32  36 ... 396
   0   0  20   0  30   0  40  45 ... 495
  12  18  24  30  36  42  48  54 ... 594
   0   0  28   0  42   0  56  63 ... 693
  16  24  32  40  48  56  64  72 ... 792
  18  27  36  45  54  63  72  81 ... 891
   .                                .
   .                                .
   .                                .
 198 297 396 495 594 693 792 891 ...9801
```

This zeros out all products of two primes.

Note: Cubes of primes have two unique factors and could be eliminated similarly. (See Appendix)

P: Following Serena's thinking (above), I can also zero out the corresponding possible sums in my table s:

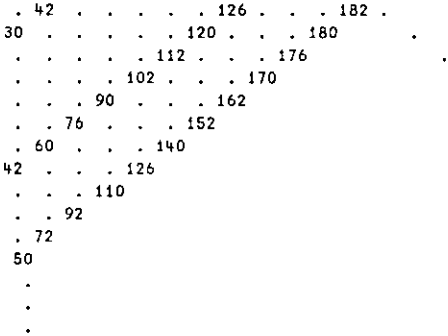
```
s =: s * zero                              s is s times zero
s                                          Updated s
   0   0   6   0   8   0  10  11 ... 101
   0   0   7   0   9   0  11  12 ... 102
   6   7   8   9  10  11  12  13 ... 103
   0   0   9   0  11   0  13  14 ... 104
   8   9  10  11  12  13  14  15 ... 105
   0   0  11   0  13   0  15  16 ... 106
  10  11  12  13  14  15  16  17 ... 107
  11  12  13  14  15  16  17  18 ... 108
   .                                   .
   .                                   .
   .                                   .
 101 102 103 104 105 106 107 108 ... 198
```

P: Now, is there just one sum corresponding to my product? Unfortunately, no. I know there are several possible sums remaining. So I cannot announce the answers.

Note: If Pete does not know the answers at this point, we could look for unique sums (such as 6) in s and eliminate them. Similarly, if Serena does not know the answers, we could look for unique products (such as 8) in p and eliminate them. This could be repeated until there is no change in p and s. (See Appendix) But this may not be necessary.

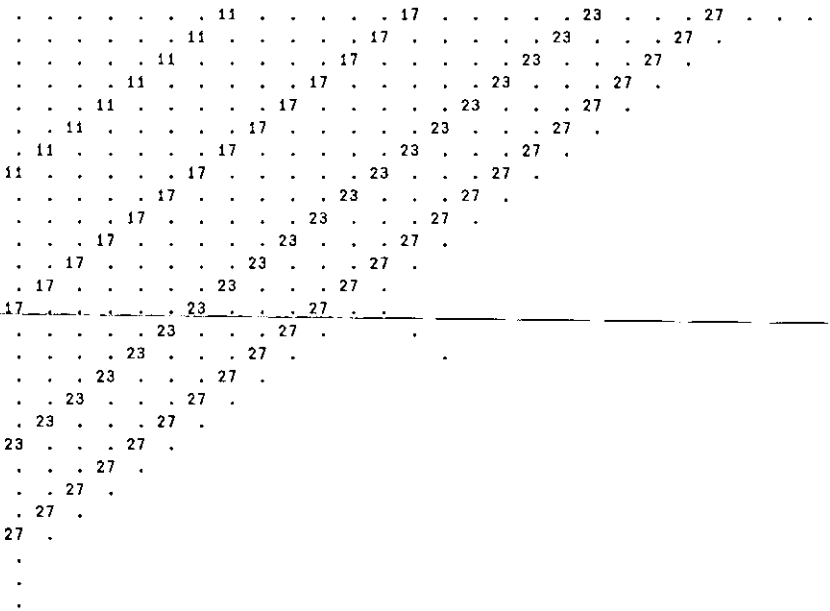
S: Is there just one product corresponding to my sum? No. I know there are several possible products remaining in p along the corresponding diagonal in s with my sum. So I cannot announce the answers. Actually, the diagonal in p corresponding to my sum has <u>all</u> possible products remaining. And there are several of these "full" diagonals (a few are shown here with dots as background):

```
. . . . . . . 18 . . . . . 30 . . . . . 42 . . . 50 . . .
. . . . . . 24 . . . . . 42 . . . . . 60 . . . 72
. . . . . 28 . . . . . 52 . . . . . 76 . . . 92
. . . . 30 . . . . . 60 . . . . . 90 . . . 110
. . . 30 . . . . . 66 . . . . . 102 . . . 126
. . 28 . . . . . 70 . . . . . 112 . . . 140
. 24 . . . . . 72 . . . . . 120 . . . 152
18 . . . . . 72 . . . . . 126 . . . 162
. . . . . 70 . . . . . 130 . . . 170
. . . . 66 . . . . . 132 . . . 176
. . . 60 . . . . . 132 . . . 180
. . 52 . . . . . 130 . . . 182
```



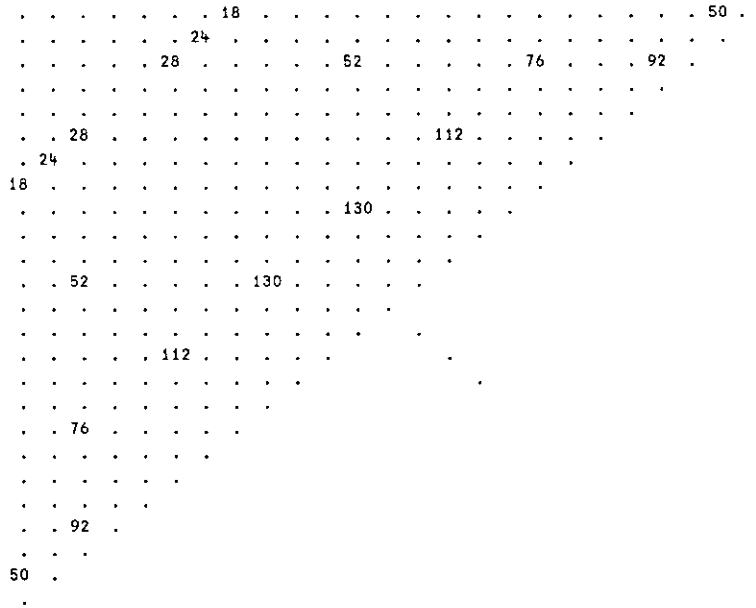
So I can say "I know you don't know".

P: If Serena doesn't know the answers, from looking at her p, all possible products in the corresponding diagonal in s containing her sum must be factorable into primes in different ways. So, I can eliminate all diagonals containing a 0. Now I know that her sum is one with a corresponding "full" diagonal in s: 11, 17, 23, 27, 29, 35, 37, 41, 47... . (The first few are shown here with dots as background.)



Actually, my product has two prime factors which sum to only one of these sums in s. So, I can say "Now I know".

S: If Pete knows the answers, his product cannot have two different prime factorings. His product must appear on only one full diagonal in p. So I can eliminate multi-factorable products which appear in more than one full diagonal: 30, 42, 60, 66, 70, 72, etc. Finally, that leaves only one product with my sum alone on a diagonal of p:



Now I can say "So do I".

P & S: Pete's product is 52. Serena's sum is 17. The answers are 4 and 13.

Review

With the answers in mind, let's review the problem from the beginning, where Pete knows only his product and Serena knows only her sum:

Pete's product (52) factors into 2*26 or 4*13. Possible corresponding sums are 28 or 17. So Pete must say "I don't know".

Serena's sum (17) partitions into 2+15, 3+14, 4+13, 5+12, 6+11, 7+10, or 8+9. Possible corresponding products are 30, 42, 52, 60, 66, 70, or 72. So Serena doesn't know which is Pete's product. But she realizes that each of these products can be factored in at least two different ways—that is, they are not unique products of two primes (nor cubes of a prime). So, she says "I know you don't know".

Pete then reasons that if Serena's sum were 28, it can be partitioned into two primes 5+23 and she would not know for sure that he didn't know the answers. In other words, if Pete's product were 5*23, he would know the answers. So Pete concludes that Serena's sum is not 28 and says "Now I know".

Finally, Serena concludes that if Pete knows the answers, his product can't have more than two different factorings. Since the possible products 30, 42, 60, 66, 70, and 72 each have three or more prime factors, his product must be the only other one - 52 - which has two factorings: 2*26 and 4*13. With her corresponding sum of 17, she knows the answers are 4 and 13 and says "So do I".

Extensions

Readers may wish to write a J program to solve this "Now I Know Problem" generally and to consider the following questions:

1. Is it necessary to remove products of two primes in order to solve this problem?
2. Is it necessary to remove the cubes of primes in order to solve this problem? (See Note 1 in Appendix.)
3. Is it necessary to eliminate unique sums and unique products successively in order to solve this problem? (See Note 2 in the Appendix.)
4. What is the smallest upper limit (<100) for which this problem can be solved?
5. What is the largest upper limit (>100) for which this problem can be solved?

Reference

[1] www.mathematik.uni-bielefeld.de/~sillke/PUZZLES/logic_sum_product

Appendix

Note 1: Cubes of primes also have two unique factors and could be eliminated:

```
p3 =: primes ^ 3              Cubes of primes

zero =: -. p e. p3            0 where p is in p3

p =: p * zero                 Update p

s =: s * zero                 Update s
```

Note 2: Pete and Serena could look in s and p and eliminate any possible unique sums and products, respectively:

```
                                A utility program to calculate
Freq =: (+/) @ (+/) @ (=/~)     frequencies of each item

zero =: (Freq s) > 2          0 where s frequencies not > 2
s =: s * zero
p =: p * zero
zero =: (Freq p) > 2          0 where p frequencies not > 2
p =: p * zero
s =: s * zero
```

This elimination procedure could be repeated (as necessary) until there is no change in p and s.

Towards a General Theory of Formula Inversion

by Ian A. Clark

Introduction

Microsoft Excel™ allows a designated cell in a grid to be computed according to a user-provided formula based on other cells, which we may call its *juniors*. If a junior cell is altered by overtyping its value, all computed cells automatically recompute their values to accommodate the change. However if you overtype the value in a given computed cell, Excel does not propagate this change back to the juniors. Instead it discards the cell's formula and converts the cell to an uncomputed one.

Some grid-based software products, notably CPA (Cognos Management Series Planning Analyst™, formerly: Adaytum Planning™) [1], do preserve the user's formula, adjusting some or all of the juniors so that the computed result equals the new value in the formula cell. To do so it must compute some sort of 'inverse function' to that defined by the formula. We call this process *formula inversion*. The proprietary formula inversion feature of CPA is called 'break-back'.

In this paper we apply the differential calculus plus some elementary functional analysis to propose a general theory of formula inversion. This permits a simple algorithm (for which we can demonstrate an APL prototype called TABULA) that does not need to analyse, or even look at, the contents of the formula for computing the new juniors. We observe in passing that 'break-back' in CPA is based on a fundamentally different algorithm - one that does need to analyse the formula (which it does into individual arithmetic operations). Its derivation owes nothing to the differential calculus. However in many instances the two approaches do give the same answers - which helps establish the plausibility of both. And where they don't give the same answers, the reasons are interesting.

One well-known method of computing an inverse function from a given forward-transfer function is by the method of *successive-approximation*, aka *Newton-Raphson*, which CPA's 'break-back' does not use. The algorithm we shall develop here broadly follows the standard treatment, see for example [2], but with the apparently novel extension of replacing a multivariate function with a function of a single variable t by judicious choice of so-called "model" functions.

The algorithm discussed here has been put to work in an eco-calculator, TABULA, which allows the user to set up a net of dependent cells in terms of their formulas. When the value of a given calculated cell is overtyped, the alterands in the net are recomputed to honour the new value, even though only forward-calculation formulas are stored and used.

Theory

Suppose a given cell in a grid takes the value z. This may either be independent of other cells, being inserted directly by the user (whence the cell is known as a *detail* cell), or it may be calculated from the values: $\{x_1, x_2, \ldots x_n\}$ of several other cells, called its *juniors* (whence the cell itself is known as a *total* cell). In the most general case we may consider z to be defined by:

(1): $z = f(x_1, x_2, \ldots x_n)$

where f is a multivariate function defined by the collection of formulas used to calculate the given total. We say 'formulas' (plural) because the definition of f may be multi-level, the juniors of z themselves being computed by formulas from yet more juniors.

If z is overtyped with a new value: z', a grid exhibiting *formula inversion* adjusts some or all of the junior cell values, the *alterands*, to preserve the above relationship, namely so that:

(1a): $z' = f(x_1', x_2', \ldots x_n')$

An essential feature of any useful formula inversion facility offers the ability to fix or *hold* one or more alterands: $\{x_1, x_2, \ldots x_n\}$ to prevent them varying when: z is replaced by: z'. Doing this may of course hinder formula inversion from taking place – as it obviously does if *all* alterands are held.

We offer a precise mathematical meaning for the term: *formula inversion.*

Definition 1 (inversion demand)

Let: $\Gamma = \{\Delta z, f, \{x_1, x_2, \ldots x_n\}\}$ be an ordered triple consisting of a change to the original value z of a given total cell, expressed as an increment: Δz, a function: f, and a list $\{x_1, x_2, \ldots x_n\}$ of arguments to f (being the values of the alterands). The last two items define the original value z of the total cell by the equation: $z = f(x_1, x_2, \ldots x_n)$. We call Γ an *inversion demand*, or simply a *demand.*

Definition 2 (formula inversion operator, formula inversion)

Let Ξ be a mapping $\Gamma \rightarrow \{\Delta x_1, \Delta x_2, \ldots \Delta x_n\}$ of the *demand* Γ onto a set of *increments* $\{\Delta x_1, \Delta x_2, \ldots \Delta x_n\}$ of $\{x_1, x_2, \ldots x_n\}$ such that:

(2): $z' = (z + \Delta z) = f((x_1 + \Delta x_1), (x_2 + \Delta x_2), \ldots (x_n + \Delta x_n))$

where z is defined by equation (1). We call Ξ a *formula inversion operator*. The actual *formula inversion* $\Xi\Gamma$ is the ordered set of increments resulting from applying Ξ to Γ. It follows that

$\Xi\Gamma = \{\Delta x_1, \Delta x_2, \ldots \Delta x_n\}$.

In other words, the act of substituting: z' for: z causes the grid to replace the alterands $\{x_1, x_2, \ldots x_n\}$ by a corresponding set $\{x_1', x_2', \ldots x_n'\}$ given by:

$x_i' = (x_i + \Delta x_i), \; i = 1, 2, \ldots n.$

This model of formula inversion differs from that implicit in the APL code of CPA. The latter computes a *ratio* applied to each of the $\{x_1, x_2, \ldots x_n\}$ in order to get $\{x_1', x_2', \ldots x_n'\}$, not a set of *increments*. This is a weaker principle than ours. In particular it does not readily accommodate *held cells* (i.e. those whose values are constrained not to change under formula inversion) nor zero values of z. This is because there is no finite ratio which will change a zero value to a non-zero value, nor any ratio except 1 which will leave a held value unchanged.

Formula inversion as defined above, is by no means unique. Quite apart from the freedom to choose one or more alterands to *hold*, one can come up with many different strategies for adjusting the alterands of a given computed cell, all giving substantially different answers. Thus, to take an extremely simple case, let: $z = f(x_1, x_2) = x_1 / x_2$. If, say, we were to double: z to get: z', should we choose a Ξ_j which doubles x_1, or one that halves x_2, or one that allocates some part of the necessary change to each? We must seek some rational basis for choosing from among the vast ensemble of conceivable formula inversion operators $\{\Xi_j\}$.

What rational basis can there possibly be to arrive at a generally accepted choice of Ξ_j ? Somehow the alterands $\{x_1, x_2, \ldots x_n\}$ must all be adjusted 'in-step' in some sense. We may formalise this principle by considering each alterand x_i, $i = 1, 2, \ldots n$, to be a function of some single notional underlying parameter t, thus:

$x_i = g_i(t), \; i = 1, 2, \ldots n.$

Substituting for $\{x_1, x_2, \ldots x_n\}$ in expression (1) gives:

$z = f(g_1(t), g_2(t), \ldots g_n(t))$

which re-expresses z as a function of a single variable t. This offers a general model of formula inversion. As yet we have made no arbitrary assumptions, since any particular model of formula inversion can be accommodated by a suitable choice of ordered set of functions $\{g_i\}$.

We note in passing the possibility of offering a whole library of choices of $\{g_i\}$ to the model-builder or end-user. These might be based upon truncating a convergent polynomial expansion of $\{g_i\}$, which is one way of expressing a wide range of candidate g-functions:

(3): $g_i(t) = a_i + b_i t + c_i t^2 + d_i t^3 + \ldots$

where a_i, b_i, c_i, d_i ... are constant terms. However it so happens that just two choices are all we need to consider here:

(4): $\hat{g}_i(t) = a_i$

(5): $\check{g}_i(t) = a_i + b_i t$

which represent truncating the polynomial in expression (3) to 1 and 2 terms respectively. Note that expression (4) is equivalent to specifying that the value of x_i must be *held* at its present value, viz. $x_i = a_i$. Thus our general theory accommodates *held cells* in a natural fashion without making a special case, and we can reasonably expect no need for special APL code to handle held cells. It also holds out possibilities for various kinds of 'sloppy hold', defined by some special case of equation (3). Such a 'sloppy hold' might be chosen to constrain x_i for some given i to lie within some given interval, or alternatively to cause x_i to vary slower than the other alterands.

Differentiating (4) and (5) with respect to t:

(4a): $d\hat{g}_i / dt = 0$

(5a): $d\check{g}_i / dt = b_i$

Equation (5a) asserts that the first derivative of $\check{g}_i$ is a constant term. We are at liberty to choose to set $b_i = x_i$, the current value of the alterand i. We discover we may also want to set $b_i = -x_i$ on occasions, to make f decrease, not increase, when we increase the given alterand i.

Let us now assume that f possesses a first total derivative at the given value: z. The first derivative of f with respect to t is given by the standard formula:

$$df/dt = (\partial f/\partial x_1)(dg_1/dt) + (\partial f/\partial x_2)(dg_2/dt) + \ldots + (\partial f/\partial x_n)(dg_n/dt)$$

Provided $df/dt \neq 0$ (an important case we shall treat later) and provided all f-derivatives of order >1 are small compared with the first derivative (arising as a

consequence of applying Taylor's Theorem to f, which we shall not delve into here) we obtain the following finite-difference approximation:

$$(6): \qquad \Delta f \approx \frac{df}{dt}\Delta t \;=\; \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} \times \frac{dg_i}{dt}\Delta t$$

This gives us the means to estimate Δt given $\Delta f = (z' - z)$, the difference between the new and the old values of the given total cell. From Δt we can estimate $\{\Delta x_1, \Delta x_2, \ldots \Delta x_n\}$ and thence $\{x_1', x_2', \ldots x_n'\}$ from finite-difference equations derived from (4a) and (5a), namely:

(4b): $\Delta x_i \approx (d\hat{g}_i / dt)\,\Delta t = 0$

(5b): $\Delta x_i \approx (d\check{g}_i / dt)\,\Delta t = b_i \Delta t$

Then to calculate the ordered set of partial derivatives $(\partial f / \partial x_i: i = 1, 2, \ldots n)$.

We can estimate these terms one at a time by incrementing each x_i, $i = 1, 2, \ldots n$. in turn by a small amount Δx_i and finding the matching change in: f, thus:

$$\partial f / \partial x_i = (f(x_1, x_2, \ldots (x_i + \Delta x_i), \ldots x_n) - f(x_1, x_2, \ldots x_i, \ldots x_n))/\Delta x_i \;+\; \delta$$

where $\delta \rightarrow 0$ as $\Delta x_i \rightarrow 0$.

What does it mean "a small amount Δx_i"? Here we abandon mathematics for heuristics. An APLer will think to employ ⌻CT, but this turns out to be rather too small. TABULA effectively uses `1E¯10×1⌈x` where `x` is the initial value of the alterand. In other words a fixed value of `1E¯10` is fine for values greater than one and is not critical, but scientific calculations can entail very small numbers (which financial ones generally don't) so some scaling is wise in these cases.

Thus we have a way of calculating an approximate result $\Xi\Gamma \approx \{\Delta x_1, \Delta x_2, \ldots \Delta x_n\}$ of inverting the demand: $\Gamma = \{\Delta z, f, \{x_1, x_2, \ldots x_n\}\}$ without needing to know the actual formula defining the forward-calculation function f of Γ.

If f is linear, the approximation is exact, i.e.:

$$(2): \quad z' = (z + \Delta z) = f(\,(x_1 + \Delta x_1), (x_2 + \Delta x_2), \ldots (x_n + \Delta x_n)\,)$$

Otherwise, for more general f, the new forward calculation will result in a value $\check{z}$ which is neither z nor z'. Hopefully it will be closer to z', so it is possible to repeat the whole process, this time inverting $\Gamma_2, \Gamma_3, \ldots$ where at each iteration we replace Δz in Γ by $\Delta\check{z} = z' - \check{z}$. Typically no more than 4 iterations are necessary on a 32-bit computer with the typically simple sorts of formulas arising in scientific and financial systems before underflow takes place and the equality (2) is sufficiently closely approximated.

Choosing a Different Formula Inversion Model

Now what if $df/dt = 0$? If this happens it means we have failed to come up with a suitable underlying parameter t which, when varied, causes f to vary, and so our algorithm fails. Fortunately it is an easy situation to detect before we actually commit ourselves to changing alterands. The remedy is to go back and review our choices of 'model' functions $\{g_1, g_2, ... g_n\}$ because it is these that determine the nature of t. In other words we must choose a different formula inversion model for some or all of: $\{x_1, x_2, ... x_n\}$ until we find one for which $df/dt \neq 0$.

The situation commonly arises with formulas like this:

$f_+(x_1, x_2) = x_1 + x_2$ where: $x_1 = -x_2$

$f_-(x_1, x_2) = x_1 - x_2$ where: $x_1 = x_2$

These of course both give $z = 0$, which causes problems for the ratio-based technique of CPA. One might expect CPA to treat $x_1 - x_2$ simply as $x_1 + x_2$ with negative x_2, but in actual fact it chooses among separate algorithms and the user has no part in the choice.

Worse is to come. Let Γ, Γ_Δ be two slightly differing formula inversion demands, defined by:

$\Gamma = \{ \Delta z, f_+, \{x_1 , (x_2 = -x_1)\} \}$

$\Gamma_\Delta = \{ \Delta z, f_+, \{x_1 , (x_2 = \Delta - x_1)\} \}$

where Δ is some small quantity which we can choose to make vanish. It is not unreasonable to expect: $\Xi\Gamma_\Delta \rightarrow \Xi\Gamma$ as $\Delta \rightarrow 0$. This is not the case with the CPA formula inversion operator Ξ for which $\Xi\Gamma_\Delta$ and $\Xi\Gamma$ differ enormously, even for small values of Δ. In fact the $\{x_i\}$ of $\Xi\Gamma_\Delta$ tend to $\pm\infty$ as $\Delta \rightarrow 0$, whereas $\Xi\Gamma$ mimics the behaviour of formula inversion for f_- not f_+.

Our general theory handles this by suitable choice of either $b_2 = x_2$ or $b_2 = -x_2$ in equation (5b). Thus we do not need separate algorithms to handle f_+ and f_-, nor to handle the cases arising with (say) f_+:

$x_2 = -x_1$

$x_2 = \Delta - x_1$

Which Direction Now?

A theory of formula-inversion based on the differential calculus provides an orderly theoretical framework for discussing the practical problems of formula-inversion problems, hitherto not apparent in commercial offerings like [1].

Specifically, how to invert the simple formula: $x_1 - x_2$. Recognising that formula inversion needs a choice of model to describe it fully leads not only to a treatment of 'held' values simply as an extension of the general case, but a recognition that there are quite different semantics attachable to the operation of subtraction. Accountants have been aware of this since antiquity, but somehow scientists tend to overlook it, thanks to an uncritical acceptance by the scientific professions of one of many possible interpretations of minus numbers. In the immortal words of Cassandra writing in the Daily Mirror in the 1960's: "show me a minus-egg. Cook it for me. Eat it!"

One needs to ask: is the subtracted quantity of the same nature as the quantity it is subtracted from, or not? Is it a credit item, say, rather than a debit one? An expenditure rather than income? Are you allowed to go into overdraft (and if so by how much)? Does it represent noise rather than information? In arithmetic terms, should $x_1 - x_2$ be handled as two summed quantities on a scale of $-\infty$ to $+\infty$, with the convention of convenience that the sign of x_2 is to be switched before addition? Or is the second quantity on the scale of $(0,\infty)$ only? Or maybe that should be $(1,\infty)$? To an APLer, is the minus really a summed hi-minus, or the negation operator, or the subtraction operator?

Such considerations remind one that the minus operator (and by analogy: division, if you logarithmically transform the scale) is far from docile, mathematically speaking and may, in the words of Bourbaki, be "abuse of notation" where it occurs. Practically speaking it determines the direction in which x_2 should move in order to achieve formula-inversion of $x_1 - x_2$, which we determine in our theory by choice of 'model' function: g_2.

Sample Application

By way of an illustration of the use of this algorithm, the innovative scientific calculator TABULA, mentioned above, will be described in a future edition of Vector. This employs a formula-inversion engine consists of 10 or so APL functions using the above principles, written to be portable across different platforms.

References

[1] *Cognos Management Series, Planning Analyst*, version 7.1. Cognos Incorporated. 2003.

[2] Press, W.H., et al. *Numerical recipes in C++*. Cambridge University Press, ISBN 0-521-75033-4. (Also available in C, Fortran 77 and Fortran 90, and on-line: *http://www.nr.com/storefront.html*)

Data Compression with Dfns

by Veli-Matti Jantunen, Statistics Finland (veli-matti.jantunen@stat.fi)

"There are zillions of programs already written in other programming languages which address data compression; some of them might even be better at the job than an APL programmer could knock up in half an hour... Build APL on top of common tools rather than write everything for ourselves all the time."
-Dick Bowman

"The ultimate compression: binary information only contains ones and zeros, and because zeros mean nothing, you can just count the ones... I'm sure that someday someone will invent the way to extract thus easily compressed data..."
—Anonymous

Why bother?

The years in the DOS world have left their scars: I still try to squeeze everything possible to smaller entities. I detest the idea of my workspace containing unnecessary white space or other compressable stuff, which unnecessarily doubles the download time or fills every diskette, not to mention the myriad of different version workspaces on my hard disk.

This article tells briefly how I developed a data compressor (actually plenty of them), but the reader should appreciate that I had the support of a dfns mailgroup behind me all the time. I could always just send my airy-fairy ideas and code fragments to the group and got useful feedback every time, either via the mailgroup or direct. Thanks to every one of you!

APL development is fun, especially with professional help ;-).

Early Habits

The simplest data packing algorithms have been in use for years. When these tools were converted to dfns, the ideas shone through nicely. Null packing simply rips off the zeroes and uses the Boolean vector to mark their places.

```
packN←{
      cmp←{mask←,⍵ 1↑0⍴⍵ ⋄ (⍴⍵)mask(mask/,⍵)}
      exp←{(shape mask items)←⍵ ⋄ shape⌷mask\items}
      ⍺←1 ⋄ ⍺:cmp ⍵ ⋄ exp ⍵
 }
```

Note that this clever programming style (shamelessly adapted from John Scholes) lets you write both the packer and the unpacker functions in the same body. Furthermore, you can write the testing operator which utilises this structure:

```
      PackTest←{⍵≡0 ⍺⍺ ⍺⍺ ⍵}
      packN PackTest 2 3 4⍴10 0 0 20 0 0 0
1
```

Unique packing is a nice and quickly implemented method (if you have cases with more than 127 but less than 256 unique elements, you may subtract 128 from the indices when compressing to abuse the internal representation; remember to add them when extracting).

```
packU←{⎕IO←0
      cmp←{u←∪,⍵ ⋄ (⍴⍵)u(u⍳,⍵)}
      exp←{(0⊃⍵)⍴(1⊃⍵)[2⊃⍵]}
      ⍺←1 ⋄ ⍺:cmp ⍵ ⋄ exp ⍵
 }
```

The third widely used tool is simple **RLE packing**, which can be found commented upon (!) in the dfns workspace (*packR*), just like the other packing functions mentioned here.

New Tricks to an Old Dog

When Adrian and Stefano revealed in their Vector article how to use the GZIP engine to squeeze data inside the workspace, I felt that surely there must be better ways to pack data than I had been using. Although the idea to use external dll is a quick and effective one, I happen to be an old stick-in-the-mud: no external parts if possible. Back to the old drawing board - and discussion about compressing algorithms ignited instantly in the dfns mailgroup.

My aim was to develop a quick compressor with a decent packing ratio and as universal use as possible. I could always use GZIP as a measurement rod, and

thus could separate interesting but useless ideas apart from those that seemed worth a closer look.

Borrowed Ideas

Stefano explained an old algorithm that was used to pack certain text data, which contained long patterns of the same character. The idea was something like this: there is one character ("esc") that is used to denote the packed parts, the following character tells how many times the third character is to be repeated. When I had finished the function it became clear that this was meant to be a packer for special purposes, not the all-round text packer, although it bears the name `packT`.

The next idea was to try to find an open algorithm (as opposed to LZW, which is patented, and anyhow seemed too complicated for me), and I came across the Shanno-Fano algorithm, which belongs to the Huffmann compression family.

The compressor (`packS`) was pretty straightforward to do (use variable length keys, use the shortest for the most frequent items), the uncompressor was a pain in the brain. Eventually I cooked up something that works, but every time I look at the `packH` code (John's version of Huffmann compressor), I feel uneasy.

An APL Approach: Do It Yourself

After furious coding I felt it a little bit awkward to try to mimic other people's ideas instead of doing something original.

The idea came to me when I was finishing this GUI application with lots of bitmaps hanging around. Although the first compression can be made by diminishing the palette to 16 colours, they still use much of the ws estate. Sixteen colours means sixteen indices which means that they can be presented with just three bits! One night with motivation, and I had it: first find the unique items and store them separately, then calculate the smallest amount of bits that those indices need, and decode them as a Boolean vector. Afterwards, I added a quick RLE coder, and the bitmap packer `packB` was born. I was astonished to see how well it performed when compared with the really professional tools. Especially the extracting speed was a pleasant surprise.

But this wasn't quite the general tool I was looking for.

After some months' quiet life, I got this simple idea: instead of packing single items, I could concentrate on paired items. The pairings can be derived with the unique vector's outer product with itself, and using Booleans and clever indexing

I realised that the packing ratio was in some cases better than with the earlier functions.

The last idea thus far came the next day: what about packing the items according to frequencies so that the first 1 will start a new key, i.e. the shortest keys would be 0, 1 0, 1 1 0, and 1 0 0. Quickly, this idea was coded as:

```
packQ←{(⎕ML ⎕IO)←3 1
     key←⊃,/{ω{⍺↓⍨⍺↑ω}¨⊂ω∘.>ω}⍳23
     cmp←{dt←,ω ⋄ u←⍬dt
          f←{+/ω=dt}¨u ⋄ u←u[⍋f]
          b←∊(1,¨(⍴u)↑key)[u⍳dt]
          u((⍴⍴ω),⍴ω)b}
     unc←{(u d b)←ω ⋄ r←(↑d)⍴1↓d
          p←b≤¯1↓0,b ⋄ r⍴u[key⍳p⊂b]}
     α←1 ⋄ α=1:cmp ω ⋄ unc ω
 }
```

Which is by far the best I have been able to put together.

A Teaser: An Enchanting Universal Algorithm

Just an idea: any simple array can be changed to the binary form and back without loss of information. Take two integers, the first is the random seed for roll, the next is the number of binaries to be created. Because there are patterns that can be really long, and thus easily created, this might be the Ultimate Packing Algorithm.

Why doesn't it work? (I don't confess I tried, though.)

Happiness is a warm compressor!

PS To subscribe to the dfns group, send email to dfns@dyalog.com with subject: subscribe

References

[1] Compressing APL arrays with GZIP, Vector 18.1, 84

[2] Workspace dfns.dws from www.dyalog.com/download/dsamples.zip.

A Variable Editor

by David Crossley (email: crossley@au-village.com)

Abstract

The paper describes an open-source utility to view or edit a variable but specifically any generalised nested array. The described version is written for Dyalog APL though other implementations are planned. The variable may be sourced from and saved to any namespace in the current workspace or any other workspace. The editor is designed as a development aid. Although available as two namespaces, options are provided to incorporate the tool as a tiny namespace within the Dyalog Session with almost zero-footprint except when active, accessible through the Session menu and toolbar buttons. The utility and installation program are available as a free download from the web.

Background

At the Madrid Conference in 2002, Jim Lucas [1] presented a paper ("Looking Back On Looking Ahead") in which he made the startling observation that there was no true editor that could handle arrays of any arbitrary depth, fundamental building blocks of APL applications. Whilst APL versions usually include some form of built-in editor, they are limited to simple arrays or at best to text vectors which can be treated as simple arrays.

The development of the editor described in this paper was inspired by that observation. Although not exactly rocket-science, it proved to be a non-trivial undertaking. Former experience with handling large multi-dimensional data in a spreadsheet manner offered an immediate design strategy on how to traverse the structures of arbitrary nested arrays using the Grid object.

In addition, I wanted to provide simplified means to view and edit commonly-used structures but at the discretion of the user; in particular, arrays of text vectors and "parallel" arrays, of which more later.

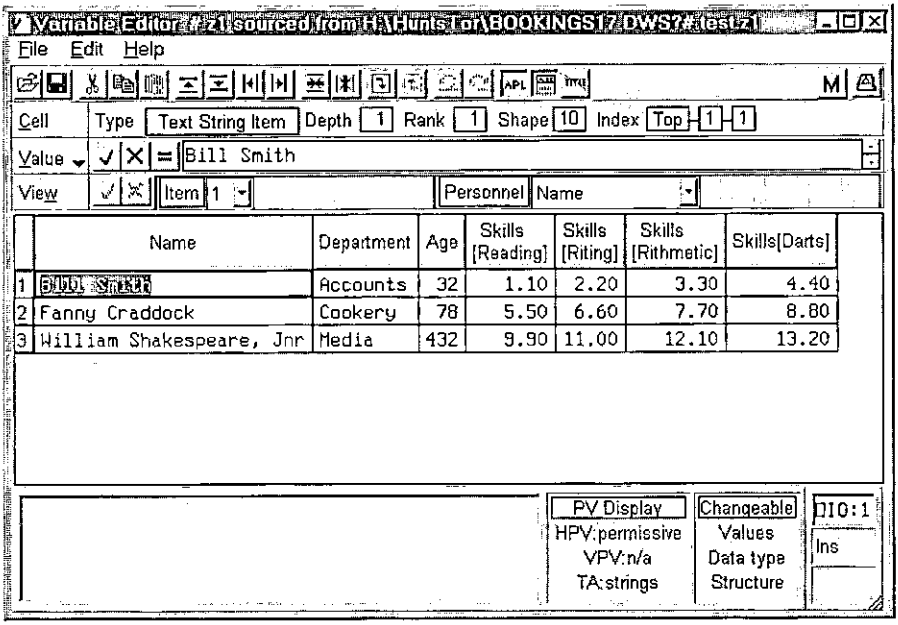
As a further aid to frequent editing of data structures, I felt that the ability to provide alternative text labels for axes and axis subscripts should be included.

And finally, for a variable editor to be worth its salt, it must be easily accessible, and it should be able to source and save variables anywhere. This led me to provide a way to integrate the editor into the session, and to develop an ancillary

product, available as a standalone utility, modelled on the Windows FileBox object, a utility that I have called WSBox.

The Variable Editor Dialog

The screen-shot below illustrates the dialog in which an array structure has been opened for editing.



The Caption Bar identifies the name of the variable and its source, in this case a workspace named BOOKINGS17.DWS whose full file path is shown followed by a separator (?) and then the workspace path to the source variable. As this is Dyalog APL the namespace path within the workspace is shown from root (#).

The dialog is normally non-modal and multiple instances of the dialog may be open simultaneously. This permits users to interact with the workspace session. However, a property may be included when the function is called to specify a modal dialog which might be appropriate if the utility were to be incorporated within a user application.

I will firstly describe the fundamentals of how the editor works and then briefly cover its salient features.

The Grid Display

The top level of a generalised array is an orthogonal structure of any rank. For a non-empty array, each element of the array is a scalar. If the element is a simple scalar, either a character or a number, there is no further depth to that element. However, if it is non-simple, the item contained in that element is a further array which is said to have a depth of 1. Each non-simple element within that array contains a further array with a depth of 2. And so on, until all elements of the array at the lowest level are simple. Even for an empty array, the prototype of that item may be simple or non-simple.

The structure is visually represented by the editor using the Grid object. At any level when navigating a structure, the array at that level is displayed initially showing its last two axes as the rows and columns respectively of the Grid. By default, the axis subscripts are consecutively numbered in the row and column titles using the preferred origin. If the array is a scalar, just a single cell is shown with no titles. If it is a vector, it is usually shown row-wise with just row titles.

Axes are also identified in the pane labelled *View* just above the Grid. Next to each axis, a drop-down list contains the subscripts for that axis. The pane is split into up to three zones. The first (or *slice*) zone on a plane background contains all of the axes other than the last two axes. The middle (or *row*) zone on a background of horizontal stripes contains the penultimate axis (shown as the Grid rows). And the third (or *column*) zone on a background of vertical stripes contains the last dimension (shown as the Grid columns). Of course, if the array is less than two-dimensional, the redundant zones will not be shown, and no zone will be displayed for a scalar array.

By opening and selecting a subscript from the drop-down list for an axis, the Grid cell corresponding to a row or column axis will become current, i.e. it will be highlighted with the cursor placed in that cell, or otherwise, if the axis occurs in the slice zone, the Grid will be refreshed with the contents of the selected plane (or slice) of the sub-array.

However, the real power of this design is that axis icons may be dragged from one zone to another or re-arranged within a zone. Multiple axes may be dragged into a single zone. Thus, the view of the array may be manipulated in any way desired. An array of rank greater than 2 may be shown in its entirety within the two-dimensional Grid. It is simple to invert the view, for example, by exchanging the row and column zones. Knowledge of the chosen orientation of the array at any

particular depth is retained so that it may be re-applied when the array is re-visited. Note that such manipulations make no change to the actual variable.

Subject to declared permissions, the value that appears in a cell is directly editable if it is a simple scalar or a special-form text string (see below). Otherwise it is displayed as an icon indicating the type and rank of the enclosed item.

Plumbing The Depths

As befits an editor designed for nested arrays, there are several ways in which the structure may be navigated, either by "drilling down" to a lower level from a selected cell in the Grid, or by moving up to a higher level in the structure. These actions are supported by tool bar buttons, by short-cut keys, and by use of a popup context menu within the Grid. Drilling-down may also be achieved by double-clicking on a cell. Navigating back to a higher level may also be achieved by clicking on an index icon in the Cell/Array details pane (see below).

Attributes

When the top-level array or any sub-array is opened, the top pane beneath the tool bar is labelled *Array*, and the attributes of the array or sub-array are displayed in the pane, showing type, depth, rank and shape. In addition, the Pick index of the sub-array is shown, i.e. the vector of indices that would select the sub-array when applied as the index of the edited variable. The first index is a dummy labelled "Top", or "Whole array" at the topmost level. When a cell is selected, the pane label is *Cell* and the attributes refer to the item contained by the selected cell. By clicking on this label, or by using shortcuts Ctrl+A or Ctrl+C, one may flip between details for the current sub-array or the active cell. The current display in the pane is important when using the edit box of the Value pane.

Editing Facilities

A property may be set in the initial function call to set various restrictions on how the array may be edited, or indeed permission to edit may be denied whilst allowing the user to view the array. Editing may be limited to changing just cell data. It may additionally permit row or column insertion and deletion. Or it may permit changes to the structure. The permissions may also be set dynamically which would allow a user to protect against accidental changes. However, a full, file-based Undo/Re-do mechanism is implemented. Editing features are made available according to the permissions.

Individual cells may be edited directly providing that the cell is editable. When a cell is made active, its value is also displayed in the edit box of the Value pane. Either this value or the cell value may be edited.

If permission is set to alter the structure, the button with the equal (=) symbol may be pressed. The sub-array or the item in the current cell, whichever is shown in the Attributes pane, is then displayed in the edit box as an executable APL expression. This expression may be edited or replaced. The box may be extended to provide more rows for input. The expression is accepted (or rejected) using the buttons to the left. The result of the expression, if it is valid, will replace either the entire sub-array or the current cell item, enclosed if non-scalar. In the former case, confirmation is required.

Tool bar options allow rows or columns to be inserted before or after the current row or column, and also to delete selected rows or columns.

Cut and Paste facilities are supported between other applications, e.g. Excel, if the data is simple using the Windows Clipboard. For nested data, Cut and Paste is supported within and between other instances of the editor using a custom buffer that co-operates with the Windows Clipboard, though the support precludes export to external applications.

Data Transforms

Several transforms are optionally available to make editing and viewing certain data structures easier.

Text Vectors. Any array of text vectors may be treated as though the vectors (or text strings) were scalar items. Strictly, each character of a text vector is a scalar item and would logically be displayed in a separate Grid cell. In an array of text vectors, each vector is a nested element that would require the user to drill down to that single vector in order to view or edit it. It is obviously much more convenient to present each vector as an editable text string in a single Grid cell, in effect to treat the text string as atomic. This is in fact the default setting. Note also that a vector of text vectors (VTV) is presented by default in the rows of a single Grid column, the most natural way to view such data.

Simple Text Arrays. For the same reasoning as applied to VTVs, it is usually more convenient to present the last axis of text arrays as text strings; that is, to display each text sting in its own Grid cell. This again is the default. Trailing blanks are preserved in the strings and reconciled after edit changes.

Horizontal Parallel Vector (HPV). This is my description for a commonly used data structure in which a nested vector contains items related along the first axis. For example, the vector may represent personnel records consisting of a vector of text names, a text matrix of departments, an integer vector of ages, a text vector indicating sex, a numeric matrix indicating skill ratings, and so forth. The n'th record from each item refers to a single person. The vector may be conveniently presented with the items side-by-side across the columns and the rows indicating record numbers, greatly facilitating the editing of data. Such a layout is illustrated in the screen-shot example.

For this transform to be applied, all of the items must be of rank-2 or less, though a rank-3 text array will be treated as rank-2 if last axis strings are transformed to scalar strings as described above. The columns of a rank-2 array become sub-numbered columns of the Grid display.

A further refinement is provided in that the lengths of the first axis items need not be the same. Items shorter than the maximum will be padded with nulls that will be reconciled as prototype elements if necessary when a null item is edited. An internal mask records null items. This is termed a *permissive* HPV.

This option is not set by default, but preferences are saved and recalled in future uses of the editor.

Vertical Parallel Vector (VPV). This structure is similar to an HPV except that the items of the vector are related along the last axis. Records are displayed as the columns of the Grid, and items as the rows of the Grid. Note that for a transformed text array, records are counted along the penultimate axis, not the last axis, since the last-axis items are treated as atomic strings. The permissive form is also supported.

This option is not set by default. HPVs and VPVs are mutually exclusive though both can be set as options, the former taking precedence.

Axis and Subscript Titles

By default, axes and axis subscripts are numbered consecutively using origin-0 or origin-1 according to preference. Origin is initially chosen according to that in the calling environment and is displayed in the status bar. The `⎕IO` status is active and may be altered by clicking on it.

At any level in the structure, titles may be given for the axes and axis subscripts of the sub-array. This can be done by double-clicking on an axis icon or an axis subscript drop-down box respectively. A small edit box is presented which is

scrollable for subscripts. Alternatively, titles may be entered through a Properties dialog available through the Edit menu or status bar. A tool bar button flips between displaying numbers and titles.

Titles may be saved in the same namespace as the associated variable with the name *var_VETitles* where *var* is the name of the variable. It will be detected automatically when the variable is subsequently opened by the editor.

Help and Information

There is no Windows help file as yet. However, the status bar provides a large text area in which hints are provided for just about everything in the dialog in sufficient detail to use the particular control. In addition, two buttons to the right of the tool bar provide pop-up dialogs that provide detailed information about the menus, tool bar buttons, the various panes, Grid, status bar and all keyboard short-cuts. The web-site cited below also provides substantial help.

The VEdit Function

The calling syntax of the top-level function is as follows:

```
{a} VEdit b
```

b is the name with optional namespace path of a variable, or null. If no path is given, it is taken to be in the calling namespace. If it is null, the WSBox dialog is invoked to choose a variable from the active or any other workspace.

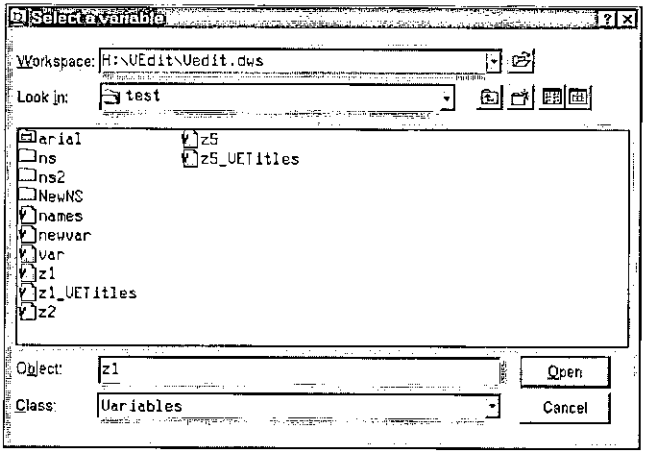
a is an optional list of properties observing the usual rules for `⎕WC`'s right argument. The keywords are:

> *Modal* , either 0 or 1, to indicate whether or not the dialog should be modal; defaults to 0.
>
> *ReadOnly*, either 0, 1 or 2; defaults to 0. If set to 2, access permission is read-only and it may not be altered from the dialog.
>
> *TypeChangeable*, either 0 or 1, to indicate whether the data type of an edited cell may be altered; defaults to 1.
>
> *StructureChangeable*, either 0 or 1, to indicate whether the structure of the variable may be changed; defaults to 1 which supersedes *TypeChangeable*.

The *VEdit* function and secondary functions reside in the *vedit* namespace except for the *WSBox* function and its associated functions which reside in the *wsbox* namespace.

The WSBox Dialog and Function

The WSBox dialog is modelled closely on the standard Windows FileBox as illustrated below.



It contains a drop-down selection list labelled *Workspace* that lists the most recently used workspaces. It has a browse button to the right that invokes the standard FileBox to select another workspace from the file system. The workspace defaults to the current active workspace.

The remainder of the dialog controls identify the namespaces and contents within the selected workspace. Namespaces and GUI objects are equivalent to FileBox folders and namespace objects are equivalent to FileBox files. Since it may be valid to select a namespace or GUI object, the convention is adopted to click on the icon to open up the namespace or GUI object and to click on the name to select it and place it in the *Object* edit box. For other objects, clicking on either icon or name will select the object.

Just as FileBox allows wildcard expressions to select files, the *Object* edit box permits wildcard expressions to identify objects, including a more refined set of name class codes to select objects as shown in Table 1. The syntax for this is to enclose single character codes in square brackets to identify the classes of objects required, e.g. *[ad] to select active and dynamic functions from the namespace.

Of course, in the context of the Variable Editor, the Class of selectable objects is limited to variables. However, WSBox is a stand-alone utility in its own namespace that may be used anywhere.

Class Nr	Code	Description
1	L	Label
20	V	Variable
21	R	Array of References (not yet implemented)
30	F	User-defined Function
31	L	Locked function
32	A	Assigned Function
33	D	Dynamic Function
34	E	External Function
40	O	User-defined Operator
41	K	Locked Operator
43	Y	Dynamic Operator
90	N	Namespace
91	G	GUI Object
92	S	Reference (not yet implemented)

Table 1: Extended Name Classes

WSBox Syntax

The WSBox function has the following syntax:

```
r←{a} WSBox b
```

a is the optional name of the parent GUI for the WSBox dialog (a Form object). It defaults to the calling namespace given by `⎕CS''`.

b is a list of properties using standard rules as for `⎕WC` similar to those for FileBox as follows:

Caption. The text to appear in the Caption Bar of the dialog.

Workspace. The full file path of the initial workspace to be selected. If this is null, the active workspace is selected. If it is 0, selection will be limited to the current workspace and the Workspace drop-down list and browse button will be omitted.

Path. The full path that identifies a namespace within the workspace. If the namespace does not exist, the contents of root will be displayed.

Objects. The name or names of objects that should be initially selected. Acceptance of multiple names depends on the *Style* setting. If a name is given with a full path, the *Path* property is superseded.

Classes. This property identifies permitted classes to appear in the Class drop-down list. It is expressed as nested pairs of items that define the text to appear in the drop-down list and a scalar or vector of numbers identifying the extended name class numbers as defined in Table 1 that should be included when that class is chosen. Standard `⎕NC` class numbers may also be included; thus 3 would identify any type of function, whereas 33 would identify just dynamic functions.

Mode. Either '*Read*' or '*Write*' where case is not relevant. In Read mode, only existing objects may be selected. In Write mode, an object name that does not exist may be specified in the Object edit box. Defaults to Read.

Style. Either '*Single*' or '*Multi*' where case is not relevant. This indicates whether just a single or multiple objects may be selected. Defaults to Single.

The result `r` provides information about the selected object or objects as follows:

The name of the selected workspace, or '' for the current workspace.

The full path from root of each of the selected objects.

The extended name classification of each object as defined in Table 1.

The Type of each object if it is a GUI object, otherwise ''.

Integration Within the Dyalog Session

An installation program is packaged with the Variable Editor (and WSBox) that displays a dialog using Property Sheets when the distribution workspace is loaded. The recommended procedure is to install a small namespace called *aftools* in the Dyalog Session (`⎕SE`). This namespace contains a few service functions plus tiny cover functions that define the Variable Editor and WSBox namespaces within the Session (*vedit* and *wsbox*) when activated and remove the namespaces when they are no longer required. Thus the normal footprint of the utilities within the workspace is almost zero. However, the utilities will work just as well if the actual namespaces are permanently defined in the Session.

The namespaces are actually stored as efficient object representations in a component file established normally in a sub-folder of the Dyalog APL folder

named *APLForum*, though an alternative location may be given. The actual location is stored in the Windows Registry under key HKEY_CURRENT_USER/Software/APLForum. Sub-keys VEdit and WSBox store user preferences such as dialog position and most-recently-used lists.

In addition, the Variable Editor may be set up in the Session Action menu and also as a tool bar button. When actioned, if the current session object is a variable it will be selected for editing; otherwise, the WSBox utility will be called to select a variable to edit. Note that double-clicking on a name in the session will invoke the standard Dyalog variable editor since this cannot be intercepted.

Facilities using the WSBox utility may also be established in the Session Action menu and as two tool bar buttons: the first to retrieve one or more objects from any workspace and define them in the current environment; and the second to copy one or more objects from any workspace to a namespace in any other. The latter makes two separate calls to FileBox and performs the copy.

In order to copy to another workspace, a small intermediary workspace will be installed in the *APLForum* folder that uses a shared-variable interface to control the process. Unfortunately, Dyalog APL does not provide a means of saving a workspace under program control without leaving an SI stack, and also possibly impairing a latent expression defined in the workspace. The final save requires user intervention through the desktop to complete the process, actually no more than executing the pre-output command: `)SAVE wsname`.

The APL Forum Web-site

The Variable Editor and WSBox utilities are available for download from a new open-source web-site committed to the free distribution of APL software contributed by members of the APL community. The utilities are available as a zipped Dyalog workspace. Information about the utilities may be viewed on the web-site and may also be downloaded as a single zipped file containing HTML and image files that may be viewed through a browser.

The web-site address is http://www.aplforum.com/.

Reference

[1] Jim Lucas, *Looking Back On Looking Ahead*, APL'2000 Madrid Proceedings, APL Quote Quad, June 2002 - Volume 32, Number 4 (ACM/SIGAPL)

At Play with J: The Magical Matrix

by Eugene McDonnell

Christ! What are patterns for?
Amy Lowell, "Patterns"

Books on combinatorial subjects seem to believe that results are obtained seriatim, that the problem is to find the next combination or permutation or partition. Such is the case in the book *Combinatorial Algorithms* [1].It is also the way the latest chapters in Knuth's *Art of Computer Programming* treat these topics [2]. Roger Hui, on the other hand, takes a more organic view, and his programs all grow an entire table from a seed. This article discusses `perm`, his algorithm for obtaining permutation tables, using a *magical matrix*. I call it that because, in studying his algorithm I stumbled hard against the critical part of his algorithm that used it, and puzzled over it for a long, long time before I could see how it worked. When I did finally understand it, all I could say was that it was magic.

Here are the tables of all the permutations from one to four:

```
+-+---+-----+-------+
|0|0 1|0 1 2|0 1 2 3|
| |1 0|0 2 1|0 1 3 2|
| |   |1 0 2|0 2 1 3|
| |   |1 2 0|0 2 3 1|
| |   |2 0 1|0 3 1 2|
| |   |2 1 0|0 3 2 1|
| |   |     |1 0 2 3|
| |   |     |1 0 3 2|
| |   |     |1 2 0 3|
| |   |     |1 2 3 0|
| |   |     |1 3 0 2|
| |   |     |1 3 2 0|
| |   |     |2 0 1 3|
| |   |     |2 0 3 1|
| |   |     |2 1 0 3|
| |   |     |2 1 3 0|
| |   |     |2 3 0 1|
| |   |     |2 3 1 0|
| |   |     |3 0 1 2|
| |   |     |3 0 2 1|
| |   |     |3 1 0 2|
| |   |     |3 1 2 0|
| |   |     |3 2 0 1|
| |   |     |3 2 1 0|
+-+---+-----+-------+
```

The bold portion of the 4-table is one plus the 3-table; that of the 3-table is one plus the 2-table; and even, quite trivially, that of the 2-table is 1 plus the 1-table. Hui's algorithm is recursive for arguments 2 or greater. For arguments 0 and 1 it simply returns `,.y.$0`, which gives an empty table with shape 1 0 when `y.` is 0, and a table of shape 1 1, having the single value 0, when it is 1. The table of order 1 is the seed used to grow all the larger tables. If we want the table of order 4 this means that we have to go back to the seed to get the table of order 2, then the table of order 3, before we work on the one we want, of order 4.

Assuming we have the table of order 3, we can get what we need by adding 1 to it, then prefixing 0 to each row:

```
   ] p3=:0,.>:perm 3    NB. prefix 0 to rows after adding 1
0 1 2 3
0 1 3 2
0 2 1 3
0 2 3 1
0 3 1 2
0 3 2 1
```

After studying the code that produced this, I was sure it was going to be worked on in such a way that three more analogous tables would be made and strung together with it to give the desired result. But as I looked more at Hui's function I couldn't believe what I saw: it appeared that it was going to be used as an *index*! What in the world was happening? The thing being indexed was table `mm`:

```
   ] mm =: \: i=i.4
0 1 2 3
1 0 2 3
2 0 1 3
3 0 1 2
```

I was sure that `mm` should be the index and `p3` the thing indexed. I tried various ways of doing this, ending with:

```
  qw=:mm{"2 1 p3
   $qw
6 4 4
   ;/qw
+-------+-------+-------+-------+-------+-------+
|0 1 2 3|0 1 3 2|0 2 1 3|0 2 3 1|0 3 1 2|0 3 2 1|
|1 0 2 3|1 0 3 2|2 0 1 3|2 0 3 1|3 0 1 2|3 0 2 1|
|2 0 1 3|3 0 1 2|1 0 2 3|3 0 2 1|1 0 3 2|2 0 3 1|
|3 0 1 2|2 0 1 3|3 0 2 1|1 0 2 3|2 0 3 1|1 0 3 2|
+-------+-------+-------+-------+-------+-------+
```

This can't be right, because, for example, the row 1 0 2 3 appears three times. In order to show you how I was at last able to understand this strange indexing, I'll show the proper table of order 4 with its four sections side by side:

```
   ;/_6]\perm 4
+-------+-------+-------+-------+
|0 1 2 3|1 0 2 3|2 0 1 3|3 0 1 2|
|0 1 3 2|1 0 3 2|2 0 3 1|3 0 2 1|
|0 2 1 3|1 2 0 3|2 1 0 3|3 1 0 2|
|0 2 3 1|1 2 3 0|2 1 3 0|3 1 2 0|
|0 3 1 2|1 3 0 2|2 3 0 1|3 2 0 1|
|0 3 2 1|1 3 2 0|2 3 1 0|3 2 1 0|
+-------+-------+-------+-------+
```

Studying this I at last saw the pattern that Hui was using. The actors are typecast in the first section, with 0, 1, 2 and 3 playing themselves. In the second, actors 0 and 1 change roles; in the third, the three actors 0, 1 and 2 play the roles of 1, 2 and 0; in the last all four actors are in disguise: 0 acts as 1, 1 acts as 2, 2 acts as 3, and 3 acts as 0. If you look at the first rows of each of the four sections, you'll see that they are exactly the rows of mm, the *magical matrix*!

Now we have to study how the magical matrix was produced. The first part gets the cast of actors:

```
   i.4
0 1 2 3
```

Classifying this gives an identity matrix:

```
   =i.4
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

And last, each row is replaced by its downgrade:

```
   ] mm=:\:"1=i.4
0 1 2 3
1 0 2 3
2 0 1 3
3 0 1 2
```

Now, if we index each row of mm with all of p3, we get an array of four 6 by 4 tables:

```
   p3{"2 1 mm
0 1 2 3
0 1 3 2
0 2 1 3
0 2 3 1
0 3 1 2
0 3 2 1

1 0 2 3
1 0 3 2
1 2 0 3
1 2 3 0
1 3 0 2
1 3 2 0

2 0 1 3
2 0 3 1
2 1 0 3
2 1 3 0
2 3 0 1
2 3 1 0

3 0 1 2
3 0 2 1
3 1 0 2
3 1 2 0
3 2 0 1
3 2 1 0
```

Last, join the tables:

```
   ,/p3{"2 1 mm
0 1 2 3
0 1 3 2
0 2 1 3
0 2 3 1
0 3 1 2
0 3 2 1
1 0 2 3
1 0 3 2
1 2 0 3
1 2 3 0
1 3 0 2
1 3 2 0
2 0 1 3
2 0 3 1
```

```
2 1 0 3
2 1 3 0
2 3 0 1
2 3 1 0
3 0 1 2
3 0 2 1
3 1 0 2
3 1 2 0
3 2 0 1
3 2 1 0
```

And this gives the desired result. The same process works for tables of all sizes greater than one.

Here is `perm` in all its flabbergasting entirety:

```
perm=: 3 : 'if.1>:y.do.,:y.$0 else.,/(0,.perm &.<:y.){"2 1\:"1=i.y. end.'
```

Sometimes I feel that Hui has an advantage over the rest of us, even more than is given him by his native intelligence. Since he wrote it all, and keeps improving it, he must have an instinctive knowledge of the performance of each of its parts, and thus can (usually) write functions that are faster than those written by the rest of us. I suspect that indexing is one of the fastest ways to do selecting, and thus `perm` is likely to be the fastest way to build permutation tables.

References

[1] *Combinatorial Algorithms*, Albert Nijenhuis and Herbert S. Wilf, Academic Press, New York, 1978

[2] Donald Knuth, home page:

http://www-cs-faculty.stanford.edu/~knuth/news.html

Pre-Fascicle 2a: Generating all n-tuples (version of 29 Aug 2003)
Pre-Fascicle 2b: Generating all permutations (version of 29 Aug 2003)
Pre-Fascicle 2c: Generating all combinations (version of 29 Aug 2003)

Taking the RTF Challenge

by Adrian Smith

The Challenge

Somewhere in your application, you might want to allow your user to type in some little notes, with basic bolding and italic. Easy stuff these days – just add a RichEdit control and save the *rtftext* property in a text variable.

The RichEdit even has a Print method, so the user can print it out. Great.

Now you want to create a nice web-report, with some headings, the user's notes, and a table of relevant data – *all in one simple stream of HTML*. Does this RichEdit have anything useful like an HTML property? Of course not. So the challenge is quite simple – take the typical output from a RichEdit control (a simple text vector with rtf markup) and convert it to an equally simple text vector of HTML.

Well, RTF is a documented format. Let's have a look at some, and see how we go.

Some Simple RTF Explained

The next block of text was inserted from a file called simple.rtf, so you can see roughly how it might have looked to the user in the RichEdit:

Sample Richtext (this is 14pt)

This is the kind of Richtext you could expect from a Richedit control. It may have some formatting like **bold** and *italic* text, and even ***both in the same place***. It may also use underlining and ~~strikeout~~, which is patchy in HTML. It is sized at 10pt (the Wordpad default) which should probably map to size=3 in the browser.

> Simple indented sections are possible, and some use of colour can be expected. This is in teal, for example. That pretty well covers it, I think.

You can do tabs in a Richedit, but we won't try these as we know there is no chance of them working in HTML, so this can stand as a reasonable testpiece for the HTML compatibility.

This was actually typed into Wordpad, but the RTF output is very similar (but not quite the same, as you will see).

Here is what it looks like if we paste it in 'native' as plain text:

```
{\rtf1\ansi\ansicpg1252\deff0\deflang1033{\fonttbl{\f0\fswiss\fcharset0
Arial;}{\f1\fswiss\fprq2\fcharset0 Arial;}}

{\colortbl ;\red0\green0\blue0;\red0\green128\blue128;}

\viewkind4\uc1\pard\f0\fs28 Sample Richtext (this is 14pt)\fs20\par
\par

This is the kind of Richtext you could expect from a Richedit control. It
may have some formatting like \b bold \b0 and \i italic \i0 text, and even
\b\i both in the same place\b0\i0 . It may also use \ul underlining \ulnone
and \cf1\strike\f1 strikeout\cf0\strike0\f0 , which is patchy in HTML. It is
sized at 10pt (the Wordpad default) which should probably map to size=3 in
the browser.\par
\par

\pard\li720\tx720\tx7920 Simple indented sections are possible, and some use
of colour can be expected. \cf2\f1 This is in teal, for example\cf0\f0 .
That pretty well covers it, I think.\par
\pard\par

You can do tabs in a Richedit, but we won't try these as we know there is no
chance of them working in HTML, so this can stand as a reasonable testpiece
for the HTML compatibility.\par
}
```

I added a couple of extra newlines for clarity, but this is pretty well what you will see if you save a document from WordPad and reopen it in Notepad.

So what can we deduce by simple inspection?

- there is some stuff at the top which tells us what fonts and colours to expect
- there is plenty of recognisable text, and markup is done with backslash characters (rather in the style of TEX, for those of you with long memories)
- those numbers look like twips, but what are the font sizes doing? They all seem to have doubled somehow??
- there are some odd inconsistencies. Clearly \b and \b0 surround bold, but what is that \ulnone about?

Time to open the manual. Actually, the best manual is now well out of print, as what you want is a very old RTF specification, ideally one that was made about the time of WinWord2 or even DOSWord4. The problem with the recent documentation is that Word2000 puts so much fancy stuff into its RTF that it has become very hard to unpick. However dear old WordPad (and by association, the RichEdit) is using a nice, ancient RTF standard, and is reasonably easy to crack.

The other thing you realise is that there is no way to parse this other than walking along it one tag at a time, maintaining state information as you go. This was designed in the days of restricted memory, and is meant to be parsed end to end by a simple character-eating machine. The exact opposite of XML which is designed to be mapped into a tree structure in memory. So don't expect things to be nicely paired up - they won't be.

Basic Grammar

There is a rudimentary block structure defined by pairs of braces. The outer level encloses the entire document, and inner levels enclose various lists. Also you can set and cancel elements by enclosing the setting in braces, for example `{\b this is bold}` is an alternative way to set a short phrase in bold. RichEdit doesn't do it like this, but WordPad does, so we should probably handle it.

Anything following \ is a control word of some sort, and does not appear in the output. The word is ended by the next space, \ or { character. The control words often consist of an alpha part concatenated to a number. The numbers are always integers, but may be negative. There may be several ways to reverse a setting, for example you could cancel bold with `\b0` or `\plain` or by wrapping it in a pair of braces, as above.

Measurements are integer twips (1 twip = 1/1440 of an inch) and font sizes are integer half-points. Ever wonder why Word limits you to half-point font-sizes? Now you know.

Colours are expressed as 0-origin indices into a global table set up at the start of the document. Fonts are effectively indices, but \f0 is reserved as 'reset to default' so your fonts number upwards from 1.

RTF is entirely written in 7-bit ASCII. Any characters above 128 appear as \'xx (where xx is the hex value) and Word has a habit of naming specials like \endash rather than using the hex code. Fortunately the Richedit just uses the \'xxx syntax so we don't need to research the names for all these.

Let's Walk Through It

At this point, it is probably best just to present my parsing code with some brief commentary interspersed. This has been adapted from the MakeHTM workspace that comes with NewLeaf, but can be run stand-alone in this form. Hopefully the actual RTF-parsing part is kept separate enough that you could easily re-use it to generate any format you need.

Setting up the state information

```
     ∇  htm←RTFParse rtf; ...
[1]    ⍝ Run vector of 'WordPad' richtext as HTML.
[2]    ⍝  <rtf> is an nl-delimited simple vector and <htm> is the same.
[3]
[4]     (nl lf)←⎕TCNL ⎕TCLF ⋄ htm←'' ⋄ →(0∈⍴rtf)↑0   ⍝ Quit if empty
[5]
[6]    ⍝ Initialise html state variables to default values
[7]     htm_tag←,c,'p' ⋄ htm_indents←3⍴0 ⋄ htm_bullet←0
          htm_last←0 ⋄ htm_level←''
```

Line-7 is the last mention of html for a while – the next part is very generic.

```
[9]    ⍝ Grab font and colour tables and then discard the rest of the header
[10]    ftbl←('{\fonttbl'⍷rtf)↓1 ⋄ ftbl←(ftbl+8)↓rtf ⋄ ftbl←(('}}'⍷ftbl)↑1)↑ftbl
[11]    ftbl←(1=+\(ftbl='{')-(ftbl='}'))/ftbl  ⍝ dump Panose etc (Word RTF)
[12]    ftbl←¯1↓(+\1,';'=ftbl)⊂';',ftbl
[13]    ftbl←(ftbl↓⍨' ')↓⍨¨ftbl          ⍝ 1st blank delimits name
[14]
[15]   ⍝ Colours - just split at the semi-colons and convert to #FF style.
[16]    ctbl←('{\colortbl'⍷rtf)↓1 ⋄ ctbl←(ctbl+9)↓rtf ⋄ ctbl←(¯2+ctbl↓'}')↑ctbl
[17]    mm←ctbl∈'0123456789;' ⋄ ctbl←mm\mm/ctbl
[18]    ctbl←1↓¨(+\1,';'=ctbl)⊂';',ctbl
[19]    ((2>⍴¨ctbl)/ctbl)←⊂'0 0 0'  ⍝ WordPad sets a default first entry as null
[20]    ctbl←⊃⍪¨ctbl                ⍝ n by 3 matrix of RGB values
[21]    ctbl←(¯2↑1 1,⍴ctbl)⍴ctbl     ⍝ may be only one colour
[22]    ctbl←⊂[2]'#',((1↑⍴ctbl),6)⍴'0123456789ABCDEF'[1+16 16⊤,ctbl] ⍝ HTML style
```

The assumption is that there could be all sorts of rubbish in here (have a look at a Word2000-generated RTF some day) so we will just look specifically for these tables, which will be in there somewhere. Here is what the tables look like:

```
      ctbl
#000000 #000000 #008080
      ftbl
Arial Arial
```

This has processed the colours into HTML format, but you could easily hold them as numbers here if that is what you needed later on.

```
[24]   ⍝ Top and tail the RTF content, ignoring stuff we can't handle
[25]   ⍝ Lose the tail, including the closing '}'
[26]    rtf←(-(⌽rtf)↓'}')↓rtf
[27]   ⍝ Now clip the rest of the garbage down to the first paragraph
[28]    rtf←(4+('\pard'⍷rtf)↓1)↓rtf ⋄ rtf←rtf~lf  ⍝ Safe to lose all these
[29]    state←'' ⋄ thispara←'' ⋄ biu←0 0 0
[30]    fcolour←1⊃ctbl ⋄ fsize←3 ⋄ font←1⊃ftbl
[31]    basefnt←(font fsize fcolour)
```

```
[32]  ⍝ Return APL symbols and other hibit things to their proper condition
[33]   hi←('\''⍷rtf)/⍳⍴rtf
[34]   :If 0<⍴hi
[35]     nn←rtf[hi∘.+2 3] ⍝ hex digits (lower case)
[36]     rtf[hi]←'∓'      ⍝ lose the \ (later, when we know we can)
[37]     rtf[hi+1]←htm⍙hibit[¯127+16 16⊥¯1+⍸'0123456789abcdef'⍳tolower nn]
[38]     rtf←rtf[(⍳⍴rtf)~,hi∘.+2 3]
[39]   :End
[40]  ⍝ Maybe rescue \endash and so on here as string replacements
[41]  ⍝ Hide escaped \ for now - they get rescued later
[42]   rtf←rtf ⍙ssr '\\' '⍙'
[43]   rtf←rtf ⍙ssr '\{' '⍗' ⋄ rtf←rtf ⍙ssr '\}' '⍔'
[44]
```

We now have all the state variables at their default values, and the only \ characters left are 'real' ones which actually mark tags. The choice of APL symbols to act as placeholders is a little arbitrary, but you will notice that all these are 'blots' in the Arial font, so are most unlikely to show up in a real document.

Processing the content

Now to eat the text, one tag at a time.

```
[45]  ⍝ Now we walk down the string - either process tag or place text
[46]   :While 0<⍴rtf
[47]     :If '\'=1↑rtf  ⍝ Handle next tag and drop it
[48]       rtf←1↓rtf ⋄ cut←¯1+⍸'/rtf⍳' \{}∓',⎕nl
[49]       tag←cut↑rtf ⋄ rtf←cut↓rtf ⋄ rtf←((1↑rtf)∈' ',⎕nl)↓rtf
[50]       cut←¯1+⍸'/tag⍳'1234567890-' ⋄ arg←cut↓tag ⋄ tag←cut↑tag
[51]      ⍝ RTF only has integers (half-point fonts and twips)
[52]       :If 0<⍴arg
[53]       :AndIf ∧/arg∈'1234567890-'
[54]         arg←⍎arg
[55]       :End
[56]
```

This has split out the tag into the text token, and the argument as a number. Now we just take them as they come, starting with the paragraph formatters:

```
[57]       :Select tolower tag ⍝ should all qualify!
[58]       :Case ,'*'      ⍝ Forget it
[59]       :Case 'pard'    ⍝ Reset to all defaults
[60]         htm_tag←,⊂,'p' ⋄ htm_indents[]←0 ⋄ htm_bullet←0
[61]       :Case 'par'     ⍝ New para - flush all output
[62]         :if 0<⍴thispara
[63]           thispara←Flush thispara
[64]           htm←htm,thispara,⎕nl ⋄ thispara←''
[65]         :end
[66]       :Case 'qc'      ⍝ Centered paragraph
```

```
[67]          (1⊃htm_tag)←(1⊃htm_tag),' align=center'
[68]        :Case 'qr'      ⍝ Right paragraph
[69]          (1⊃htm_tag)←(1⊃htm_tag),' align=center'
[70]        :Case 'li'      ⍝ Left indent in twips
[71]          htm_indents[1]←⌊arg÷20
[72]        :Case 'ri'      ⍝ Right indent (not used in basic HTML)
[73]          htm_indents[2]←⌊arg÷20
[74]        :Case 'fi'      ⍝ Firstline indent (also ignored)
[75]          htm_indents[3]←⌊arg÷20
```

I keep the indent information, although it is not currently interesting for basic HTML. With inline CSS styles, it could be used to get more detailed control if you thought it worth the effort. Indents are saved in points, but actually all we care about is stepping in and stepping out. We will come back to `Flush` shortly.

Now for the character formatters:

```
[76]        :Case ,'b'      ⍝ Bold on/off is b or b0
[77]          :If 0=1⊃arg
[78]            thispara←thispara,'</b>' ⋄ biu[1]←0
[79]          :Else
[80]            biu[1]←1 ⋄ thispara←thispara,'<b>'
[81]          :End
[82]        :Case ,'i'      ⍝ Italic on/off
[83]          :If 0=1⊃arg
[84]            thispara←thispara,'</i>' ⋄ biu[2]←0
[85]          :Else
[86]            biu[2]←1 ⋄ thispara←thispara,'<i>'
[87]          :End
[88]        :Case 'ul'      ⍝ underline on
[89]          biu[3]←1 ⋄ thispara←thispara,'<u>'
[90]        :Case 'ulnone' ⍝ underline off
[91]          biu[3]←0 ⋄ thispara←thispara,'</u>'
[92]        :Case 'strike' ⍝ strikethrough is ignored
[93]        :Case 'plain'  ⍝ cancel anything we set before
[94]          thispara←thispara,∈(⊂'</'),¨(biu/'biu'),¨⊂'>' ⋄ biu[]←0
[95]        :Case 'tx'      ⍝ Tab settings are futile, so ignore
[96]        :Case 'tab'     ⍝ Skip to next tab stop (dummy also)
[97]          thispara←thispara,' ... '
[98]        :Case 'cf'      ⍝ Set the colour (as #FF style)
[99]          fcolour←(arg+1)⊃ctbl
[100]       :Case ,'f'      ⍝ Set the font with emphasis and size
[101]         font←(arg+1)⊃ftbl
[102]       :Case 'fs'      ⍝ Set the font size (relative 1-7)
[103]         arg←0.5×arg ⋄ fsize←1⌈7⌊arg+.≥6 8 10 14 24 32 44
```

The only tricky one is /plain which turns off any of bold, italic and underlined. The mapping from font sizes to internet sizes is very subjective – adjust the numbers to taste.

Finally a couple of oddballs – bulleted paragraphs in WordPad are marked with a specific tag which we process as a special case:

```
[104]      :Case 'pntxtb' ⍝ Use simple unordered list
[105]        rtf←(rtf↓'}')↓rtf ⋄ htm_bullet←1
[106]      :Case 'pntext' ⍝ Ignore symbol bullets - use built-in capability
[107]        rtf←(rtf↓'}')↓rtf
[108]      :CaseList 'pnlvlblt' 'pnindent' 'pn' 'pnf' 'lang'  ⍝ Ignore these
[109]      :Else
[110]        'unprocessed tag'tag arg
[111]      :EndSelect
```

Anything we missed just gets an echo to the session.

Groups must push and pop a state stack

Now we deal with the groups (marked out by braces) and discard any spare carriage returns, which are just decoration in RTF:

```
[112]    :ElseIf '{'=1↑rtf ⍝ Push state
[113]      state←(⊂fsize biu),state
[114]      rtf←1↓rtf
[115]    :ElseIf '}'=1↑rtf ⍝ Pop state (cancel open tags)
[116]      (fsize biu)←1⊃state ⋄ state←1↓state
[117]      rtf←1↓rtf
[118]    :ElseIf (1↑rtf)∈nl,lf   ⍝ May be regarded as noise here!
[119]      rtf←1↓rtf
```

Wrap it with font info if we need to

Finally, we are down to a block of text. The substituted escapees can be restored, and the placeholder for restored hi-bit text can be removed. If the font does not match the document default, we wrap it all in an HTML font setting, otherwise just add it to out current paragraph and move on.

```
[120]    :Else  ⍝ Buffer up plain text up to next tag
[121]      cut←¯1+(rtf='\')⍳1 ⋄ txt←cut↑rtf ⋄ rtf←cut↓rtf ⋄ txt←txt~lf,nl
[122]      ((txt='Ā')/txt)←'\' ⋄ ((txt='Ē')/txt)←'{' ⋄ ((txt='Ī')/txt)←'}'
[123]      txt←txt~'Ŧ'
[124]      :If ~(font fsize fcolour)≡basefnt ⍝ Throw out a font change
[125]        thispara←thispara,'<font'
[126]        thispara←thispara,(~font≡1⊃basefnt)/' face="',font,'"'
[127]        thispara←thispara,(~fsize≡2⊃basefnt)/' size=',⍕fsize
[128]        thispara←thispara,(~fcolour≡3⊃basefnt)/' color="',fcolour,'"'
[129]        thispara←thispara,'>',txt,'</font>'
[130]      :Else  ⍝ Just keep on truckin'
[131]        thispara←thispara,txt
```

```
[132]       :End
[133]     :End
[134]   :EndWhile
      ∇
```

Basically, this is just mind-numbingly tedious. You would expect it to be fairly slow, but actually it processes quite acceptably as long as the majority of the content is just plain text – it skips from tag to tag rather than checking character by character.

Wrapping the completed paragraph

If you scan back a couple of pages, you will notice that in the processing block for \par we actually convert the accumulated paragraph to HTML, wrapped as required. For plain text this is easy enough, but various indent styles need careful handling as the HTML needs paired markers. Here are the functions to do this part:

```
      ∇ htm←Flush txt
[1]     ⍝ Generate a paragraph of HTML, noting various state variables
[2]     ⍝ These all preface htm_ and are maintained in the calling code.
[3]
[4]     htm←Makelist ⍝ Null unless we have some indents to handle
[5]
[6]     htm←htm,(,/'<',¨htm_tag,¨'>'),¨(⊂txt),¨,/'<',¨(⍬⍴Backout¨htm_tag),¨'>'
[7]
[8]     htm←⊂htm
      ∇


      ∇ r←Backout tag
[1]     ⍝ Reverse effect of any tag
[2]     r←'/',(¯1+tag⍳' ')↑tag
      ∇


      ∇ htm←Makelist;style;tag;nl
[1]     ⍝ Process list tags, comparing with previous indent
[2]     ⍝ Effect is to step in and out of <ul> etc
[3]
[4]     htm←'' ⋄ nl←⎕TCNL
[5]
[6]     :if htm_indents[1]>htm_last
[7]         :If htm_bullet
[8]             htm_tag←,⊂'li' ⋄ tag←'ul type=disc'
[9]         :else
[10]            tag←'blockquote'
[11]        :end
[12]        htm_level←htm_level,⊂(¯1+tag⍳' ')↑tag
[13]        htm←⊂'<',tag,'>',nl
```

```
[14]      htm_last←htm_indents[1]
[15]      :return
[16]   :end
[17]
[18]   :if (htm_last>0)^htm_indents[1]=0  ⍝ Reset to normal text
[19]      htm_last←0 ⋄ htm_tag←,⊂,'p'
[20]      htm←(∊(⊂'</'),¨(⍕htm_level),¨'>'),⎕nl
[21]      htm_level←0⌈htm_level
[22]      :return
[23]   :end
[24]
[25]   :if htm_indents[1]<htm_last       ⍝ Back out one level
[26]      :if 0<htm_level
[27]         htm←∊'</',(↑⍕htm_level),'>',⎕nl
[28]      :end
[29]      htm_last←htm_indents[1] ⋄ htm_level←¯1↓htm_level
[30]      :if 0=⌈htm_level ⋄ htm_tag←,⊂,'p' ⋄ :end
[31]      :return
[32]   :end
    ∇
```

The Proof of the Pudding

Is the html we get back from it ...

```
<p><font size=4>Sample Richtext (this is 14pt)</font></p>
<p>This is the kind of Richtext you could expect from a Richedit control. It
may have some formatting like <b>bold </b>and <i>italic </i>text, and even
<b><i>both in the same place</b></i>. It may also use <u>underlining </u>and
strikeout, which is patchy in HTML. It is sized at 10pt (the Wordpad
default) which should probably map to size=3 in the browser.</p>
<blockquote>
<p>Simple indented sections are possible, and some use of colour can be
expected. <font color="#008080">This is in teal, for example</font>. That
pretty well covers it, I think.</p>
</blockquote>
<p>You can do tabs in a Richedit, but we won't try these as we know there is
no chance of them working in HTML, so this can stand as a reasonable
testpiece for the HTML compatibility.</p>
```

Which looks fine in the browser. If you want to grab the workspace from the Vector resources page and test it with some tougher examples, please do. There are a few interesting possibilities for enhancements, for example you could implement tabs with HTML tables, or do rather more with the formatting using CSS capabilities. However the basic RTF chomper should not have to change at all to add more bells and whistles to the generated output.

R.net: A Reverse Polish Interpreter for .Net

by Richard Smith (richard@redcorona.com)

Introduction

Some of you may already know about the original R, which was essentially my 'learning exercise' for Java. For those of you (probably most) who haven't, I will run through the basic features of the language here. (If you want to try it out as you go through, you can download the latest copy from *http://www.apl-385.demon.co.uk/rcs/programs/r/net.htm*.)

Firstly, it is a reverse Polish (or stack-based) interpreter, like PostScript, meaning that everything is pushed onto a stack and functions run on the items at the top of the stack. For example, to perform 3+5 in R, you would type `3 5 +`. This pushes a 3, pushes a 5, and then runs the function `+` on the top two items, pushing the result (8) onto the stack as a result. (In effect, everything only has left arguments.) This is because (a) it is much easier to program, and (b) it is nice to be different. I don't believe it would be too hard to adapt Roger Hui's amazing technicolor parse table (*Vector* 9.4, p.85) to get APL syntax.

Secondly, it has native support for vectors (as befits a language written in a house exposed to APL for so many years). For example, try `[3 4] [2 3] *` and you will see `(6 12)`. It will scalar extend, too; `5 #i 1 +` will produce `(2 3 4 5 6)` as you would expect.

Finally, because it is in genuine .Net assemblies, you have access to all the classes in the Framework; for example `$System.Console "Hello, world" :WriteLine` will work just as it should.

Arrays

Arrays in R are bounded by *square brackets*, and can be nested as deeply as you like. You can create heterogeneous arrays – such as `["Richard Smith" 19 "Gilling"]` – but there is little you can do with these apart from use them as a data store. Homogenous arrays (which are vectors, vectors of vectors etc), particularly of numbers, are much more useful as any simple primitives will also work on arbitrarily nested arrays; for example:

```
  [2 3 4 [5 6 7]] [[3 4 5] 1 1 [2 1 [2 1]]] *
[(6 8 10) 3 4 [10 6 (14 7) ] ]
```

R has the usual array-manipulation features - `#get`, `#take`, `#drop`, `#resize` and `#reverse` do what you would expect. You can also flatten an array with `#apop` (effectively removing one level of nesting), and wrap up a number of values with `#apush`, making it possible to build arrays in the code as well.

Functions and Variables

Of course no language is practical without being able to store and process data. R's workhorse equivalent to the assignment arrow is `#def`, which is used in exactly the same way to define variables (`"qq" 13.5 #def`) and functions (`"AddTwo" {2 +} #def`). The difference is that function blocks are delimited by *curly brackets*; any assignment of a function block results in a function. When a token is resolved, a variable will have its content placed on the stack, whereas a function will be run.

You may well want to create multi-line functions, and both the command-line and windowed versions of R let you do this at the session (as long as there is a brace open, anything you type is added to the block). You can also use `#ed` in the windowed version, which lets you edit pre-existing code as well as create new functions.

Function blocks have other uses besides simply being assigned to names, although that is the most common. Control structures also use them, as do operators (e.g. `10 #i {+} #reduce`) and exception handlers (see Using .Net).

Control Structures

R supports two flow-of-control functions (`#if` and `#ifelse`) and four types of loop (`#for`, `#forall`, `#loop` and `#repeat`). If and if-else blocks should be familiar (for example `1000 > {$System.OutOfMemoryException "You can't ask for that many!" #create #error} #if`), but some of the loops will be less so (except to PostScript aficionados). `#for` takes 3 numbers - a start value, a step and an end value - rather like Basic's `FOR I = 1 TO 10 STEP 2`. For example:

```
   1 2.5 8 {} #for
1 3.5 6
```

`#forall` works like a C# `foreach` statement, executing the block once for each item in an array. For example:

```
   [1 2.5 8] {2 +} #forall
3 4.5 10
```

`#loop` repeats the block indefinitely, and `#repeat` repeats it a given number of times. Now, at this point you may well be saying "Why would you want to run an infinite loop?", and yes, you can work the CPU quite hard with `{} #loop`. Of course, the answer is that you may well want to keep going until something happens, and when it does you will want to escape. The escape is provided by `#exit`, which leaves the innermost enclosing loop (think `break` in C); you can also use `#continue` which jumps back to the start of the loop (once again, think of the C `continue`).

Using .Net

Because R.net is in real .Net assemblies, you can get at all the classes Microsoft kindly provides as part of the Framework. The name of a class starts with a $ sign (e.g. `$System.Console`), and you can access static methods and fields directly using *colon syntax;* for example `$System.Console "Hello, world, from R!" :WriteLine` will call the WriteLine method with one parameter, the string *Hello, world, from R!.* If you want to supply more than one parameter, use an array. For example:

```
   "System" #using
   $Int32 ["7F" $Globalization.NumberStyles:HexNumber] :Parse
127
```

This example also shows that you can reference fields without including a parameter list between the class and the colon (this works for methods with no argument list too), and the `#using` function, similar to C#'s `using` statement, to make use of the `System` namespace.

Of course you will want to do more than just use static members of classes - you also need to be able to instantiate a class (create an object). You do this with the `#create` function, which takes a class and the arguments to its constructor and returns an instance of that class. You can then use colon syntax to get at the non-static members of the object. For example:

```
   "RCS" $System.DateTime [1983 10 04] #create #def

   RCS
04/10/1983 00:00:00
   RCS:Year
1983
   RCS 20:AddYears
04/10/2003 00:00:00
```

Exception Handling

All errors in R.net are thrown as .Net exceptions. For example:

```
   2 "a" +
System.InvalidCastException: Operands of type String and Integer
      cannot be passed to Plus
   at R.CoreFunctionality.FastPrimitives.Plus(IEngine engine, Symbol
      op1, Symbol op2, Boolean invert) in
c:\DATA\SharpDevelop\R\CoreFunctionality\FastPrimitives.cs:line 523
   at R.CoreFunctionality.ExposedInterface.Plus(IEngine engine) in
c:\DATA\SharpDevelop\R\CoreFunctionality\ExposedInterface.cs:line 23
   at R.Interpreter.Engine.ExecuteBlock(ArrayList tokens) in
c:\DATA\SharpDevelop\R\Interpreter\R.cs:line 492
   at Session[1]
```

This is a complete .Net exception stack trace (including debugging info for the actual C# code), followed by the R function stack at the moment the exception was thrown. It may seem trivial here, but if we have an error somewhere deep in our code it can be very useful:

```
System.MissingMemberException: Undefined token .exit
   at R.Interpreter.Engine.ExecSymbol(String name) in
c:\DATA\SharpDevelop\R\Interpreter\R.cs:line 405
   at R.Interpreter.Engine.ExecuteBlock(ArrayList tokens) in
c:\DATA\SharpDevelop\R\Interpreter\R.cs:line 489
   at #if[1]
   at #loop[2]
   at Eratos[13]
   at Session[1]
```

This also means that exceptions from external calls fit into the R error model, and produce a similar stack trace:

```
   $System.Int32 "hello" :Parse
System.FormatException: Input string was not in a correct format.
   at R.Interpreter.Engine.ExecuteBlock(ArrayList tokens) in
c:\DATA\SharpDevelop\R\Interpreter\R.cs:line 573
   at Session[1]
```

Now, as well as seeing what you did wrong, you might want to catch the exception and do some other processing. This is accomplished with the `#trap` function, which works in a similar way to the `try ... catch` blocks in other languages - you can trap for a number of different exceptions and provide handlers for each. A good example is reading properties that are supposed to be numbers, where you might do something like `{ "num" $Int32 str :Parse #def }`

`[[$FormatException { "num" 0 #def}]] #trap` to use a default value where it was not numeric. Or you might want a file loading routine like this:

```
"LoadFile" {
 // Load a text file and return its content
 #pushdict
 "fnm" #exch #def
 {
  "sr" $IO.File fnm :OpenText #def
  "res" "" #def
  {
   "thisline" sr:ReadLine #def
   thisline #null = {#exit}{"res" res thisline ; '\n' ; #def} #ifelse
  } #loop
  res
 } [ [$IO.FileNotFoundException { "File " fnm ; " was not found" ;
#output "" }]
    [$Exception {"Unexpected error occured!" #output ""} ] ] #trap
 #popdict } #def
```

Dictionaries

The `LoadFile` example also shows how scope is managed in R. Instead of having localisation at the function level, it uses *dictionary scope* (like PostScript), where names are saved in the topmost dictionary. The `#pushdict` function pushes a new dictionary onto the top of the stack, effectively localising everything following it, and `#popdict` removes the dictionary *and its contents*, leaving the symbol table in the state it was before `#pushdict` was run.

This means that you can control (and see) exactly where localised areas of code are; however, it has a couple of bad points – firstly, if an error occurs inside a local region, you have to manually `#popdict` the dictionaries; and secondly, it is not possible to save data in the base dictionary from within a local region (you have to leave it on the stack until the region is left, and then save it). Neither of these drawbacks is terribly severe.

Namespaces

R.net supports 'shallow' namespaces, i.e. there is only one symbol table and system state, which pervades all the namespaces. In fact a namespace is just a way of avoiding name clashes, and a namespace exists only because it contains data. For example, type `"one.qq" 23 #def` and you will automatically create a `one` namespace (it appears in the tree in the IDE); erase the variable and it will disappear again.

A function can call functions and variables in other namespaces by using a relative path between them (using dots to separate namespaces, @ for root and @@ for parent) – for example:

```
    "base" { one.fn } #def
    "one.fn" { two.fn } #def
    "one.two.fn" { @@.var } #def
    "one.var" 17 #def
    base
17
```

You can move around the namespace tree using #cs (similar to Dyalog's)cs), and find out where you are with #curns. For example, you could have coded the previous example as:

```
    "base" { one.fn } #def
    "one" #cs
    "fn" { two.fn } #def
    "two" #cs
    "fn" { @@.var } #def
    "@@.var" 17 #def
    "@" #cs
    base
17
```

You can see what namespaces are defined by using #spaces, or by looking at the tree in the IDE.

Loading and Saving

R uses scripts as its storage system, and a script is just a plain text file containing executable R code. For example, here is a script which gets prime numbers by a very well-known route:

```
// R Script created by the ScriptWriter 2.1
// Saved at 07/09/2003 20:48:39

"@" #cs
"Eratos" {#pushdict
"max" #exch #def

"primes" [1] #def
"candidates" max #i 1 #drop #def

{
 candidates #size [0] = {#exit} #if
 "check" candidates 1 #get #def
```

```
  check "" ;#setstatus
  "primes" primes check , #def
  "candidates" 0 candidates check % < candidates #exch #compress #def
 } #loop

 primes
 #popdict} #def
```

As you can see, this one was actually saved by the ScriptWriter (the `#save` function in the interpreter), which saves all functions and variables present in a script file. You can also see that it is perfectly possible to edit the scripts in a standard text editor. To run a script, use the `#run` function.

Of course, there is no reason why a script cannot contain executable code as well as (or instead of) `#def` calls - in this case the code will be run when `#run` is executed, and then discarded.

Conclusion

R.net has certainly fulfilled its primary objective, which was to teach me how to use C# and get to grips with the .Net Framework. I think it also does quite well on its secondary, namely to be a moderately useful programming tool. With its support for vectors and native calls to the Framework it fills a gap which is currently unfilled, since although Dyadic are moving towards .Net they do not yet have a fully managed, small (the whole of R is only 220Kb, 50Kb of which is the documentation) interpreter in which it is easy to call the rest of the Framework.

There are a few things still on the wish list - support for delegates, multidimensional arrays and deep arrays in the interface to the Framework, images in the treeview in the IDE (although this is actually SharpDevelop's bug!) and better resumption of execution after error - but overall I am pleased with it.

[Note on efficiency - Adrian Smith] When Richard first put this together, he naturally used all the nice stuff in C# that lets you handle 'any old array'. It worked nicely on `12 #i` and we were pretty pleased with it. Then I suggested the Eratos test, which will run `qq←Eratos 40000` in about 3.2 sec in APL+WIN. It ran in around 2 minutes 4 sec. Interesting - I think we learned something.

Now what it does is use 'template' code and build variations of the source-code for the primitives to match exactly each combination of types in the incoming data. The resulting DLL is a little larger, but it does `40000 Eratos "qq" #def` in under 4 sec now, so it is definitely back in the right ballpark. Phew!

Index to Advertisers

MicroAPL Ltd	2
Strand Software	6
Vector Back Numbers	8

All queries regarding advertising in VECTOR should be made to Gill Smith, at 01439-788385, Email: apl385@compuserve.com.

Submitting Material to Vector

The Vector working group meets towards the end of the month in which Vector appears; we review material for issue n+1 and discuss themes for issues n+2 onwards. Please send the text of submitted articles (hardcopy with diskette as appropriate) to the Vector Working Group via:

Vector Administration, c/o Gill Smith
Brook House
Gilling East
YORK, YO62 4JJ
Tel: +44 (0) 1439-788385
Email: apl385@compuserve.com

Authors wishing to use Word for Windows should contact Vector Production for a copy of the APL2741 TrueType font, and a suitable Winword template. These may also be downloaded from the Vector web site at www.vector.org.uk

Camera-ready artwork (e.g. advertisements) and diskettes of 'standard' material (e.g. sustaining members' news) should be sent to Vector Production, Brook House, Gilling East, YORK YO62 4JJ. Please also copy us with all electronically submitted material so that we have early warning of possible problems.

Subscribing to Vector

Your Vector subscription includes membership of the British APL Association, which is open to anyone interested in APL or related languages. The membership year runs from 1st May to 30th April. The British APL Association is a special interest group of the British Computer Society, Reg. Charity No. 292,786

Name: _____
Address: _____
_____
Postcode / Country: _____
Telephone Number: _____
Email Address: _____

UK private membership	£20	☐
Overseas private membership	£22	☐
Airmail supplement (not needed for Europe)	£4	☐
UK Corporate membership	£100	☐
Corporate membership overseas	£135	☐
Sustaining membership	£500	☐
Non-voting UK member (student/OAP/unemployed)	£10	☐

PAYMENT – in Sterling or by Visa/Mastercard/JCB only

Payment should be enclosed with membership applications in the form of a UK Sterling cheque to "The British APL Association", or you may quote your Mastercard, Visa or JCB number.

I authorise you to debit my Visa/Mastercard/JCB account

Number: ⌊⌊⌊⌊⌋ ⌊⌊⌊⌊⌋ ⌊⌊⌊⌊⌋ ⌊⌊⌊⌊⌋ Expiry date: ⌊⌊⌋/⌊⌊⌋

for the membership category indicated above,

☐ annually, at the prevailing rate, until further notice
☐ one year's subscription only

Data Protection Act:
The information supplied may be stored on computer and processed in accordance with the registration of the British Computer Society.

Signature: _____

Send the completed form to:
BAA, c/o Rowena Small, 12 Cambridge Road, Waterbeach, CAMBRIDGE CB5 9NJ, UK
Fax: +44 (0) 1653 697719

The British APL Association

The British APL Association is a Specialist Group of the British Computer Society. It is administered by a Committee of officers who are elected by a postal ballot of Association members prior to the Annual General Meeting. Working groups are also established in areas such as activity planning and journal production. Offers of assistance and involvement with any Association matters are welcomed and should be addressed in the first instance to the Secretary.

2003/2004 Committee

Chairman	Adrian Smith 01439-788385 apl385@compuserve.com	Brook House Gilling East YORK YO62 4JJ
Secretary	Anthony Camacho 0117-973-0036 acam@blueyonder.co.uk	11 Auburn Road Redland BRISTOL, BS6 6LS
Treasurer	Nicholas Small 01223-570850 treas.apl@bcs.org.uk	12 Cambridge Road, Waterbeach, Cambridge CB5 9NJ
Journal Editor	Stefano Lanzavecchia stf@apl.it	c/o APL Italiana Corso Vercelli 54 20145 - Milano Italy
Projects and Publicity	Dr Alan Mayer 01792-205678x4274 a.d.mayer@swansea.ac.uk	European Business Management School, Swansea University, Singleton Park, SWANSEA SA2 8PP
Webmaster	Bob Hoekstra 01483-771028 bob.hoekstra@hoekstrasystems.ltd.uk	Dominique, Salisbury Road WOKING, Surrey GU33 7UR
Activities	Jon Sandles 01904-612882 jon_sandles@csi.com	138 Burton Stone Lane, YORK YO30 6DF
School Project	Stephen Taylor sjt@lambenttechnology.com	
Administration	Rowena Small 01223-570850 treas.apl@bcs.org.uk	12 Cambridge Road, Waterbeach, Cambridge CB5 9NJ

Journal Working Group

Editor:	Stefano Lanzavecchia	see above
Production:	Adrian & Gill Smith	Brook House, Gilling East, YORK (01439-788385)
Advertising:	Gill Smith	Brook House, Gilling East, YORK (01439-788385)

Support Team: Jonathan Barman (01488-648575), Anthony & Sylvia Camacho (0117-973-0036), Ray Cannon (01252-874697), Stephen Taylor (077-1340-0852) Bob Hoekstra (01483-771028), Jon Sandles (01904-612882), Marc Griffiths

Typeset by APL-385 with MS Word for Windows

Printed in England by Short-Run Press Ltd, Exeter

VECTOR

VECTOR is the quarterly journal of the British APL Association and is distributed to Association members in the UK and overseas. The British APL Association is a Specialist Group of the British Computer Society. APL stands for 'A Programming Language' - an interactive computer language noted for its elegance, conciseness and fast development speed. It is supported on most mainframes, workstations and personal computers.

SUSTAINING MEMBERS

The Committee of the British APL Association wish to acknowledge the generous financial support of the following Association Sustaining Members. In many cases these organisations also provide manpower and administrative assistance to the Association at their own cost.